I-DO: A “Deformable Organisms” framework for ITK

Release 0.50

Chris McIntosh and Ghassan Hamarneh

July 21, 2006

Medical Image Analysis Lab
School of Computing Science, Simon Fraser University
Burnaby, BC, Canada
{cmcintos,hamarneh}@cs.sfu.ca

Abstract

Medical image analysis is an important problem relating to the study of various diseases. Since their introduction to MICCAI in 2001, ”deformable organisms” have emerged as a fruitful methodology with examples ranging from 2D corpus callosum segmentation to 3D vasculature and spinal cord segmentation. Essentially we previously have developed an artificial life framework that complements the geometrical and physical layers of classical deformable models (snakes and deformable meshes) with high-level behavioral and cognitive layers that facilitate anatomically-driven control mechanisms. This paper describes the integration of deformable organisms into the Insight Toolkit (ITK) www.itk.org. In our proposed implementation we attempt to bridge the ITK framework and coding style with deformable organism design methodologies. In the interest of open science, as the framework develops it will serve as a basis for the community to develop new deformable organisms as well as experiment with those recently published by our group. Further, as the design of the ITK Deformable Organisms (I-DO) is highly modular, researchers and developers can exchange components (spatial objects, dynamic simulation engines, image sensors, etc) allowing in the future for fast development of new custom deformable organisms for different clinical applications.

Contents

1 Introduction **2**
1.1 ITK Deformable Organisms: Motivation and Introduction 4
1.2 DOs Requirements 4

2 Implementation **4**
2.1 Organism 4
2.2 Control Center 5
2.3 Sensor 6
2.4 Behavior 6
2.5 Physics 7
2.6 Deformations 7
2.7 Geometric 7

3 Conclusions **8**

4 Acknowledgements **8**

A Requirements **8**

B Examples **8**
B.1 Layer Examples 8
B.2 Deformable Organism Examples 9

C The Visual Interface to I-DO **9**

D Guide to users **10**
Hello I-DO 10
Building A Deformable Organism 10
Extending Existing DOs 13
Creating New DOs and Layers 13

1 Introduction

In medical image analysis strategies based on deformable models, controlling the deformations of the models is a desirable goal to produce proper segmentations. Incorporating expert knowledge to automatically guide deformations cannot be easily and elegantly achieved using the classical deformable model low-level energy-based fitting mechanisms. Deformable Organisms (DOs), are a decision-making framework for medical image analysis that complements bottom-up, data-driven deformable models with top-down, knowledge-driven mode-fitting strategies in a layered fashion inspired by artificial life modeling concepts. Intuitive and controlled deformations are carried out through behaviors. Sensory input from image data and contextual knowledge about the analysis problem govern these different behaviors.

Since their introduction in 2001 [3], various DOs-based approaches for medical image analysis have been developed (Figure 1). In this original work, a variety of DOs where demonstrated with applications to locating the lateral ventricles, caudate nuclei, and putamina structures in transversal brain magnetic resonance image (MRI) slices, as well as DOs for the segmentation of vessels in 2D angiography. In [4], DOs were augmented to include physically-based and controlled deformations demonstrating an application to corpus callosum segmentation in mid-sagittal magnetic resonance images (MRI). Recently, DOs were extended to 3D and applied to vascular segmentation and analysis. The so called ‘vessel crawlers’ were equipped with sensors, decision modules, and deformation layers suited for vasculature [7]. An extension of that work introduces DOs for spinal cord segmentation and analysis and demonstrates the ability to efficiently replace modules of existing DOs to create new solutions. The ‘spinal crawlers’ no longer possessed a decision module to detect branching and their sensors were adapted to detect elliptical cross sections [6]. In each case DOs have demonstrated their key advantages over other leading techniques. Namely, their ability to produce increased accuracy, allow intuitive user-interaction to control or repair the segmentation where other methods would require being restarted with some type of parameter adjustment, facilitate greater analysis and labeling abilities than those methods producing binary outputs, the ready ability to incorporate image or shape-based prior-knowledge, and a modular framework allowing for incorporating new sensors (image filters), decision models, shape representations, and deformation mechanisms.

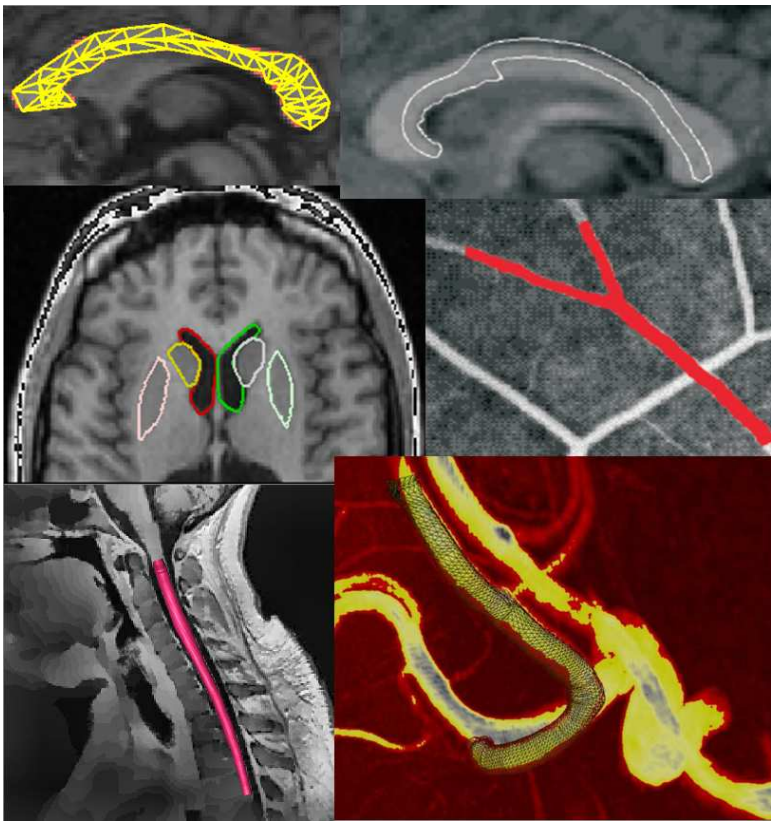


Figure 1: An assortment of deformable organisms showing(left to right, top to bottom): Physically-based corpus callosum, Geometrically-based corpus callosum, Putamina and ventricle organisms, 2D Angiography, 3D 'spinal crawler', and 3D 'vessel crawler'. Related images and videos can be found at http://mial.fas.sfu.ca/researchProject.php?s=157

Though a summary is provided here, complete research-oriented look at DOs can be found in [5]. DOs are built following a multilevel AL modelling approach consisting of four primary layers: ***cognitive, behavioral, physical, and geometrical***. Specifically, the cognitive layer makes decisions based on the DOs current state, anatomical knowledge, and its surrounding environment (the image). Decisions could be made to sense information, to deform based on sensory data, to illicit help from the user, or to terminate the segmentation process. All of these actions are described under the behavioral layer of the organism, and they rely upon both the physical and geometrical layers for implementation. For example, in the context of our 'vessel crawlers' [7], the act of moving towards a sensed target location is described by the *'growing'* behavioral method. The cognitive center gathers sensory input using the *'sense-to-grow'* sensory module, decides the correct location via the *'where-to-grow'* decision module, elicits the act of *'growing'* , and then conforms to the vascular walls by *'fitting'*. In turn, each of these methods relies upon the physical and geometrical layers to carry out tasks, such as maintaining model stability. Consequently, we have a framework with many independent layers of abstraction, each built upon the implementation of independent modules and or processes.

We begin with a motivation of our ITK-Deformable Organisms (I-DO) framework in section 1.1, and a discussion of the general requirements of DOs that the framework is set out to meet in 1.2. Sections (2.1-2.7) provide an overview of how each layer is designed and implemented in the framework. We summarize in section 3. The appendices provide the most information on using the framework with a requirements listing

(section A), examples of layers and organisms (section B), a description of our visual interface (section C), a guide to building and running your first organism (section D), and information on extending organisms and the framework (section D).

1.1 ITK Deformable Organisms: Motivation and Introduction

Previously, the major drawback of DOs has been their restriction to a closed-source MATLAB framework. Though straightforward and intuitive in design they are not readily extendable by the general medical image analysis community in this form. ITK, however, enjoys a large user base and exemplifies the notion of an open-source, adoptable, and extendable framework. Furthermore, the incorporation of ITK grants DOs access to faster processing, multi-threading, additional image processing functions and libraries, and straightforward compatibility with the powerful visualization capabilities of the Visualization Toolkit (VTK) `www.vtk.org`.

1.2 DOs Requirements

DOs are constructed through the realization of many abstract and independent concepts/layers (cognitive, behavioral, physical, geometrical, sensors). As such, a DO framework must reflect this modular design by allowing users to replace one implementation (layer) for another. For example, new shape representations should be introducible without re-designing existing cognitive layers. To this end, the interface between layers must be consistent across implementations (plug and play), and clearly defined.

The framework must also be extendable, allowing it to grow and advance as the concept of DOs does. That is to say, it should support current research into new types of DOs designed for different applications, with increasingly advanced decision making and deformation abilities.

2 Implementation

This section provides details on the implementation of the I-DO framework. Each section (2.1-2.7) describes a DO layer in detail within the context of our I-DO framework. A high level overview of the DOs framework is shown in Figure 2.

2.1 Organism

The `organism` is the abstract base class (ABC) that acts as a container for most of the framework. Each organism posses its world, a control center, a physics layer, and a geometrical layer. It provides public interfaces through which users can add deformations and behaviors, as well as attach the cognitive, physical, and geometrical layers. Its important to understand that as an ABC, the organism class itself is not instantiated. It is designed as such so that no matter the derivation (type of organism), a DO application can simply call its associated public interface. Consequently, of most interest are the derived classes themselves.

The `itkOrganism` derived class can be instantiated and used as a fully functional organism, or can be used as a base class of another more specialized organism. It inherits from both the Organism ABC, and ITK's ImageToImageFilter class. Though many other classes could be used, the ImageToImageFilter class allows these particular DOs to be incorporated as autonomous tools in existing ITK filtering pipelines (taking as

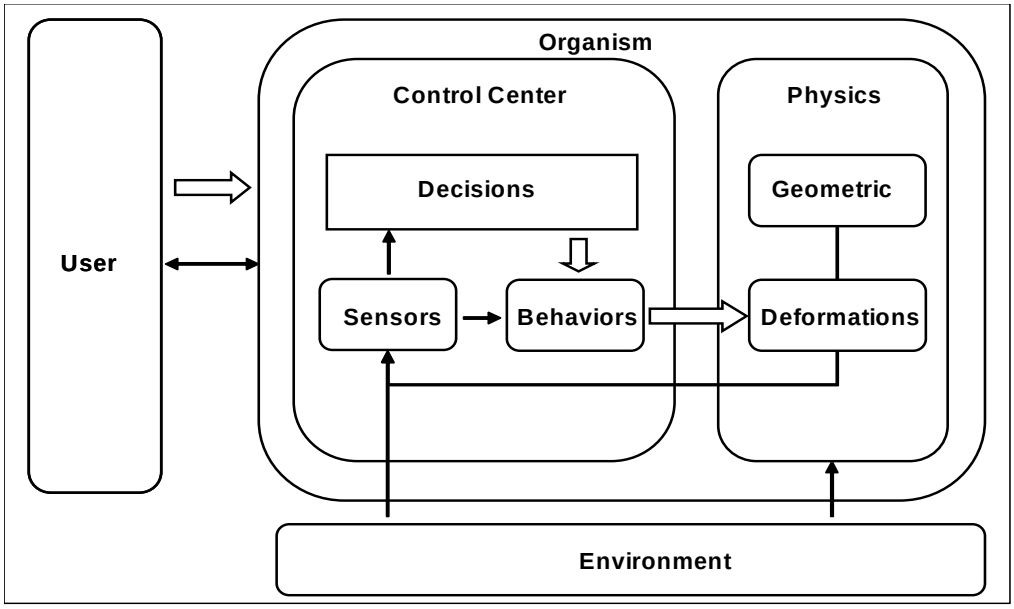


Figure 2: The basic outline of the deformable organism framework. Dark arrows represent directions of communication between objects, while hollow arrows represent one class running another's public `run` method, and encapsulation represents one class containing another. For example, the behavior class controls the deformations class through the physics class.

input an image and producing as output a segmented image). More details on this derived class are provided at http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_organism.html.

2.2 Control Center

The control center is designed to handle all "intelligent" aspects of the organism. It has associated behaviors and sensory modules, and provides the organism with its ability to make decisions (e.g. next behavior to run, image data to sense, etc.). It monitors the status of the behaviors, deformations, and sensors, then makes decisions based upon their states and outputs.

Consequently, this class exploits much of the complex versatility of the framework obtained through the use of ABCs, streams, and structures. Through a single list of sensors and behaviors, the cognitive center can perform a variety of actions on any defined geometrical or physical type regardless of the varying input requirements they may have. For example, the decision to "translate" will trigger a spatial translation behavior, which will in turn trigger the appropriate translate deformation as it pertains to the particular physical layer of the model. All without the cognitive layer having any regard for which derived physical layer and deformation class, or geometrical layer and shape representation is being called.

The control center accomplishes this by using a "run-by-name" design methodology, where once it decides upon (or is asked to run) a particular named behavior it will search its list of known behaviors for one with the matching name.

By calling a control center's `Update` method the organism will conceptually cause the control center to do its thinking. If no current behavior exists it will decide on one (via the derived classes provided `DecideNextBehavior` method). Otherwise, it will check the status of the behavior (via its `IsFinished` method), then clean up (`CleanUp` method) and decide on a new behavior if it has finished, or update it

(`Update` method) if it has not.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_control_center.html

2.3 Sensor

Organisms perceive their surroundings through sensory modules. They provide a means by which to gather statistics and characteristics of its own geometry and the world (image data) in which it resides. At any given time a decision function may possess many different sensory objects, each of which can report back different sensory information (e.g. gray level intensity, gradient magnitude and direction, texture features, etc). It is important to note that some sensors will be implementation dependent, while others will not. For example, it makes no sense to run a vasculature bifurcation sensory module on a corpus callosum organism because the latter is only 2D and has a completely different topology and appearance characteristics.

In order to run a sensor one must use its publicly defined `sensorIn` and `sensorOut` types to create the input arguments and receive the output. This allows maximum flexibility in the parameters a sensor can have, while still enabling any sensor to be ran abstractly. Through this flexibility users can setup and run complex pipelines of ITK filters within the sensors, while passing their variety of input requirements in via the sensorIn type.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_sensor.html

2.4 Behavior

Behaviors are basically actions, or sequences of actions. As such, each behavior has a name, a state, a pointer to the physical layer, and multiple sub-behaviors, and deformations. To ensure meaningful interaction with other organisms and users each behavior has a name. So for example, despite the action “running” being carried out differently by different animals each can always be told to run, or report that it is running. Upon being executed the behavior simply begins executing its main body. Again, the behavior class is simply an ABC. So let’s consider a few example derived classes to illustrate the subtleties of this class.

The first simple example behavior is ‘inflate for 30 cycles’. The act of the organism inflating itself is physics system dependant, so the behavior runs its associated inflate deformation by calling the `runDeformation` method of the physics object. The behavior then sets its status to incomplete. At the next run of its `decideNextBehavor` method the control center checks the status of the inflate behavior, and upon seeing `incomplete` runs the behavior’s `update` method. Now upon executing, the behavior checks to see if its ran for 30 cycles by examining the physics objects time counter, if so it sets its status to `complete`. Now suppose a more complex behavior inflates then moves forward. First it runs its inflate sub-behavior by checking its list of behaviors for one with a matching name, then checks its status. Upon confirming that its first sub-behavior is complete it moves forward, and sets its own status to `complete`.

It is also possible for the `decideNextBehavior` method to use a decision function to decide that a given behavior is finished executing, regardless of its current status. Of course, a behavior may also fail, resulting in some action by the control center.

Sub-behaviors are smaller behaviors performed as part of a larger action. This enables significant levels of abstraction, allowing users to issue single commands and carry out vast and complex sequences of actions, or small exact ones. For example, one could instruct the organism to simply inflate, or one could tell it to segment which includes inflation [2].

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_behavior.html

2.5 Physics

The `Physics` layer is responsible for simulating the deformations and handling the organisms interaction with its environment through external forces. Each physics object possesses a list of executable deformations and a geometric object. The main public interface of interest is the simulate method, which actually causes forces to be calculated and exerted. Again, as the physics layer is merely an ABC, it is of much more interest to discuss this class through an example of one of its derived classes.

An example derived class is the `Phys_Euler` physics object. This implementation relies on the simulation of a spring-mass system to perform deformations. When the organism calls the simulation method, the `Phys_Euler` object runs its simulation cycle for a set number of times, and then increments the global timer. During the simulation cycle the physics layer has control of the CPU, and can not be interrupted. Consequently, the length of this cycle should be kept short in order to allow the organism to check behavior status states, run decision functions, etc. If the length of the cycle is longer than the time required to run a single behavior, then the organism will basically be idle for the remaining iterations. However, the running deformation also has a runtime set by its calling behavior. So the physics object can stop simulating after that runtime has expired.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_physics.html

2.6 Deformations

The `Deformation` classes manipulate the geometry of the organism. For example, in a physically-based spring-mass implementation deformations move nodes, actuate springs, apply forces, and basically deform the geometrical model. Much like behaviors, each deformation has an associated status and runtime, as well as run method for its public interface. However, in this case deformations do not posses many sub-deformations.

As an example let us consider the inflate deformation. Upon being executed by an associated behavior it begins applying forces normal to the model's surface, causing it to inflate. In the case of a spring-mass system these forces may be carried out by applying forces on individual nodes, or by increasing the rest-lengths of springs. The concept of reversing the inflation to a deflation once the organism has passed from dark to bright (for example when segmenting dark object on a white background) is delegated to the control center of the organism, and does not take place here. Instead only low-level tasks like actuating springs, moving nodes, etc are carried out. This enables the execution of both prior and learned deformations [8], where learned deformations are carried out by the associated learned behavior causing a sequence of spring actuations. However, if the underlying shape representation is level sets based the inflation takes the form of adding a constant to the embedding function in order to expand the zero-level set.

2.7 Geometric

The `Geometric` object houses the the actual topology of the organism. It handles adding and removing nodes, as well as reading and writing the meshes to file. Consider two different example derived classes: the VectorGeometry class and the TubularGeometry class. The VectorGeometry class is implemented entirely with vector geometry, while the TubularGeometry class is also derived from an ITK spatialobjects class. Both classes provide the same public interface in terms of getting nodes, setting nodes, writing to file, reading from file, etc. However, they each allow the user to take advantage of their inherit properties. So the user can write a custom sensory class, that uses the additional functionality of the TubularGeometry class

without having to modify any internal code of the organism itself. In essence, the user can be dependent on the implementation when they want to be, and remain totaly independent in other situations by sticking to the `Geometric` base class interface.

3 Conclusions

We have developed a powerful new framework for medical image segmentation and analysis that offers both great flexibility and rigid design enforcement, thereby, ensuring maximum reusability, portability and sustainability. Our framework makes use of many powerful features in ITK including filters, meshes, file IO, and spatial objects. We have also created a robust spring-mass physically-based deformations layer, which can be seen as a contribution in itself. With the geometrical layers `binaryImageToMesh` functionality, one can easily create deformable models and deform them using our spring-mass deformation system or our level-set implementation. Furthermore, the added ability to convert BYU surfaces into `itk::MeshSpatialObjects` and consequently, into deformable organisms should prove a useful tool allowing level-set refinement, or physics-based interaction with segmentation results of various existing projects.

4 Acknowledgements

We would like to thank Andy Rova for his development of the Phys_LevelSet class, Vincent Chu for his role as lead developer of the KWWidgets viewer application (section C), and Aaron Ward for his technical expertise and discussions on fundamental framework design choices.

A Requirements

Though the framework itself only requires ITK 2.4 or greater, building the provided viewer (section C), has additional requirements:

- VTK 5.0.0 `http://www.vtk.org`
- SOViewer (Feb 8, 2006) `http://www.vtk.org/Wiki/SOViewer`
- KWWidgets (Feb 8, 2006) `http://www.kwwidgets.org/Wiki/KWWidgets`

B Examples

B.1 Layer Examples

Various examples of the layers/modules explained in section 2 are available, with details provided in the frameworks online documentation.

- Geom_MeshSpatialObject<dType,nDims, MType, VType>
  `http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_geom___mesh_spatial_object.htm`

- Phys_Euler<DataType,TGradientImage,nDims,MType,VType>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_phys___euler.html
- Phys_LevelSet<DataType,InputImageType,nDims,MType,VType>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_phys___level_set.html
- Beh_TranslateAll<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_beh___translate_all.html
- Beh_UniformScale<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_beh___uniform_scale.html
- Beh_SearchForObject<Type,TInputImage,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_beh___search_for_object.html
- Def_TranslateAll<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_def___translate_all.html
- Def_UniformScale<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_def___uniform_scale.html
- Ctrl_ScheduleDriven<class Type, int nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_ctrl___schedule_driven.html
- Sense_Gradient<DataType,TInputImage, TGradientImage, nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_sense___gradient.html

B.2 Deformable Organism Examples

There are numerous example DOs included with the framework.

- **itkOrganism**<**ImageType, ImageType, GradientImageType, dType, nDims**> A derived organism based on a `itk::ImageToImageFilter` that contains no default layers.
- **Org_LevelSetSchedule**<**ImageType, ImageType, GradientImageType, dType, nDims**> A geodesic active contours [1] based DO that uses a schedule driven cognitive layer.
- **Org_EulerSchedule**<**ImageType, ImageType, GradientImageType, dType, nDims**> A 3D spring-mass [7] based DO that uses a schedule driven cognitive layer.

C The Visual Interface to I-DO

We have also developed a graphical user interface to the I-DO framework, that allows its users to visualize the geometry of created DOs as well as observe their deformations in real time. It gives the user the ability to load DOs as dll files, while allowing the developer to define customized interfaces via the `DefOrgAdapter` class. The GUI is based on, and therefore requires, KWWidegets, VTK, and SOViewer. Future versions

will facilitate interaction with DOs through mouse click driven forces, and possibly other forms of input. Complete documentation of the viewer will be made available at a later date, but many details reside in its doxygen.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_def_org_viewer_adapter_base.html

D Guide to users

This section provides information to those who wish to use, or contribute to the framework.

Hello I-DO

In this section we present a simple “Hello [I-DO] World” example that provides a step by step guide to how a new user can build and run a simple DO.

1. Download and compile ITK 2.4 or greater (see www.itk.org).
2. Download (http://mial.fas.sfu.ca/researchProject.php?s=157) and configure the I-DO framework using CMake (www.cmake.org) and the CMakeLists.txt file found in the root-most directory. Make sure to leave ”Build Examples” set to ”ON”.
3. Compile the created project. This will build the I-DO library, and two executables.
4. Run YourBuildDirectory/examples/basic/defOrg_basic from command line, providing input and output image names, a schedule name, and a mesh name. (e.g. cube.mhd out.mhd eulerSchedule3d.txt cubeMesh3d.meta)
5. The DO will run, and output a final binary image using the file name provided.

Users can follow these procedures for any of the provided examples in the examples directory.

- Basic - The same example as shown in “Building A Deformable Organism”. An physically-based DO using a schedule driven cognitive layer along with a few example behaviors and deformations.
- Advanced - A multi-organism application that uses two pre-made DOs in sequence. `Org_EulerSchedule` begins the segmentation process and initializes `Org_LevelSetSchedule` with its output, which then proceeds to refine the segmentation results before writing out to file.

Building A Deformable Organism

This example walks the reader through creating a DO by individualy instantiating and attaching the layers. This is contrast to using an already created DO, which can be instantiated, setup, and used just like any ITK filter.

The first step is to choose and instantiate a DO shell (one having no built in layers) using the standard ITK `itk::SmartPointer` approach. In this case the DO is an ITK `itk::ImageToImageFilter`, and must be provided with an input image via the `SetInput` method.

```
typedef itk::ItkOrganism <ImageType, ImageType, GradientImageType, float, 3> organismType;
organismType::Pointer testOrg = organismType::New();
std::cout << "Organism created..." << std::endl;
testOrg->SetInput(reader->GetOutput());
```

Next we will instantiate a sensor to calculate the gradient information used as an external force during the deformation simulations by the `Physics` layer.

```
typedef Sense_Gradient<float,ImageType,GradientImageType,3> gradientSensorType;
gradientSensorType::Pointer gradientSensor = gradientSensorType::New();
```

The sensor requires its publicly defined `sensorIn` as input. Here we create a pointer to the class, and set its values. This allows all sensors to be ran from a common `run` method, with their own customized input.

```
gradientSensorType::sensorIn::Pointer input = gradientSensorType::sensorIn::New();
input->sigma = 1.0;
reader->Update();
input->imageIn = reader->GetOutput();
```

The gradient sensor can then be ran. Note that at this time `Sensors` themselves do not fit into the ITK pipeline, and thus the reader's `Update()` method must be called prior to running the sensor.

```
gradientSensor->run(input);
```

Finally, its output can be obtained by constructing a `sensorOut` `itk::SmartPointer` and providing the appropriate downcast on the pointer returned by the `getOutput` method.

```
gradientSensorType::sensorOut::Pointer output = (gradientSensorType::sensorOut *) (gradientSensor->
```

Next create the `Physics` and `Geometrical` layers. Notice that the type of external force image is provided as an input type to the `Physics` layer.

```
typedef Phys_Euler<float,GradientImageType,3> PhysLayerType;
typedef Geom_MeshSpatialObject<float,3> GeometricType;

PhysLayerType::Pointer physLayer = PhysLayerType::New();
GeometricType::Pointer geomLayer = GeometricType::New();
```

Then set the `Physics` layer to use the external force image calculated by the gradient sensor and the newly constructed `Geometrical` layer, and setup the topology of the `Geometric` layer (in this case an ITK `itk::MeshSpatialObject`). Finally, attach both to the `Organism`.

```
physLayer->setExternalForces((void *) &(output->imageOut));
physLayer->setGeometry(geomLayer);
std::cout << "External forces set." << std::endl;

geomLayer->readTopologyFromFile(topologyInputFileName);
std::cout << "Topology read from '" << topologyInputFileName << "'..." << std::endl;

testOrg->setPhysicsLayer(physLayer);
testOrg->setGeometricLayer(geomLayer);
std::cout << "Physics and Geometric layers added..." << std::endl;
```

Create a Cogntive layer, set its appropriate options, and attach it to the DO. In this case it only requires a Schedule text file (e.g. eulerSchedule3D.txt).

```
Ctrl_ScheduleDriven<float, 3>::Pointer cgL = Ctrl_ScheduleDriven<float, 3>::New();
cgL->setSchedule(scheduleFileName);
testOrg->setCognitiveLayer(cgL);
```

Now begin creating and attaching simple behaviors, and deformations. Note in this case, the behaviors and deformations do not require any additional parameters or settings.

```
Beh_TranslateAll<float, 3>::Pointer beh1 = Beh_TranslateAll<float,3>::New();
Beh_UniformScale<float, 3>::Pointer beh2 = Beh_UniformScale<float,3>::New();
Def_Translation<float, 3>::Pointer def1 = Def_Translation<float,3>::New();
Def_UniformScale<float, 3>::Pointer def2 = Def_UniformScale<float,3>::New();

testOrg->addBehaviour(beh1);
testOrg->addBehaviour(beh2);
testOrg->addDeformation(def1);
testOrg->addDeformation(def2);
```

Attach a more advanced behavior and set its additional parameters. In this case it needs an image and a Geometric pointer for its internal Sense_AvgIntensity sensor.

```
Beh_SearchForObject<float,ImageType,3>::Pointer beh3 = Beh_SearchForObject<float,ImageType,3>::New(
beh3->image = reader->GetOutput();
beh3->geomLayer = geomLayer;
testOrg->addBehaviour(beh3);
```

The Organism is ready to run. Calling Update() on the writer will cause the DO to simulate for a set amount of DO time. Here we set the DO to run for 25 iterations with a single Update().

```
testOrg->setRunTime(120);
writer->SetInput(testOrg->GetOutput());
try
{
    writer->Update();
}
catch(itk::ExceptionObject & err)
{
    std::cout << "ExceptionObject caught!" << std::endl;
    std::cout << err << std::endl;
    return -1;
}
```

Finally, in addition to the binary output available on the writer the DO's mesh can be written back to file.

```
testOrg->writeNodesToFile(nodeOutputFileName);
std::cout << "Nodes written to '" << nodeOutputFileName << "'." << std::endl;
```

Extending Existing DOs

Extending existing organisms is as easy as following the *Building A Deformable Organism* example and attaching additional layers.

Creating New DOs and Layers

Detailed information about creating new DOs and layers will be included in this document in a later revision. In the mean time, interested users are referred to the doxygen documentation which outlines how each pure virtual function of the ABCs should be defined in a derived class. We will also provide skeleton code generators, that will give those wishing to create new layers a "fill in the blanks" option.

`http://www.sfu.ca/~cmcintos/IDO/doxygen/html/index.html`

References

[1] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In *ICCV*, pages 694–699, 1995. B.2

[2] Laurent D. Cohen. On active contour models and balloons. *CVGIP: Image Underst.*, 53(2):211–218, 1991. 2.4

[3] Ghassan Hamarneh, Tim McInerney, and Demetri Terzopoulos. Deformable organisms for automatic medical image analysis. In *MICCAI*, pages 66–76, 2001. 1

[4] Ghassan Hamarneh and Chris McIntosh. Physics-based deformable organisms for medical image analysis. *SPIE Medical Imaging*, 5747:326–335, 2005. 1

[5] G. Hamarnerh and C. McIntosh. *Parametric and Geometric Deformable Models: An application in Biomaterials and Medical Imagery*, chapter 12: Deformable Organisms for Medical Image Analysis. Springer Publishers, 1 edition, 2006. 1

[6] C. McIntosh and G. Hamarnerh. Spinal crawlers: Deformable organisms for spinal cord segmentation and analysis. *MICCAI*, 2006. 1

[7] C. McIntosh and G. Hamarnerh. Vessel crawlers: 3d physically-based deformable organisms for vasulature segmentation and analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 1, 1, B.2

[8] D. Terzopoulos, X. Tu, and R. Grzeszczuk. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994. 2.6

I-DO: A “Deformable Organisms” framework for ITK

Release 0.50

Chris McIntosh and Ghassan Hamarneh

July 23, 2006

Medical Image Analysis Lab
School of Computing Science, Simon Fraser University
Burnaby, BC, Canada
{cmcintos,hamarneh}@cs.sfu.ca

Abstract

Medical image analysis is an important problem relating to the study of various diseases. Since their introduction to MICCAI in 2001, ”deformable organisms” have emerged as a fruitful methodology with examples ranging from 2D corpus callosum segmentation to 3D vasculature and spinal cord segmentation. Essentially we previously have developed an artificial life framework that complements the geometrical and physical layers of classical deformable models (snakes and deformable meshes) with high-level behavioral and cognitive layers that facilitate anatomically-driven control mechanisms. This paper describes the integration of deformable organisms into the Insight Toolkit (ITK) www.itk.org. In our proposed implementation we attempt to bridge the ITK framework and coding style with deformable organism design methodologies. In the interest of open science, as the framework develops it will serve as a basis for the community to develop new deformable organisms as well as experiment with those recently published by our group. Further, as the design of the ITK Deformable Organisms (I-DO) is highly modular, researchers and developers can exchange components (spatial objects, dynamic simulation engines, image sensors, etc) allowing in the future for fast development of new custom deformable organisms for different clinical applications.

Contents

1 Introduction **2**
1.1 ITK Deformable Organisms: Motivation and Introduction 4
1.2 DOs Requirements 4

2 Implementation **4**
2.1 Organism 4
2.2 Control Center 5
2.3 Sensor 6
2.4 Behavior 6
2.5 Physics 7
2.6 Deformations 7
2.7 Geometric 7

3 Conclusions **8**

4 Acknowledgements **8**

A Requirements **8**

B Examples **8**
B.1 Layer Examples 8
B.2 Deformable Organism Examples 9

C The Visual Interface to I-DO **10**

D Guide to users **10**
Hello I-DO 10
Building A Deformable Organism 11
Extending Existing DOs 13
Creating New DOs and Layers 13

1 Introduction

In medical image analysis strategies based on deformable models, controlling the deformations of the models is a desirable goal to produce proper segmentations. Incorporating expert knowledge to automatically guide deformations cannot be easily and elegantly achieved using the classical deformable model low-level energy-based fitting mechanisms. Deformable Organisms (DOs), are a decision-making framework for medical image analysis that complements bottom-up, data-driven deformable models with top-down, knowledge-driven mode-fitting strategies in a layered fashion inspired by artificial life modeling concepts. Intuitive and controlled deformations are carried out through behaviors. Sensory input from image data and contextual knowledge about the analysis problem govern these different behaviors.

Since their introduction in 2001 [3], various DOs-based approaches for medical image analysis have been developed (Figure 1). In this original work, a variety of DOs where demonstrated with applications to locating the lateral ventricles, caudate nuclei, and putamina structures in transversal brain magnetic resonance image (MRI) slices, as well as DOs for the segmentation of vessels in 2D angiography. In [4], DOs were augmented to include physically-based and controlled deformations demonstrating an application to corpus callosum segmentation in mid-sagittal magnetic resonance images (MRI). Recently, DOs were extended to 3D and applied to vascular segmentation and analysis. The so called ‘vessel crawlers’ were equipped with sensors, decision modules, and deformation layers suited for vasculature [7]. An extension of that work introduces DOs for spinal cord segmentation and analysis and demonstrates the ability to efficiently replace modules of existing DOs to create new solutions. The ‘spinal crawlers’ no longer possessed a decision module to detect branching and their sensors were adapted to detect elliptical cross sections [6]. In each case DOs have demonstrated their key advantages over other leading techniques. Namely, their ability to produce increased accuracy, allow intuitive user-interaction to control or repair the segmentation where other methods would require being restarted with some type of parameter adjustment, facilitate greater analysis and labeling abilities than those methods producing binary outputs, the ready ability to incorporate image or shape-based prior-knowledge, and a modular framework allowing for incorporating new sensors (image filters), decision models, shape representations, and deformation mechanisms.

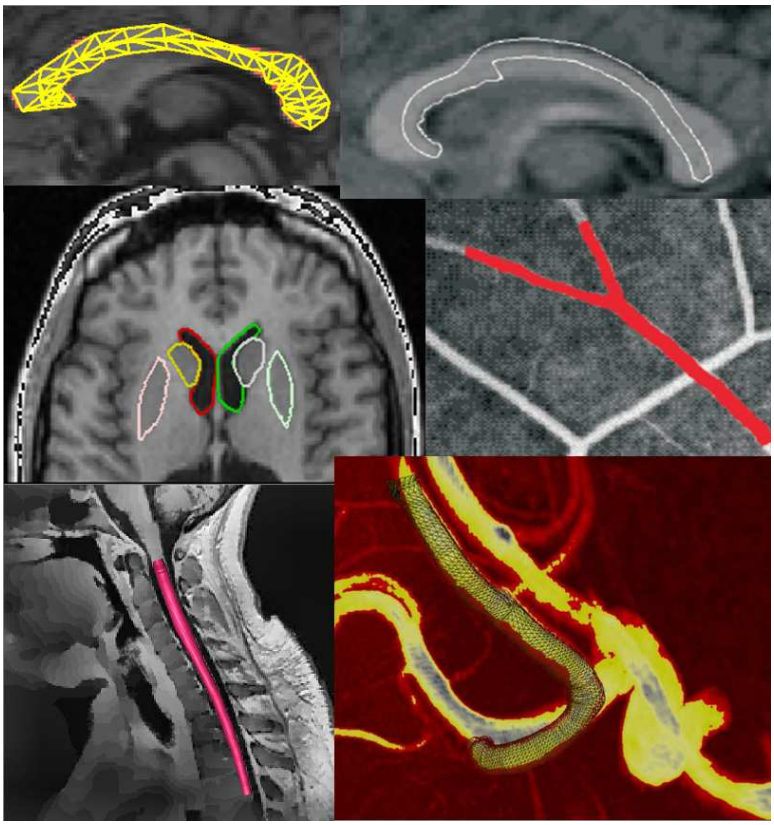


Figure 1: An assortment of deformable organisms showing(left to right, top to bottom): Physically-based corpus callosum, Geometrically-based corpus callosum, Putamina and ventricle organisms, 2D Angiography, 3D 'spinal crawler', and 3D 'vessel crawler'. Related images and videos can be found at http://mial.fas.sfu.ca/researchProject.php?s=157

Though a summary is provided here, a complete research-oriented look at DOs can be found in [5]. DOs are built following a multilevel AL modelling approach consisting of four primary layers: ***cognitive, behavioral, physical, and geometrical***. Specifically, the cognitive layer makes decisions based on the DOs current state, anatomical knowledge, and its surrounding environment (the image). Decisions could be made to sense information, to deform based on sensory data, to illicit help from the user, or to terminate the segmentation process. All of these actions are described under the behavioral layer of the organism, and they rely upon both the physical and geometrical layers for implementation. For example, in the context of our 'vessel crawlers' [7], the act of moving towards a sensed target location is described by the *'growing'* behavioral method. The cognitive center gathers sensory input using the *'sense-to-grow'* sensory module, decides the correct location via the *'where-to-grow'* decision module, elicits the act of *'growing'* , and then conforms to the vascular walls by *'fitting'*. In turn, each of these methods relies upon the physical and geometrical layers to carry out tasks, such as maintaining model stability. Consequently, we have a framework with many independent layers of abstraction, each built upon the implementation of independent modules and or processes.

We begin with a motivation of our ITK-Deformable Organisms (I-DO) framework in section 1.1, and a discussion of the general requirements of DOs that the framework is set out to meet in 1.2. Sections (2.1-2.7) provide an overview of how each layer is designed and implemented in the framework. We summarize in section 3. The appendices provide the most information on using the framework with a requirements listing

(section A), examples of layers and organisms (section B), a description of our visual interface (section C), a guide to building and running your first organism (section D), and information on extending organisms and the framework (section D).

1.1 ITK Deformable Organisms: Motivation and Introduction

Previously, the major drawback of DOs has been their restriction to a closed-source MATLAB framework. Though straightforward and intuitive in design they are not readily extendable by the general medical image analysis community in this form. ITK, however, enjoys a large user base and exemplifies the notion of an open-source, adoptable, and extendable framework. Furthermore, the incorporation of ITK grants DOs access to faster processing, multi-threading, additional image processing functions and libraries, and straightforward compatibility with the powerful visualization capabilities of the Visualization Toolkit (VTK) `www.vtk.org`.

1.2 DOs Requirements

DOs are constructed through the realization of many abstract and independent concepts/layers (cognitive, behavioral, physical, geometrical, sensors). As such, a DO framework must reflect this modular design by allowing users to replace one implementation (layer) for another. For example, new shape representations should be introducible without re-designing existing cognitive layers. To this end, the interface between layers must be consistent across implementations (plug and play), and clearly defined.

The framework must also be extendable, allowing it to grow and advance as the concept of DOs does. That is to say, it should support current research into new types of DOs designed for different applications, with increasingly advanced decision making and deformation abilities.

2 Implementation

This section provides details on the implementation of the I-DO framework. Each section (2.1-2.7) describes a DO layer in detail within the context of our I-DO framework. A high level overview of the DOs framework is shown in Figure 2.

2.1 Organism

The `organism` is the abstract base class (ABC) that acts as a container for most of the framework. Each organism posses its world, a control center, a physics layer, and a geometrical layer. It provides public interfaces through which users can add deformations and behaviors, as well as attach the cognitive, physical, and geometrical layers. Its important to understand that as an ABC, the organism class itself is not instantiated. It is designed as such so that no matter the derivation (type of organism), a DO application can simply call its associated public interface. Consequently, of most interest are the derived classes themselves.

The `itkOrganism` derived class can be instantiated and used as a fully functional organism, or can be used as a base class of another more specialized organism. It inherits from both the Organism ABC, and ITK's ImageToImageFilter class. Though many other classes could be used, the ImageToImageFilter class allows these particular DOs to be incorporated as autonomous tools in existing ITK filtering pipelines (taking as

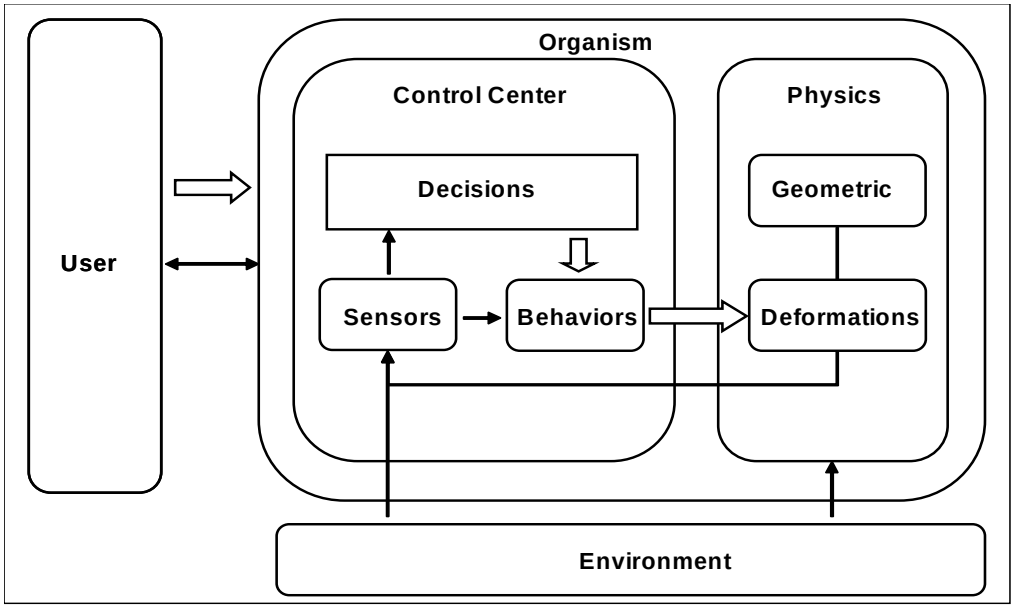


Figure 2: The basic outline of the deformable organism framework. Dark arrows represent directions of communication between objects, while hollow arrows represent one class running another's public `run` method, and encapsulation represents one class containing another. For example, the behavior class controls the deformations class through the physics class.

input an image and producing as output a segmented image). More details on this derived class are provided at http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_organism.html.

2.2 Control Center

The control center is designed to handle all "intelligent" aspects of the organism. It has associated behaviors and sensory modules, and provides the organism with its ability to make decisions (e.g. next behavior to run, image data to sense, etc.). It monitors the status of the behaviors, deformations, and sensors, then makes decisions based upon their states and outputs.

Consequently, this class exploits much of the complex versatility of the framework obtained through the use of ABCs, streams, and structures. Through a single list of sensors and behaviors, the cognitive center can perform a variety of actions on any defined geometrical or physical type regardless of the varying input requirements they may have. For example, the decision to "translate" will trigger a spatial translation behavior, which will in turn trigger the appropriate translate deformation as it pertains to the particular physical layer of the model. All without the cognitive layer having any regard for which derived physical layer and deformation class, or geometrical layer and shape representation is being called.

The control center accomplishes this by using a "run-by-name" design methodology, where once it decides upon (or is asked to run) a particular named behavior it will search its list of known behaviors for one with the matching name.

By calling a control center's `Update` method the organism will conceptually cause the control center to do its thinking. If no current behavior exists it will decide on one (via the derived classes provided `DecideNextBehavior` method). Otherwise, it will check the status of the behavior (via its `IsFinished` method), then clean up (`CleanUp` method) and decide on a new behavior if it has finished, or update it

(`Update` method) if it has not.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_control_center.html

2.3 Sensor

Organisms perceive their surroundings through sensory modules. They provide a means by which to gather statistics and characteristics of its own geometry and the world (image data) in which it resides. At any given time a decision function may possess many different sensory objects, each of which can report back different sensory information (e.g. gray level intensity, gradient magnitude and direction, texture features, etc). It is important to note that some sensors will be implementation dependent, while others will not. For example, it makes no sense to run a vasculature bifurcation sensory module on a corpus callosum organism because the latter is only 2D and has a completely different topology and appearance characteristics.

In order to run a sensor one must use its publicly defined `sensorIn` and `sensorOut` types to create the input arguments and receive the output. This allows maximum flexibility in the parameters a sensor can have, while still enabling any sensor to be ran abstractly. Through this flexibility users can setup and run complex pipelines of ITK filters within the sensors, while passing their variety of input requirements in via the sensorIn type.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_sensor.html

2.4 Behavior

Behaviors are basically actions, or sequences of actions. As such, each behavior has a name, a state, a pointer to the physical layer, and multiple sub-behaviors, and deformations. To ensure meaningful interaction with other organisms and users each behavior has a name. So for example, despite the action “running” being carried out differently by different animals each can always be told to run, or report that it is running. Upon being executed the behavior simply begins executing its main body. Again, the behavior class is simply an ABC. So let’s consider a few example derived classes to illustrate the subtleties of this class.

The first simple example behavior is ‘inflate for 30 cycles’. The act of the organism inflating itself is physics system dependant, so the behavior runs its associated inflate deformation by calling the `runDeformation` method of the physics object. The behavior then sets its status to incomplete. At the next run of its `decideNextBehavor` method the control center checks the status of the inflate behavior, and upon seeing `incomplete` runs the behavior’s `update` method. Now upon executing, the behavior checks to see if its ran for 30 cycles by examining the physics objects time counter, if so it sets its status to `complete`. Now suppose a more complex behavior inflates then moves forward. First it runs its inflate sub-behavior by checking its list of behaviors for one with a matching name, then checks its status. Upon confirming that its first sub-behavior is complete it moves forward, and sets its own status to `complete`.

It is also possible for the `decideNextBehavior` method to use a decision function to decide that a given behavior is finished executing, regardless of its current status. Of course, a behavior may also fail, resulting in some action by the control center.

Sub-behaviors are smaller behaviors performed as part of a larger action. This enables significant levels of abstraction, allowing users to issue single commands and carry out vast and complex sequences of actions, or small exact ones. For example, one could instruct the organism to simply inflate, or one could tell it to segment which includes inflation [2].

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_behavior.html

2.5 Physics

The `Physics` layer is responsible for simulating the deformations and handling the organisms interaction with its environment through external forces. Each physics object possesses a list of executable deformations and a geometric object. The main public interface of interest is the simulate method, which actually causes forces to be calculated and exerted. Again, as the physics layer is merely an ABC, it is of much more interest to discuss this class through an example of one of its derived classes.

An example derived class is the `Phys_Euler` physics object. This implementation relies on the simulation of a spring-mass system to perform deformations. When the organism calls the simulation method, the `Phys_Euler` object runs its simulation cycle for a set number of times, and then increments the global timer. During the simulation cycle the physics layer has control of the CPU, and can not be interrupted. Consequently, the length of this cycle should be kept short in order to allow the organism to check behavior status states, run decision functions, etc. If the length of the cycle is longer than the time required to run a single behavior, then the organism will basically be idle for the remaining iterations. However, the running deformation also has a runtime set by its calling behavior. So the physics object can stop simulating after that runtime has expired.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_physics.html

2.6 Deformations

The `Deformation` classes manipulate the geometry of the organism. For example, in a physically-based spring-mass implementation deformations move nodes, actuate springs, apply forces, and basically deform the geometrical model. Much like behaviors, each deformation has an associated status and runtime, as well as run method for its public interface. However, in this case deformations do not posses many sub-deformations.

As an example let us consider the inflate deformation. Upon being executed by an associated behavior it begins applying forces normal to the model's surface, causing it to inflate. In the case of a spring-mass system these forces may be carried out by applying forces on individual nodes, or by increasing the rest-lengths of springs. The concept of reversing the inflation to a deflation once the organism has passed from dark to bright (for example when segmenting dark object on a white background) is delegated to the control center of the organism, and does not take place here. Instead only low-level tasks like actuating springs, moving nodes, etc are carried out. This enables the execution of both prior and learned deformations [8], where learned deformations are carried out by the associated learned behavior causing a sequence of spring actuations. However, if the underlying shape representation is level sets based the inflation takes the form of adding a constant to the embedding function in order to expand the zero-level set.

2.7 Geometric

The `Geometric` object houses the the actual topology of the organism. It handles adding and removing nodes, as well as reading and writing the meshes to file. Consider two different hypothetical derived classes: a VectorGeometry class and a TubularGeometry class. The VectorGeometry class would be implemented entirely with vector geometry, while the TubularGeometry class would also derived from an ITK spatialobjects class. Both classes would provide the same public interface in terms of getting nodes, setting nodes, writing to file, reading from file, etc. However, they each would allow the user to take advantage of their inherit properties. So the user can write a custom sensory class, that uses the additional functionality of the

TubularGeometry class without having to modify any internal code of the organism itself. In essence, the user can be dependent on the implementation when they want to be, and remain totaly independent in other situations by sticking to the `Geometric` base class interface.

3 Conclusions

We have developed a powerful new framework for medical image segmentation and analysis that offers both great flexibility and rigid design enforcement, thereby, ensuring maximum reusability, portability and sustainability. Our framework makes use of many powerful features in ITK including filters, meshes, file IO, smart pointers, and spatial objects. We have also created a robust spring-mass physically-based deformation layer, which can be seen as a contribution in itself.

Furthermore, the added ability to convert BYU surfaces or binary volumes into `itk::MeshSpatialObjects` and consequently, into deformable organisms should prove a useful tool allowing level-set refinement, or physics-based interaction with segmentation results of various existing projects. For example, both explicit physically based (spring mass) and implicit level set based classical deformable models are special cases of DOs and their implementation is a special case of the IDO framework. They now simply emerge in IDO by setting the proper geometrical and physical layers (spring mass vs level set) and having behavioral and cognitive layers that simply simulate the deformation dynamics without any top down control or scheduling.

4 Acknowledgements

We would like to thank Andy Rova for his development of the Phys_LevelSet class, Vincent Chu for his role as lead developer of the KWWidgets viewer application (section C), and Aaron Ward for his technical expertise and discussions on fundamental framework design choices.

A Requirements

Though the framework itself only requires ITK 2.4 or greater, building the provided viewer (section C), has additional requirements:

- VTK 5.0.0 `http://www.vtk.org`
- SOViewer (Feb 8, 2006) `http://www.vtk.org/Wiki/SOViewer`
- KWWidgets (Feb 8, 2006) `http://www.kwwidgets.org/Wiki/KWWidgets`

B Examples

B.1 Layer Examples

Various examples of the layers/modules explained in section 2 are available, with details provided in the frameworks online documentation.

- Geom_MeshSpatialObject<dType,nDims, MType, VType>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_geom___mesh_spatial_object.htm
- Phys_Euler<DataType,TGradientImage,nDims,MType,VType>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_phys___euler.html
- Phys_LevelSet<DataType,InputImageType,nDims,MType,VType>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_phys___level_set.html
- Beh_TranslateAll<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_beh___translate_all.html
- Beh_UniformScale<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_beh___uniform_scale.html
- Beh_SearchForObject<Type,TInputImage,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_beh___search_for_object.html
- Def_TranslateAll<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_def___translate_all.html
- Def_UniformScale<Type,nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_def___uniform_scale.html
- Ctrl_ScheduleDriven<class Type, int nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_ctrl___schedule_driven.html
- Sense_Gradient<DataType,TInputImage, TGradientImage, nDims>
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_sense___gradient.html

B.2 Deformable Organism Examples

There are numerous example DOs included with the framework.

- **itkOrganism<ImageType, ImageType, GradientImageType, dType, nDims>** A derived organism based on a `itk::ImageToImageFilter` that contains no default layers.
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classitk_1_1_itk_organism.html
- **Org_LevelSetSchedule<ImageType, ImageType, GradientImageType, dType, nDims>** A geodesic active contours [1] based DO that uses a schedule driven cognitive layer.
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classitk_1_1_org___level_set_schedule.html
- **Org_EulerSchedule<ImageType, ImageType, GradientImageType, dType, nDims>** A 3D spring-mass [7] based DO that uses a schedule driven cognitive layer.
  http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classitk_1_1_org___euler_schedule.html

C The Visual Interface to I-DO

We have also developed a graphical user interface to the I-DO framework, that allows its users to visualize the geometry of created DOs as well as observe their deformations in real time. It gives the user the ability to load DOs as dll files, while allowing the developer to define customized interfaces via the `DefOrgAdapter` class. The GUI is based on, and therefore requires, KWWidegets, VTK, and SOViewer. Future versions will facilitate interaction with DOs through mouse click driven forces, and possibly other forms of input. Complete documentation of the viewer will be made available at a later date, but many details reside in its doxygen. A binary of the viewer is available for Windows at http://hdl.handle.net/1926/228/viewerApplication.zip.

http://www.sfu.ca/~cmcintos/IDO/doxygen/html/classmial_1_1_def_org_viewer_adapter_base.html

D Guide to users

This section provides information to those who wish to use, or contribute to the framework.

Hello I-DO

In this section we present a simple “Hello [I-DO] World” example that provides a step by step guide to how a new user can build and run a simple DO.

1. Download and compile ITK 2.4 or greater (see www.itk.org).
2. Download (http://hdl.handle.net/1926/228/IDO.zip) and configure the I-DO framework using CMake (www.cmake.org) and the CMakeLists.txt file found in the root-most directory. Make sure to leave ”Build Examples” set to ”ON”.
3. Compile the created project. This will build the I-DO library, and two executables.
4. Run YourBuildDirectory/examples/basic/defOrg_basic from command line, providing input and output image names, a schedule name, and a mesh name. (e.g. cube.mhd out.mhd eulerSchedule3d.txt cubeMesh3d.meta)
5. The DO will run, and output a final binary image using the file name provided.

Users can follow these procedures for any of the provided examples in the examples directory.

- Basic - The same example as shown in “Building A Deformable Organism”. A spring-mass DO using a schedule driven cognitive layer along with a few example behaviors and deformations (Figure 3 top).
- Advanced - A multi-organism application that uses two pre-made DOs in sequence. `Org_EulerSchedule` begins the segmentation process and initializes `Org_LevelSetSchedule` with its output, which then proceeds to refine the segmentation results before writing out to file (Figure 3 bottom).

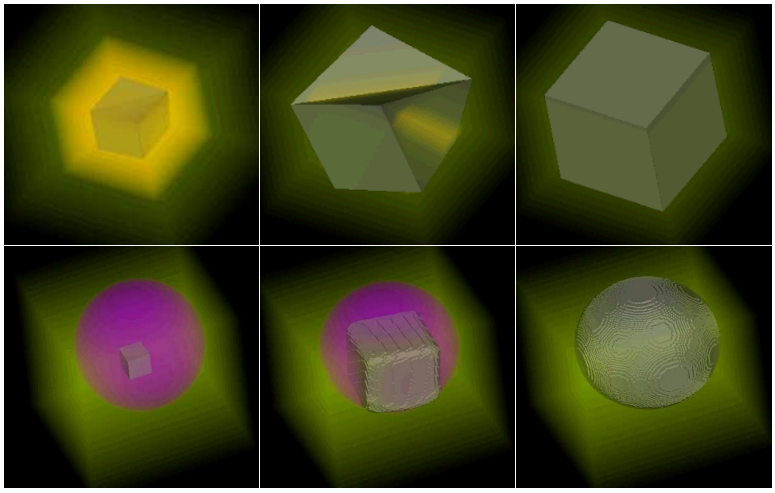


Figure 3: Two example DOs progressing from left to right. Top: The basic example initialized with a cube, performing a `Beh_UniformScale`, and coming to rest. Bottom: The advanced example initialized with a cube, smoothing under `Phys_LevelSet` after a `Beh_UniformScale` under `Phys_Euler`, and coming to rest via image forces. The complete videos are available at http://hdl.handle.net/1926/228/{basic,advanced}.wmv

Building A Deformable Organism

This example walks the reader through creating a DO by individually instantiating and attaching the layers. This is in contrast to using an already created DO, which can be instantiated, setup, and used just like any ITK filter.

The first step is to choose and instantiate a DO shell (one having no built in layers) using the standard ITK `itk::SmartPointer` approach. In this case the DO is an ITK `itk::ImageToImageFilter`, and must be provided with an input image via the `SetInput` method.

```
typedef itk::ItkOrganism <ImageType, ImageType, GradientImageType, float, 3> organismType;
organismType::Pointer testOrg = organismType::New();
std::cout << "Organism created..." << std::endl;
testOrg->SetInput(reader->GetOutput());
```

Now we can begin instantiating and attaching implementations of the layers/components the DO needs to function. For simplicity, all derived classes of a particular layer are prefixed with an abbreviation of that layer (Org for Organism, Ctrl for Control, Beh for Behavior, Sense for Sensor, Phys for Physics, Def for Deformation, and Geom for Geometrical). Next we will instantiate a sensor to calculate the gradient information used as an external force during the deformation simulations by the `Physics` layer.

```
typedef Sense_Gradient<float,ImageType,GradientImageType,3> gradientSensorType;
gradientSensorType::Pointer gradientSensor = gradientSensorType::New();
```

The sensor requires its publicly defined `sensorIn` as input. Here we create a pointer to the class, and set its values. This allows all sensors to be ran from a common `run` method, with their own customized input.

```
gradientSensorType::sensorIn::Pointer input = gradientSensorType::sensorIn::New();
input->sigma = 1.0;
reader->Update();
input->imageIn = reader->GetOutput();
```

The gradient sensor can then be ran. Note that at this time Sensors themselves do not fit into the ITK pipeline, and thus the reader's Update() method must be called prior to running the sensor.

```
gradientSensor->run(input);
```

Finally, its output can be obtained by constructing a sensorOut itk::SmartPointer and providing the appropriate downcast on the pointer returned by the getOutput method.

```
gradientSensorType::sensorOut::Pointer output = (gradientSensorType::sensorOut *) (gradientSensor->
```

Next create the Physics and Geometrical layers. Notice that the type of external force image is provided as an input type to the Physics layer.

```
typedef Phys_Euler<float,GradientImageType,3> PhysLayerType;
typedef Geom_MeshSpatialObject<float,3> GeometricType;

PhysLayerType::Pointer physLayer = PhysLayerType::New();
GeometricType::Pointer geomLayer = GeometricType::New();
```

Then set the Physics layer to use the external force image calculated by the gradient sensor and the newly constructed Geometrical layer, and setup the topology of the Geometric layer (in this case an ITK itk::MeshSpatialObject). Finally, attach both to the Organism.

```
physLayer->setExternalForces((void *) &(output->imageOut));
physLayer->setGeometry(geomLayer);
std::cout << "External forces set." << std::endl;

geomLayer->readTopologyFromFile(topologyInputFileName);
std::cout << "Topology read from '" << topologyInputFileName << "'..." << std::endl;

testOrg->setPhysicsLayer(physLayer);
testOrg->setGeometricLayer(geomLayer);
std::cout << "Physics and Geometric layers added..." << std::endl;
```

Create a Cogntive layer, set its appropriate options, and attach it to the DO. In this case it only requires a Schedule text file (e.g. eulerSchedule3D.txt).

```
Ctrl_ScheduleDriven<float, 3>::Pointer cgL = Ctrl_ScheduleDriven<float, 3>::New();
cgL->setSchedule(scheduleFileName);
testOrg->setCognitiveLayer(cgL);
```

Now begin creating and attaching simple behaviors, and deformations. Note in this case, the behaviors and deformations do not require any additional parameters or settings.

```
Beh_TranslateAll<float, 3>::Pointer beh1 = Beh_TranslateAll<float,3>::New();
Beh_UniformScale<float, 3>::Pointer beh2 = Beh_UniformScale<float,3>::New();
Def_Translation<float, 3>::Pointer def1 = Def_Translation<float,3>::New();
Def_UniformScale<float, 3>::Pointer def2 = Def_UniformScale<float,3>::New();
```

```
testOrg->addBehaviour(beh1);
testOrg->addBehaviour(beh2);
testOrg->addDeformation(def1);
testOrg->addDeformation(def2);
```

Attach a more advanced behavior and set its additional parameters. In this case it needs an image and a `Geometric` pointer for its internal `Sense_AvgIntensity` sensor.

```
Beh_SearchForObject<float,ImageType,3>::Pointer beh3 = Beh_SearchForObject<float,ImageType,3>::New(
beh3->image = reader->GetOutput();
beh3->geomLayer = geomLayer;
testOrg->addBehaviour(beh3);
```

The `Organism` is ready to run. Calling `Update()` on the writer will cause the DO to simulate for a set amount of DO time. Here we set the DO to `run` for 25 iterations with a single `Update()`.

```
testOrg->setRunTime(120);
writer->SetInput(testOrg->GetOutput());
try
{
    writer->Update();
}
catch(itk::ExceptionObject & err)
{
    std::cout << "ExceptionObject caught!" << std::endl;
    std::cout << err << std::endl;
    return -1;
}
```

Finally, in addition to the binary output available on the writer the DO's mesh can be written back to file.

```
testOrg->writeNodesToFile(nodeOutputFileName);
std::cout << "Nodes written to '" << nodeOutputFileName << "'." << std::endl;
```

Extending Existing DOs

Extending existing organisms is as easy as following the *Building A Deformable Organism* example and attaching additional layers.

Creating New DOs and Layers

Detailed information about creating new DOs and layers will be included in this document in a later revision. In the mean time, interested users are referred to the doxygen documentation which outlines how each pure virtual function of the ABCs should be defined in a derived class. We will also provide skeleton code generators, that will give those wishing to create new layers a "fill in the blanks" option.

http://hdl.handle.net/1926/228/doxygenManual.pdf
or
http://www.sfu.ca/~cmcintos/IDO/doxygen/html/index.html

References

[1] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In *ICCV*, pages 694–699, 1995. B.2

[2] Laurent D. Cohen. On active contour models and balloons. *CVGIP: Image Underst.*, 53(2):211–218, 1991. 2.4

[3] Ghassan Hamarneh, Tim McInerney, and Demetri Terzopoulos. Deformable organisms for automatic medical image analysis. In *MICCAI*, pages 66–76, 2001. 1

[4] Ghassan Hamarneh and Chris McIntosh. Physics-based deformable organisms for medical image analysis. *SPIE Medical Imaging*, 5747:326–335, 2005. 1

[5] G. Hamarnerh and C. McIntosh. *Parametric and Geometric Deformable Models: An application in Biomaterials and Medical Imagery*, chapter 12: Deformable Organisms for Medical Image Analysis. Springer Publishers, 1 edition, 2006. 1

[6] C. McIntosh and G. Hamarnerh. Spinal crawlers: Deformable organisms for spinal cord segmentation and analysis. *MICCAI*, 2006. 1

[7] C. McIntosh and G. Hamarnerh. Vessel crawlers: 3d physically-based deformable organisms for vasulature segmentation and analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 1, 1, B.2

[8] D. Terzopoulos, X. Tu, and R. Grzeszczuk. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994. 2.6

IDO Reference Manual

Generated by Doxygen 1.4.7

Fri Jul 21 00:37:49 2006

Contents

1 IDO Namespace Index **1**

1.1 IDO Namespace List 1

2 IDO Hierarchical Index **3**

2.1 IDO Class Hierarchy 3

3 IDO Class Index **7**

3.1 IDO Class List 7

4 IDO Namespace Documentation **11**

4.1 itk Namespace Reference 11

4.2 mial Namespace Reference 12

5 IDO Class Documentation **15**

5.1 BasicDefOrg_DefOrgViewerAdapter Class Reference 15

5.2 mial::BasicDefOrg_DefOrgViewerAdapter Class Reference 18

5.3 mial::Beh_Fitting< Type, nDims > Class Template Reference 22

5.4 mial::Beh_Growing< Type, CrawlerPhysType > Class Template Reference 23

5.5 mial::Beh_Growing< Type, CrawlerPhysType >::behaviorIn Struct Reference 24

5.6 mial::Beh_SearchForObject< Type, TInputImage, nDims > Class Template Reference . . 25

5.7 mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn Struct Reference . 28

5.8 mial::Beh_Spawning< Type, nDims > Class Template Reference 30

5.9 mial::Beh_TranslateAll< Type, nDims > Class Template Reference 31

5.10 mial::Beh_TranslateAll< Type, nDims >::behaviorIn Struct Reference 34

5.11 mial::Beh_UniformScale< Type, nDims > Class Template Reference 36

5.12 mial::Beh_UniformScale< Type, nDims >::behaviorIn Struct Reference 39

5.13 mial::Behavior< Type, nDims > Class Template Reference 41

5.14 mial::Behavior< Type, nDims >::behaviorIn Struct Reference 45

5.15 mial::Behavior< Type, nDims >::Error Struct Reference 47

5.16 mial::Blank_DefOrgViewerAdapter Class Reference 48

5.17 mial::ControlCenter< Type, nDims > Class Template Reference 51
5.18 mial::Ctrl_ScheduleDriven< Type, nDims > Class Template Reference 55
5.19 mial::Ctrl_SensoryDriven< Type, nDims > Class Template Reference 57
5.20 mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType > Class Template Reference . 58
5.21 mial::Def_Translation< DataType, nDims, MType, VType > Class Template Reference . . 59
5.22 mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn Struct Reference 61
5.23 mial::Def_UniformScale< DataType, nDims, MType, VType > Class Template Reference 63
5.24 mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn Struct Reference 65
5.25 mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType > Class Template Refer
5.26 mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn Struc
5.27 mial::DefOrgAdapter_VesselCrawler Class Reference 71
5.28 mial::DefOrgLayerStruct Struct Reference 75
5.29 mial::DefOrgPropertyStruct Struct Reference 76
5.30 DefOrgViewer Class Reference 77
5.31 DefOrgViewerAdapter Class Reference 79
5.32 mial::DefOrgViewerAdapterBase Class Reference 80
5.33 mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE > Class Template Reference 86
5.34 mial::DefOrgViewerAdapterDynamicLoader Class Reference 89
5.35 DefOrgViewerGUI Class Reference 90
5.36 mial::Deformation< DataType, nDims, MType, VType > Class Template Reference . . . 91
5.37 mial::Deformation< DataType, nDims, MType, VType >::DefArgSet Struct Reference . . 94
5.38 mial::Deformation< DataType, nDims, MType, VType >::deformationIn Struct Reference 95
5.39 mial::Deformation< DataType, nDims, MType, VType >::Error Struct Reference 96
5.40 fftw_iodim_do_not_use_me Struct Reference 97
5.41 mial::GenerateDefOrgHelpers Class Reference 98
5.42 mial::Geom_MeshSpatialObject< dType, nDims, MType, VType > Class Template Reference 99
5.43 mial::Geom_VesselCrawler< dType, nDims, MType, VType > Class Template Reference 105
5.44 mial::Geom_vGeometry< dType, nDims, MType, VType > Class Template Reference . . 107
5.45 mial::Geometric< dType, nDims, MType, VType > Class Template Reference 109
5.46 mial::Geometric< dType, nDims, MType, VType >::Error Struct Reference 117
5.47 mial::ImageViewerDescriptor Struct Reference 118
5.48 itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Class Template Referen
5.49 itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType > Class Template Reference 1
5.50 mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType > Class Template Reference 1
5.51 mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::DefArgSet Struct Referenc
5.52 itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Class Template F

5.53 itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Class Templa
5.54 mial::Organism< DataType, nDims > Class Template Reference 135
5.55 mial::OrganismScheduler Class Reference 139
5.56 mial::OutputImageDescriptorStruct Struct Reference 140
5.57 mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType > Class Template Reference 141
5.58 mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::Error Struct Reference 148
5.59 mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType > Class Template Reference 149
5.60 mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType > Class Template Reference 15
5.61 mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::Error Struct Reference 154
5.62 mial::Physics< Type, nDims, MType, VType > Class Template Reference 155
5.63 mial::Physics< Type, nDims, MType, VType >::Error Struct Reference 159
5.64 mial::Sense_AvgIntensity< DataType, TInputImage, nDims > Class Template Reference . 160
5.65 mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorIn Struct Reference 162
5.66 mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorOut Struct Reference 163
5.67 mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims > Class Template Reference 164
5.68 mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorIn Struct Reference 166
5.69 mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorOut Struct Reference 167
5.70 mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims > Class Template Reference 168
5.71 mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::In Struct Reference 170
5.72 mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::Out Struct Reference 171
5.73 mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims > Class Template Reference 172
5.74 mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::In Struct Reference 174
5.75 mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::Out Struct Reference 175
5.76 mial::Sense_SenseToGrow< DataType, TInputImage, nDims > Class Template Reference 176
5.77 mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::In Struct Reference . . . 178
5.78 mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::Out Struct Reference . . 179
5.79 mial::Sensor Class Reference 180
5.80 mial::Sensor::sensorIn Struct Reference 182
5.81 mial::Sensor::sensorOut Struct Reference 183
5.82 mial::SOViewerDescriptor Struct Reference 184
5.83 mial::SpatialObjectDescriptorStruct Struct Reference 185
5.84 mial::SpringMassDeformation< DataType, nDims, MType, VType > Class Template Reference 186
5.85 mial::SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet Struct Reference 188
5.86 mial::UnixOS Class Reference 190
5.87 VistVTKCellsClass Class Reference 191
5.88 mial::VistVTKCellsClass< MESHTYPE > Class Template Reference 192

5.89 vtkDefOrgViewerWithKW Class Reference 193
5.90 mial::vtkDefOrgViewerWithKWState Class Reference 200
5.91 vtkFlRenderWindowInteractor Class Reference 202
5.92 vtkGenerateDefOrgDialog Class Reference 203
5.93 vtkImageImport Class Reference 204
5.94 vtkMyCallback Class Reference 206
5.95 mial::WindowsOS Class Reference 207

Chapter 1

IDO Namespace Index

1.1 IDO Namespace List

Here is a list of all documented namespaces with brief descriptions:

itk 11
mial 12

Chapter 2

IDO Hierarchical Index

2.1 IDO Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

mial::Behavior< Type, nDims > . . . 41
mial::Beh_Fitting< Type, nDims > . . . 22
mial::Beh_SearchForObject< Type, TInputImage, nDims > . . . 25
mial::Beh_Spawning< Type, nDims > . . . 30
mial::Beh_TranslateAll< Type, nDims > . . . 31
mial::Beh_UniformScale< Type, nDims > . . . 36
mial::Behavior< Type, nDims >::behaviorIn . . . 45
mial::Beh_Growing< Type, CrawlerPhysType >::behaviorIn . . . 24
mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn . . . 28
mial::Beh_TranslateAll< Type, nDims >::behaviorIn . . . 34
mial::Beh_UniformScale< Type, nDims >::behaviorIn . . . 39
mial::Behavior< Type, nDims >::Error . . . 47
mial::Behavior< float, nDims > . . . 41
mial::Behavior< Type, 3 > . . . 41
mial::Beh_Growing< Type, CrawlerPhysType > . . . 23
mial::ControlCenter< Type, nDims > . . . 51
mial::Ctrl_ScheduleDriven< Type, nDims > . . . 55
mial::Ctrl_SensoryDriven< Type, nDims > . . . 57
mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType > . . . 58
mial::DefOrgLayerStruct . . . 75
mial::DefOrgPropertyStruct . . . 76
DefOrgViewerAdapter . . . 79
mial::DefOrgViewerAdapterBase . . . 80
mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE > . . . 86
BasicDefOrg_DefOrgViewerAdapter . . . 15
mial::BasicDefOrg_DefOrgViewerAdapter . . . 18
mial::Blank_DefOrgViewerAdapter . . . 48
mial::DefOrgAdapter_VesselCrawler . . . 71
mial::DefOrgViewerAdapterBaseTemplated< unsigned char, float > . . . 86
mial::DefOrgViewerAdapterDynamicLoader . . . 89
DefOrgViewerGUI . . . 90
DefOrgViewer . . . 77

mial::Deformation< DataType, nDims, MType, VType > 91
mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType > . . 126
mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType > 67
mial::LevelSetDeformation< float, nDims, TDistanceImageType, MType, VType > 126
mial::SpringMassDeformation< DataType, nDims, MType, VType > 186
mial::Def_Translation< DataType, nDims, MType, VType > 59
mial::Def_UniformScale< DataType, nDims, MType, VType > 63
mial::SpringMassDeformation< float, nDims, MType, VType > 186
mial::Deformation< DataType, nDims, MType, VType >::DefArgSet 94
mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::DefArgSet 128
mial::SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet 188
mial::Deformation< DataType, nDims, MType, VType >::deformationIn 95
mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn 61
mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn 65
mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn 69
mial::Deformation< DataType, nDims, MType, VType >::Error 96
mial::Deformation< float, nDims, MType, VType > 91
fftw_iodim_do_not_use_me 97
mial::GenerateDefOrgHelpers 98
mial::Geometric< dType, nDims, MType, VType > 109
mial::Geom_MeshSpatialObject< dType, nDims, MType, VType > 99
mial::Geom_VesselCrawler< dType, nDims, MType, VType > 105
mial::Geom_vGeometry< dType, nDims, MType, VType > 107
mial::Geometric< dType, nDims, MType, VType >::Error 117
mial::ImageViewerDescriptor 118
itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, float, nDims > 119
mial::Organism< DataType, nDims > 135
itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > 119
itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType > 123
itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > 129
itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > 132
mial::Organism< float, nDims > 135
mial::OrganismScheduler 139
mial::UnixOS 190
mial::WindowsOS 207
mial::OutputImageDescriptorStruct 140
mial::Physics< Type, nDims, MType, VType > 155
mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType > 141
mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType > 152
mial::Phys_Euler< float, TGradientImage, nDims, MType, VType > 141
mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType > 149
mial::Physics< Type, nDims, MType, VType >::Error 159
mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::Error 148
mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::Error 154
mial::Physics< float, nDims, MType, VType > 155

mial::Physics< Type, nDims > . . . 155
mial::Sensor . . . 180
mial::Sense_AvgIntensity< DataType, TInputImage, nDims > . . . 160
mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims > . . . 164
mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims > . . . 168
mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims > . . . 172
mial::Sense_SenseToGrow< DataType, TInputImage, nDims > . . . 176
mial::Sensor::sensorIn . . . 182
mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorIn . . . 162
mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorIn . . . 166
mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::In . . . 170
mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::In . 174
mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::In . . . 178
mial::Sensor::sensorOut . . . 183
mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorOut . . . 163
mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorOut . . . 167
mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::Out . . . 171
mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::Out 175
mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::Out . . . 179
mial::SOViewerDescriptor . . . 184
mial::SpatialObjectDescriptorStruct . . . 185
VistVTKCellsClass . . . 191
mial::VistVTKCellsClass< MESHTYPE > . . . 192
vtkDefOrgViewerWithKW . . . 193
mial::vtkDefOrgViewerWithKWState . . . 200
vtkFlRenderWindowInteractor . . . 202
vtkGenerateDefOrgDialog . . . 203
vtkImageImport . . . 204
vtkMyCallback . . . 206

Chapter 3

IDO Class Index

3.1 IDO Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BasicDefOrg_DefOrgViewerAdapter . . . 15
mial::BasicDefOrg_DefOrgViewerAdapter . . . 18
mial::Beh_Fitting< Type, nDims > (Fit the front-most layer of a vessel crawler to the local vasculature) . . . 22
mial::Beh_Growing< Type, CrawlerPhysType > (Add a new layer to vessel crawler) . . . 23
mial::Beh_Growing< Type, CrawlerPhysType >::behaviorIn (A structure defining the inputs for the behavior) . . . 24
mial::Beh_SearchForObject< Type, TInputImage, nDims > . . . 25
mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn (A structure defining the inputs for the behavior) . . . 28
mial::Beh_Spawning< Type, nDims > (A behavior designed to spawn new vessel crawlers upon the detection of a bifurcation [1]) . . . 30
mial::Beh_TranslateAll< Type, nDims > . . . 31
mial::Beh_TranslateAll< Type, nDims >::behaviorIn (A structure defining the inputs for the behavior) . . . 34
mial::Beh_UniformScale< Type, nDims > . . . 36
mial::Beh_UniformScale< Type, nDims >::behaviorIn (A structure defining the inputs for the behavior) . . . 39
mial::Behavior< Type, nDims > (Perform one or many sequences of actions) . . . 41
mial::Behavior< Type, nDims >::behaviorIn (A structure defining the inputs of a Behavior) . . . 45
mial::Behavior< Type, nDims >::Error (The error structure thrown by behaviors run method when an error is encountered) . . . 47
mial::Blank_DefOrgViewerAdapter . . . 48
mial::ControlCenter< Type, nDims > (The brain of the organism responsible for making decisions and taking action based upon their outcome) . . . 51
mial::Ctrl_ScheduleDriven< Type, nDims > (ControlCenter that reads the schedule and peforms the listed behaviors sequentially) . . . 55
mial::Ctrl_SensoryDriven< Type, nDims > . . . 57
mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType > (ControlCenter that specific for vessel crawlers [1]) . . . 58
mial::Def_Translation< DataType, nDims, MType, VType > (An example spring-mass deformation that translates the mesh) . . . 59

mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn (A structure defining the inputs for the deformation) 61
mial::Def_UniformScale< DataType, nDims, MType, VType > (An example spring-mass deformation that scales the organism by increasing the rest lengths of all its springs) . . . 63
mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn (A structure defining the inputs for the deformation) 65
mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType > (An example level set deformation that scales the deformable organism by adding a single user-provided value to each voxel of the level set) 67
mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn (A structure defining the inputs for the deformation) 69
mial::DefOrgAdapter_VesselCrawler (A deforg adapter for vessel crawlers [1]) 71
mial::DefOrgLayerStruct 75
mial::DefOrgPropertyStruct 76
DefOrgViewer 77
DefOrgViewerAdapter 79
mial::DefOrgViewerAdapterBase 80
mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE > 86
mial::DefOrgViewerAdapterDynamicLoader 89
DefOrgViewerGUI 90
mial::Deformation< DataType, nDims, MType, VType > (An abstract base class for deformations) 91
mial::Deformation< DataType, nDims, MType, VType >::DefArgSet (These define the standard "hidden" argument set for a deformation that allow to manipulate the model) . . . 94
mial::Deformation< DataType, nDims, MType, VType >::deformationIn (A structure defining the inputs of a Deformation) 95
mial::Deformation< DataType, nDims, MType, VType >::Error (The error structure) 96
fftw_iodim_do_not_use_me 97
mial::GenerateDefOrgHelpers 98
mial::Geom_MeshSpatialObject< dType, nDims, MType, VType > (Derived geometric class based on a spatial object) 99
mial::Geom_VesselCrawler< dType, nDims, MType, VType > (Derived geometric class based on a spatial object that includes a specific notion of layers) 105
mial::Geom_vGeometry< dType, nDims, MType, VType > 107
mial::Geometric< dType, nDims, MType, VType > (Topological characteristics of the organism) 109
mial::Geometric< dType, nDims, MType, VType >::Error 117
mial::ImageViewerDescriptor 118
itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > (A derived class that implements a deformable organism as an itk::ImageToImageFilter) . 119
itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType > (A vessel crawler [1] deformable organism for the segmentation and analysis of vasculature in 3D images) 123
mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType > (This class extends the basic deformation class with specific functionality for level set systems)126
mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::DefArgSet (A customized argument set) 128
itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > (A derived class that implements an itkOrganism that posses built in Phys_Euler and Ctrl_Schedule layers, along with corresponding behaviors and deformations) 129
itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > (A derived class that implements an itkOrganism that posses built in Phys_LevelSet and Ctrl_Schedule layers, along with corresponding behaviors and deformations) . . . 132
mial::Organism< DataType, nDims > (The abstract base container class that houses and connects all layers of the deformable organism) 135

mial::OrganismScheduler . . . 139
mial::OutputImageDescriptorStruct . . . 140
mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType > (A derived physics class capable of carrying out deformations of a spring mass system) . . . 141
mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::Error . . . 148
mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType > (A derived physics class capable of carrying out deformations of a spring mass system) . . . 149
mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType > (A derived physics class capable of carrying out deformations of a layered spring mass system) . . . 152
mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::Error 154
mial::Physics< Type, nDims, MType, VType > (Simluates the deformation dynamics, thereby, modifying actual positions of nodes belonging to the organism) . . . 155
mial::Physics< Type, nDims, MType, VType >::Error (A structure containing error information that should be filled and thrown whenever an error in the simulation occurs) . . . 159
mial::Sense_AvgIntensity< DataType, TInputImage, nDims > (Derived sensory class for computing image AvgIntensity) . . . 160
mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorIn . . . 162
mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorOut . . . 163
mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims > (Derived sensory class for computing image gradient) . . . 164
mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorIn . . . 166
mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorOut . . . 167
mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims > (Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer) . . . 168
mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::In . . . 170
mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::Out . . . 171
mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims > (Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer) . . . 172
mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::In . . . 174
mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::Out . . . 175
mial::Sense_SenseToGrow< DataType, TInputImage, nDims > (Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer) . . . 176
mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::In . . . 178
mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::Out . . . 179
mial::Sensor (Sensors provide the organism with its view of the world) . . . 180
mial::Sensor::sensorIn (A structure defining the inputs of a sensor) . . . 182
mial::Sensor::sensorOut (A structure defining the output of a sensor) . . . 183
mial::SOViewerDescriptor . . . 184
mial::SpatialObjectDescriptorStruct . . . 185
mial::SpringMassDeformation< DataType, nDims, MType, VType > (This class extends the basic deformation class with specific functionality for spring mass systems) . . . 186
mial::SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet (A customized argument set) . . . 188
mial::UnixOS . . . 190
VistVTKCellsClass . . . 191
mial::VistVTKCellsClass< MESHTYPE > . . . 192
vtkDefOrgViewerWithKW . . . 193
mial::vtkDefOrgViewerWithKWState . . . 200
vtkFlRenderWindowInteractor . . . 202
vtkGenerateDefOrgDialog . . . 203

vtkImageImport 204
vtkMyCallback 206
mial::WindowsOS 207

Chapter 4

IDO Namespace Documentation

4.1 itk Namespace Reference

Classes

- class ItkVesselCrawler

  A vessel crawler [1] deformable organism for the segmentation and analysis of vasculature in 3D images.

- class ItkOrganism

  A derived class that implements a deformable organism as an itk::ImageToImageFilter.

- class Org_EulerSchedule

  A derived class that implements an itkOrganism that posses built in Phys_Euler and Ctrl_Schedule layers, along with corresponding behaviors and deformations.

- class Org_LevelSetSchedule

  A derived class that implements an itkOrganism that posses built in Phys_LevelSet and Ctrl_Schedule layers, along with corresponding behaviors and deformations.

4.2 mial Namespace Reference

Classes

- class BasicDefOrg_DefOrgViewerAdapter
- class Blank_DefOrgViewerAdapter
- struct DefOrgLayerStruct
- struct DefOrgPropertyStruct
- struct SpatialObjectDescriptorStruct
- struct OutputImageDescriptorStruct
- class DefOrgViewerAdapterBase
- class DefOrgViewerAdapterBaseTemplated
- class VistVTKCellsClass
- class DefOrgViewerAdapterDynamicLoader
- class GenerateDefOrgHelpers
- struct SOViewerDescriptor
- struct ImageViewerDescriptor
- class vtkDefOrgViewerWithKWState
- class Beh_Fitting

  Fit the front-most layer of a vessel crawler to the local vasculature.

- class Beh_Growing

  Add a new layer to vessel crawler.

- class Beh_Spawning

  A behavior designed to spawn new vessel crawlers upon the detection of a bifurcation [1].

- class Ctrl_VesselCrawler

  ControlCenter that specific for vessel crawlers [1].

- class DefOrgAdapter_VesselCrawler

  A deforg adapter for vessel crawlers [1].

- class Geom_VesselCrawler

  Derived geometric class based on a spatial object that includes a specific notion of layers.

- class Phys_VesselCrawlerEuler

  A derived physics class capable of carrying out deformations of a layered spring mass system.

- class Sense_HessianBased

  Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

- class Sense_ProjectiveSpherical

  Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

- class Sense_SenseToGrow

  Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

- class Behavior

  Perform one or many sequences of actions.

- class Beh_SearchForObject
- class Beh_TranslateAll
- class Beh_UniformScale
- class ControlCenter

  The brain of the organism responsible for making decisions and taking action based upon their outcome.

- class Ctrl_ScheduleDriven

  ControlCenter that reads the schedule and peforms the listed behaviors sequentially.

- class Ctrl_SensoryDriven
- class Geometric

  Topological characteristics of the organism.

- class Geom_MeshSpatialObject

  Derived geometric class based on a spatial object.

- class Geom_vGeometry
- class Organism

  The abstract base container class that houses and connects all layers of the deformable organism.

- class OrganismScheduler
- class UnixOS
- class WindowsOS
- class Deformation

  An abstract base class for deformations.

- class LevelSetDeformation

  This class extends the basic deformation class with specific functionality for level set systems.

- class Physics

  Simluates the deformation dynamics, thereby, modifying actual positions of nodes belonging to the organism.

- class SpringMassDeformation

  This class extends the basic deformation class with specific functionality for spring mass systems.

- class Def_Translation

  An example spring-mass deformation that translates the mesh.

- class Def_UniformScale

  An example spring-mass deformation that scales the organism by increasing the rest lengths of all its springs.

- class Def_UniformScaleLevelSet

  An example level set deformation that scales the deformable organism by adding a single user-provided value to each voxel of the level set.

- class Phys_Euler

  A derived physics class capable of carrying out deformations of a spring mass system.

- class Phys_LevelSet

  A derived physics class capable of carrying out deformations of a spring mass system.

- class Sensor

  Sensors provide the organism with its view of the world.

- class Sense_AvgIntensity

  Derived sensory class for computing image AvgIntensity.

- class Sense_Gradient

  Derived sensory class for computing image gradient.

Typedefs

- typedef unsigned char PixelType
- typedef float DataType
- typedef unsigned char PixelType
- typedef float DataType
- typedef itk::SceneSpatialObject< N_DIMS >::Pointer itkScenePointer
- typedef DefOrgViewerAdapterBase *(*) DEFORG_LOAD_FUNCTION ()
- typedef unsigned char PixelType
- typedef float DataType

Functions

- int BYUToMeta (std::string BYUFileName, std::string metaFileName)

4.2.1 Function Documentation

4.2.1.1 int mial::BYUToMeta (std::string *BYUFileName*, std::string *metaFileName*)

Convert BYU meshes into .meta meshes

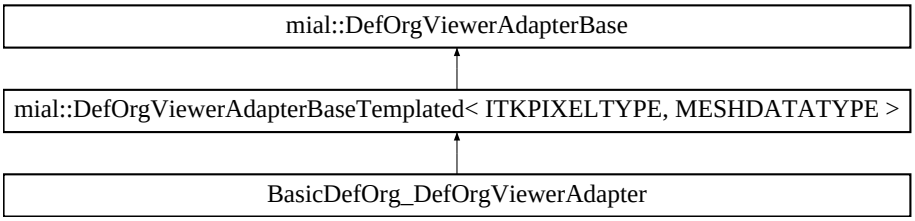
Definition at line 4 of file BYUToMeta.cxx.

Chapter 5

IDO Class Documentation

5.1 BasicDefOrg_DefOrgViewerAdapter Class Reference

Inheritance diagram for BasicDefOrg_DefOrgViewerAdapter::



Public Types

- typedef Geom_MeshSpatialObject< DataType, N_DIMS > GeometricType
- typedef itk::CovariantVector< DataType, N_DIMS > GradientPixelType
- typedef itk::Image< GradientPixelType, N_DIMS > GradientImageType
- typedef Phys_LevelSet< DataType, ImageType, N_DIMS > LevelSetPhysicsType
- typedef Phys_Euler< DataType, GradientImageType, N_DIMS > EulerPhysicsType
- typedef Sense_Gradient< DataType, ImageType, GradientImageType, N_DIMS > GradientSensorType
- typedef Ctrl_ScheduleDriven< DataType, N_DIMS > CognitiveType
- typedef Beh_TranslateAll< DataType, N_DIMS > Beh_TranslateAllType
- typedef Def_Translation< DataType, N_DIMS > Def_TranslateAllType
- typedef itk::ItkOrganism< ImageType, ImageType, GradientImageType, DataType, N_DIMS > OrganismType
- typedef itk::DefaultDynamicMeshTraits< DataType, N_DIMS, N_DIMS > MeshTrait
- typedef itk::Mesh< DataType, N_DIMS, MeshTrait > MeshType
- typedef MeshType::Pointer MeshTypePointer

Public Member Functions

- virtual void SetupOrganism ()
- virtual void UpdateOrganism ()
- virtual void PopulateVtkImage ()
- virtual void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid *vtkGrid)
- virtual void PopulateItkScene ()
- virtual int MaxNumberOfOutputItkSpatialObjects ()
- virtual int MaxNumberOfOutputImages ()

5.1.1 Detailed Description

Definition at line 30 of file BasicDefOrg_DefOrgViewerAdapter.h.

5.1.2 Member Function Documentation

5.1.2.1 virtual int BasicDefOrg_DefOrgViewerAdapter::MaxNumberOfOutputImages () [virtual]

Derived class should override this method to let DefOrgViewer know how many output Image Volumes this DefOrg wants

Implements mial::DefOrgViewerAdapterBase.

5.1.2.2 virtual int BasicDefOrg_DefOrgViewerAdapter::MaxNumberOfOutputItkSpatialObjects () [virtual]

Derived class should override this method to let DefOrgViewer know how many output Spatial Objects this DefOrg wants

Implements mial::DefOrgViewerAdapterBase.

5.1.2.3 virtual void BasicDefOrg_DefOrgViewerAdapter::PopulateItkScene () [virtual]

Derived class should provide a itkScene that it wants to display when the mesh is first loaded (optional)

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

5.1.2.4 virtual void BasicDefOrg_DefOrgViewerAdapter::PopulateVtkImage () [virtual]

Derived class should provide a vtkImage that it wants to display when the Image Volumes is first loaded (optional)

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

5.1.2.5 virtual void BasicDefOrg_DefOrgViewerAdapter::PopulateVtkUnstructuredGrid (vtkUnstructuredGrid * *vtkGrid*) [virtual]

Currently not used. Replaced with ItkScene and SOViewer

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

5.1.2.6 virtual void BasicDefOrg_DefOrgViewerAdapter::SetupOrganism () [virtual]

Viewer will call this method as instructed from GUI. Derived class should initialize the organism itself using the DefOrg properties (via GetOrganismProperty) supplied through the GUI

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

5.1.2.7 virtual void BasicDefOrg_DefOrgViewerAdapter::UpdateOrganism () [virtual]

During each step of simulation, viewer will call this method. It is the derived class responsibility to update the OutputItkScenes and/or OutputImageVolumes using implemented helper methods

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

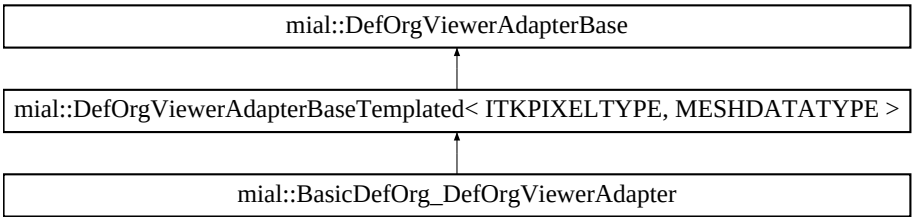
The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/BasicDefOrg_DefOrgViewer-Adapter.h

5.2 mial::BasicDefOrg_DefOrgViewerAdapter Class Reference

`#include <BasicDefOrg_DefOrgViewerAdapter.h>`

Inheritance diagram for mial::BasicDefOrg_DefOrgViewerAdapter::



Public Types

- typedef Geom_MeshSpatialObject< DataType, N_DIMS > GeometricType
- typedef itk::CovariantVector< DataType, N_DIMS > GradientPixelType
- typedef itk::Image< GradientPixelType, N_DIMS > GradientImageType
- typedef Phys_LevelSet< DataType, ImageType, N_DIMS > LevelSetPhysicsType
- typedef Phys_Euler< DataType, GradientImageType, N_DIMS > EulerPhysicsType
- typedef Sense_Gradient< DataType, ImageType, GradientImageType, N_DIMS > GradientSensorType
- typedef Sense_AvgIntensity< DataType, ImageType, N_DIMS > AvgIntensitySensorType
- typedef Ctrl_ScheduleDriven< DataType, N_DIMS > CognitiveType
- typedef Beh_SearchForObject< DataType, ImageType, N_DIMS > Beh_SearchForObjectType
- typedef Beh_TranslateAll< DataType, N_DIMS > Beh_TranslateAllType
- typedef Def_Translation< DataType, N_DIMS > Def_TranslateAllType
- typedef Beh_UniformScale< DataType, N_DIMS > Beh_UniformScaleType
- typedef Def_UniformScale< DataType, N_DIMS > Def_UniformScaleEulerType
- typedef Def_UniformScaleLevelSet< DataType, N_DIMS, typename LevelSetPhysicsType::InternalImageType > Def_UniformScaleLevelSetType
- typedef itk::ItkOrganism< ImageType, ImageType, GradientImageType, DataType, N_DIMS > OrganismType
- typedef itk::DefaultDynamicMeshTraits< DataType, N_DIMS, N_DIMS > MeshTrait
- typedef itk::Mesh< DataType, N_DIMS, MeshTrait > MeshType
- typedef MeshType::Pointer MeshTypePointer

Public Member Functions

- BasicDefOrg_DefOrgViewerAdapter ()
- virtual void SetupOrganism ()
- virtual void UpdateOrganism ()
- virtual void PopulateVtkImage ()
- virtual void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid *vtkGrid)
- virtual void PopulateItkScene ()

- virtual void HandleUserMouseInteraction (vtkTransform *userTransformation)
- virtual int MaxNumberOfOutputItkSpatialObjects ()
- virtual unsigned int MaxNumberOfOutputImages ()

5.2.1 Detailed Description

This is an example adapter

Definition at line 36 of file BasicDefOrg_DefOrgViewerAdapter.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 BasicDefOrg_DefOrgViewerAdapter::BasicDefOrg_DefOrgViewerAdapter ()

Constructor should define the property the organism exposes at run-time using AddOrganismProperty helper method

Definition at line 3 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::AddOrganismProperty().

5.2.3 Member Function Documentation

5.2.3.1 void BasicDefOrg_DefOrgViewerAdapter::HandleUserMouseInteraction (vtkTransform * *userTransformation*) [virtual]

This method is called when the user changes the DefOrg. It is the responsibility of the adapter to update the internal data structure of the DefOrg. The user transformation argument supplies the transformation that is specified visually by the user

Implements mial::DefOrgViewerAdapterBase.

Definition at line 46 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

5.2.3.2 unsigned int BasicDefOrg_DefOrgViewerAdapter::MaxNumberOfOutputImages () [virtual]

Should return n, the number of Image volumes this organism exports. The viewer would create n-1 secondary windows for images. The output of the image volumes are usually assigned to during PopulateVtkImage() and UpdateOrganism() The primary output will be displayed in the primary window

Implements mial::DefOrgViewerAdapterBase.

Definition at line 14 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

5.2.3.3 int BasicDefOrg_DefOrgViewerAdapter::MaxNumberOfOutputItkSpatialObjects () [virtual]

Should return n, the number of ItkSpatialObjects this organism exports. The viewer would create n-1 secondary windows for Spatial objects. The output of the spatialoutputs are usually assigned to during PopulateItkScene() and UpdateOrganism() The primary output will be displayed in the primary window

Implements mial::DefOrgViewerAdapterBase.

Definition at line 9 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

5.2.3.4 void BasicDefOrg_DefOrgViewerAdapter::PopulateItkScene () [virtual]

This method is called when the user presses Load Mesh

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 18 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_MeshFileName, and mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects.

5.2.3.5 void BasicDefOrg_DefOrgViewerAdapter::PopulateVtkImage () [virtual]

This method is called when the user presses Load Image

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 28 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_ImageFileName, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::m_InputImageReader, mial::DefOrgViewerAdapterBase::m_OutputImages, and mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateVtkImageHelper().

5.2.3.6 void BasicDefOrg_DefOrgViewerAdapter::PopulateVtkUnstructuredGrid (vtkUnstructuredGrid * *vtkGrid*) [virtual]

This method is currently not used. This method is originally for displaying meshes when SOViewer was not used

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 97 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_MeshFileName, and mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::MeshToUnstructuredGrid().

5.2.3.7 void BasicDefOrg_DefOrgViewerAdapter::SetupOrganism () [virtual]

This method is called when the user presses "Initialize/Setup organism from the GUI. The adapter should initialize the organism appropriately.

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 48 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::GetOrganismProperty(), mial::DefOrgViewerAdapterBase::m_Initialized, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::m_InputImageReader, and mial::DefOrgViewerAdapterBase::m_ScheduleFileName.

5.2.3.8 void BasicDefOrg_DefOrgViewerAdapter::UpdateOrganism () [virtual]

This method is called during each organism step. In this method, the adapter should indicate which outputs are modified, and call run on the organism.

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 40 of file BasicDefOrg_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects.

The documentation for this class was generated from the following files:

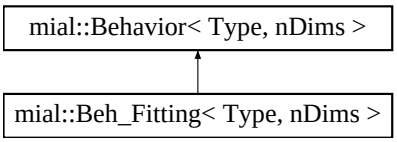
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgAdapter/BasicDefOrgAdapter/BasicDefOrg_DefOrgViewerAdapter.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/BasicDefOrg_DefOrgViewerAdapter.cxx
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgAdapter/BasicDefOrgAdapter/BasicDefOrg_DefOrgViewerAdapter.cxx

5.3 mial::Beh_Fitting< Type, nDims > Class Template Reference

Fit the front-most layer of a vessel crawler to the local vasculature.

`#include <Beh_Fitting.h>`

Inheritance diagram for mial::Beh_Fitting< Type, nDims >::



Public Member Functions

- virtual bool run (void *i, bool stream=false)
- Beh_Fitting ()

5.3.1 Detailed Description

template<class Type, int nDims> class mial::Beh_Fitting< Type, nDims >

Fit the front-most layer of a vessel crawler to the local vasculature.

See [1] for details.

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Definition at line 18 of file Beh_Fitting.h.

The documentation for this class was generated from the following files:

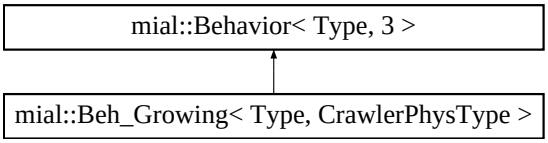
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Fitting.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Fitting.cxx

5.4 mial::Beh_Growing< Type, CrawlerPhysType > Class Template Reference

Add a new layer to vessel crawler.

`#include <Beh_Growing.h>`

Inheritance diagram for mial::Beh_Growing< Type, CrawlerPhysType >::



Public Member Functions

- virtual bool run (typename Behavior< Type, 3 >::behaviorIn *i, std::stringstream *s=NULL)
- Beh_Growing ()

Classes

- struct behaviorIn

  A structure defining the inputs for the behavior.

5.4.1 Detailed Description

template<class Type, class CrawlerPhysType> class mial::Beh_Growing< Type, CrawlerPhysType >

Add a new layer to vessel crawler.

Adds a new layer to vessel crawler at the control-center provided location and orientation. See [1] for details.

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Definition at line 19 of file Beh_Growing.h.

The documentation for this class was generated from the following files:

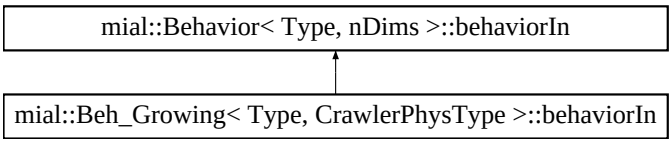
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Growing.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Growing.cxx

5.5 mial::Beh_Growing< Type, CrawlerPhysType >::behaviorIn Struct Reference

A structure defining the inputs for the behavior.

`#include <Beh_Growing.h>`

Inheritance diagram for mial::Beh_Growing< Type, CrawlerPhysType >::behaviorIn::



Public Member Functions

- behaviorIn ()

Public Attributes

- VectorType newAxis
- VectorType locXYZ
- Type radius
- Type growDistance
- Geom_VesselCrawler< Type, 3 > * vesselGeom

5.5.1 Detailed Description

template<class Type, class CrawlerPhysType> struct mial::Beh_Growing< Type, CrawlerPhysType >::behaviorIn

A structure defining the inputs for the behavior.

Since structures support public inheritance derived class must inherit from this class in their definitions of behaviorIn.

Definition at line 30 of file Beh_Growing.h.

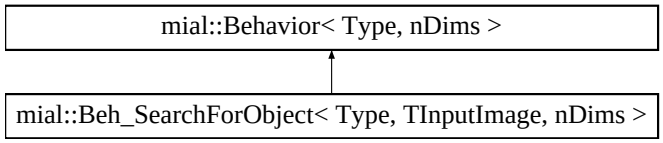
The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Growing.h

5.6 mial::Beh_SearchForObject< Type, TInputImage, nDims > Class Template Reference

`#include <Beh_SearchForObject.h>`

Inheritance diagram for mial::Beh_SearchForObject< Type, TInputImage, nDims >::



Public Types

- typedef Beh_SearchForObject Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef vnl_vector< Type > VectorType
- typedef Behavior< Type, nDims >::Error Error

Public Member Functions

- virtual bool run (typename Behavior< Type, nDims >::behaviorIn *i, std::stringstream *s)
- bool isFinished ()
- virtual bool update ()

  Public pure virtual function. This method allows behaviors to run in multiple stages.

- virtual void cleanUp ()

  Public pure virtual function. Method of cleaning up after the behavior.

Public Attributes

- TInputImage::ConstPointer image

  A pointer to the image used for the Sense_AvgIntensity sensor.

- Geometric< Type, nDims >::Pointer geomLayer

  A pointer for the geometric layer used for the Sense_AvgIntensity sensor.

Protected Member Functions

- Beh_SearchForObject ()

Classes

- struct behaviorIn

  A structure defining the inputs for the behavior.

5.6.1 Detailed Description

template<class Type, class TInputImage, int nDims> class mial::Beh_SearchForObject< Type, TInputImage, nDims >

A derived class of the behavioral ABC this class performs a translation towards the user provided location. Then, for a user provided duration it begins searching that area for a structure thats intensity is above the user provided threshold (via the Sense_AvgIntensity sensor). Upon finding an area with the minimum intensity level the Beh_UniformScale sub-behavior is launched, which scales up the deformable organism until the intensity condition is broken.

This behavior does not has Beh_TranslateAll and Beh_UniformScale sub-behaviors.

Parameters:

Type the internal type used for storage

nDims the number of dimensions

Definition at line 30 of file Beh_SearchForObject.h.

5.6.2 Member Function Documentation

5.6.2.1 template<class Type, class TInputImage, int nDims> void mial::Beh_SearchForObject< Type, TInputImage, nDims >::cleanUp () [virtual]

Public pure virtual function. Method of cleaning up after the behavior.

Since behaviors may be ran multiple times before being destructed they must provide a method to clean up for the next run. This method will be ran after the behavior asserts itself as finished.

Implements mial::Behavior< Type, nDims >.

Definition at line 157 of file Beh_SearchForObject.cxx.

5.6.2.2 template<class Type, class TInputImage, int nDims> bool mial::Beh_SearchForObject< Type, TInputImage, nDims >::update () [virtual]

Public pure virtual function. This method allows behaviors to run in multiple stages.

This method will be ran after the run method, until the isFinished() returns true.

Implements mial::Behavior< Type, nDims >.

Definition at line 79 of file Beh_SearchForObject.cxx.

The documentation for this class was generated from the following files:

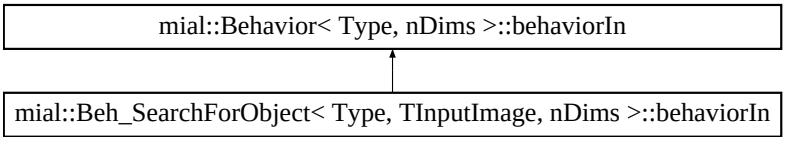
- C:/cmcintos/defOrgs/source/behavioral/Beh_SearchForObject.h
- C:/cmcintos/defOrgs/source/behavioral/Beh_SearchForObject.cxx

5.7 mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn Struct Reference

A structure defining the inputs for the behavior.

`#include <Beh_SearchForObject.h>`

Inheritance diagram for mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn::



Public Types

- typedef behaviorIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- bool fillFromStream (std::stringstream &args)

  A method for converting stream-based arguments into the structures members (marshalling).

Public Attributes

- double intensityRequirement

  The internal intensity requirement to be met.

- double duration

  The duration of the translation in deformable organism time.

- VectorType translateLoc

  The location to translate too.

Protected Member Functions

- behaviorIn ()

5.7.1 Detailed Description

template<class Type, class TInputImage, int nDims> struct mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn

A structure defining the inputs for the behavior.

Since structures support public inheritance derived class must inherit from this class in their definitions of behaviorIn.

Definition at line 49 of file Beh_SearchForObject.h.

5.7.2 Member Function Documentation

5.7.2.1 template<class Type, class TInputImage, int nDims> bool mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn::fillFromStream (std::stringstream & *args*) [inline]

A method for converting stream-based arguments into the structures members (marshalling).

Stream is expected as "duration intensityLevel xPosition yPosition zPosition(3d only)"

Definition at line 72 of file Beh_SearchForObject.h.

References mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn::duration, mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn::intensityRequirement, and mial::Beh_SearchForObject< Type, TInputImage, nDims >::behaviorIn::translateLoc.

The documentation for this struct was generated from the following file:

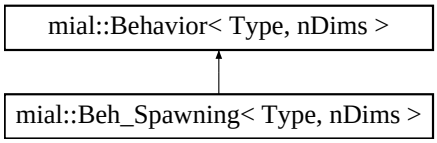
- C:/cmcintos/defOrgs/source/behavioral/Beh_SearchForObject.h

5.8 mial::Beh_Spawning< Type, nDims > Class Template Reference

A behavior designed to spawn new vessel crawlers upon the detection of a bifurcation [1].

`#include <Beh_Spawning.h>`

Inheritance diagram for mial::Beh_Spawning< Type, nDims >::



Public Member Functions

- virtual bool run (void *, bool stream=false)
- Beh_Spawning ()

5.8.1 Detailed Description

template<class Type, int nDims> class mial::Beh_Spawning< Type, nDims >

A behavior designed to spawn new vessel crawlers upon the detection of a bifurcation [1].

This behavior is used by the control center to spawn new vessel crawlers once a bifurcation has been detected.

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Definition at line 17 of file Beh_Spawning.h.

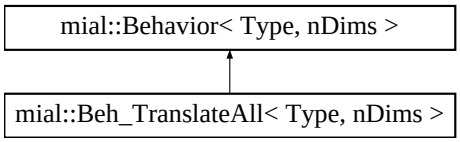
The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Spawning.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Beh_Spawning.cxx

5.9 mial::Beh_TranslateAll< Type, nDims > Class Template Reference

`#include <Beh_TranslateAll.h>`

Inheritance diagram for mial::Beh_TranslateAll< Type, nDims >::



Public Types

- typedef Beh_TranslateAll Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef Behavior< Type, nDims >::Error Error

Public Member Functions

- virtual bool run (typename Behavior< Type, nDims >::behaviorIn *i, std::stringstream *s)
- bool isFinished ()
- virtual bool update ()

  Public pure virtual function. This method allows behaviors to run in multiple stages.

- virtual void cleanUp ()

  Public pure virtual function. Method of cleaning up after the behavior.

Public Attributes

- double startTime

  Internal recording of when the behavior started running.

- double endTime

  Internal recording of when the behavior is set to end.

Protected Member Functions

- Beh_TranslateAll ()

Protected Attributes

- behaviorIn::Pointer input

  A pointer for the input.

Classes

- struct behaviorIn

  A structure defining the inputs for the behavior.

5.9.1 Detailed Description

template<class Type, int nDims> class mial::Beh_TranslateAll< Type, nDims >

A derived class of the behavioral ABC this class performs a translation with user provided direction and magnitude of the organism. The power of the framework is demonstrated here in that by eliciting its attached physics layer's translate deformation this behavior may be performed on any physics layer providing that deformation.

This behavior does not have any sub-behaviors.

Parameters:

Type the internal type used for storage

nDims the number of dimensions

Definition at line 26 of file Beh_TranslateAll.h.

5.9.2 Member Function Documentation

5.9.2.1 template<class Type, int nDims> void mial::Beh_TranslateAll< Type, nDims >::cleanUp () [virtual]

Public pure virtual function. Method of cleaning up after the behavior.

Since behaviors may be ran multiple times before being destructed they must provide a method to clean up for the next run. This method will be ran after the behavior asserts itself as finished.

Implements mial::Behavior< Type, nDims >.

Definition at line 71 of file Beh_TranslateAll.cxx.

References mial::Beh_TranslateAll< Type, nDims >::endTime, and mial::Beh_TranslateAll< Type, nDims >::startTime.

5.9.2.2 template<class Type, int nDims> bool mial::Beh_TranslateAll< Type, nDims >::update () [virtual]

Public pure virtual function. This method allows behaviors to run in multiple stages.

This method will be ran after the run method, until the isFinished() returns true.

Implements mial::Behavior< Type, nDims >.

Definition at line 58 of file Beh_TranslateAll.cxx.

References mial::Beh_TranslateAll< Type, nDims >::input, mial::Behavior< Type, nDims >::physLayer, and mial::Physics< Type, nDims, MType, VType >::runDeformation().

The documentation for this class was generated from the following files:

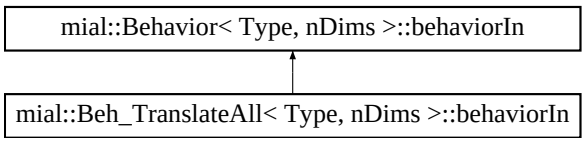
- C:/cmcintos/defOrgs/source/behavioral/Beh_TranslateAll.h
- C:/cmcintos/defOrgs/source/behavioral/Beh_TranslateAll.cxx

5.10 mial::Beh_TranslateAll< Type, nDims >::behaviorIn Struct Reference

A structure defining the inputs for the behavior.

`#include <Beh_TranslateAll.h>`

Inheritance diagram for mial::Beh_TranslateAll< Type, nDims >::behaviorIn::



Public Types

- typedef behaviorIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- bool fillFromStream (std::stringstream &args)

  A method for converting stream-based arguments into the structures members (marshalling).

Public Attributes

- double translateAmount [nDims]

  The amount in each dimension to translate by.

- int duration

  The duration of the translation in deformable organism time.

Protected Member Functions

- behaviorIn ()

5.10.1 Detailed Description

template<class Type, int nDims> struct mial::Beh_TranslateAll< Type, nDims >::behaviorIn

A structure defining the inputs for the behavior.

Since structures support public inheritance derived class must inherit from this class in their definitions of behaviorIn.

Definition at line 43 of file Beh_TranslateAll.h.

5.10.2 Member Function Documentation

5.10.2.1 template<class Type, int nDims> bool mial::Beh_TranslateAll< Type, nDims >::behaviorIn::fillFromStream (std::stringstream & *args*) [inline]

A method for converting stream-based arguments into the structures members (marshalling).

Stream is expected as "duration xAmount yAmount zAmount(3d only)"

Definition at line 61 of file Beh_TranslateAll.h.

References mial::Beh_TranslateAll< Type, nDims >::behaviorIn::duration, and mial::Beh_TranslateAll< Type, nDims >::behaviorIn::translateAmount.

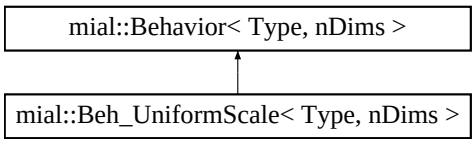
The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/behavioral/Beh_TranslateAll.h

5.11 mial::Beh_UniformScale< Type, nDims > Class Template Reference

`#include <Beh_UniformScale.h>`

Inheritance diagram for mial::Beh_UniformScale< Type, nDims >::



Public Types

- typedef Beh_UniformScale Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef Behavior< Type, nDims >::Error Error

Public Member Functions

- virtual bool run (typename Behavior< Type, nDims >::behaviorIn *i, std::stringstream *s)

  Example run method that scales the organism.

- bool isFinished ()
- virtual bool update ()

  Public pure virtual function. This method allows behaviors to run in multiple stages.

- virtual void cleanUp ()

  Public pure virtual function. Method of cleaning up after the behavior.

Public Attributes

- double startTime

  Internal recording of when the behavior started running.

- double endTime

  Internal recording of when the behavior is set to end.

Protected Member Functions

- Beh_UniformScale ()

Classes

- struct behaviorIn

  A structure defining the inputs for the behavior.

5.11.1 Detailed Description

template<class Type, int nDims> class mial::Beh_UniformScale< Type, nDims >

A derived class of the behavioral ABC this class performs a uniform scaling of the deformable organism with user provided duration and magnitude. The power of the framework is demonstrated here in that by eliciting its attached physics layer's scale deformation this behavior may be performed on any physics layer providing that deformation.

This behavior does not have any sub-behaviors.

Parameters:

Type the internal type used for storage

nDims the number of dimensions

Definition at line 24 of file Beh_UniformScale.h.

5.11.2 Member Function Documentation

5.11.2.1 template<class Type, int nDims> void mial::Beh_UniformScale< Type, nDims >::cleanUp () [virtual]

Public pure virtual function. Method of cleaning up after the behavior.

Since behaviors may be ran multiple times before being destructed they must provide a method to clean up for the next run. This method will be ran after the behavior asserts itself as finished.

Implements mial::Behavior< Type, nDims >.

Definition at line 56 of file Beh_UniformScale.cxx.

References mial::Beh_UniformScale< Type, nDims >::endTime, and mial::Beh_UniformScale< Type, nDims >::startTime.

5.11.2.2 template<class Type, int nDims> virtual bool mial::Beh_UniformScale< Type, nDims >::update () [inline, virtual]

Public pure virtual function. This method allows behaviors to run in multiple stages.

This method will be ran after the run method, until the isFinished() returns true.

Implements mial::Behavior< Type, nDims >.

Definition at line 76 of file Beh_UniformScale.h.

The documentation for this class was generated from the following files:

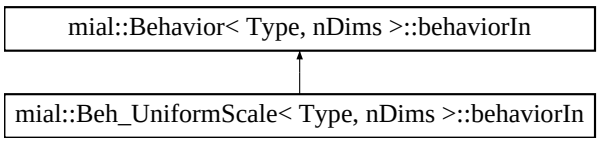
- C:/cmcintos/defOrgs/source/behavioral/Beh_UniformScale.h
- C:/cmcintos/defOrgs/source/behavioral/Beh_UniformScale.cxx

5.12 mial::Beh_UniformScale< Type, nDims >::behaviorIn Struct Reference

A structure defining the inputs for the behavior.

`#include <Beh_UniformScale.h>`

Inheritance diagram for mial::Beh_UniformScale< Type, nDims >::behaviorIn::



Public Types

- typedef behaviorIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- bool fillFromStream (std::stringstream &args)

  A method for converting stream-based arguments into the structures members (marshalling).

Public Attributes

- double scaleAmount

  The amount to scale the deformable organism by (passed onto the deformation).

- int duration

  The duration of the translation in deformable organism time.

Protected Member Functions

- behaviorIn ()

5.12.1 Detailed Description

template<class Type, int nDims> struct mial::Beh_UniformScale< Type, nDims >::behaviorIn

A structure defining the inputs for the behavior.

Since structures support public inheritance derived class must inherit from this class in their definitions of behaviorIn.

Definition at line 42 of file Beh_UniformScale.h.

5.12.2 Member Function Documentation

5.12.2.1 template<class Type, int nDims> bool mial::Beh_UniformScale< Type, nDims >::behaviorIn::fillFromStream (std::stringstream & *args*) [inline]

A method for converting stream-based arguments into the structures members (marshalling).

Stream is expected as "duration amount"

Definition at line 60 of file Beh_UniformScale.h.

References mial::Beh_UniformScale< Type, nDims >::behaviorIn::duration, and mial::Beh_UniformScale< Type, nDims >::behaviorIn::scaleAmount.

The documentation for this struct was generated from the following file:

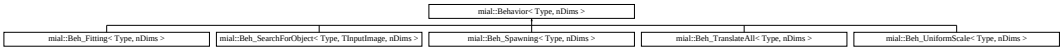
- C:/cmcintos/defOrgs/source/behavioral/Beh_UniformScale.h

5.13 mial::Behavior< Type, nDims > Class Template Reference

Perform one or many sequences of actions.

`#include <Behavior.h>`

Inheritance diagram for mial::Behavior< Type, nDims >::



Public Types

- typedef Behavior Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- virtual bool run (typename Behavior< Type, nDims >::behaviorIn *i, std::stringstream *s=NULL)=0

  Public pure virtual function. Implementation required to provide cognitive layer with a method to run the behavior.

- virtual bool update ()=0

  Public pure virtual function. This method allows behaviors to run in multiple stages.

- virtual void cleanUp ()=0

  Public pure virtual function. Method of cleaning up after the behavior.

- std::string getName ()

  Public accessor to the name string.

- bool setPhysLayer (Physics< Type, nDims > *phys)

  Public modifier to set the physics layer the behavior will be run on.

Protected Types

- typedef Physics< Type, nDims > PhysicsType
- typedef PhysicsType::MatrixType MatrixType
- typedef PhysicsType::VectorType VectorType

Protected Member Functions

- Behavior (std::string name)

  Default constructor, must provide a name.

Protected Attributes

- Behavior * subBehaviors

  List of sub-behaviors available.

- behaviorIn * input

  A pointer for the input.

- std::string name

  The name of the behavior used to identify it to other organisms, users, and behaviors.

- Physics< Type, nDims > * physLayer

  The physics layer used to simulate the actions.

Classes

- struct behaviorIn

  A structure defining the inputs of a Behavior.

- struct Error

  The error structure thrown by behaviors run method when an error is encountered.

5.13.1 Detailed Description

template<class Type, int nDims> class mial::Behavior< Type, nDims >

Perform one or many sequences of actions.

Behaviors are particular actions or sequences of actions that a deformable organism can perform. They may elicit sensory operations, deformations, and decisions. To ensure meaningful interaction with other organisms and users each behavior has a name. So for example, despite the action "running" being carried out differently by different animals each can always be told to run, or report that it is running. This class is currently classified as unstable.

Sub-behaviors are smaller behaviors performed as part of a larger action. This enables significant levels of abstraction, allowing users to issue single commands and carry out vast and complex sequences of actions, or small exact ones. For example, one could tell the organism to simply inflate, or one could tell it to segment which includes inflation.

Behaviors are responsible for un-marshaling their own run-time arguments.

Upon completion a behavior will set its state to finished.

Error reporting is handled by the Error structure, which reports an error message as well as the name of the behavior involved.

Wherever possible Behaviors should not contain "if" statements, as any such decision should be left up to the control center.

Parameters:

Type the internal storage type

nDims the number of dimensions

Definition at line 46 of file Behavior.h.

5.13.2 Member Function Documentation

5.13.2.1 template<class Type, int nDims> virtual void mial::Behavior< Type, nDims >::cleanUp () [pure virtual]

Public pure virtual function. Method of cleaning up after the behavior.

Since behaviors may be ran multiple times before being destructed they must provide a method to clean up for the next run. This method will be ran after the behavior asserts itself as finished.

Implemented in mial::Beh_SearchForObject< Type, TInputImage, nDims >, mial::Beh_TranslateAll< Type, nDims >, mial::Beh_UniformScale< Type, nDims >, and mial::Beh_SearchForObject< float, TInputImage, nDims >.

5.13.2.2 template<class Type, int nDims> virtual bool mial::Behavior< Type, nDims >::run (typename Behavior< Type, nDims >::behaviorIn * *i*, std::stringstream * *s* = NULL) [pure virtual]

Public pure virtual function. Implementation required to provide cognitive layer with a method to run the behavior.

Note: This method will only be ran once per behavior, subsequent calls will be made to the update method.

Note:that this function must take as an argument the basemost behaviorIn class, implementations can then downcast to their derived behaviorIn classes.

Parameters:

i the input behavior struct. Typically, setting to NULL indicates stream should be used.

s the input stream, typically only used if input struct is NULL.

Implemented in mial::Beh_SearchForObject< float, TInputImage, nDims >.

5.13.2.3 template<class Type, int nDims> virtual bool mial::Behavior< Type, nDims >::update () [pure virtual]

Public pure virtual function. This method allows behaviors to run in multiple stages.

This method will be ran after the run method, until the isFinished() returns true.

Implemented in mial::Beh_SearchForObject< Type, TInputImage, nDims >, mial::Beh_TranslateAll< Type, nDims >, mial::Beh_UniformScale< Type, nDims >, and mial::Beh_SearchForObject< float, TInputImage, nDims >.

The documentation for this class was generated from the following files:

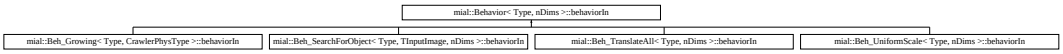
- C:/cmcintos/defOrgs/source/behavioral/abc/Behavior.h
- C:/cmcintos/defOrgs/source/behavioral/abc/Behavior.cxx

5.14 mial::Behavior< Type, nDims >::behaviorIn Struct Reference

A structure defining the inputs of a Behavior.

`#include <Behavior.h>`

Inheritance diagram for mial::Behavior< Type, nDims >::behaviorIn::



Public Types

- typedef behaviorIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- virtual bool fillFromStream (std::istream &args)

  Overide if you want to convert from stream input (be ran by name).

Protected Member Functions

- behaviorIn ()

5.14.1 Detailed Description

template<class Type, int nDims> struct mial::Behavior< Type, nDims >::behaviorIn

A structure defining the inputs of a Behavior.

Since structures support public inheritance derived class must inherit from this class in their definitions of behaviorIn.

Definition at line 72 of file Behavior.h.

5.14.2 Member Function Documentation

5.14.2.1 template<class Type, int nDims> virtual bool mial::Behavior< Type, nDims >::behaviorIn::fillFromStream (std::istream & *args*) [inline, virtual]

Overide if you want to convert from stream input (be ran by name).

Return false if method not provided or unsuccessful.

Definition at line 86 of file Behavior.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/behavioral/abc/Behavior.h

5.15 mial::Behavior< Type, nDims >::Error Struct Reference

The error structure thrown by behaviors run method when an error is encountered.

`#include <Behavior.h>`

Public Attributes

- std::string msg
- std::string name

5.15.1 Detailed Description

template<class Type, int nDims> struct mial::Behavior< Type, nDims >::Error

The error structure thrown by behaviors run method when an error is encountered.

Parameters:

msg A message to be sent to the users, should also be placed on std::cerr

name The name of the behavior.

Definition at line 63 of file Behavior.h.

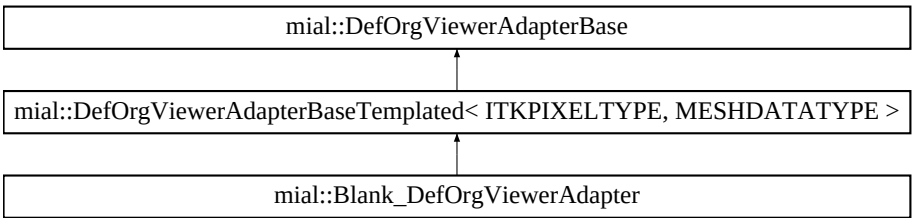
The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/behavioral/abc/Behavior.h

5.16 mial::Blank_DefOrgViewerAdapter Class Reference

`#include <Blank_DefOrgViewerAdapter.h>`

Inheritance diagram for mial::Blank_DefOrgViewerAdapter::



Public Member Functions

- Blank_DefOrgViewerAdapter ()
- virtual void SetupOrganism ()
- virtual void UpdateOrganism ()
- virtual void PopulateVtkImage ()
- virtual void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid ∗vtkGrid)
- virtual void PopulateItkScene ()
- virtual void HandleUserMouseInteraction (vtkTransform ∗userTransformation)
- virtual int MaxNumberOfOutputItkSpatialObjects ()
- virtual int MaxNumberOfOutputImages ()

5.16.1 Detailed Description

This is the template for generating new DefOrg skeleton code

Definition at line 31 of file Blank_DefOrgViewerAdapter.h.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 mial::Blank_DefOrgViewerAdapter::Blank_DefOrgViewerAdapter ()

Constructor should define the property the organism exposes at run-time using AddOrganismProperty helper method

Definition at line 15 of file Blank_DefOrgViewerAdapter.cxx.

5.16.3 Member Function Documentation

5.16.3.1 void mial::Blank_DefOrgViewerAdapter::HandleUserMouseInteraction (vtkTransform * *userTransformation*) [virtual]

This method is called when the user changes the DefOrg. It is the responsibility of the adapter to update the internal data structure of the DefOrg. The user transformation argument supplies the transformation that is specified visually by the user

Implements mial::DefOrgViewerAdapterBase.

Definition at line 40 of file Blank_DefOrgViewerAdapter.cxx.

5.16.3.2 int mial::Blank_DefOrgViewerAdapter::MaxNumberOfOutputImages () [virtual]

Should return n, the number of Image volumes this organism exports. The viewer would create n-1 secondary windows for images. The output of the image volumes are usually assigned to during PopulateVtkImage() and UpdateOrganism() The primary output will be displayed in the primary window

Implements mial::DefOrgViewerAdapterBase.

Definition at line 33 of file Blank_DefOrgViewerAdapter.cxx.

5.16.3.3 int mial::Blank_DefOrgViewerAdapter::MaxNumberOfOutputItkSpatialObjects () [virtual]

Should return n, the number of ItkSpatialObjects this organism exports. The viewer would create n-1 secondary windows for Spatial objects. The output of the spatialoutputs are usually assigned to during PopulateItkScene() and UpdateOrganism() The primary output will be displayed in the primary window

Implements mial::DefOrgViewerAdapterBase.

Definition at line 25 of file Blank_DefOrgViewerAdapter.cxx.

5.16.3.4 void mial::Blank_DefOrgViewerAdapter::PopulateItkScene () [virtual]

This method is called when the user presses Load Mesh

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 47 of file Blank_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects.

5.16.3.5 void mial::Blank_DefOrgViewerAdapter::PopulateVtkImage () [virtual]

This method is called when the user presses Load Image

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 67 of file Blank_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_OutputImages, and mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateVtkImageHelper().

5.16.3.6 void mial::Blank_DefOrgViewerAdapter::PopulateVtkUnstructuredGrid (vtkUnstructuredGrid * *vtkGrid*) [virtual]

This method is currently not used. This method is originally for displaying meshes when SOViewer was not used

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 111 of file Blank_DefOrgViewerAdapter.cxx.

5.16.3.7 void mial::Blank_DefOrgViewerAdapter::SetupOrganism () [virtual]

This method is called when the user presses "Initialize/Setup organism from the GUI. The adapter should initialize the organism appropriately.

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 105 of file Blank_DefOrgViewerAdapter.cxx.

5.16.3.8 void mial::Blank_DefOrgViewerAdapter::UpdateOrganism () [virtual]

This method is called during each organism step. In this method, the adapter should indicate which outputs are modified, and call run on the organism.

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 94 of file Blank_DefOrgViewerAdapter.cxx.

References mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects.

The documentation for this class was generated from the following files:

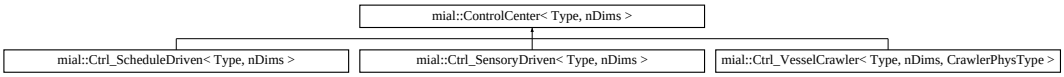
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgAdapter/BlankDefOrgAdapter/Blank_DefOrgViewerAdapter.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgAdapter/BlankDefOrgAdapter/Blank_DefOrgViewerAdapter.cxx

5.17 mial::ControlCenter< Type, nDims > Class Template Reference

The brain of the organism responsible for making decisions and taking action based upon their outcome.

`#include <ControlCenter.h>`

Inheritance diagram for mial::ControlCenter< Type, nDims >::



Public Types

- typedef ControlCenter Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- ~ControlCenter ()

  Default destructor responsible for correctly destroying both behavior and sensor lists.

- virtual bool update ()

  Update the cognitive layer (think!).

- virtual bool setAllPhysics (Physics< Type, nDims > *p)

  Set all the behaviors to have the provided physics layer.

- virtual Behavior< Type, nDims > * findBehavior (const std::string name)

  finds and returns a behavior with a particular name

- virtual int getNumBehaviors ()

  Accessor returning the number of known behaviors.

- virtual bool addBehavior (Behavior< Type, nDims > *b)

  Add a behavior to the list of known behaviors.

Protected Member Functions

- ControlCenter ()

  Default constructor.

- void setBehaviorToRunStream (std::stringstream *a)

  Set behavior to run stream.

- virtual bool decideNextBehavior ()=0

  Pure virtual member function, decides what behavior will be run after the current behavior is completed.

- virtual bool runNextBehavior ()

  Run the next behavior decided upon by decideNextBehavior(). Will also run the current running behaviors clean up method (if the behavior is finished).

- virtual bool updateCurrentBehavior ()

  Update the current behavior.

Protected Attributes

- std::vector< typename Behavior< Type, nDims >::Pointer > behaviorList

  The list of behaviors known to the cognitive center.

- std::vector< Sensor::Pointer > sensorList

  A list of sensors available to the control center.

- Behavior< Type, nDims >::Pointer behaviorToRun

  The behavior to run next.

- Behavior< Type, nDims >::behaviorIn * behaviorToRunStruct

  The arguments for the behavior to run next. This is the default behavior.

- std::stringstream * behaviorToRunStream

  The arguments for the behavior to run next in string format.

- int state

  The current state of the cognitive center.

- int numBehaviors

  The number of known behaviors.

5.17.1 Detailed Description

template<class Type, int nDims> class mial::ControlCenter< Type, nDims >

The brain of the organism responsible for making decisions and taking action based upon their outcome.

The cognitive layer of deformable organism is responsible for the decision making process. Essentially, this is the brain of the organism. It monitors the status, of the behaviors, deformations and sensors and makes decisions based upon their states and outputs.

This class exploits much of the complex versatility of the framework obtained through the use of ABCs, and IO streams and structures. Through a single list of sensors and behaviors, the cognitive center can perform a variety of actions on any defined geometrical or physical type. For example, the decision to

"translate" will trigger a translate behavior, which will in turn trigger the appropriate translate deformation as it pertains to the particular physical layer of the model. By that same notion, the decision to sense the eccentricity of the model will trigger the appropriate sensor.

Definition at line 36 of file ControlCenter.h.

5.17.2 Member Function Documentation

5.17.2.1 template<class Type, int nDims> bool mial::ControlCenter< Type, nDims >::addBehavior (Behavior< Type, nDims > * *b*) [virtual]

Add a behavior to the list of known behaviors.

Parameters:

b The behavior to be added.

Definition at line 37 of file ControlCenter.cxx.

References mial::ControlCenter< Type, nDims >::behaviorList, and mial::ControlCenter< Type, nDims >::numBehaviors.

5.17.2.2 template<class Type, int nDims> Behavior< Type, nDims > * mial::ControlCenter< Type, nDims >::findBehavior (const std::string *name*) [virtual]

finds and returns a behavior with a particular name

Parameters:

name The name of the behavior to locate

Definition at line 22 of file ControlCenter.cxx.

References mial::ControlCenter< Type, nDims >::behaviorList, and mial::ControlCenter< Type, nDims >::numBehaviors.

Referenced by mial::Ctrl_ScheduleDriven< Type, nDims >::decideNextBehavior(), and mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType >::decideNextBehavior().

5.17.2.3 template<class Type, int nDims> virtual bool mial::ControlCenter< Type, nDims >::update () [inline, virtual]

Update the cognitive layer (think!).

If there is no current behavior, decide on one and run. Otherwise, if the behavior is finished clean it up then decide and run a new one. Otherwise keep running the current.

Definition at line 52 of file ControlCenter.h.

5.17.3 Member Data Documentation

5.17.3.1 template<class Type, int nDims> std::vector<typename Behavior<Type,nDims>::Pointer> mial::ControlCenter< Type, nDims >::behaviorList [protected]

The list of behaviors known to the cognitive center.

The cognitive center maintains a list of named behaviors that it may run. Consequently, each behavior must be of the appropriate data type and dimensionality.

Definition at line 87 of file ControlCenter.h.

Referenced by mial::ControlCenter< Type, nDims >::addBehavior(), mial::ControlCenter< Type, nDims >::findBehavior(), and mial::ControlCenter< float, nDims >::setAllPhysics().

The documentation for this class was generated from the following files:

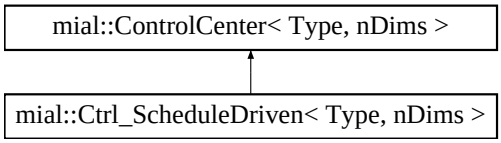
- C:/cmcintos/defOrgs/source/cognitive/abc/ControlCenter.h
- C:/cmcintos/defOrgs/source/cognitive/abc/ControlCenter.cxx

5.18 mial::Ctrl_ScheduleDriven< Type, nDims > Class Template Reference

ControlCenter that reads the schedule and peforms the listed behaviors sequentially.

`#include <Ctrl_ScheduleDriven.h>`

Inheritance diagram for mial::Ctrl_ScheduleDriven< Type, nDims >::



Public Types

- typedef Ctrl_ScheduleDriven Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Member Functions

- virtual bool decideNextBehavior ()

  Decide the next behavior by reading the next line of the schedule file.

- virtual bool setSchedule (std::string n)

  Set the schedule to be used.

Public Attributes

- std::stringstream behaviorToRunStream

  This class runs everything via streams.

Protected Member Functions

- Ctrl_ScheduleDriven ()

  Default constructor.

5.18.1 Detailed Description

template<class Type, int nDims> class mial::Ctrl_ScheduleDriven< Type, nDims >

ControlCenter that reads the schedule and peforms the listed behaviors sequentially.

A derived class of the ControlCenter ABC, it provides a schedule driven form of control wherein the deformable organisms proceeds sequentially through the actions listed in a file. One could also pipe the command prompt into the deformable organism instead and illicit direct sequential control over each behavior.

Definition at line 22 of file Ctrl_ScheduleDriven.h.

5.18.2 Member Function Documentation

5.18.2.1 template<class Type, int nDims> virtual bool mial::Ctrl_ScheduleDriven< Type, nDims >::setSchedule (std::string *n*) [inline, virtual]

Set the schedule to be used.

Parameters:

n The name of the schedule to be used.

Definition at line 40 of file Ctrl_ScheduleDriven.h.

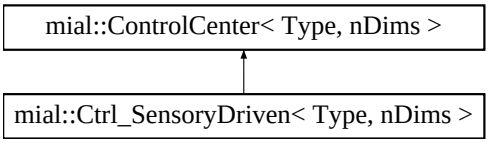
Referenced by DefOrgViewerAdapter::SetupOrganism(), and DefOrgViewer::SetupOrganism().

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/source/cognitive/Ctrl_ScheduleDriven.h
- C:/cmcintos/defOrgs/source/cognitive/Ctrl_ScheduleDriven.cxx

5.19 mial::Ctrl_SensoryDriven< Type, nDims > Class Template Reference

Inheritance diagram for mial::Ctrl_SensoryDriven< Type, nDims >::



5.19.1 Detailed Description

template<class Type, int nDims> class mial::Ctrl_SensoryDriven< Type, nDims >

Definition at line 15 of file Ctrl_SensoryDriven.h.

The documentation for this class was generated from the following files:

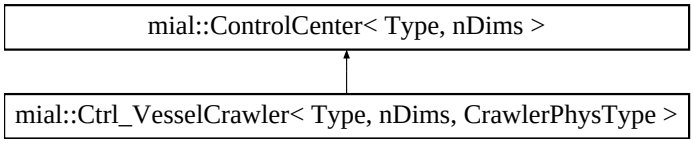
- C:/cmcintos/defOrgs/source/cognitive/Ctrl_SensoryDriven.h
- C:/cmcintos/defOrgs/source/cognitive/Ctrl_SensoryDriven.cxx

5.20 mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType > Class Template Reference

ControlCenter that specific for vessel crawlers [1].

`#include <Ctrl_VesselCrawler.h>`

Inheritance diagram for mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType >::



Public Member Functions

- virtual bool decideNextBehavior ()

  Pure virtual member function, decides what behavior will be run after the current behavior is completed.

- bool terminate ()
- bool bifurcation ()
- virtual bool setAllPhysics (Physics< Type, nDims > *p)

  Set all the behaviors to have the provided physics layer.

- virtual bool setGeom (Geom_VesselCrawler< Type, nDims > *g)

5.20.1 Detailed Description

template<class Type, int nDims, class CrawlerPhysType> class mial::Ctrl_VesselCrawler< Type, nDims, CrawlerPhysType >

ControlCenter that specific for vessel crawlers [1].

A derived class of the ControlCenter ABC, it provides a vascular segmentation specific control structure described in [1].

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Definition at line 21 of file Ctrl_VesselCrawler.h.

The documentation for this class was generated from the following files:

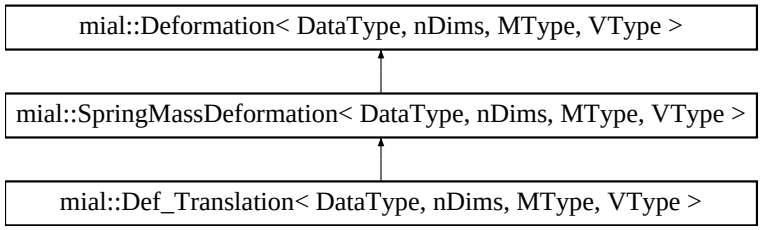
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Ctrl_VesselCrawler.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Ctrl_VesselCrawler.cxx

5.21 mial::Def_Translation< DataType, nDims, MType, VType > Class Template Reference

An example spring-mass deformation that translates the mesh.

`#include <Def_Translation.h>`

Inheritance diagram for mial::Def_Translation< DataType, nDims, MType, VType >::



Public Types

- typedef Def_Translation Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef SpringMassDeformation< DataType, nDims, MType, VType >::Error Error
- typedef SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet DefArgSet

Public Member Functions

- virtual bool run (typename Deformation< DataType, nDims >::deformationIn *i, typename Deformation< DataType, nDims >::DefArgSet *org, std::stringstream *s=NULL)

  An implementation of the run method that places translation forces on all nodes in the mesh.

Protected Member Functions

- Def_Translation ()

Classes

- struct deformationIn

  A structure defining the inputs for the deformation.

5.21.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::Def_Translation< DataType, nDims, MType, VType >

An example spring-mass deformation that translates the mesh.

Parameters:

DataType the type of container

nDims the dimensionality of the deformation

MType The matrix type used

VType The vector type used

Definition at line 24 of file Def_Translation.h.

The documentation for this class was generated from the following files:

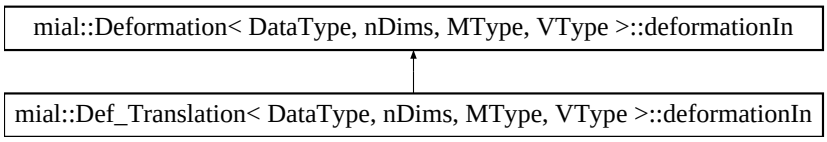
- C:/cmcintos/defOrgs/source/physical/Def_Translation.h
- C:/cmcintos/defOrgs/source/physical/Def_Translation.cxx

5.22 mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn Struct Reference

A structure defining the inputs for the deformation.

`#include <Def_Translation.h>`

Inheritance diagram for mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn::



Public Types

- typedef deformationIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer

Public Member Functions

- bool fillFromStream (std::stringstream &args)

  A method for converting stream-based arguments into the structures members (marshalling).

Public Attributes

- double amplitude [nDims]

  The amplitude of the translation in each dimension. Added directly as a force to all nodes of the spring-mass system.

Protected Member Functions

- deformationIn ()

5.22.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn

A structure defining the inputs for the deformation.

Since structures support public inheritance derived class must inherit from this class in their definitions of deformationIn.

Definition at line 44 of file Def_Translation.h.

5.22.2 Member Function Documentation

5.22.2.1 template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> bool mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn::fillFromStream (std::stringstream & *args*) [inline]

A method for converting stream-based arguments into the structures members (marshalling).

Stream is expected as "amplitudeX amplitudeY amplitudeZ(3D only)"

Definition at line 59 of file Def_Translation.h.

References mial::Def_Translation< DataType, nDims, MType, VType >::deformationIn::amplitude.

The documentation for this struct was generated from the following file:

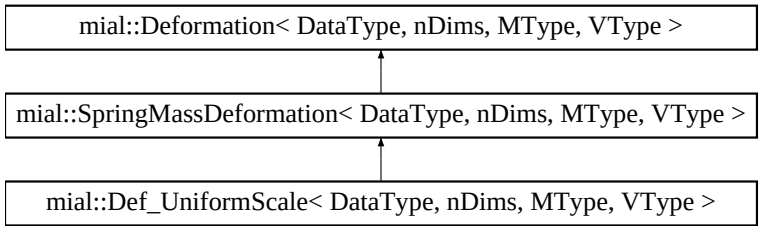
- C:/cmcintos/defOrgs/source/physical/Def_Translation.h

5.23 mial::Def_UniformScale< DataType, nDims, MType, VType > Class Template Reference

An example spring-mass deformation that scales the organism by increasing the rest lengths of all its springs.

`#include <Def_UniformScale.h>`

Inheritance diagram for mial::Def_UniformScale< DataType, nDims, MType, VType >::



Public Types

- typedef Def_UniformScale Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef SpringMassDeformation< DataType, nDims, MType, VType >::Error Error
- typedef SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet DefArgSet

Public Member Functions

- virtual bool run (typename Deformation< DataType, nDims >::deformationIn *i, typename Deformation< DataType, nDims >::DefArgSet *org, std::stringstream *s=NULL)

  Example run method that scales a deformable organism by increasing the rest lengths of all its springs.

Protected Member Functions

- Def_UniformScale ()

Classes

- struct deformationIn

  A structure defining the inputs for the deformation.

5.23.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::Def_UniformScale< DataType, nDims, MType, VType >

An example spring-mass deformation that scales the organism by increasing the rest lengths of all its springs.

Parameters:

DataType the type of container

nDims the dimensionality of the deformation

MType The matrix type used

VType The vector type used

Definition at line 24 of file Def_UniformScale.h.

The documentation for this class was generated from the following files:

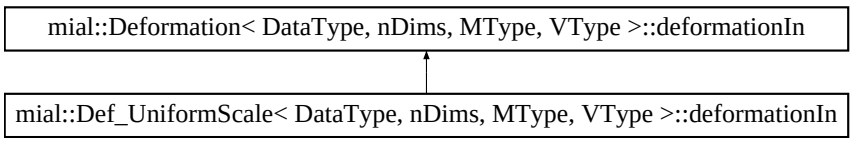
- C:/cmcintos/defOrgs/source/physical/Def_UniformScale.h
- C:/cmcintos/defOrgs/source/physical/Def_UniformScale.cxx

5.24 mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn Struct Reference

A structure defining the inputs for the deformation.

`#include <Def_UniformScale.h>`

Inheritance diagram for mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn::



Public Types

- typedef deformationIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer

Public Member Functions

- bool fillFromStream (std::stringstream &args)

  fills the structure using a stream as input. (Unmarshalls the data).

Public Attributes

- double scaleAmount

  The amount to scale each spring by.

Protected Member Functions

- deformationIn ()

5.24.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn

A structure defining the inputs for the deformation.

Since structures support public inheritance derived class must inherit from this class in their definitions of deformationIn.

Definition at line 45 of file Def_UniformScale.h.

5.24.2 Member Function Documentation

5.24.2.1 template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> bool mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn::fillFromStream (std::stringstream & *args*) [inline]

fills the structure using a stream as input. (Unmarshalls the data).

Stream is expected as "scaleAmount"

Parameters:

args the stream to be read.

Definition at line 61 of file Def_UniformScale.h.

References mial::Def_UniformScale< DataType, nDims, MType, VType >::deformationIn::scaleAmount.

The documentation for this struct was generated from the following file:

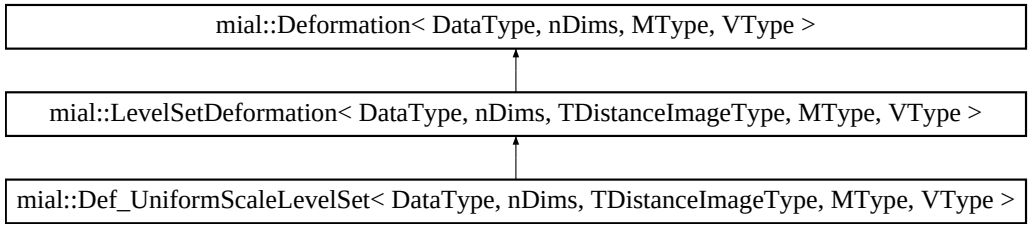
- C:/cmcintos/defOrgs/source/physical/Def_UniformScale.h

5.25 mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType > Class Template Reference

An example level set deformation that scales the deformable organism by adding a single user-provided value to each voxel of the level set.

`#include <Def_UniformScaleLevelSet.h>`

Inheritance diagram for mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::



Public Types

- typedef Def_UniformScaleLevelSet Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::ImageRegionIterator< typename LevelSetDeformation::DistanceImageType > IteratorType

Public Member Functions

- virtual bool run (typename Deformation::deformationIn *i, typename Deformation::DefArgSet *org, std::stringstream *s=NULL)

  Example run method that scales a deformable organism by adding a user-provided constant value to each voxel of the level set.

Protected Member Functions

- Def_UniformScaleLevelSet ()

Classes

- struct deformationIn

  A structure defining the inputs for the deformation.

5.25.1 Detailed Description

template<class DataType, int nDims, class TDistanceImageType, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >

An example level set deformation that scales the deformable organism by adding a single user-provided value to each voxel of the level set.

Parameters:

DataType the type of container

nDims the dimensionality of the deformation

MType The matrix type used

VType The vector type used

Definition at line 24 of file Def_UniformScaleLevelSet.h.

The documentation for this class was generated from the following files:

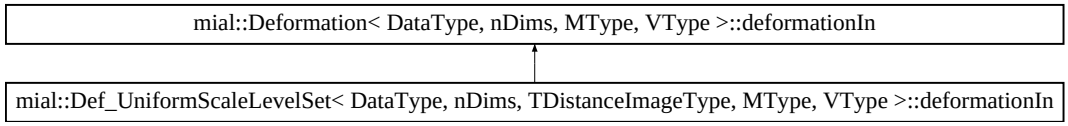
- C:/cmcintos/defOrgs/source/physical/Def_UniformScaleLevelSet.h
- C:/cmcintos/defOrgs/source/physical/Def_UniformScaleLevelSet.cxx

5.26 mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn Struct Reference

A structure defining the inputs for the deformation.

`#include <Def_UniformScaleLevelSet.h>`

Inheritance diagram for mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn::



Public Types

- typedef deformationIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer

Public Member Functions

- bool fillFromStream (std::stringstream &args)

  fills the structure using a stream as input. (Unmarshalls the data).

Public Attributes

- double scaleAmount

  The amount to scale by adding its value to each voxel of the level set.

Protected Member Functions

- deformationIn ()

5.26.1 Detailed Description

template<class DataType, int nDims, class TDistanceImageType, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn

A structure defining the inputs for the deformation.

Since structures support public inheritance derived class must inherit from this class in their definitions of deformationIn.

Definition at line 44 of file Def_UniformScaleLevelSet.h.

5.26.2 Member Function Documentation

5.26.2.1 template<class DataType, int nDims, class TDistanceImageType, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> bool mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn::fillFromStream (std::stringstream & *args*) [inline]

fills the structure using a stream as input. (Unmarshalls the data).

Stream is expected as "scaleAmount"

Parameters:

args the stream to be read.

Definition at line 59 of file Def_UniformScaleLevelSet.h.

References mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >::deformationIn::scaleAmount.

The documentation for this struct was generated from the following file:

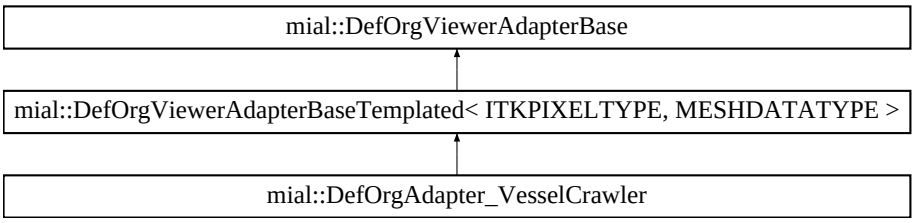
- C:/cmcintos/defOrgs/source/physical/Def_UniformScaleLevelSet.h

5.27 mial::DefOrgAdapter_VesselCrawler Class Reference

A deforg adapter for vessel crawlers [1].

`#include <DefOrgAdapter_VesselCrawler.h>`

Inheritance diagram for mial::DefOrgAdapter_VesselCrawler::



Public Types

- typedef itk::CovariantVector< DataType, N_DIMS > GradientPixelType
- typedef itk::Image< GradientPixelType, N_DIMS > GradientImageType
- typedef itk::ItkVesselCrawler< ImageType, ImageType, GradientImageType, DataType > VesselCrawlerType

Public Member Functions

- virtual bool IsAllInputFilesSet ()
- DefOrgAdapter_VesselCrawler ()
- virtual void SetupOrganism ()
- virtual void UpdateOrganism ()
- virtual void PopulateVtkImage ()
- virtual void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid *vtkGrid)
- virtual void PopulateItkScene ()
- virtual void HandleUserMouseInteraction (vtkTransform *userTransformation)
- virtual int MaxNumberOfOutputItkSpatialObjects ()
- virtual unsigned int MaxNumberOfOutputImages ()

5.27.1 Detailed Description

A deforg adapter for vessel crawlers [1].

This class describes a viewer adapter (to be loaded by the DefOrg Viewer) for a vessel crawler. Essentially it provides the functionaly for which the vessel crawlers can obtain input from the user, and display output.

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Definition at line 38 of file DefOrgAdapter_VesselCrawler.h.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 mial::DefOrgAdapter_VesselCrawler::DefOrgAdapter_VesselCrawler ()

Constructor should define the property the organism exposes at run-time using AddOrganismProperty helper method

Definition at line 15 of file DefOrgAdapter_VesselCrawler.cxx.

References mial::DefOrgViewerAdapterBase::AddOrganismProperty(), and mial::DefOrgViewerAdapterBase::m_OutputImages.

5.27.3 Member Function Documentation

5.27.3.1 void mial::DefOrgAdapter_VesselCrawler::HandleUserMouseInteraction (vtkTransform * *userTransformation*) [virtual]

This method is called when the user changes the DefOrg. It is the responsibility of the adapter to update the internal data structure of the DefOrg. The user transformation argument supplies the transformation that is specified visually by the user

Implements mial::DefOrgViewerAdapterBase.

Definition at line 53 of file DefOrgAdapter_VesselCrawler.cxx.

5.27.3.2 virtual bool mial::DefOrgAdapter_VesselCrawler::IsAllInputFilesSet () [inline, virtual]

Viewer will query this method to see if all three required files are set. If they are, it means the DefOrg can now be initialized

Reimplemented from mial::DefOrgViewerAdapterBase.

Definition at line 42 of file DefOrgAdapter_VesselCrawler.h.

References mial::DefOrgViewerAdapterBase::m_ImageFileName.

5.27.3.3 unsigned int mial::DefOrgAdapter_VesselCrawler::MaxNumberOfOutputImages () [virtual]

Should return n, the number of Image volumes this organism exports. The viewer would create n-1 secondary windows for images. The output of the image volumes are usually assigned to during PopulateVtkImage() and UpdateOrganism() The primary output will be displayed in the primary window

Implements mial::DefOrgViewerAdapterBase.

Definition at line 46 of file DefOrgAdapter_VesselCrawler.cxx.

5.27.3.4 int mial::DefOrgAdapter_VesselCrawler::MaxNumberOfOutputItkSpatialObjects () [virtual]

Should return n, the number of ItkSpatialObjects this organism exports. The viewer would create n-1 secondary windows for Spatial objects. The output of the spatialoutputs are usually assigned to during PopulateItkScene() and UpdateOrganism() The primary output will be displayed in the primary window

Implements mial::DefOrgViewerAdapterBase.

Definition at line 38 of file DefOrgAdapter_VesselCrawler.cxx.

5.27.3.5 void mial::DefOrgAdapter_VesselCrawler::PopulateItkScene () [virtual]

This method is called when the user presses Load Mesh

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 60 of file DefOrgAdapter_VesselCrawler.cxx.

References mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects.

5.27.3.6 void mial::DefOrgAdapter_VesselCrawler::PopulateVtkImage () [virtual]

This method is called when the user presses Load Image

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 77 of file DefOrgAdapter_VesselCrawler.cxx.

References mial::DefOrgViewerAdapterBase::m_ImageFileName, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::m_InputImageReader, mial::DefOrgViewerAdapterBase::m_OutputImages, and mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateVtkImageHelper().

5.27.3.7 void mial::DefOrgAdapter_VesselCrawler::PopulateVtkUnstructuredGrid (vtkUnstructuredGrid * *vtkGrid*) [virtual]

This method is currently not used. This method is originally for displaying meshes when SOViewer was not used

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 135 of file DefOrgAdapter_VesselCrawler.cxx.

5.27.3.8 void mial::DefOrgAdapter_VesselCrawler::SetupOrganism () [virtual]

This method is called when the user presses "Initialize/Setup organism from the GUI. The adapter should initialize the organism appropriately.

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 125 of file DefOrgAdapter_VesselCrawler.cxx.

References mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::m_InputImageReader.

5.27.3.9 void mial::DefOrgAdapter_VesselCrawler::UpdateOrganism () [virtual]

This method is called during each organism step. In this method, the adapter should indicate which outputs are modified, and call run on the organism.

Implements mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >.

Definition at line 110 of file DefOrgAdapter_VesselCrawler.cxx.

References mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/DefOrgAdapter_VesselCrawler.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/DefOrgAdapter_VesselCrawler.cxx

5.28 mial::DefOrgLayerStruct Struct Reference

Public Attributes

- const char * layerName
- std::vector< const char * > options
- const char * chosenOption

5.28.1 Detailed Description

Definition at line 47 of file DefOrgViewerAdapterBase.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase.h

5.29 mial::DefOrgPropertyStruct Struct Reference

`#include <DefOrgViewerAdapterBase.h>`

Public Member Functions

- DefOrgPropertyStruct (double _lowerBound, double _upperBound, double _defaultValue, double _currentValue, double _resolution, std::string _helpString)
- DefOrgPropertyStruct ()

Public Attributes

- double lowerBound
- double upperBound
- double defaultValue
- double currentValue
- double resolution
- std::string helpString

5.29.1 Detailed Description

Structure for storing DefOrg numeric property. Define the range, default value and resolution for displaying the numeric property as a slicer in the GUI

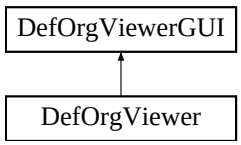
Definition at line 58 of file DefOrgViewerAdapterBase.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase.h

5.30 DefOrgViewer Class Reference

Inheritance diagram for DefOrgViewer::



Public Types

- typedef float DataType
- typedef unsigned char PixelType
- typedef itk::Image< PixelType, N_DIMS > ImageType
- typedef Geom_MeshSpatialObject< DataType, N_DIMS > GeometricType
- typedef itk::CovariantVector< DataType, N_DIMS > GradientPixelType
- typedef itk::Image< GradientPixelType, N_DIMS > GradientImageType
- typedef Phys_LevelSet< DataType, ImageType, N_DIMS > LevelSetPhysicsType
- typedef Phys_Euler< DataType, GradientImageType, N_DIMS > EulerPhysicsType
- typedef Sense_Gradient< DataType, ImageType, GradientImageType, N_DIMS > GradientSensorType
- typedef Ctrl_ScheduleDriven< DataType, N_DIMS > CognitiveType
- typedef Beh_TranslateAll< DataType, N_DIMS > Beh_TranslateAllType
- typedef Def_Translation< DataType, N_DIMS > Def_TranslateAllType
- typedef itk::ItkOrganism< ImageType, ImageType, GradientImageType, DataType, N_DIMS > OrganismType
- typedef itk::DefaultDynamicMeshTraits< DataType, N_DIMS, N_DIMS > MeshTrait
- typedef itk::Mesh< DataType, N_DIMS, MeshTrait > MeshType
- typedef MeshType::Pointer MeshTypePointer
- typedef vtkRenderer RendererType
- typedef RendererType * RendererPointer
- typedef itk::VTKImageExport< ImageType > itkImageExportType
- typedef vtkImageImport vtkImageImportType
- typedef itk::ImageFileReader< ImageType > ImageReaderType
- typedef itk::CellInterfaceVisitorImplementation< DataType, MeshType::CellTraits, itk::TriangleCell< itk::CellInterface< MeshType::PixelType, MeshType::CellTraits > >, VistVTKCellsClass > TriangleVisitor
- typedef itk::CellInterfaceVisitorImplementation< DataType, MeshType::CellTraits, itk::QuadrilateralCell< itk::CellInterface< MeshType::PixelType, MeshType::CellTraits > >, VistVTKCellsClass > QuadrilateralVisitor

Public Member Functions

- DefOrgViewer ()
- DefOrgViewer (MeshTypePointer itkMesh)
- void Quit ()
- void Show ()
- void Update ()
- void LoadImage ()
- void LoadMeta ()
- void TogglePlayPause ()
- void ToggleVolumeRendering ()
- vtkUnstructuredGrid * MeshToUnstructuredGrid (MeshType::Pointer itkMesh)
- void ConnectPipelines (itkImageExportType::Pointer exporter, vtkImageImportType *importer)
- void SetupOrganism ()
- void UpdateOrganism ()

Public Attributes

- ImageReaderType::Pointer imageReader

5.30.1 Detailed Description

Definition at line 88 of file DefOrgViewer.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/DefOrgViewer.h
- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/DefOrgViewer.cxx

5.31 DefOrgViewerAdapter Class Reference

Public Types

- typedef float DataType
- typedef Geom_MeshSpatialObject< DataType, N_DIMS > GeometricType
- typedef itk::CovariantVector< DataType, N_DIMS > GradientPixelType
- typedef itk::Image< GradientPixelType, N_DIMS > GradientImageType
- typedef Phys_LevelSet< DataType, ImageType, N_DIMS > LevelSetPhysicsType
- typedef Phys_Euler< DataType, GradientImageType, N_DIMS > EulerPhysicsType
- typedef Sense_Gradient< DataType, ImageType, GradientImageType, N_DIMS > GradientSensorType
- typedef Ctrl_ScheduleDriven< DataType, N_DIMS > CognitiveType
- typedef Beh_TranslateAll< DataType, N_DIMS > Beh_TranslateAllType
- typedef Def_Translation< DataType, N_DIMS > Def_TranslateAllType
- typedef itk::ItkOrganism< ImageType, ImageType, GradientImageType, DataType, N_DIMS > OrganismType
- typedef itk::DefaultDynamicMeshTraits< DataType, N_DIMS, N_DIMS > MeshTrait
- typedef itk::Mesh< DataType, N_DIMS, MeshTrait > MeshType
- typedef MeshType::Pointer MeshTypePointer
- typedef itk::CellInterfaceVisitorImplementation< DataType, MeshType::CellTraits, itk::TriangleCell< itk::CellInterface< MeshType::PixelType, MeshType::CellTraits > >, VistVTKCellsClass > TriangleVisitor
- typedef itk::CellInterfaceVisitorImplementation< DataType, MeshType::CellTraits, itk::QuadrilateralCell< itk::CellInterface< MeshType::PixelType, MeshType::CellTraits > >, VistVTKCellsClass > QuadrilateralVisitor

Public Member Functions

- DefOrgViewerAdapter ()
- void SetupOrganism ()
- void UpdateOrganism (itkScenePointer itkScene)
- void PopulateVtkImage (vtkImageImport *vtkImporter)
- void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid *vtkGrid)
- void PopulateItkScene (itkScenePointer itkScene)
- bool IsAllInputFilesSet ()

5.31.1 Detailed Description

Definition at line 59 of file DefOrgViewerAdapter.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapter.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapter.cxx

5.32 mial::DefOrgViewerAdapterBase Class Reference

`#include <DefOrgViewerAdapterBase.h>`

Inheritance diagram for mial::DefOrgViewerAdapterBase::

Public Member Functions

- DefOrgViewerAdapterBase (int maxNumImages)
- virtual bool IsAllInputFilesSet ()
- virtual void SetupOrganism ()=0
- virtual void UpdateOrganism ()=0
- virtual void PopulateVtkImage ()=0
- virtual void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid ∗vtkGrid)=0
- virtual void PopulateItkScene ()=0
- virtual void HandleUserMouseInteraction (vtkTransform ∗userTransformation)=0
- virtual int MaxNumberOfOutputItkSpatialObjects ()=0
- virtual unsigned int MaxNumberOfOutputImages ()=0
- virtual void AddOutputItkSpatialObjects (itkScenePointer itkScene)
- virtual void AddOutputImages (vtkImageImport ∗vtkImporter, int Offset)

Public Attributes

- std::map< std::string, DefOrgPropertyStruct > m_PropertyBag
- std::vector< DefOrgLayerStruct > m_LayerBag
- std::vector< SpatialObjectDescriptorStruct > m_OutputItkSpatialObjects
- std::vector< OutputImageDescriptorStruct > m_OutputImages
- std::ostringstream theDefOrgTextOutput
- std::stringstream theDefOrgTextInput

Protected Member Functions

- virtual void AddOrganismProperty (const char ∗propertyName, double lowerBound, double upperBound, double defaultValue, double resolution=1.0, const char ∗helpString="")
- virtual double GetOrganismProperty (const char ∗propertyName)
- virtual void AddLayer (const char ∗layerName, const char ∗∗options, int numChoices)
- virtual const char ∗ GetLayer (const char ∗layerName)

Protected Attributes

- bool m_Initialized
- std::string m_ImageFileName
- std::string m_ScheduleFileName
- std::string m_MeshFileName

5.32.1 Detailed Description

Abstract base class for DefOrgViewerAdapter. Do not inherit directly from this. Instead, inherit from DefOrgViewerAdapterBaseTemplate. This class is for internal use in DefOrgViewer.

Definition at line 137 of file DefOrgViewerAdapterBase.h.

5.32.2 Member Function Documentation

5.32.2.1 void mial::DefOrgViewerAdapterBase::AddLayer (const char * *layerName*, const char ** *options*, int *numChoices*) [protected, virtual]

Helper method to allow a combox to be displayed, allowing the user to choose which instance of a layer to attach

Definition at line 4 of file DefOrgViewerAdapterBase.cxx.

References mial::DefOrgLayerStruct::layerName, m_LayerBag, and mial::DefOrgLayerStruct::options.

5.32.2.2 virtual void mial::DefOrgViewerAdapterBase::AddOrganismProperty (const char * *propertyName*, double *lowerBound*, double *upperBound*, double *defaultValue*, double *resolution* = 1.0, const char * *helpString* = "") [inline, protected, virtual]

Helper method to allow numeric properties to be added in derived adapter's constructor

Parameters:

propertyName Name of the property

lowerBound Lower bound of the numeric property

upperBound Upper bound of the numeric property

defaultValue Default Value of the numeric property

resolution By what value will the current value be incremented when the slicer is adjusted

helpString Optional help string describing this property

Definition at line 223 of file DefOrgViewerAdapterBase.h.

References m_PropertyBag.

Referenced by mial::BasicDefOrg_DefOrgViewerAdapter::BasicDefOrg_DefOrgViewerAdapter(), and mial::DefOrgAdapter_VesselCrawler::DefOrgAdapter_VesselCrawler().

5.32.2.3 virtual void mial::DefOrgViewerAdapterBase::AddOutputImages (vtkImageImport * *vtkImporter*, int *Offset*) [inline, virtual]

For viewer. Derived class should not call this method

Definition at line 203 of file DefOrgViewerAdapterBase.h.

References m_OutputImages.

Referenced by vtkDefOrgViewerWithKW::InitializeState().

5.32.2.4 virtual void mial::DefOrgViewerAdapterBase::AddOutputItkSpatialObjects (itkScenePointer *itkScene*) [inline, virtual]

For viewer. Derived class should not call this method

Definition at line 198 of file DefOrgViewerAdapterBase.h.

References m_OutputItkSpatialObjects.

Referenced by vtkDefOrgViewerWithKW::InitializeState().

5.32.2.5 const char * mial::DefOrgViewerAdapterBase::GetLayer (const char * *layerName*) [protected, virtual]

Helper method to get the name of the layer assigned

Definition at line 13 of file DefOrgViewerAdapterBase.cxx.

References m_LayerBag.

5.32.2.6 virtual double mial::DefOrgViewerAdapterBase::GetOrganismProperty (const char * *propertyName*) [inline, protected, virtual]

Return the value of the property as obtained through the GUI

Parameters:

propertyName Name of the property

Definition at line 229 of file DefOrgViewerAdapterBase.h.

References m_PropertyBag.

Referenced by mial::BasicDefOrg_DefOrgViewerAdapter::SetupOrganism().

5.32.2.7 virtual void mial::DefOrgViewerAdapterBase::HandleUserMouseInteraction (vtkTransform * *userTransformation*) [pure virtual]

Derived class should override this method to make use of the supplied userTransformation matrix

Implemented in mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

Referenced by vtkMyCallback::Execute().

5.32.2.8 virtual bool mial::DefOrgViewerAdapterBase::IsAllInputFilesSet () [inline, virtual]

Viewer will query this method to see if all three required files are set. If they are, it means the DefOrg can now be initialized

Reimplemented in mial::DefOrgAdapter_VesselCrawler.

Definition at line 151 of file DefOrgViewerAdapterBase.h.

References m_ImageFileName, m_MeshFileName, and m_ScheduleFileName.

5.32.2.9 virtual unsigned int mial::DefOrgViewerAdapterBase::MaxNumberOfOutputImages () [pure virtual]

Derived class should override this method to let DefOrgViewer know how many output Image Volumes this DefOrg wants

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

Referenced by vtkDefOrgViewerWithKW::CreateVolumeRenderingFrame(), vtkDefOrgViewerWithKW::InitializeState(), vtkDefOrgViewerWithKW::SetVolumeRenderingControlBoxCallback(), and vtkDefOrgViewerWithKW::UpdateImageViewers().

5.32.2.10 virtual int mial::DefOrgViewerAdapterBase::MaxNumberOfOutputItkSpatialObjects () [pure virtual]

Derived class should override this method to let DefOrgViewer know how many output Spatial Objects this DefOrg wants

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

Referenced by vtkDefOrgViewerWithKW::AddActorsFromSOViewers(), vtkDefOrgViewerWithKW::InitializeState(), vtkDefOrgViewerWithKW::LoadMeshDialogCallback(), vtkDefOrgViewerWithKW::RemoveActorsFromSOViewers(), vtkDefOrgViewerWithKW::RenderSOViewers(), and vtkDefOrgViewerWithKW::SetScenesToSOViewers().

5.32.2.11 virtual void mial::DefOrgViewerAdapterBase::PopulateItkScene () [pure virtual]

Derived class should provide a itkScene that it wants to display when the mesh is first loaded (optional)

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >, mial::DefOrgAdapter_VesselCrawler, and mial::DefOrgViewerAdapterBaseTemplated< unsigned char, float >.

Referenced by vtkDefOrgViewerWithKW::LoadMeshDialogCallback().

5.32.2.12 virtual void mial::DefOrgViewerAdapterBase::PopulateVtkImage () [pure virtual]

Derived class should provide a vtkImage that it wants to display when the Image Volumes is first loaded (optional)

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >, mial::DefOrgAdapter_VesselCrawler, and mial::DefOrgViewerAdapterBaseTemplated< unsigned char, float >.

Referenced by vtkDefOrgViewerWithKW::LoadImageDialogCallback().

5.32.2.13 virtual void mial::DefOrgViewerAdapterBase::PopulateVtkUnstructuredGrid (vtkUnstructuredGrid * *vtkGrid*) [pure virtual]

Currently not used. Replaced with ItkScene and SOViewer

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >, mial::DefOrgAdapter_VesselCrawler, and mial::DefOrgViewerAdapterBaseTemplated< unsigned char, float >.

5.32.2.14 virtual void mial::DefOrgViewerAdapterBase::SetupOrganism () [pure virtual]

Viewer will call this method as instructed from GUI. Derived class should initialize the organism itself using the DefOrg properties (via GetOrganismProperty) supplied through the GUI

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >, mial::DefOrgAdapter_VesselCrawler, and mial::DefOrgViewerAdapterBaseTemplated< unsigned char, float >.

Referenced by vtkDefOrgViewerWithKW::InitOrganismButtonCallback().

5.32.2.15 virtual void mial::DefOrgViewerAdapterBase::UpdateOrganism () [pure virtual]

During each step of simulation, viewer will call this method. It is the derived class responsibility to update the OutputItkScenes and/or OutputImageVolumes using implemented helper methods

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >, mial::DefOrgAdapter_VesselCrawler, and mial::DefOrgViewerAdapterBaseTemplated< unsigned char, float >.

Referenced by vtkDefOrgViewerWithKW::StepOrganismButtonCallback().

5.32.3 Member Data Documentation

5.32.3.1 std::vector< OutputImageDescriptorStruct > mial::DefOrgViewerAdapterBase::m_OutputImages

Number of vtkImageImport* will depend on MaxNumberOfOutputImages(). The derived class would then populate the imageVolumes as it finds appropriate

Definition at line 195 of file DefOrgViewerAdapterBase.h.

Referenced by AddOutputImages(), mial::DefOrgAdapter_VesselCrawler::DefOrgAdapter_VesselCrawler(), DefOrgViewerAdapterBase(), mial::vtkDefOrgViewerWithKWState::GetCurrentImageViewer(), mial::vtkDefOrgViewerWithKWState::GetCurrentImageViewerRenderWidget(), vtkDefOrgViewerWithKW::LoadImageDialogCallback(), mial::DefOrgAdapter_VesselCrawler::PopulateVtkImage(), mial::Blank_DefOrgViewerAdapter::PopulateVtkImage(), mial::BasicDefOrg_DefOrgViewerAdapter::PopulateVtkImage(), vtkDefOrgViewerWithKW::SetupImageViewerPipeline(), and vtkDefOrgViewerWithKW::UpdateImageViewers().

5.32.3.2 std::vector< SpatialObjectDescriptorStruct > mial::DefOrgViewerAdapterBase::m_OutputItkSpatialObjects

Number of itkScenePointer will depend on MaxNumberOfOutputItkSpatialObjects(). The derived class would then populate the Scenes as it finds appropriate

Definition at line 191 of file DefOrgViewerAdapterBase.h.

Referenced by vtkDefOrgViewerWithKW::AddActorsFromSOViewers(), AddOutputItkSpatialObjects(), mial::vtkDefOrgViewerWithKWState::GetCurrentSOViewer(), mial::vtkDefOrgViewerWithKWState::GetCurrentSOViewerRenderWidget(), mial::DefOrgAdapter_VesselCrawler::PopulateItkScene(), mial::Blank_DefOrgViewerAdapter::PopulateItkScene(), mial::BasicDefOrg_DefOrgViewerAdapter::PopulateItkScene(), vtkDefOrgViewerWithKW::RemoveActorsFromSOViewers(), vtkDefOrgViewerWithKW::RenderSOViewers(), vtkDefOrgViewerWithKW::SetScenesToSOViewers(), mial::DefOrgAdapter_VesselCrawler::UpdateOrganism(), mial::Blank_DefOrgViewerAdapter::UpdateOrganism(), and mial::BasicDefOrg_DefOrgViewerAdapter::UpdateOrganism().

5.32.3.3 std::map<std::string,DefOrgPropertyStruct> mial::DefOrgViewerAdapterBase::m_PropertyBag

Property Bag storing initialization properties (changeable by user from the GUI)

Definition at line 185 of file DefOrgViewerAdapterBase.h.

Referenced by AddOrganismProperty(), vtkDefOrgViewerWithKW::CreateOrganismDataFrame(), GetOrganismProperty(), and vtkDefOrgViewerWithKW::InitOrganismButtonCallback().

5.32.3.4 std::stringstream mial::DefOrgViewerAdapterBase::theDefOrgTextInput

Derived class can read from this stream for text input collected from the viewer

Definition at line 211 of file DefOrgViewerAdapterBase.h.

Referenced by vtkDefOrgViewerWithKW::SendOrganismMessageCallback().

5.32.3.5 std::ostringstream mial::DefOrgViewerAdapterBase::theDefOrgTextOutput

For viewer. Derived class should not access this variable. To produce text output to be displayed, use std::cout

Definition at line 209 of file DefOrgViewerAdapterBase.h.

Referenced by DefOrgViewerAdapterBase(), and vtkDefOrgViewerWithKW::UpdateOrganismTextOutputCallback().

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase.cxx

5.33 mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE > Class Template Reference

`#include <DefOrgViewerAdapterBaseTemplated.h>`

Inheritance diagram for mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::

Public Types

- typedef ITKPIXELTYPE PixelType
- typedef MESHDATATYPE DataType
- typedef itk::Image< PixelType, N_DIMS > ImageType
- typedef ImageType::Pointer ImageTypePointer
- typedef itk::ImageFileReader< ImageType > ImageFileReader
- typedef ImageFileReader::Pointer ImageFileReaderPointer
- typedef itk::VTKImageExport< ImageType > itkImageExportType
- typedef itkImageExportType::Pointer itkImageExportTypePointer
- typedef itk::SceneSpatialObject< N_DIMS >::Pointer itkScenePointer
- typedef itk::DefaultDynamicMeshTraits< DataType, N_DIMS, N_DIMS > MeshTrait
- typedef itk::Mesh< DataType, N_DIMS, MeshTrait > MeshType
- typedef MeshType::CellTraits MeshTypeCellTraits
- typedef MeshType::Pointer MeshTypePointer
- typedef itk::TriangleCell< typename itk::CellInterface< MESHDATATYPE, MeshTypeCellTraits > > TriangleCell
- typedef itk::CellInterfaceVisitorImplementation< DataType, MeshTypeCellTraits, TriangleCell, VistVTKCellsClass< typename DefOrgViewerAdapterBaseTemplated::MeshType > > TriangleVisitor
- typedef itk::QuadrilateralCell< typename itk::CellInterface< MESHDATATYPE, MeshTypeCellTraits > > QuadrilateralCell
- typedef itk::CellInterfaceVisitorImplementation< DataType, MeshTypeCellTraits, QuadrilateralCell, VistVTKCellsClass< typename DefOrgViewerAdapterBaseTemplated::MeshType > > QuadrilateralVisitor

Public Member Functions

- DefOrgViewerAdapterBaseTemplated (int maxNumImages)
- virtual void SetupOrganism ()=0
- virtual void UpdateOrganism ()=0
- virtual void PopulateVtkImage ()=0
- virtual void PopulateVtkImageHelper (ImageTypePointer itkImage, vtkImageImport *vtkImporter)
- virtual void PopulateVtkUnstructuredGrid (vtkUnstructuredGrid *vtkGrid)=0
- virtual void PopulateItkScene ()=0

Protected Member Functions

- virtual void ConnectPipelines (itkImageExportTypePointer exporter, vtkImageImport *importer)
- virtual void MeshToUnstructuredGrid (MeshTypePointer mesh, vtkUnstructuredGrid *vgrid)

Protected Attributes

- ImageFileReaderPointer m_InputImageReader

5.33.1 Detailed Description

template<class ITKPIXELTYPE, class MESHDATATYPE> class mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >

Provide convience templated helper methods for derived classes

Definition at line 32 of file DefOrgViewerAdapterBaseTemplated.h.

5.33.2 Member Function Documentation

5.33.2.1 template<class ITKPIXELTYPE, class MESHDATATYPE> void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::ConnectPipelines (itkImageExportTypePointer *exporter*, vtkImageImport * *importer*) [protected, virtual]

Connect vtk and itk pipelines for image volumes

Definition at line 18 of file DefOrgViewerAdapterBaseTemplated.cxx.

Referenced by mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateVtkImageHelper().

5.33.2.2 template<class ITKPIXELTYPE, class MESHDATATYPE> void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::MeshToUnstructuredGrid (MeshTypePointer *mesh*, vtkUnstructuredGrid * *vgrid*) [protected, virtual]

Convert itk mesh into vtk unstructured grid. Currently not used because SOViewer is used instead

Definition at line 36 of file DefOrgViewerAdapterBaseTemplated.cxx.

Referenced by mial::BasicDefOrg_DefOrgViewerAdapter::PopulateVtkUnstructuredGrid().

5.33.2.3 template<class ITKPIXELTYPE, class MESHDATATYPE> virtual void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateItkScene () [pure virtual]

Derived class should provide a itkScene that it wants to display when the mesh is first loaded (optional)

Implements mial::DefOrgViewerAdapterBase.

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

5.33.2.4 template<class ITKPIXELTYPE, class MESHDATATYPE> virtual void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateVtkImage () [pure virtual]

Derived class should provide a vtkImage that it wants to display when the Image Volumes is first loaded (optional)

Implements mial::DefOrgViewerAdapterBase.

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

5.33.2.5 template<class ITKPIXELTYPE, class MESHDATATYPE> virtual void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::PopulateVtkUnstructuredGrid (vtkUnstructuredGrid * *vtkGrid*) [pure virtual]

Currently not used. Replaced with ItkScene and SOViewer

Implements mial::DefOrgViewerAdapterBase.

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

5.33.2.6 template<class ITKPIXELTYPE, class MESHDATATYPE> virtual void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::SetupOrganism () [pure virtual]

Viewer will call this method as instructed from GUI. Derived class should initialize the organism itself using the DefOrg properties (via GetOrganismProperty) supplied through the GUI

Implements mial::DefOrgViewerAdapterBase.

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

5.33.2.7 template<class ITKPIXELTYPE, class MESHDATATYPE> virtual void mial::DefOrgViewerAdapterBaseTemplated< ITKPIXELTYPE, MESHDATATYPE >::UpdateOrganism () [pure virtual]

During each step of simulation, viewer will call this method. It is the derived class responsibility to update the OutputItkScenes and/or OutputImageVolumes using implemented helper methods

Implements mial::DefOrgViewerAdapterBase.

Implemented in BasicDefOrg_DefOrgViewerAdapter, mial::BasicDefOrg_DefOrgViewerAdapter, mial::Blank_DefOrgViewerAdapter, and mial::DefOrgAdapter_VesselCrawler.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase-Templated.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase-Templated.cxx

5.34 mial::DefOrgViewerAdapterDynamicLoader Class Reference

`#include <DefOrgViewerAdapterDynamicLoader.h>`

Public Member Functions

- DefOrgViewerAdapterBase * LoadLibrary (char *path)
- DefOrgViewerAdapterDynamicLoader ()
- ~DefOrgViewerAdapterDynamicLoader ()

5.34.1 Detailed Description

Helper class for loading DefOrg Adapters dynamically

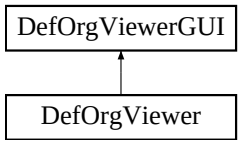
Definition at line 8 of file DefOrgViewerAdapterDynamicLoader.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterDynamic-Loader.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterDynamic-Loader.cxx

5.35 DefOrgViewerGUI Class Reference

Inheritance diagram for DefOrgViewerGUI::



Public Member Functions

- DefOrgViewerGUI ()
- virtual ~DefOrgViewerGUI ()
- virtual void Quit ()
- virtual void LoadImage ()
- virtual void Show ()
- virtual void Hide ()
- virtual void TogglePlayPause ()
- virtual void LoadMeta ()
- virtual void ToggleVolumeRendering ()

Public Attributes

- Fl_Double_Window * mainWindow
- vtkFlRenderWindowInteractor * display
- Fl_Button * playPauseButton

5.35.1 Detailed Description

Definition at line 13 of file DefOrgViewerGUI.h.

The documentation for this class was generated from the following files:

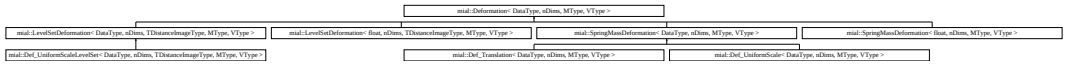
- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/DefOrgViewerGUI.h
- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/DefOrgViewerGUI.cxx

5.36 mial::Deformation< DataType, nDims, MType, VType > Class Template Reference

An abstract base class for deformations.

`#include <Deformation.h>`

Inheritance diagram for mial::Deformation< DataType, nDims, MType, VType >::



Public Types

- typedef Deformation Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef MType MatrixType

  Public typedef for the internal matrix type.

- typedef VType VectorType

  Public typedef for the internal vector type.

Public Member Functions

- virtual bool run (deformationIn *i, DefArgSet *org, std::stringstream *s=NULL)=0

  Run method for running this deformation.

- virtual int getStatus ()

  Get the status of the deformation.

- virtual std::string getName ()

  Get the name of the deformation.

Protected Member Functions

- Deformation ()

Protected Attributes

- std::string name

  The name of the deformation used to identify itself in a list of deformations.

Classes

- struct DefArgSet

  These define the standard "hidden" argument set for a deformation that allow to manipulate the model.

- struct deformationIn

  A structure defining the inputs of a Deformation.

- struct Error

  The error structure.

5.36.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_-vector<DataType>> class mial::Deformation< DataType, nDims, MType, VType >

An abstract base class for deformations.

Each deformation is in charge of unmarshaling its own arguments.

Parameters:

DataType the type of container

nDims the dimensionality of the deformation

MType The matrix type used

VType The vector type used

Definition at line 31 of file Deformation.h.

5.36.2 Member Function Documentation

5.36.2.1 template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> virtual bool mial::Deformation< DataType, nDims, MType, VType >::run (deformationIn * *i*, DefArgSet * *org*, std::stringstream * *s* = NULL) [pure virtual]

Run method for running this deformation.

Note that one and only one of i,s can be non-NULL.

Parameters:

i the deformationIn structure.

s the deformation arguments as a stream. To support, one would typically define a conversion method in the deformationIN class of the derived deformation class.

Implemented in mial::Def_UniformScaleLevelSet< DataType, nDims, TDistanceImageType, MType, VType >.

The documentation for this class was generated from the following files:

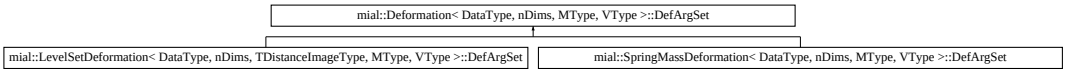
- C:/cmcintos/defOrgs/source/physical/abc/Deformation.h
- C:/cmcintos/defOrgs/source/physical/abc/Deformation.cxx

5.37 mial::Deformation< DataType, nDims, MType, VType >::DefArgSet Struct Reference

These define the standard "hidden" argument set for a deformation that allow to manipulate the model.

`#include <Deformation.h>`

Inheritance diagram for mial::Deformation< DataType, nDims, MType, VType >::DefArgSet::



5.37.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Deformation< DataType, nDims, MType, VType >::DefArgSet

These define the standard "hidden" argument set for a deformation that allow to manipulate the model.

Contains the Physics layer specific arguments that a deformation uses to initiate deformations.

Definition at line 52 of file Deformation.h.

The documentation for this struct was generated from the following file:

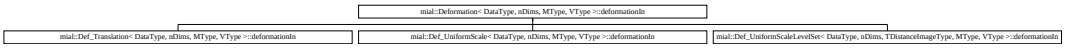
- C:/cmcintos/defOrgs/source/physical/abc/Deformation.h

5.38 mial::Deformation< DataType, nDims, MType, VType >::deformationIn Struct Reference

A structure defining the inputs of a Deformation.

`#include <Deformation.h>`

Inheritance diagram for mial::Deformation< DataType, nDims, MType, VType >::deformationIn::



Public Types

- typedef deformationIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer

Protected Member Functions

- deformationIn ()

5.38.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Deformation< DataType, nDims, MType, VType >::deformationIn

A structure defining the inputs of a Deformation.

Since structures support public inheritance derived class must inherit from this class in their definitions of deformationIn.

Definition at line 67 of file Deformation.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/physical/abc/Deformation.h

5.39 mial::Deformation< DataType, nDims, MType, VType >::Error Struct Reference

The error structure.

`#include <Deformation.h>`

Public Attributes

- std::string name
- std::string msg

5.39.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Deformation< DataType, nDims, MType, VType >::Error

The error structure.

Definition at line 57 of file Deformation.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/physical/abc/Deformation.h

5.40 fftw_iodim_do_not_use_me Struct Reference

Public Attributes

- int n
- int is
- int os

5.40.1 Detailed Description

Definition at line 58 of file fftw3.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/include/fftw3.h

5.41 mial::GenerateDefOrgHelpers Class Reference

Public Member Functions

- GenerateDefOrgHelpers (const char *className)
- void SetupMappings ()
- std::string ReplaceTokens (std::string fileContent)
- std::string LookupFilename (std::string originalFileName)

Public Attributes

- std::string ClassName

5.41.1 Detailed Description

Definition at line 8 of file GenerateDefOrgHelpers.h.

The documentation for this class was generated from the following files:

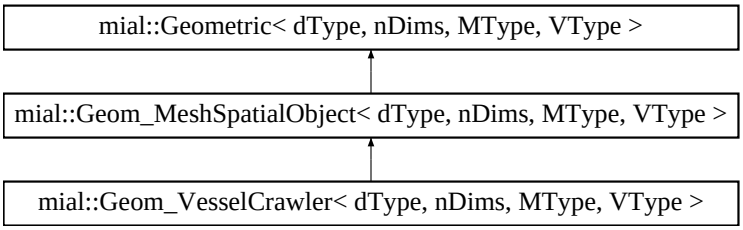
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/GenerateDefOrgHelpers.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/GenerateDefOrgHelpers.cxx

5.42 mial::Geom_MeshSpatialObject< dType, nDims, MType, VType > Class Template Reference

Derived geometric class based on a spatial object.

`#include <Geom_MeshSpatialObject.h>`

Inheritance diagram for mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::



Public Types

- typedef Geom_MeshSpatialObject Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef MType MatrixType

  The internal matrix type.

- typedef VType VectorType

  The internal vector type.

- typedef itk::DefaultDynamicMeshTraits< dType, nDims, nDims > MeshTrait
- typedef itk::Mesh< dType, nDims, MeshTrait > MeshType
- typedef MeshType::PointType PointType
- typedef itk::MeshSpatialObject< MeshType > MeshSpatialObjectType
- typedef MeshSpatialObjectType::Pointer MeshSpatialObjectPointerType
- typedef MeshType::CellType CellType
- typedef CellType::CoordRepType CoordRepType
- typedef itk::TriangleCell< CellType > TriangleType
- typedef CellType::CellAutoPointer CellAutoPointer
- typedef MeshType::CellsContainer::ConstIterator CellIterator
- typedef itk::GroupSpatialObject< nDims > GroupType
- typedef itk::SpatialObjectReader< nDims, dType, MeshTrait > ReaderType
- typedef itk::SpatialObjectWriter< nDims, dType, MeshTrait > WriterType
- typedef Geometric< dType, nDims >::BinaryImageType BinaryImageType

  The type of binary image.

- typedef BinaryImageType::Pointer BinaryImageTypePointer

Public Member Functions

- virtual MatrixType getMatrixNodePositions ()
  Pure virtual member function. Returns the matrix node positions.
- virtual void writeNodesToFile (std::string fileName)
  Pure virtual member function. Writes the nodes to a file.
- virtual void readNodesFromFile (std::string fileName)
  Pure virtual member function. Reads the nodes from a file.
- virtual bool readTopologyFromFile (std::string fileName)
- virtual bool setMatrixNodePositions (MatrixType, int *)
  Pure virtual member.
- virtual bool setMatrixNodePositions (MatrixType)
  Pure virtual member.
- virtual void writeObjectToFile (std::string fileName)
- virtual itk::Image< unsigned char, nDims >::Pointer generateBinaryImageFromTopology (typename BinaryImageType::SizeType size)
  Pure virtual member function. Creates a binary mask image from the organism.
- virtual void generateTopologyFromBinaryImage (BinaryImageTypePointer binaryInputImage)
  Pure virtual member function. Creates an organism from a binary mask.
- virtual bool removeNode (int nodeNumber)
  Pure virtual member function.
- virtual bool addNodes (MatrixType node, VectorType classes)
  Pure virtual member function.
- virtual bool addConnection (int, int)
  Pure virtual member function.
- virtual MatrixType getMatrixConnections ()
  Pure virtual member function. Returns the matrix of connections.
- virtual bool isInside (VectorType p)

Public Attributes

- MeshSpatialObjectType::Pointer theMeshSpatialObject
- MeshType::Pointer theMesh
- WriterType::Pointer writer

Protected Member Functions

- Geom_MeshSpatialObject (dType p=0.001)
  The default constructor.

5.42.1 Detailed Description

template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> class mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >

Derived geometric class based on a spatial object.

A derived class of the Geometrical ABC, this class provides the framework with the functionality of ITK mesh spatial objects. Creating a deformable organism is made easy with this class, as it can import any triangulated mesh spatial object contained in a meta file. It can also convert triangulated BYU surfaces to meta format, and import those. The primary restriction of this class is that it must be a triangulated mesh and, as such, parts of this classes functionality are currently restricted to 3D. Specifically, its ability to generate binary images from mesh objects. This class is also designed specifically to complement the Phys_LevelSet class. This class is currently classified as unstable, since it is constantly evolving to meet the changing needs of our framework.

Parameters:

dType The data type used for numerical storage.

nDims The number of dimensions.

MType The matrix type used for internal matrix storage. Defaults to vnl_matrix<dType>. Only vnl is supported at this time.

VType The vector type used for internal vector storage. Defaults to vnl_vector<dType>. Only vnl is supported at this time.

Definition at line 43 of file Geom_MeshSpatialObject.h.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 template<class dType, int nDims, class MType, class VType> mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::Geom_MeshSpatialObject (dType *p* = `0.001`) `[protected]`

The default constructor.

Parameters:

p The precision used to check if a point lies inside the object for conversion to a binary image.

Definition at line 10 of file Geom_MeshSpatialObject.cxx.

5.42.3 Member Function Documentation

5.42.3.1 template<class dType, int nDims, class MType, class VType> bool mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::addConnection (int, int) `[virtual]`

Pure virtual member function.

Adds a connection between nodes in the organism, should be ran in conjunction with addNode. Implementations should correctly update the topology, and change the state variables.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 596 of file Geom_MeshSpatialObject.cxx.

5.42.3.2 template<class dType, int nDims, class MType, class VType> bool mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::addNodes (MatrixType *node*, VectorType *classes*) [virtual]

Pure virtual member function.

Adds nodes to the organism, should be ran in conjunction with addConnection. Implementations should correctly update the topology, and change the state variables.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 56 of file Geom_MeshSpatialObject.cxx.

5.42.3.3 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual itk::Image< unsigned char, nDims>::Pointer mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::generateBinaryImageFromTopology (typename BinaryImageType::SizeType *size*) [virtual]

Pure virtual member function. Creates a binary mask image from the organism.

Any implementation of this method should a binary image.

Implements mial::Geometric< dType, nDims, MType, VType >.

5.42.3.4 template<class dType, int nDims, class MType, class VType> void mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::generateTopologyFromBinaryImage (BinaryImageTypePointer *binaryInputImage*) [virtual]

Pure virtual member function. Creates an organism from a binary mask.

Any implementation of this method should produce a deformable organism from a binary image.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 444 of file Geom_MeshSpatialObject.cxx.

5.42.3.5 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual MatrixType mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::getMatrixConnections () [inline, virtual]

Pure virtual member function. Returns the matrix of connections.

Implementation should update the matrix connections before returning, hence why this is a pure virtual.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 116 of file Geom_MeshSpatialObject.h.

5.42.3.6 template<class dType, int nDims, class MType, class VType> MType mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::getMatrixNodePositions () [virtual]

Pure virtual member function. Returns the matrix node positions.

Implementation should update the matrix node positions before returning, hence why this is a pure virtual.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 21 of file Geom_MeshSpatialObject.cxx.

Referenced by mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::simulate().

5.42.3.7 template<class dType, int nDims, class MType, class VType> void mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::readNodesFromFile (std::string *fileName*) [virtual]

Pure virtual member function. Reads the nodes from a file.

Derived class will need to update its own internal topology representation.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 226 of file Geom_MeshSpatialObject.cxx.

5.42.3.8 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual bool mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::removeNode (int *nodeNumber*) [inline, virtual]

Pure virtual member function.

Removes a node from the organism. Implementations must make sure to correctly update the topology, and set the state variables.

Parameters:

index The index of the node to be removed.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 113 of file Geom_MeshSpatialObject.h.

5.42.3.9 template<class dType, int nDims, class MType, class VType> bool mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::setMatrixNodePositions (MatrixType) [virtual]

Pure virtual member.

Set the matrix node positions for all nodes.

Parameters:

pos the positions

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 90 of file Geom_MeshSpatialObject.cxx.

5.42.3.10 template<class dType, int nDims, class MType, class VType> bool mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::setMatrixNodePositions (MatrixType, int *) [virtual]

Pure virtual member.

Set the matrix node positions at particular rows.

Parameters:

pos The new positions.

rows The rows to change.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 45 of file Geom_MeshSpatialObject.cxx.

Referenced by mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::simulate().

5.42.3.11 template<class dType, int nDims, class MType, class VType> void mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >::writeNodesToFile (std::string *fileName*) [virtual]

Pure virtual member function. Writes the nodes to a file.

Implementation should update the matrix positions before calling, hence why this is a pure virtual.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 197 of file Geom_MeshSpatialObject.cxx.

The documentation for this class was generated from the following files:

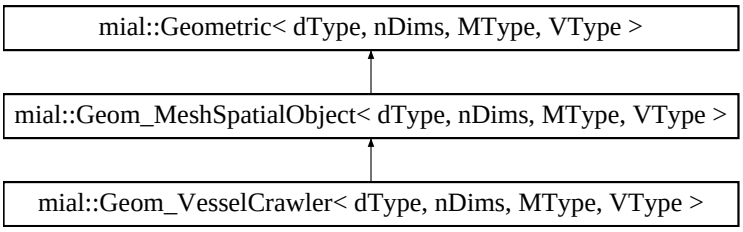
- C:/cmcintos/defOrgs/source/geometrical/Geom_MeshSpatialObject.h
- C:/cmcintos/defOrgs/source/geometrical/Geom_MeshSpatialObject.cxx

5.43 mial::Geom_VesselCrawler< dType, nDims, MType, VType > Class Template Reference

Derived geometric class based on a spatial object that includes a specific notion of layers.

`#include <Geom_VesselCrawler.h>`

Inheritance diagram for mial::Geom_VesselCrawler< dType, nDims, MType, VType >::



Public Types

- typedef MType MatrixType

  The internal matrix type.

- typedef VType VectorType

  The internal vector type.

Public Member Functions

- Geom_VesselCrawler (int numNPL=32)
- virtual bool addConnection (int, int, int=-1, int=-1)
- virtual bool addLayer (MType nodes, VType classes)
- virtual int getNumNodesPerLayer ()
- virtual VectorType getActiveSprings (int c=-1)
- virtual VectorType getActiveNodes (int c=-1)

5.43.1 Detailed Description

template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> class mial::Geom_VesselCrawler< dType, nDims, MType, VType >

Derived geometric class based on a spatial object that includes a specific notion of layers.

The layered geometric class used for vessel crawlers. Note that we did not specifically use TubularSpatialObjects because for each medial point we have explicit boundary nodes along with a radial value.

For details see [1].

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Parameters:

dType The data type used for numerical storage.

nDims The number of dimensions.

MType The matrix type used for internal matrix storage. Defaults to vnl_matrix<dType>. Only vnl is supported at this time.

VType The vector type used for internal vector storage. Defaults to vnl_vector<dType>. Only vnl is supported at this time.

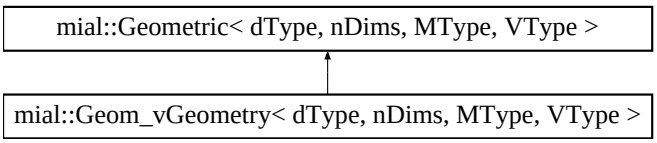
Definition at line 22 of file Geom_VesselCrawler.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Geom_VesselCrawler.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Geom_VesselCrawler.cxx

5.44 mial::Geom_vGeometry< dType, nDims, MType, VType > Class Template Reference

Inheritance diagram for mial::Geom_vGeometry< dType, nDims, MType, VType >::



Public Types

- typedef MType MatrixType

  The internal matrix type.

- typedef VType VectorType

  The internal vector type.

Public Member Functions

- Geom_vGeometry ()
- virtual MatrixType getVectorNodePositions ()
- virtual bool setNodePosition (int index, dType *pos)
- virtual void getNodePosition (int index, dType *)
- virtual void writeNodesToFile (std::string fileName)

  Pure virtual member function. Writes the nodes to a file.

- virtual void readNodesFromFile (std::string fileName)

  Pure virtual member function. Reads the nodes from a file.

- virtual unsigned int numNodes ()

5.44.1 Detailed Description

template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> class mial::Geom_vGeometry< dType, nDims, MType, VType >

Definition at line 18 of file Geom_vGeometry.h.

5.44.2 Member Function Documentation

5.44.2.1 template<class dType, int nDims, class MType, class VType> void mial::Geom_vGeometry< dType, nDims, MType, VType >::readNodesFromFile (std::string *fileName*) [virtual]

Pure virtual member function. Reads the nodes from a file.

Derived class will need to update its own internal topology representation.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 53 of file Geom_vGeometry.cxx.

5.44.2.2 template<class dType, int nDims, class MType, class VType> void mial::Geom_vGeometry< dType, nDims, MType, VType >::writeNodesToFile (std::string *fileName*) [virtual]

Pure virtual member function. Writes the nodes to a file.

Implementation should update the matrix positions before calling, hence why this is a pure virtual.

Implements mial::Geometric< dType, nDims, MType, VType >.

Definition at line 42 of file Geom_vGeometry.cxx.

The documentation for this class was generated from the following files:

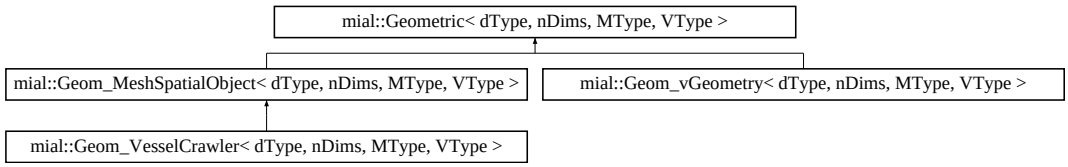
- C:/cmcintos/defOrgs/source/geometrical/Geom_vGeometry.h
- C:/cmcintos/defOrgs/source/geometrical/Geom_vGeometry.cxx

5.45 mial::Geometric< dType, nDims, MType, VType > Class Template Reference

Topological characteristics of the organism.

`#include <Geometric.h>`

Inheritance diagram for mial::Geometric< dType, nDims, MType, VType >::



Public Types

- typedef Geometric Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef VType VectorType

  The internal vector type.

- typedef MType MatrixType

  The internal matrix type.

- typedef itk::Image< unsigned char, nDims > BinaryImageType

  The type of binary image.

Public Member Functions

- virtual VectorType getCentroid ()
- virtual bool setMatrixNodePositions (MatrixType pos, int *rows)=0

  Pure virtual member.

- virtual bool setMatrixNodePositions (MatrixType pos)=0

  Pure virtual member.

- unsigned int getNumNodes ()

  Accessor returning the number of nodes.

- virtual bool addNodes (MatrixType nodes, VectorType classes)=0

  Pure virtual member function.

- virtual bool addConnection (int a, int b)=0

  Pure virtual member function.

- virtual bool removeNode (int index)=0

  Pure virtual member function.

- virtual MatrixType getMatrixNodePositions ()=0

  Pure virtual member function. Returns the matrix node positions.

- virtual MatrixType getMatrixConnections ()=0

  Pure virtual member function. Returns the matrix of connections.

- virtual void writeNodesToFile (std::string fileName)=0

  Pure virtual member function. Writes the nodes to a file.

- virtual void readNodesFromFile (std::string fileName)=0

  Pure virtual member function. Reads the nodes from a file.

- virtual void generateTopologyFromBinaryImage (typename BinaryImageType::Pointer binaryInputImage)=0

  Pure virtual member function. Creates an organism from a binary mask.

- virtual itk::Image< unsigned char, nDims >::Pointer generateBinaryImageFromTopology (typename BinaryImageType::SizeType)=0

  Pure virtual member function. Creates a binary mask image from the organism.

- unsigned int getNumConnections ()

  Accessor for the number of connections.

- bool didTopologyChange ()

  Accessor for topology change.

- bool didNodesChange ()

  Accessor for node position change.

Protected Member Functions

- Geometric ()

  Default constructor.

Protected Attributes

- MatrixType mNodes

  An numNodes by nDims matrix containing all the nodes of the organism.

- VectorType nodeClass

  A vector containing classifiers for all nodes.

- MatrixType mConnections

  A numConnections by 2 matrix containing connections between nodes as rows.

- VectorType connectionClass

  A vector containing classifiers for all connections.

- unsigned int numNodes

  The number of nodes. Must be updated by derived classes when new nodes are added.

- unsigned int numConnections

  The number of connections between nodes. Must be updated by derived classes when new connections are added.

- bool topologyChange

  A state variable denoting a change in topology.

- bool nodesChange

  A state variable denoting a change in node positions.

Classes

- struct Error

5.45.1 Detailed Description

template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> class mial::Geometric< dType, nDims, MType, VType >

Topological characteristics of the organism.

The geometrical layer of a deformable organism is responsible for managing the physical instantiation of the organisms body. Hence, the geometrical ABC serves as the storage point for the organisms topology.

Parameters:

dType The data type used for numerical storage.

nDims The number of dimensions.

MType The matrix type used for internal matrix storage. Defaults to vnl_matrix<dType>. Only vnl is supported at this time.

VType The vector type used for internal vector storage. Defaults to vnl_vector<dType>. Only vnl is supported at this time.

Definition at line 28 of file Geometric.h.

5.45.2 Member Function Documentation

5.45.2.1 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual bool mial::Geometric< dType, nDims, MType, VType >::addConnection (int *a*, int *b*) [pure virtual]

Pure virtual member function.

Adds a connection between nodes in the organism, should be ran in conjunction with addNode. Implementations should correctly update the topology, and change the state variables.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.2 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual bool mial::Geometric< dType, nDims, MType, VType >::addNodes (MatrixType *nodes*, VectorType *classes*) [pure virtual]

Pure virtual member function.

Adds nodes to the organism, should be ran in conjunction with addConnection. Implementations should correctly update the topology, and change the state variables.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.3 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> bool mial::Geometric< dType, nDims, MType, VType >::didNodesChange () [inline]

Accessor for node position change.

TODO: Decide if should be smart enough to know who is asking. Keep a list of clients.

Definition at line 240 of file Geometric.h.

Referenced by mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::simulate().

5.45.2.4 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> bool mial::Geometric< dType, nDims, MType, VType >::didTopologyChange () [inline]

Accessor for topology change.

TODO: Decide if should be smart enough to know who is asking. Keep a list of clients.

Definition at line 234 of file Geometric.h.

Referenced by mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::simulate().

5.45.2.5 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual itk::Image< unsigned char, nDims>::Pointer mial::Geometric< dType, nDims, MType, VType >::generateBinaryImageFromTopology (typename BinaryImageType::SizeType) [pure virtual]

Pure virtual member function. Creates a binary mask image from the organism.

Any implementation of this method should a binary image.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.6 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual void mial::Geometric< dType, nDims, MType, VType >::generateTopologyFromBinaryImage (typename BinaryImageType::Pointer *binaryInputImage*) [pure virtual]

Pure virtual member function. Creates an organism from a binary mask.

Any implementation of this method should produce a deformable organism from a binary image.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.7 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual MatrixType mial::Geometric< dType, nDims, MType, VType >::getMatrixConnections () [pure virtual]

Pure virtual member function. Returns the matrix of connections.

Implementation should update the matrix connections before returning, hence why this is a pure virtual.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.8 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual MatrixType mial::Geometric< dType, nDims, MType, VType >::getMatrixNodePositions () [pure virtual]

Pure virtual member function. Returns the matrix node positions.

Implementation should update the matrix node positions before returning, hence why this is a pure virtual.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.9 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual void mial::Geometric< dType, nDims, MType, VType >::readNodesFromFile (std::string *fileName*) [pure virtual]

Pure virtual member function. Reads the nodes from a file.

Derived class will need to update its own internal topology representation.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, mial::Geom_vGeometry< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix<

5.45.2.10 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual bool mial::Geometric< dType, nDims, MType, VType >::removeNode (int *index*) [pure virtual]

Pure virtual member function.

Removes a node from the organism. Implementations must make sure to correctly update the topology, and set the state variables.

Parameters:

index The index of the node to be removed.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.11 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual bool mial::Geometric< dType, nDims, MType, VType >::setMatrixNodePositions (MatrixType *pos*) [pure virtual]

Pure virtual member.

Set the matrix node positions for all nodes.

Parameters:

pos the positions

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.12 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual bool mial::Geometric< dType, nDims, MType, VType >::setMatrixNodePositions (MatrixType *pos*, int * *rows*) [pure virtual]

Pure virtual member.

Set the matrix node positions at particular rows.

Parameters:

pos The new positions.

rows The rows to change.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >.

5.45.2.13 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> virtual void mial::Geometric< dType, nDims, MType, VType >::writeNodesToFile (std::string *fileName*) [pure virtual]

Pure virtual member function. Writes the nodes to a file.

Implementation should update the matrix positions before calling, hence why this is a pure virtual.

Implemented in mial::Geom_MeshSpatialObject< dType, nDims, MType, VType >, mial::Geom_vGeometry< dType, nDims, MType, VType >, and mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix<

5.45.3 Member Data Documentation

5.45.3.1 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> VectorType mial::Geometric< dType, nDims, MType, VType >::connectionClass [protected]

A vector containing classifiers for all connections.

Numerous defines are supported, and can be overridden at risk of breaking compatibility with existing functionality.

0 - medialNode 1 - boundaryNode 2 - internalNode

Definition at line 98 of file Geometric.h.

5.45.3.2 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> MatrixType mial::Geometric< dType, nDims, MType, VType >::mConnections [protected]

A numConnections by 2 matrix containing connections between nodes as rows.

This must be updated anytime a derived class changes its own representation of the topology. This ensures a consistent data representation is available to all classes. However, for greater efficiency one could override the accessors to this function and therefore avoid doing the update; as the variable would never be accessed.

Definition at line 86 of file Geometric.h.

Referenced by mial::Geom_MeshSpatialObject< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >::getMatrixConnections().

5.45.3.3 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> MatrixType mial::Geometric< dType, nDims, MType, VType >::mNodes [protected]

An numNodes by nDims matrix containing all the nodes of the organism.

This must be updated anytime a derived class changes its own representation of nodes. This ensures a consistent data representation is available to all classes. However, for greater efficiency one could override the accessors to this function and therefore avoid doing the update; as the variable would never be accessed.

Definition at line 50 of file Geometric.h.

Referenced by mial::Geometric< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >::getCentroid().

5.45.3.4 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> VectorType mial::Geometric< dType, nDims, MType, VType >::nodeClass [protected]

A vector containing classifiers for all nodes.

Numerous defines are supported, and can be overridden at risk of breaking compatibility with existing functionality.

0 - medialNode 1 - boundaryNode 2 - internalNode

Definition at line 74 of file Geometric.h.

5.45.3.5 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> bool mial::Geometric< dType, nDims, MType, VType >::nodesChange [protected]

A state variable denoting a change in node positions.

Set this variable true whenever a change in position or count of the nodes occurs. Any objects keeping local copies of topology (physic layers, etc) should check this state before running.

Definition at line 124 of file Geometric.h.

Referenced by mial::Geometric< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >::didNodesChange().

5.45.3.6 template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> bool mial::Geometric< dType, nDims, MType, VType >::topologyChange [protected]

A state variable denoting a change in topology.

Set this variable true whenever a change in topology occurs. Any objects keeping local copies of topology (physic layers, etc) should check this state before running.

Definition at line 115 of file Geometric.h.

Referenced by mial::Geometric< Type, nDims, vnl_matrix< Type >, vnl_vector< Type > >::didTopologyChange().

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/source/geometrical/abc/Geometric.h

5.46 mial::Geometric< dType, nDims, MType, VType >::Error Struct Reference

Public Attributes

- std::string msg

5.46.1 Detailed Description

template<class dType, int nDims, class MType = vnl_matrix<dType>, class VType = vnl_vector<dType>> struct mial::Geometric< dType, nDims, MType, VType >::Error

Definition at line 127 of file Geometric.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/geometrical/abc/Geometric.h

5.47 mial::ImageViewerDescriptor Struct Reference

`#include <vtkDefOrgViewerWithKWState.h>`

Public Member Functions

- ImageViewerDescriptor (vtkImageViewer2 *_theImageViewerPointer, vtkKWRenderWidget *_theRenderWidget)
- ImageViewerDescriptor ()
- ~ImageViewerDescriptor ()

Public Attributes

- vtkImageViewer2 * theImageViewerPointer
- vtkKWRenderWidget * theRenderWidget

5.47.1 Detailed Description

Structure for storing the correspondance between ImageViewer and RenderWidget. Each ImageViewer has a corresponding RenderWidget

Definition at line 42 of file vtkDefOrgViewerWithKWState.h.

The documentation for this struct was generated from the following file:

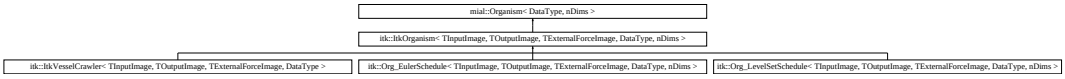
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKWState.h

5.48 itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Class Template Reference

A derived class that implements a deformable organism as an itk::ImageToImageFilter.

`#include <itkOrganism.h>`

Inheritance diagram for itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::



Public Types

- typedef TInputImage InputImageType
- typedef TOutputImage OutputImageType
- typedef TExternalForceImage ExternalForceImageType
- typedef ItkOrganism Self
- typedef ImageToImageFilter< InputImageType, OutputImageType > Superclass
- typedef SmartPointer< Self > Pointer
- typedef SmartPointer< const Self > ConstPointer
- typedef InputImageType::PixelType InputPixelType
- typedef OutputImageType::PixelType OutputPixelType
- typedef NumericTraits< InputPixelType >::RealType InputRealType
- typedef InputImageType::RegionType InputImageRegionType
- typedef OutputImageType::RegionType OutputImageRegionType
- typedef InputImageType::SizeType InputSizeType
- typedef Organism< DataType, nDims >::GeometricType GeometricType

  The type used for the geometric layer.

Public Member Functions

- itkStaticConstMacro (InputImageDimension, unsigned int, TInputImage::ImageDimension)
- itkNewMacro (Self)
- itkTypeMacro (ItkOrganism, ImageToImageFilter)
- virtual void GenerateInputRequestedRegion () throw (InvalidRequestedRegionError)
- virtual char * checkMessages ()
- virtual bool postMessage ()
- virtual int run ()

  Public method for simulating the organism.

- virtual bool setDefaultBehaviour ()
- virtual bool setSchedule ()
- virtual void addNode (DataType *n)
- virtual void writeNodesToFile (std::string fileName)
- virtual void readNodesFromFile (std::string fileName)

Protected Member Functions

- ItkOrganism ()
- virtual ~ItkOrganism ()
- void GenerateData ()

5.48.1 Detailed Description

template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> class itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >

A derived class that implements a deformable organism as an itk::ImageToImageFilter.

Those wishing to create new deformable organisms with image inputs/outputs should inherit directly from this class. Building a new organism essentially involves inheriting from this class, and placing customized layers directly in the organisms constructor. See Org_EulerSchedule and Org_LevelSetSchedule for details.

Parameters:

TInputImage the input image type

TOutputImage the output image type

TExternalForceImage the type of image used for external forces

DataType The data type to use (eg. float)

nDims The dimensionality of the organism

Definition at line 33 of file itkOrganism.h.

5.48.2 Member Typedef Documentation

5.48.2.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> typedef TInputImage itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::InputImageType

Convenient typedefs for simplifying declarations.

Definition at line 41 of file itkOrganism.h.

5.48.2.2 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> typedef InputImageType::PixelType itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::InputPixelType

Image typedef support.

Definition at line 59 of file itkOrganism.h.

5.48.2.3 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> typedef ItkOrganism itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Self

Standard class typedefs.

Reimplemented in itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >, itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >, and itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

Definition at line 47 of file itkOrganism.h.

5.48.3 Member Function Documentation

5.48.3.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkNewMacro (Self)

Method for creation through the object factory.

5.48.3.2 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkStaticConstMacro (InputImageDimension, unsigned *int*, TInputImage::ImageDimension)

Extract dimension from input and output image.

Reimplemented in itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >, and itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

5.48.3.3 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkTypeMacro (ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >, ImageToImageFilter)

Run-time type information (and related methods).

5.48.3.4 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> int itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::run () [virtual]

Public method for simulating the organism.

Implemenations of this method are in charge of running the cogntive and physics layers.

Implements mial::Organism< DataType, nDims >.

Definition at line 78 of file itkOrganism.cxx.

References mial::Organism< DataType, nDims >::cgLayer, and mial::Organism< DataType, nDims >::physLayer.

The documentation for this class was generated from the following files:

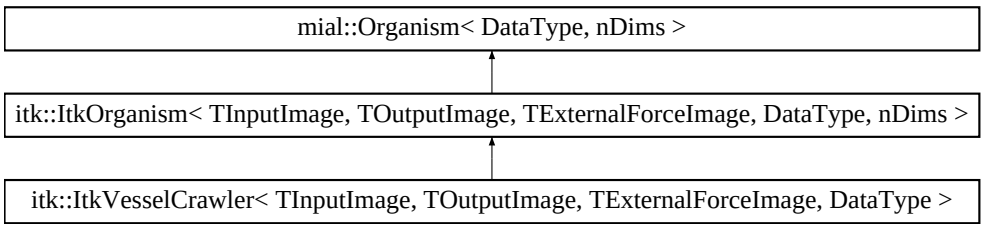
- C:/cmcintos/defOrgs/source/organism/itkOrganism.h
- C:/cmcintos/defOrgs/source/organism/itkOrganism.cxx

5.49 itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType > Class Template Reference

A vessel crawler [1] deformable organism for the segmentation and analysis of vasculature in 3D images.

`#include <itkVesselCrawler.h>`

Inheritance diagram for itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >::



Public Types

- typedef ItkVesselCrawler Self
- typedef ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Superclass
- typedef SmartPointer< Self > Pointer
- typedef SmartPointer< const Self > ConstPointer
- typedef itk::CovariantVector< DataType, nDims > GradientPixelType
- typedef itk::Image< GradientPixelType, nDims > GradientImageType
- typedef Phys_VesselCrawlerEuler< DataType, GradientImageType, nDims > PhysLayerType
- typedef Geom_VesselCrawler< DataType, nDims > GeometricType

  The type used for the geometric layer.

- typedef Sense_Gradient< DataType, InputImageType, GradientImageType, nDims > gradientSensorType
- typedef Ctrl_VesselCrawler< DataType, nDims, PhysLayerType > CognitiveLayerType

Public Member Functions

- ItkVesselCrawler ()
- ~ItkVesselCrawler ()
- itkNewMacro (Self)
- itkTypeMacro (ItkVesselCrawler, ItkOrganism)

Public Attributes

- gradientSensorType::sensorIn input
- gradientSensorType gradientSensor
- gradientSensorType::sensorOut * output
- PhysLayerType eulerPhys
- GeometricType meshGeom
- CognitiveLayerType sensoryCog

5.49.1 Detailed Description

template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType> class itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >

A vessel crawler [1] deformable organism for the segmentation and analysis of vasculature in 3D images.

For details on this deformable organism see [1].

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

Definition at line 48 of file itkVesselCrawler.h.

5.49.2 Member Typedef Documentation

5.49.2.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType> typedef ItkVesselCrawler itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >::Self

Standard class typedefs.

Reimplemented from itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

Definition at line 54 of file itkVesselCrawler.h.

5.49.3 Member Function Documentation

5.49.3.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType> itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >::itkNewMacro (Self)

Method for creation through the object factory.

5.49.3.2 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType> itk::ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >::itkTypeMacro (ItkVesselCrawler< TInputImage, TOutputImage, TExternalForceImage, DataType >, ItkOrganism)

Run-time type information (and related methods).

The documentation for this class was generated from the following files:

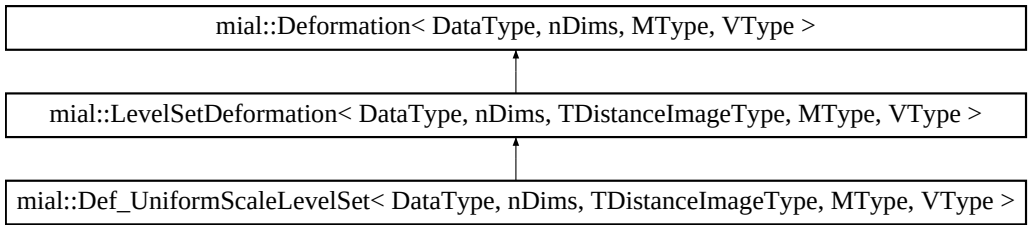
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/itkVesselCrawler.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/itkVesselCrawler.cxx

5.50 mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType > Class Template Reference

This class extends the basic deformation class with specific functionality for level set systems.

`#include <LevelSetDeformation.h>`

Inheritance diagram for mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::



Public Types

- typedef TDistanceImageType DistanceImageType

Classes

- struct DefArgSet

  A customized argument set.

5.50.1 Detailed Description

template<class DataType, int nDims, class TDistanceImageType, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >

This class extends the basic deformation class with specific functionality for level set systems.

All level set deformations should inherit from this class.

Parameters:

DataType the type of container

nDims the dimensionality of the deformation

TdistanceImage the type of distance image used

MType The matrix type used

VType The vector type used

Definition at line 22 of file LevelSetDeformation.h.

The documentation for this class was generated from the following file:

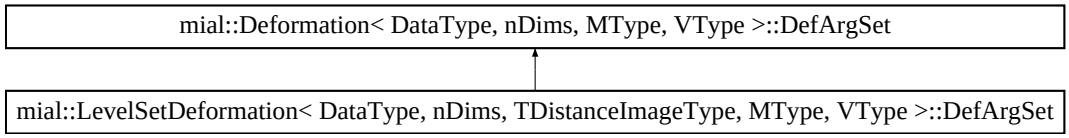
- C:/cmcintos/defOrgs/source/physical/abc/LevelSetDeformation.h

5.51 mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::DefArgSet Struct Reference

A customized argument set.

`#include <LevelSetDeformation.h>`

Inheritance diagram for mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::DefArgSet::



Public Attributes

- TDistanceImageType::Pointer distanceImg

5.51.1 Detailed Description

template<class DataType, int nDims, class TDistanceImageType, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::LevelSetDeformation< DataType, nDims, TDistanceImageType, MType, VType >::DefArgSet

A customized argument set.

Definition at line 28 of file LevelSetDeformation.h.

The documentation for this struct was generated from the following file:

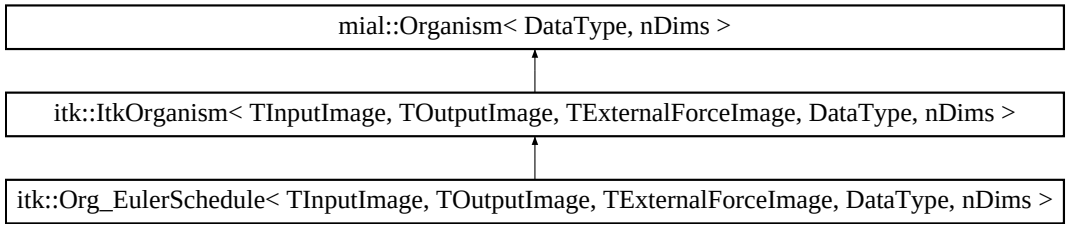
- C:/cmcintos/defOrgs/source/physical/abc/LevelSetDeformation.h

5.52 itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Class Template Reference

A derived class that implements an itkOrganism that posses built in Phys_Euler and Ctrl_Schedule layers, along with corresponding behaviors and deformations.

`#include <Org_EulerSchedule.h>`

Inheritance diagram for itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::



Public Types

- typedef Org_EulerSchedule Self
- typedef SmartPointer< Self > Pointer
- typedef SmartPointer< const Self > ConstPointer
- typedef Geom_MeshSpatialObject< DataType, nDims > GeometricType

  The type used for the geometric layer.

- typedef Phys_Euler< DataType, TExternalForceImage, nDims > PhysLayerType
- typedef Sense_Gradient< DataType, TInputImage, TExternalForceImage, nDims > gradientSensorType

Public Member Functions

- itkStaticConstMacro (InputImageDimension, unsigned int, TInputImage::ImageDimension)
- itkTypeMacro (Org_EulerSchedule, ItkOrganism)
- virtual bool setSchedule (std::string scheduleFileName)

  Set the schedule.

- virtual bool setTopology (std::string fName)

  Set the topology.

- virtual void setUp ()

  SetUp the organism.

Protected Member Functions

- Org_EulerSchedule ()
- virtual ~Org_EulerSchedule ()

Protected Attributes

- Ctrl_ScheduleDriven< DataType, nDims >::Pointer cgL
- GeometricType::Pointer geomLayer

  The Geometric layer that houses the shape of the organism.

- PhysLayerType::Pointer physLayer

  The Physics layer used to simulate the deformation dynamics.

- gradientSensorType::sensorIn::Pointer input
- gradientSensorType::Pointer gradientSensor
- Beh_SearchForObject< DataType, TInputImage, nDims >::Pointer beh0

5.52.1 Detailed Description

template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> class itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >

A derived class that implements an itkOrganism that posses built in Phys_Euler and Ctrl_Schedule layers, along with corresponding behaviors and deformations.

This class contains built in layers, and is therefore an example of a complete deformable organism. Though additional behaviors/deformations can still be added the following are provided. Beh_TranslateAll Def_-TranslateAll Beh_SearchForObject Beh_UniformScale Def_UniformScale

Parameters:

TInputImage the input image type

TOutputImage the output image type

TExternalForceImage the type of image used for external forces

DataType The data type to use (eg. DataType)

nDims The dimensionality of the organism

Definition at line 47 of file Org_EulerSchedule.h.

5.52.2 Member Typedef Documentation

5.52.2.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> typedef Org_EulerSchedule itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Self

Standard class typedefs.

Reimplemented from itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

Definition at line 55 of file Org_EulerSchedule.h.

5.52.3 Member Function Documentation

5.52.3.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkStaticConstMacro (InputImageDimension, unsigned *int*, TInputImage::ImageDimension)

Extract dimension from input and output image.

Reimplemented from itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

5.52.3.2 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkTypeMacro (Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >, ItkOrganism)

Run-time type information (and related methods).

The documentation for this class was generated from the following files:

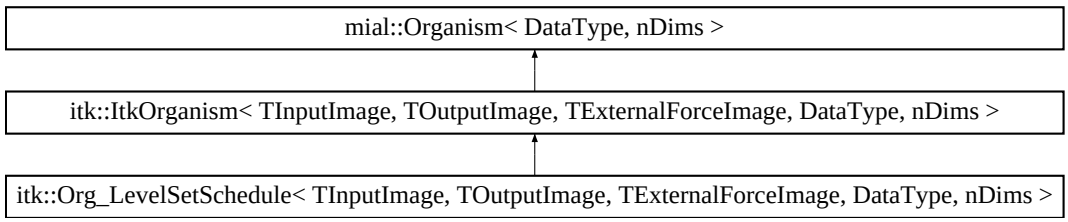
- C:/cmcintos/defOrgs/source/organism/Org_EulerSchedule.h
- C:/cmcintos/defOrgs/source/organism/Org_EulerSchedule.cxx

5.53 itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims > Class Template Reference

A derived class that implements an itkOrganism that posses built in Phys_LevelSet and Ctrl_Schedule layers, along with corresponding behaviors and deformations.

`#include <org_levelsetschedule.h>`

Inheritance diagram for itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::



Public Types

- typedef Org_LevelSetSchedule Self
- typedef SmartPointer< Self > Pointer
- typedef SmartPointer< const Self > ConstPointer
- typedef Geom_MeshSpatialObject< DataType, nDims > GeometricType

  The type used for the geometric layer.

- typedef Phys_LevelSet< DataType, TInputImage, nDims > PhysLayerType

Public Member Functions

- itkStaticConstMacro (InputImageDimension, unsigned int, TInputImage::ImageDimension)
- itkTypeMacro (Org_LevelSetSchedule, ItkOrganism)
- virtual bool setSchedule (std::string scheduleFileName)

  Set the schedule.

- virtual void setTopologyFromBinaryImage (typename GeometricType::BinaryImageType::Pointer img)

  Set the topology from a binary image.

- virtual void writeObjectToFile (std::string fname)
- virtual bool setTopology (std::string fName)

  Set the toplogy from a file.

- virtual void setUp ()

Protected Member Functions

- Org_LevelSetSchedule ()
- virtual ~Org_LevelSetSchedule ()

Protected Attributes

- Ctrl_ScheduleDriven< DataType, nDims >::Pointer cgL
- GeometricType::Pointer geomLayer

  The Geometric layer that houses the shape of the organism.

- PhysLayerType::Pointer physLayer

  The Physics layer used to simulate the deformation dynamics.

5.53.1 Detailed Description

template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> class itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >

A derived class that implements an itkOrganism that posses built in Phys_LevelSet and Ctrl_Schedule layers, along with corresponding behaviors and deformations.

This class contains built in layers, and is therefore an example of a complete deformable organism. Though additional behaviors/deformations can still be added the following are provided. Beh_UniformScale Def_-UniformScale

Parameters:

TInputImage the input image type

TOutputImage the output image type

TExternalForceImage the type of image used for external forces

DataType The data type to use (eg. DataType)

nDims The dimensionality of the organism

Definition at line 42 of file org_levelsetschedule.h.

5.53.2 Member Typedef Documentation

5.53.2.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> typedef Org_LevelSetSchedule itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Self

Standard class typedefs.

Reimplemented from itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

Definition at line 50 of file org_levelsetschedule.h.

5.53.3 Member Function Documentation

5.53.3.1 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkStaticConstMacro (InputImageDimension, unsigned *int*, TInputImage::ImageDimension)

Extract dimension from input and output image.

Reimplemented from itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

5.53.3.2 template<class TInputImage, class TOutputImage, class TExternalForceImage, class DataType, int nDims> itk::Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::itkTypeMacro (Org_LevelSetSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >, ItkOrganism)

Run-time type information (and related methods).

The documentation for this class was generated from the following files:

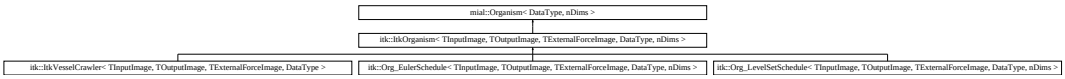
- C:/cmcintos/defOrgs/source/organism/org_levelsetschedule.h
- C:/cmcintos/defOrgs/source/organism/Org_LevelSetSchedule.cxx

5.54 mial::Organism< DataType, nDims > Class Template Reference

The abstract base container class that houses and connects all layers of the deformable organism.

`#include <Organism.h>`

Inheritance diagram for mial::Organism< DataType, nDims >::



Public Types

- typedef Geometric< DataType, nDims > GeometricType

  The type used for the geometric layer.

Public Member Functions

- Organism ()

  The default constructor.

- virtual bool addBehaviour (Behavior< DataType, nDims > *b, bool replacePhys=true)

  Public method for adding behaviors.

- virtual bool addDeformation (Deformation< DataType, nDims > *d)

  Public method for adding deformations.

- virtual bool setCognitiveLayer (ControlCenter< DataType, nDims > *c)

  Public method for setting the cogntive layer.

- virtual bool setPhysicsLayer (Physics< DataType, nDims > *p)

  Public method for setting the physics layer.

- virtual bool setGeometricLayer (Geometric< DataType, nDims > *g)

  Public method for setting the geometric layer.

- virtual int run ()=0

  Public method for simulating the organism.

- virtual void setRunTime (DataType r)

Protected Attributes

- Physics< DataType, nDims >::Pointer physLayer

  The Physics layer used to simulate the deformation dynamics.

- Geometric< DataType, nDims >::Pointer geomLayer

  The Geometric layer that houses the shape of the organism.

- ControlCenter< DataType, nDims >::Pointer cgLayer

  The ControlCenter (or cognitive layer) that is responsible for deciding what action to take, and performing that action.

- DataType runTime

5.54.1 Detailed Description

template<class DataType, int nDims> class mial::Organism< DataType, nDims >

The abstract base container class that houses and connects all layers of the deformable organism.

In medical image analysis strategies based on deformable models, controlling the deformations of the models is a desirable goal to produce proper segmentations. Incorporating expert knowledge to automatically guide deformations cannot be easily and elegantly achieved using the classical deformable model low-level energy-based fitting mechanisms. Deformable Organisms (DOs), are a decision-making framework for medical image analysis that complements bottom-up, data-driven deformable models with top-down, knowledge-driven mode-fitting strategies in a layered fashion inspired by artificial life modeling concepts. Intuitive and controlled deformations are carried out through behaviors. Sensory input from image data and contextual knowledge about the analysis problem govern these different behaviors.

Deformable Organisms are built following a multilevel AL modelling approach consisting of four primary layers: cognitive, behavioral, physical, and geometrical. Specifically, the cognitive layer makes decisions based on the DOs current state, anatomical knowledge, and its surrounding environment (the image). Decisions could be made to sense information, to deform based on sensory data, to illicit help from the user, or to terminate the segmentation process. All of these actions are described under the behavioral layer of the organism, and they rely upon both the physical and geometrical layers for implementation. For example, in the context of our 'vessel crawlers', the act of moving towards a sensed target location is described by the 'growing' behavioral method. The cognitive center gathers sensory input using the 'sense-to-grow' sensory module, decides the correct location via the 'where-to-grow' decision module, elicits the act of 'growing', and then conforms to the vascular walls by 'fitting'. In turn, each of these methods relies upon the physical and geometrical layers to carry out tasks, such as maintaining model stability. Consequently, we have a framework with many independent layers of abstraction, each built upon the implementation of independent modules and or processes.

Parameters:

DataType The data type to use (eg. float)

nDims The dimensionality of the organism

Definition at line 58 of file Organism.h.

5.54.2 Member Function Documentation

5.54.2.1 template<class DataType, int nDims> virtual bool mial::Organism< DataType, nDims >::addBehaviour (Behavior< DataType, nDims > * *b*, bool *replacePhys* = `true`) `[inline, virtual]`

Public method for adding behaviors.

Parameters:

Behavior<DataType,nDims> * b the behavior to add

bool replacePhys = true A boolean indicating whether or not to replace the physics layer of the passed in behavior with that of the organism. If set to false, the organism may run behaviors that actually simulate a different organism.

Definition at line 88 of file Organism.h.

Referenced by itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Org_EulerSchedule().

5.54.2.2 template<class DataType, int nDims> virtual bool mial::Organism< DataType, nDims >::addDeformation (Deformation< DataType, nDims > * *d*) `[inline, virtual]`

Public method for adding deformations.

Parameters:

Deformation<DataType,nDims> * d the deformation to add

Definition at line 94 of file Organism.h.

Referenced by itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Org_EulerSchedule().

5.54.2.3 template<class DataType, int nDims> virtual int mial::Organism< DataType, nDims >::run () `[pure virtual]`

Public method for simulating the organism.

Implemenations of this method are in charge of running the cogntive and physics layers.

Implemented in itk::ItkOrganism< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >.

5.54.2.4 template<class DataType, int nDims> virtual bool mial::Organism< DataType, nDims >::setCognitiveLayer (ControlCenter< DataType, nDims > * *c*) `[inline, virtual]`

Public method for setting the cogntive layer.

Parameters:

ControlCenter<DataType,nDims> * c the control center to set.

Definition at line 100 of file Organism.h.

Referenced by itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Org_EulerSchedule().

5.54.2.5 template<class DataType, int nDims> virtual bool mial::Organism< DataType, nDims >::setGeometricLayer (Geometric< DataType, nDims > * *g*) [inline, virtual]

Public method for setting the geometric layer.

Parameters:

Geometric<DataType,nDims> * c the geometric layer to set.

Definition at line 113 of file Organism.h.

Referenced by itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Org_EulerSchedule().

5.54.2.6 template<class DataType, int nDims> virtual bool mial::Organism< DataType, nDims >::setPhysicsLayer (Physics< DataType, nDims > * *p*) [inline, virtual]

Public method for setting the physics layer.

Parameters:

Physics<DataType,nDims> * p the physics layer to set.

Definition at line 107 of file Organism.h.

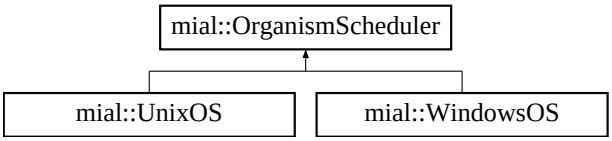
Referenced by itk::Org_EulerSchedule< TInputImage, TOutputImage, TExternalForceImage, DataType, nDims >::Org_EulerSchedule().

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/source/organism/Organism.h

5.55 mial::OrganismScheduler Class Reference

Inheritance diagram for mial::OrganismScheduler::



Public Member Functions

- simulateMultiOrganismSystem ()

Protected Attributes

- Organism * organisms

5.55.1 Detailed Description

Definition at line 13 of file OrganismScheduler.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/source/organismScheduler/OrganismScheduler.h
- C:/cmcintos/defOrgs/source/organismScheduler/OrganismScheduler.cxx

5.56 mial::OutputImageDescriptorStruct Struct Reference

`#include <DefOrgViewerAdapterBase.h>`

Public Member Functions

- OutputImageDescriptorStruct (vtkImageImport *_theImageVolume, bool _displayInSeparateFrame, int _D, bool _isModified=true)
- OutputImageDescriptorStruct ()

Public Attributes

- vtkImageImport * theImageVolume
- bool isModified
- bool isInitialized
- int dimension
- std::string windowName
- bool displayInSeparateFrame

5.56.1 Detailed Description

Structure for storing output images descriptions. Viewer assigns a vtkImageImport to be used by the adapter, the adapter populat the scene. Modified field must be set if the scene is changed. displayInSeparateFrame is currently ignored.

Definition at line 109 of file DefOrgViewerAdapterBase.h.

The documentation for this struct was generated from the following file:

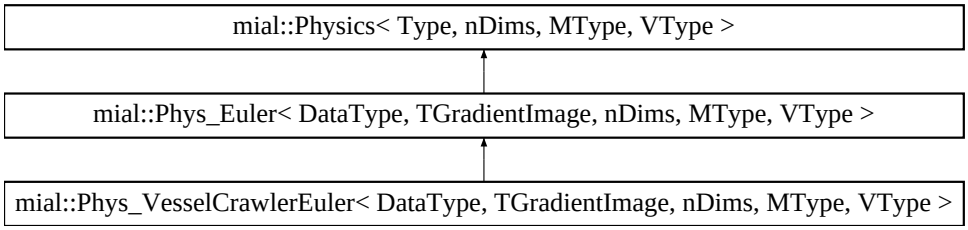
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase.h

5.57 mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType > Class Template Reference

A derived physics class capable of carrying out deformations of a spring mass system.

`#include <Phys_Euler.h>`

Inheritance diagram for mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::



Public Types

- typedef Phys_Euler Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef SpringMassDeformation< DataType, nDims, MType, VType > DeformationType

  The type of deformation's used. Notice they must be of the same internal storage types and dimension as the physics class.

- typedef MType MatrixType
- typedef VType VectorType
- typedef TGradientImage GradientImageType

Public Member Functions

- virtual bool runDeformation (const std::string defName, typename DeformationType::deformationIn *const i, std::stringstream *const s=NULL)

  Run a deformation.

- virtual void setRestLengths (int *a, DataType *values, int n)

  Set the restlengths of the specified springs to the specified values.

- virtual void setRestLengths (VectorType a, VectorType values)

  Set the restlengths of the specified springs to the specified values.

- virtual void setTimeStep (double a)

  Set the time step.

- virtual void setSpringLengths (int *a, DataType *values, int n)
  Set the lengths of the specified springs to the specified values.
- virtual void setSpringLengths (VectorType a, VectorType values)
  Set the lengths of the specified springs to the specified values.
- virtual void setSpringsK (int *a, DataType *values, int n)
  Set the Hooke's constants of the specified springs to the specified values.
- virtual void setSpringsK (VectorType a, VectorType values)
  Set the Hooke's constants of the specified springs to the specified values.
- virtual void enableImageForces ()
  enable image forces (defaultly enabled)
- virtual void disableImageForces ()
  disable image forces (defaultly enabled)
- virtual bool simulate ()
  Simulate the deformation dynamics.
- virtual void setExternalForces (void *f)
  Set the external forces to be used during the simulation.

Protected Types

- typedef GradientImageType::Pointer GradientImageTypePointer
  Typedef for the type of gradient image smart pointer used.

Protected Member Functions

- Phys_Euler (int numNodes=0, int numSprings=0, int numPossibleDeformations=0, int defK=35)

Protected Attributes

- MatrixType nodes
  A matrix of nodes.
- MatrixType nodesV
  A matrix of node velocities.
- MatrixType nodesF
  A matrix of node forces.
- MatrixType nodesFDef
  A matrix of node deformation forces.

- MatrixType nodesA

  A matrix of node accelerations.

- VectorType nodesM

  A matrix of node masses.

- DataType defaultMass

  The default mass.

- VectorType springsRest

  A vector of spring rest lengths.

- VectorType springsDamp

  A vector of spring dampening amounts.

- DataType defaultDamp

  The default dampening amount.

- MatrixType springsNodes

  A matrix describing how the springs and masses are connected. e.g. spring1: node 1 –> node 2, spring2: node 2 –> node 3.

- VectorType springLengths

  A vector of current spring lengths.

- VectorType springsK

  A vector of Hooke constants for each spring.

- DataType defaultK

  The default Hooke constant.

- DataType defaultDrag

  The default drag force.

- double timeStep

  How large a time step to take during the simulations. Position = PositionOld + (VelocityOld + acceleartion∗timeStep)∗timeStep;.

- GradientImageTypePointer gradientPointer

  The pointer to the gradient image.

- bool imageForces

  Whether or not image forces are enabled.

Classes

- struct Error

5.57.1 Detailed Description

template<class DataType, class TGradientImage, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >

A derived physics class capable of carrying out deformations of a spring mass system.

A derived class of the physical ABC, this class provides the framework with the ability to simulate deformation dynamics of any dimension on a spring mass system. It assumes any node in the geometrical layer to represent a mass and any connection present to represent a spring between connected masses. This class is considered to be relatively stable, in that additional functionality may be added but current functionality will remain.

See [1] for details on the simulation dynamics and examples of possible controled deformation types.

[1] Ghassan Hamarneh and Chris McIntosh. Physics-based deformable organisms for medical image analysis. SPIE Medical Imaging, 5747:326335, 2005.

Parameters:

Type The data type to be used for internal storage.

TGradientImage The data type used for external forces.

nDims The number of dimensions used for simulations.

MType The type of matrix used for internal storage. Defaults to vnl_matrix<Type>, the only supported type at this time.

VType The type of vector used for internal storage. Defaults to vnl_vector<Type>, the only supported type at this time.

Definition at line 46 of file Phys_Euler.h.

5.57.2 Member Function Documentation

5.57.2.1 template<class DataType, class TGradientImage, int nDims, class MType, class VType> bool mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::runDeformation (const std::string *defName*, typename DeformationType::deformationIn *const *i*, std::stringstream *const *s* = NULL) [virtual]

Run a deformation.

Assemble the neccessary inputs and run the named deformation. Typically, only behaviors call this method. Note that only one of i,s can be non-null, while s is only optionally supported.

Parameters:

defName the name of the deformation

i The deformations argument list prepared by the behavior requesting the run

s The deformations argument list in stream format.

Implements mial::Physics< Type, nDims, MType, VType >.

Definition at line 123 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::defaultMass, mial::Physics< Type, nDims, MType, VType >::Error::deformationError, mial::Physics< Type, nDims, MType, VType >::Error::deformationNumber, mial::Physics< Type, nDims, MType, VType >::deformationsList, mial::Physics< Type, nDims, MType, VType >::geom, mial::Physics< Type, nDims, MType, VType >::Error::msg, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::nodes, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::nodesA, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::nodesF, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::nodesFDef, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::nodesM, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::nodesV, mial::Physics< Type, nDims, MType, VType >::numDeformations, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springLengths, mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springsNodes, and mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springsRest.

5.57.2.2 template<class DataType, class TGradientImage, int nDims, class MType, class VType> void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setRestLengths (VectorType *a*, VectorType *values*) [virtual]

Set the restlengths of the specified springs to the specified values.

Parameters:

- ***a*** the indices of the springs
- ***values*** the values to use

Definition at line 62 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springsRest.

5.57.2.3 template<class DataType, class TGradientImage, int nDims, class MType, class VType> void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setRestLengths (int * *a*, DataType * *values*, int *n*) [virtual]

Set the restlengths of the specified springs to the specified values.

Parameters:

- ***a*** the indices of the springs
- ***values*** the values to use
- ***n*** the number of springs being set

Definition at line 49 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springsRest.

5.57.2.4 template<class DataType, class TGradientImage, int nDims, class MType, class VType> void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setSpringLengths (VectorType *a*, VectorType *values*) [virtual]

Set the lengths of the specified springs to the specified values.

Parameters:

a the indices of the springs

values the values to use

Definition at line 87 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springLengths.

5.57.2.5 template<class DataType, class TGradientImage, int nDims, class MType, class VType> void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setSpringLengths (int * *a*, DataType * *values*, int *n*) [virtual]

Set the lengths of the specified springs to the specified values.

Parameters:

a the indices of the springs

values the values to use

n the number of springs being set

Definition at line 74 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springLengths.

5.57.2.6 template<class DataType, class TGradientImage, int nDims, class MType, class VType> void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setSpringsK (VectorType *a*, VectorType *values*) [virtual]

Set the Hooke's constants of the specified springs to the specified values.

Parameters:

a the indices of the springs

values the values to use

Definition at line 112 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springsK.

5.57.2.7 template<class DataType, class TGradientImage, int nDims, class MType, class VType> void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setSpringsK (int * *a*, DataType * *values*, int *n*) [virtual]

Set the Hooke's constants of the specified springs to the specified values.

Parameters:

a the indices of the springs

values the values to use

n the number of springs being set

Definition at line 99 of file Phys_Euler.cxx.

References mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::springsK.

5.57.2.8 template<class DataType, class TGradientImage, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> virtual void mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::setTimeStep (double *a*) [inline, virtual]

Set the time step.

Parameters:

a the new time step.

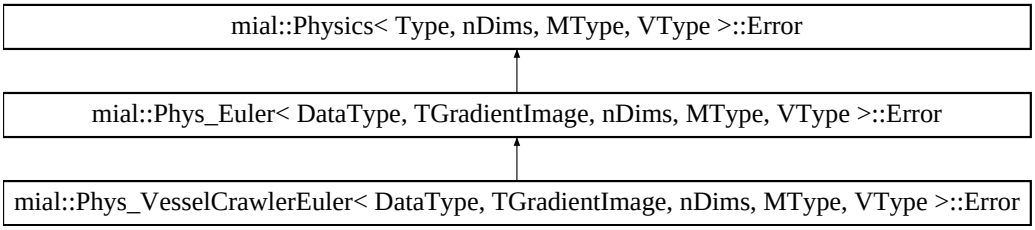
Definition at line 168 of file Phys_Euler.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/source/physical/Phys_Euler.h
- C:/cmcintos/defOrgs/source/physical/Phys_Euler.cxx

5.58 mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::Error Struct Reference

Inheritance diagram for mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::Error::



5.58.1 Detailed Description

template<class DataType, class TGradientImage, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >::Error

Definition at line 67 of file Phys_Euler.h.

The documentation for this struct was generated from the following file:

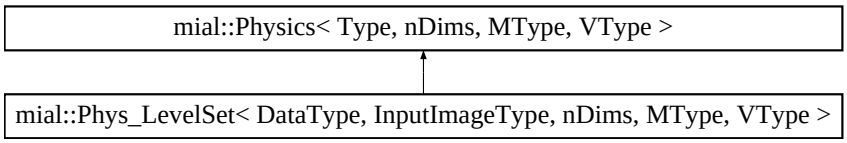
- C:/cmcintos/defOrgs/source/physical/Phys_Euler.h

5.59 mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType > Class Template Reference

A derived physics class capable of carrying out deformations of a spring mass system.

`#include <Phys_LevelSet.h>`

Inheritance diagram for mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >::



Public Types

- typedef Phys_LevelSet Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef Physics< DataType, nDims, MType, VType >::Error Error
- typedef itk::Image< DataType, nDims > InternalImageType
- typedef Geometric< DataType, nDims >::BinaryImageType BinaryImageType
- typedef LevelSetDeformation< DataType, nDims, InternalImageType > DeformationType

  The type of deformation's used. Notice they must be of the same internal storage types and dimension as the physics class.

Public Member Functions

- virtual void initializeDistanceImg ()
- virtual void initializeEdgePotentialImg ()
- virtual bool simulate ()

  Simulate the deformation dynamics.

- virtual bool runDeformation (const std::string defName, typename DeformationType::deformationIn *const i, std::stringstream *const s=NULL)

  Public member. Used by a behaviour to execute a particular deformation by name.

- virtual void setExternalForces (void *)

  Public pure virtual function for setting the external forces used during simulations of the deformation dynamics.

- virtual void setInput (typename InputImageType::ConstPointer img)

  Set the layer's input image.

- virtual void setPropagationScaling (const double ps)
- virtual void setCurvatureScaling (const double cs)
- virtual void setAdvectionScaling (const double as)
- virtual void setMaximumRMSError (const double me)
- virtual void setNumIterations (const double ni)
- virtual double getPropagationScaling ()
- virtual double getCurvatureScaling ()
- virtual double getAdvectionScaling ()
- virtual double getMaximumRMSError ()
- virtual double getNumIterations ()
- void printInternalValues ()
- InternalImageType::Pointer getDistanceImage ()
- InternalImageType::Pointer getEdgePotentialImage ()
- BinaryImageType::Pointer getOutputImage ()

Protected Member Functions

- Phys_LevelSet ()

5.59.1 Detailed Description

template<class DataType, class InputImageType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >

A derived physics class capable of carrying out deformations of a spring mass system.

A derived class of the physical ABC, this class provides the framework with the ability to simulate deformation dynamics using a geodesic active contour approach[1]. This class is considered to be relatively stable, in that additional functionality may be added but current functionality will remain.

[1] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In ICCV, pages 694699, 1995.

Parameters:

DataType The data type to be used for internal storage.

InputImageType The data type used for the input image

nDims The number of dimensions used for simulations.

MType The type of matrix used for internal storage. Defaults to vnl_matrix<Type>, the only supported type at this time.

VType The type of vector used for internal storage. Defaults to vnl_vector<Type>, the only supported type at this time.

Definition at line 54 of file Phys_LevelSet.h.

5.59.2 Member Function Documentation

5.59.2.1 template<class DataType, class InputImageType, int nDims, class MType, class VType> bool mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >::runDeformation (const std::string *defName*, typename DeformationType::deformationIn *const i*, std::stringstream *const s* = NULL) [virtual]

Public member. Used by a behaviour to execute a particular deformation by name.

Runs a deformation on the attached geometric layer.

Parameters:

args The marshaled argument list.

Implements mial::Physics< Type, nDims, MType, VType >.

Definition at line 226 of file Phys_LevelSet.cxx.

References mial::Physics< Type, nDims, MType, VType >::Error::deformationError, mial::Physics< Type, nDims, MType, VType >::Error::deformationNumber, and mial::Physics< Type, nDims, MType, VType >::Error::msg.

5.59.2.2 template<class DataType, class InputImageType, int nDims, class MType, class VType> void mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >::setExternalForces (void *) [virtual]

Public pure virtual function for setting the external forces used during simulations of the deformation dynamics.

The external force image is used to provide external forces to the organism during the deformation process. These could be gradient based, or a distance transform from a point of interest, etc.

Parameters:

img The image to be used as an external force. Derived classes must publicly define the expected type, and then typecast the input to this function.

Implements mial::Physics< Type, nDims, MType, VType >.

Definition at line 342 of file Phys_LevelSet.cxx.

5.59.2.3 template<class DataType, class InputImageType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> virtual void mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >::setInput (typename InputImageType::ConstPointer *img*) [inline, virtual]

Set the layer's input image.

Parameters:

img the image to be used by the layer for gradient magnitude calculations.

Definition at line 97 of file Phys_LevelSet.h.

The documentation for this class was generated from the following files:

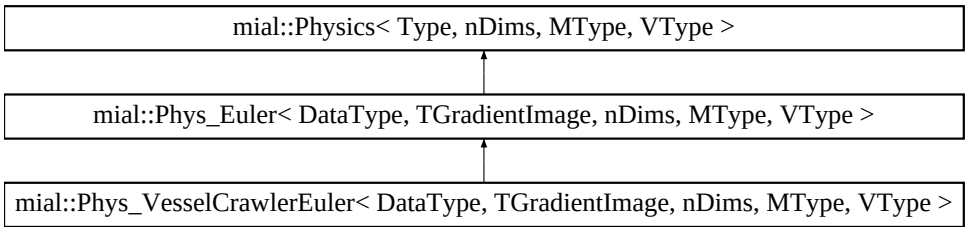
- C:/cmcintos/defOrgs/source/physical/Phys_LevelSet.h
- C:/cmcintos/defOrgs/source/physical/Phys_LevelSet.cxx

5.60 mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType > Class Template Reference

A derived physics class capable of carrying out deformations of a layered spring mass system.

`#include <Phys_VesselCrawlerEuler.h>`

Inheritance diagram for mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::



Public Types

- typedef Geom_VesselCrawler< DataType, nDims, MatrixType, VectorType > CrawlerGeometryType

Public Member Functions

- void setGeometry (GeometryType *a)
- virtual bool simulate ()

  A re-implementation of the typical Phys_Euler simulation that only simulates active sections of the mesh (layers).

Public Attributes

- CrawlerGeometryType * geom

  *A pointer to the crawler geometry, hides GeometryType *geom defined in Physics.h.*

Classes

- struct Error

5.60.1 Detailed Description

template<class DataType, class TGradientImage, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >

A derived physics class capable of carrying out deformations of a layered spring mass system.

A derived class of the physical ABC, this class provides the framework with the ability to simulate deformation dynamics of a 3D layered spring-mass system.

Parameters:

Type The data type to be used for internal storage.

TGradientImage The data type used for external forces.

nDims The number of dimensions used for simulations.

MType The type of matrix used for internal storage. Defaults to vnl_matrix<Type>, the only supported type at this time.

VType The type of vector used for internal storage. Defaults to vnl_vector<Type>, the only supported type at this time.

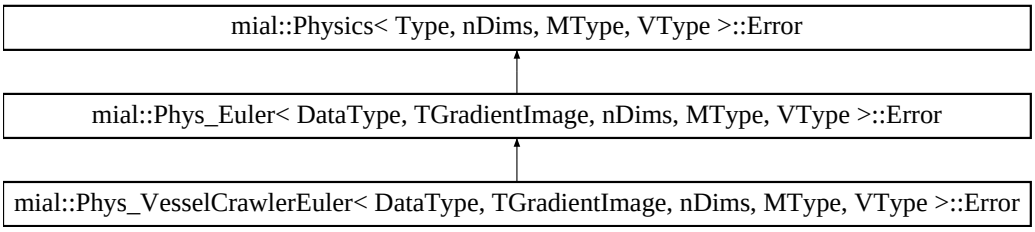
Definition at line 30 of file Phys_VesselCrawlerEuler.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Phys_VesselCrawlerEuler.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Phys_VesselCrawlerEuler.cxx

5.61 mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::Error Struct Reference

Inheritance diagram for mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::Error::



5.61.1 Detailed Description

template<class DataType, class TGradientImage, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::Phys_VesselCrawlerEuler< DataType, TGradientImage, nDims, MType, VType >::Error

Definition at line 39 of file Phys_VesselCrawlerEuler.h.

The documentation for this struct was generated from the following file:

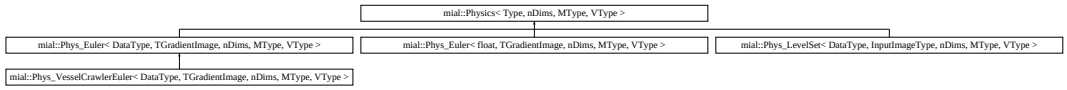
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Phys_VesselCrawlerEuler.h

5.62 mial::Physics< Type, nDims, MType, VType > Class Template Reference

Simluates the deformation dynamics, thereby, modifying actual positions of nodes belonging to the organism.

`#include <Physics.h>`

Inheritance diagram for mial::Physics< Type, nDims, MType, VType >::



Public Types

- typedef Physics Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef Deformation< Type, nDims, MType, VType > DeformationType

  The type of deformation's used. Notice they must be of the same internal storage types and dimension as the physics class.

- typedef MType MatrixType
- typedef VType VectorType
- typedef Geometric< Type, nDims, MatrixType, VectorType > GeometryType

  The type of geometric layer used. Notice that it must be of the same internal storage types and dimension as the physics class.

Public Member Functions

- ~Physics ()

  The destructor in charge of correctly removing the deformation list from the heap before an instantiation of the class is removed from memory.

- virtual bool runDeformation (const std::string defName, typename DeformationType::deformationIn *const i, std::stringstream *const s=NULL)=0

  Public member. Used by a behaviour to execute a particular deformation by name.

- virtual bool addDeformation (Deformation< Type, nDims > *def)

  Public member for adding a new deformation.

- virtual bool simulate ()=0

  Simulate the deformation dynamics.

- virtual void setExternalForces (void *img)=0

  Public pure virtual function for setting the external forces used during simulations of the deformation dynamics.

- void setGeometry (GeometryType *a)

  Public method for setting the geometric type.

- double getTime ()

Protected Member Functions

- Physics ()

  The default constructor.

Protected Attributes

- int numDeformations

  The number of deformations currently invocable.

- GeometryType::Pointer geom

  The geometric layer.

- std::vector< typename Deformation< Type, nDims >::Pointer > deformationsList

  The list of deformations invocable by a particular physics layer. A list of smart pointers.

- double time

  Keep track of the time.

Classes

- struct Error

  A structure containing error information that should be filled and thrown whenever an error in the simulation occurs.

5.62.1 Detailed Description

template<class Type, int nDims, class MType = vnl_matrix<Type>, class VType = vnl_vector<Type>> class mial::Physics< Type, nDims, MType, VType >

Simluates the deformation dynamics, thereby, modifying actual positions of nodes belonging to the organism.

The physical layer of a deformable organism handles the simulation of the deformation dynamics. That is to say, it manipulates the nodes, or surfaces housed by the geometrical layer. It also maintains a list of available deformations which can be invoked on the model via the runDeformation method. This class is currently classified as stable.

Error reporting is handled by the Error structure, containing an error string, the deformation number concerned, and the deformations Error structure itself.

Parameters:

Type The data type to be used for internal storage.

nDims The number of dimensions used for simulations.

MType The type of matrix used for internal storage. Defaults to vnl_matrix<Type>, the only supported type at this time.

VType The type of vector used for internal storage. Defaults to vnl_vector<Type>, the only supported type at this time.

Definition at line 39 of file Physics.h.

5.62.2 Member Function Documentation

5.62.2.1 template<class Type, int nDims, class MType = vnl_matrix<Type>, class VType = vnl_vector<Type>> virtual bool mial::Physics< Type, nDims, MType, VType >::runDeformation (const std::string *defName*, typename DeformationType::deformationIn *const *i*, std::stringstream *const *s* = NULL) [pure virtual]

Public member. Used by a behaviour to execute a particular deformation by name.

Runs a deformation on the attached geometric layer.

Parameters:

args The marshaled argument list.

Implemented in mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >, mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >, and mial::Phys_Euler< float, TGradientImage, nDims, MType, VType >.

Referenced by mial::Beh_UniformScale< Type, nDims >::run(), mial::Beh_TranslateAll< Type, nDims >::run(), and mial::Beh_TranslateAll< Type, nDims >::update().

5.62.2.2 template<class Type, int nDims, class MType = vnl_matrix<Type>, class VType = vnl_vector<Type>> virtual void mial::Physics< Type, nDims, MType, VType >::setExternalForces (void * *img*) [pure virtual]

Public pure virtual function for setting the external forces used during simulations of the deformation dynamics.

The external force image is used to provide external forces to the organism during the deformation process. These could be gradient based, or a distance transform from a point of interest, etc.

Parameters:

img The image to be used as an external force. Derived classes must publicly define the expected type, and then typecast the input to this function.

Implemented in mial::Phys_Euler< DataType, TGradientImage, nDims, MType, VType >, mial::Phys_LevelSet< DataType, InputImageType, nDims, MType, VType >, and mial::Phys_Euler< float, TGradientImage, nDims, MType, VType >.

The documentation for this class was generated from the following files:

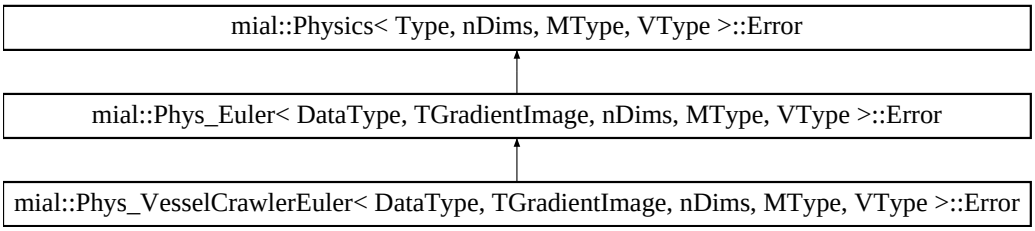
- C:/cmcintos/defOrgs/source/physical/abc/Physics.h
- C:/cmcintos/defOrgs/source/physical/abc/Physics.cxx

5.63 mial::Physics< Type, nDims, MType, VType >::Error Struct Reference

A structure containing error information that should be filled and thrown whenever an error in the simulation occurs.

`#include <Physics.h>`

Inheritance diagram for mial::Physics< Type, nDims, MType, VType >::Error::



Public Attributes

- std::string msg
- int deformationNumber
- DeformationType::Error * deformationError

5.63.1 Detailed Description

template<class Type, int nDims, class MType = vnl_matrix<Type>, class VType = vnl_vector<Type>> struct mial::Physics< Type, nDims, MType, VType >::Error

A structure containing error information that should be filled and thrown whenever an error in the simulation occurs.

Parameters:

msg A text description of the error, should also be written to std::cerr

deformationNumber The deformation's number in the deformation list. 0-numDeformations

deformationError The deformation's error information.

Definition at line 65 of file Physics.h.

The documentation for this struct was generated from the following file:

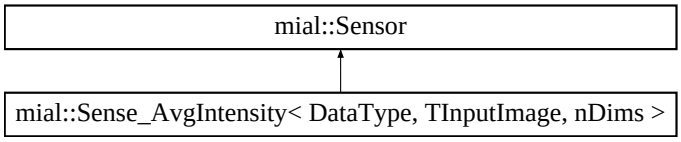
- C:/cmcintos/defOrgs/source/physical/abc/Physics.h

5.64 mial::Sense_AvgIntensity< DataType, TInputImage, nDims > Class Template Reference

Derived sensory class for computing image AvgIntensity.

`#include <Sense_AvgIntensity.h>`

Inheritance diagram for mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::



Public Types

- typedef Sense_AvgIntensity Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef TInputImage InputImageType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef itk::Image< DataType, nDims > OutputImageType
- typedef OutputImageType::ConstPointer OutputImagePointer

Public Member Functions

- virtual void run (typename Sensor::sensorIn *const i)

  Pure virtual public member function. New sensors operations are defined and run by implementing this method in a derived class.

Protected Member Functions

- Sense_AvgIntensity ()

Classes

- struct sensorIn
- struct sensorOut

5.64.1 Detailed Description

template<class DataType, class TInputImage, int nDims> class mial::Sense_AvgIntensity< DataType, TInputImage, nDims >

Derived sensory class for computing image AvgIntensity.

A derived class of the sensory ABC, this class calculates the AvgIntensity of the image and returns it in the form of TAvgIntensityImage. Calculations are performed using ITK's AvgIntensityMagnitudeRecursiveGaussianImageFilter, and AvgIntensityRecursiveGaussianImageFilter, the later of which provides the derivatives along each direction.

In order to run the sensor one must use its publicly defined sensorIn and sensorOut types to create the input arguments and receive the output. In this case the sensor takes a pointer to the input image and the standard deviation to be used for smoothing, and outputs a pointer to the AvgIntensity image.

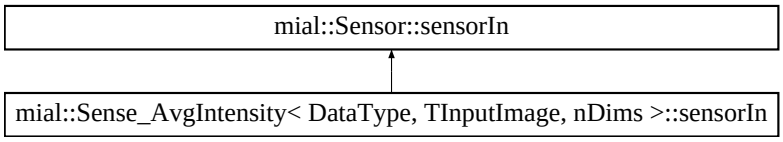
Definition at line 28 of file Sense_AvgIntensity.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/source/sensory/Sense_AvgIntensity.h
- C:/cmcintos/defOrgs/source/sensory/Sense_AvgIntensity.cxx

5.65 mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorIn Struct Reference

Inheritance diagram for mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorIn::



Public Types

- typedef sensorIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef Geometric< DataType, nDims > GeometricType

Public Attributes

- InputImagePointer imageIn
- GeometricType::Pointer geom

Protected Member Functions

- sensorIn ()

5.65.1 Detailed Description

template<class DataType, class TInputImage, int nDims> struct mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorIn

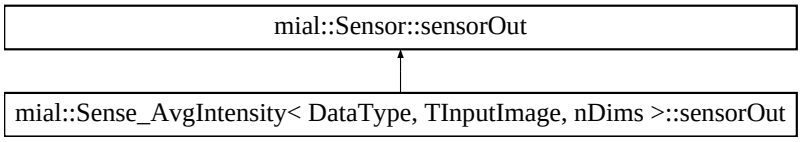
Definition at line 48 of file Sense_AvgIntensity.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/sensory/Sense_AvgIntensity.h

5.66 mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorOut Struct Reference

Inheritance diagram for mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorOut::



Public Types

- typedef sensorOut Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Attributes

- DataType avgIntensity

Protected Member Functions

- sensorOut ()

5.66.1 Detailed Description

template<class DataType, class TInputImage, int nDims> struct mial::Sense_AvgIntensity< DataType, TInputImage, nDims >::sensorOut

Definition at line 64 of file Sense_AvgIntensity.h.

The documentation for this struct was generated from the following file:

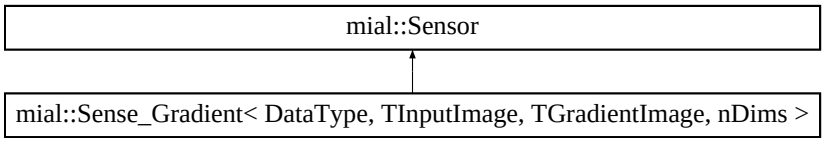
- C:/cmcintos/defOrgs/source/sensory/Sense_AvgIntensity.h

5.67 mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims > Class Template Reference

Derived sensory class for computing image gradient.

`#include <Sense_Gradient.h>`

Inheritance diagram for mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::



Public Types

- typedef Sense_Gradient Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer
- typedef TInputImage InputImageType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef itk::Image< DataType, nDims > OutputImageType
- typedef OutputImageType::ConstPointer OutputImagePointer
- typedef TGradientImage GradientImageType
- typedef GradientImageType::Pointer GradientImagePointer

Public Member Functions

- virtual void run (typename Sensor::sensorIn *const i)

  Pure virtual public member function. New sensors operations are defined and run by implementing this method in a derived class.

Protected Member Functions

- Sense_Gradient ()

Classes

- struct sensorIn
- struct sensorOut

5.67.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> class mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >

Derived sensory class for computing image gradient.

A derived class of the sensory ABC, this class calculates the gradient of the image and returns it in the form of TGradientImage. Calculations are performed using ITK's GradientMagnitudeRecursiveGaussian-ImageFilter, and GradientRecursiveGaussianImageFilter, the later of which provides the derivatives along each direction.

In order to run the sensor one must use its publicly defined sensorIn and sensorOut types to create the input arguments and receive the output. In this case the sensor takes a pointer to the input image and the standard deviation to be used for smoothing, and outputs a pointer to the gradient image.

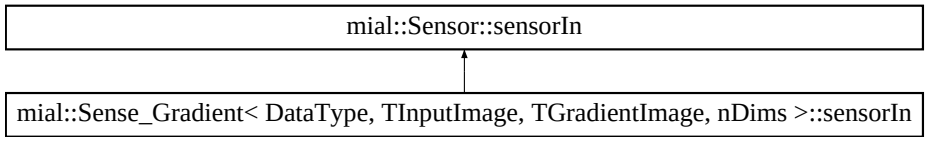
Definition at line 28 of file Sense_Gradient.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/source/sensory/Sense_Gradient.h
- C:/cmcintos/defOrgs/source/sensory/Sense_Gradient.cxx

5.68 mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorIn Struct Reference

Inheritance diagram for mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorIn::



Public Types

- typedef sensorIn Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Attributes

- InputImagePointer imageIn
- DataType sigma

Protected Member Functions

- sensorIn ()

5.68.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> struct mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorIn

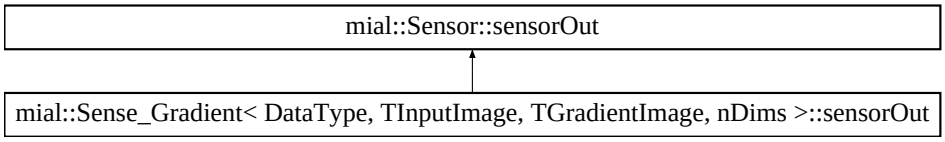
Definition at line 51 of file Sense_Gradient.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/sensory/Sense_Gradient.h

5.69 mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorOut Struct Reference

Inheritance diagram for mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorOut::



Public Types

- typedef sensorOut Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer
- typedef itk::WeakPointer< const Self > ConstWeakPointer

Public Attributes

- GradientImagePointer imageOut

Protected Member Functions

- sensorOut ()

5.69.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> struct mial::Sense_Gradient< DataType, TInputImage, TGradientImage, nDims >::sensorOut

Definition at line 66 of file Sense_Gradient.h.

The documentation for this struct was generated from the following file:

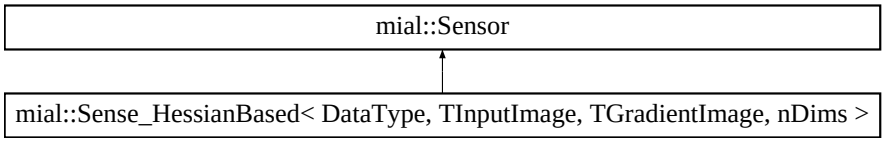
- C:/cmcintos/defOrgs/source/sensory/Sense_Gradient.h

5.70 mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims > Class Template Reference

Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

`#include <Sense_HessianBased.h>`

Inheritance diagram for mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::



Public Types

- typedef TInputImage InputImageType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef itk::Image< DataType, nDims > OutputImageType
- typedef OutputImageType::ConstPointer OutputImagePointer
- typedef TGradientImage GradientImageType
- typedef GradientImageType::Pointer GradientImagePointer
- typedef In sensorIn
- typedef Out sensorOut

Public Member Functions

- Sense_HessianBased ()
- virtual void run (void *i)
- virtual void * getOuput ()

Classes

- struct In
- struct Out

5.70.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> class mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >

Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

A derived class of the sensory ABC, this class locates a possible centroid of the vessel immediately infront of the crawler.

In order to run the sensor one must use its publicly defined sensorIn and sensorOut types to create the input arguments and receive the output.

For details on this sensor see [1].

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

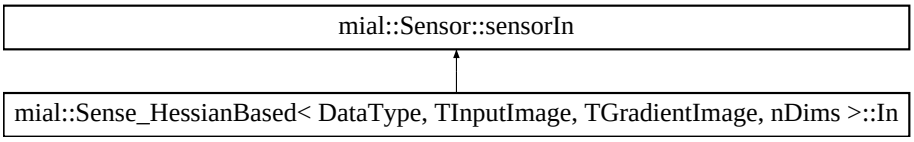
Definition at line 27 of file Sense_HessianBased.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_HessianBased.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_HessianBased.cxx

5.71 mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::In Struct Reference

Inheritance diagram for mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::In::



Public Attributes

- InputImagePointer imageIn
- DataType sigma

5.71.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> struct mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::In

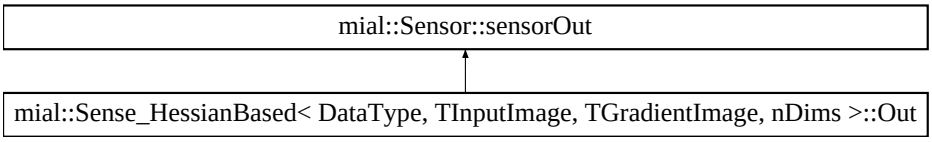
Definition at line 44 of file Sense_HessianBased.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_HessianBased.h

5.72 mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::Out Struct Reference

Inheritance diagram for mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::Out::



Public Attributes

- GradientImagePointer imageOut

5.72.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> struct mial::Sense_HessianBased< DataType, TInputImage, TGradientImage, nDims >::Out

Definition at line 51 of file Sense_HessianBased.h.

The documentation for this struct was generated from the following file:

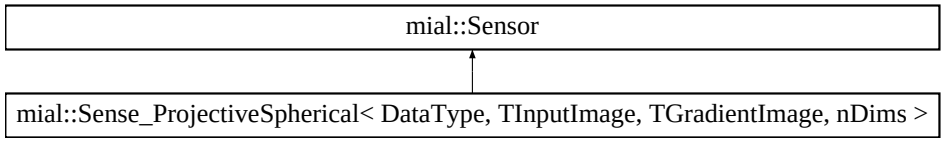
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_HessianBased.h

5.73 mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims > Class Template Reference

Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

`#include <Sense_ProjectiveSpherical.h>`

Inheritance diagram for mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::



Public Types

- typedef TInputImage InputImageType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef itk::Image< DataType, nDims > OutputImageType
- typedef OutputImageType::ConstPointer OutputImagePointer
- typedef TGradientImage GradientImageType
- typedef GradientImageType::Pointer GradientImagePointer
- typedef In sensorIn
- typedef Out sensorOut

Public Member Functions

- Sense_ProjectiveSpherical ()
- virtual void run (void *i)
- virtual void * getOuput ()

Classes

- struct In
- struct Out

5.73.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> class mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >

Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

A derived class of the sensory ABC, this class locates a possible centroid of the vessel immediately infront of the crawler.

In order to run the sensor one must use its publicly defined sensorIn and sensorOut types to create the input arguments and receive the output.

For details on this sensor see [1].

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

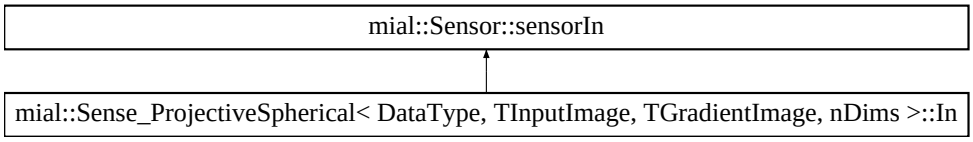
Definition at line 27 of file Sense_ProjectiveSpherical.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_ProjectiveSpherical.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_ProjectiveSpherical.cxx

5.74 mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::In Struct Reference

Inheritance diagram for mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::In::



Public Attributes

- InputImagePointer imageIn
- DataType sigma

5.74.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> struct mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::In

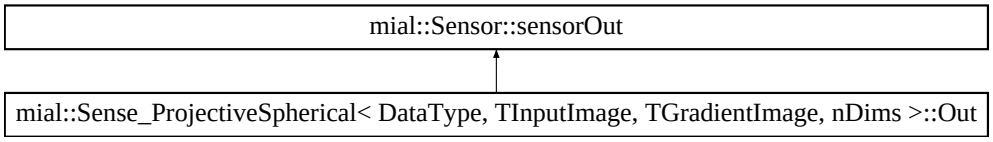
Definition at line 44 of file Sense_ProjectiveSpherical.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_ProjectiveSpherical.h

5.75 mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::Out Struct Reference

Inheritance diagram for mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::Out::



Public Attributes

- GradientImagePointer imageOut

5.75.1 Detailed Description

template<class DataType, class TInputImage, class TGradientImage, int nDims> struct mial::Sense_ProjectiveSpherical< DataType, TInputImage, TGradientImage, nDims >::Out

Definition at line 51 of file Sense_ProjectiveSpherical.h.

The documentation for this struct was generated from the following file:

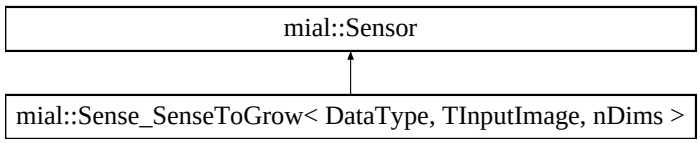
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_ProjectiveSpherical.h

5.76 mial::Sense_SenseToGrow< DataType, TInputImage, nDims > Class Template Reference

Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

`#include <Sense_SenseToGrow.h>`

Inheritance diagram for mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::



Public Types

- typedef TInputImage InputImageType
- typedef InputImageType::ConstPointer InputImagePointer
- typedef itk::Image< DataType, nDims > OutputImageType
- typedef OutputImageType::ConstPointer OutputImagePointer
- typedef TGradientImage GradientImageType
- typedef GradientImageType::Pointer GradientImagePointer
- typedef In sensorIn
- typedef Out sensorOut

Public Member Functions

- Sense_SenseToGrow ()
- virtual void run (void *i)
- virtual void * getOuput ()

Classes

- struct In
- struct Out

5.76.1 Detailed Description

template<class DataType, class TInputImage, int nDims> class mial::Sense_SenseToGrow< DataType, TInputImage, nDims >

Derived sensory class for computing the next location a crawler should grow to [1]. i.e. the centroid of the current vessel infront of the crawler's leading most layer.

A derived class of the sensory ABC, this class locates the next locates the centroid of the vessel immediately infront of the crawler.

In order to run the sensor one must use its publicly defined sensorIn and sensorOut types to create the input arguments and receive the output.

For details on this sensor see [1].

[1] C. McIntosh and G. Hamarneh, "Vessel Crawlers: 3D Physically-based Deformable Organisms for Segmentation and Analysis of Tubular Structures in Medical Images", IEEE Conference on Computer Vision and Pattern Recognition, 2006.

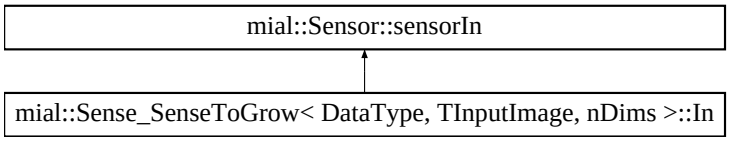
Definition at line 25 of file Sense_SenseToGrow.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_SenseToGrow.h
- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_SenseToGrow.cxx

5.77 mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::In Struct Reference

Inheritance diagram for mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::In::



Public Attributes

- InputImagePointer imageIn

  The image to run on.

5.77.1 Detailed Description

template<class DataType, class TInputImage, int nDims> struct mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::In

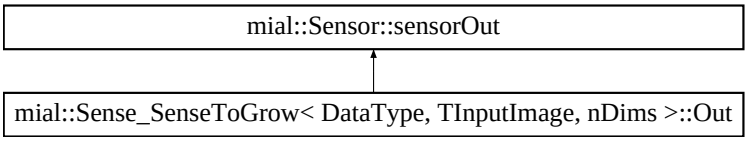
Definition at line 42 of file Sense_SenseToGrow.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_SenseToGrow.h

5.78 mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::Out Struct Reference

Inheritance diagram for mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::Out::



Public Attributes

- DataType[nDims] growLocation

5.78.1 Detailed Description

template<class DataType, class TInputImage, int nDims> struct mial::Sense_SenseToGrow< DataType, TInputImage, nDims >::Out

Definition at line 49 of file Sense_SenseToGrow.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/vesselCrawler/source/Sense_SenseToGrow.h

5.79 mial::Sensor Class Reference

Sensors provide the organism with its view of the world.

`#include <Sensor.h>`

Inheritance diagram for mial::Sensor::

Public Types

- typedef Sensor Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer

Public Member Functions

- virtual void run (sensorIn *const i)=0

  Pure virtual public member function. New sensors operations are defined and run by implementing this method in a derived class.

- virtual sensorOut::Pointer getOutput ()

Protected Member Functions

- Sensor ()

Protected Attributes

- sensorOut::Pointer sensorOutput

Classes

- struct sensorIn

  A structure defining the inputs of a sensor.

- struct sensorOut

  A structure defining the output of a sensor.

5.79.1 Detailed Description

Sensors provide the organism with its view of the world.

Sensors provide the deformable organisms with their view of the world. They provide a means by which to gather statistics and characteristics of its own geometry and the world in which it resides.

In order to run a sensor one must use its publicly defined sensorIn and sensorOut types to create the input arguments and receive the output. This allows maximum flexibility in the parameters a sensor can have, while still enabling any sensor to be ran abstractly.

Definition at line 25 of file Sensor.h.

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/source/sensory/abc/Sensor.h

5.80 mial::Sensor::sensorIn Struct Reference

A structure defining the inputs of a sensor.

`#include <Sensor.h>`

Inheritance diagram for mial::Sensor::sensorIn::

Public Types

- typedef sensorIn Self
- typedef itk::SmartPointer< Self > Pointer

Public Member Functions

- sensorIn (const sensorIn &)

Protected Member Functions

- sensorIn ()

5.80.1 Detailed Description

A structure defining the inputs of a sensor.

Since structures support public inheritance derived class must inherit from this class in their definitions of sensorIn.

Definition at line 39 of file Sensor.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/sensory/abc/Sensor.h

5.81 mial::Sensor::sensorOut Struct Reference

A structure defining the output of a sensor.

`#include <Sensor.h>`

Inheritance diagram for mial::Sensor::sensorOut::

Public Types

- typedef sensorOut Self
- typedef itk::SmartPointer< Self > Pointer
- typedef itk::SmartPointer< const Self > ConstPointer

Protected Member Functions

- sensorOut ()

5.81.1 Detailed Description

A structure defining the output of a sensor.

Since structures support public inheritance derived class must inherit from this class in their definitions of sensorOut.

Definition at line 54 of file Sensor.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/sensory/abc/Sensor.h

5.82 mial::SOViewerDescriptor Struct Reference

`#include <vtkDefOrgViewerWithKWState.h>`

Public Types

- typedef sov::VTKRenderer3D::Pointer SOViewerPointer

Public Member Functions

- SOViewerDescriptor (SOViewerPointer _theSoViewerPointer, vtkKWRenderWidget *_theRenderWidget)
- SOViewerDescriptor ()
- ~SOViewerDescriptor ()

Public Attributes

- SOViewerPointer theSoViewerPointer
- vtkKWRenderWidget * theRenderWidget

5.82.1 Detailed Description

Structure for storing the correspondance between SOViewer and RenderWidget. Each SOViewer has a RenderWidget that it is associated with

Definition at line 24 of file vtkDefOrgViewerWithKWState.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKWState.h

5.83 mial::SpatialObjectDescriptorStruct Struct Reference

`#include <DefOrgViewerAdapterBase.h>`

Public Member Functions

- SpatialObjectDescriptorStruct (itkScenePointer _theItkScene, bool _displayInSeparateFrame, bool _isModified=true)
- SpatialObjectDescriptorStruct ()

Public Attributes

- itkScenePointer theItkScene
- bool isModified
- bool displayInSeparateFrame

5.83.1 Detailed Description

Structure for storing spatial object descriptions. Viewer assigns a ITKScene to be used by the adapter, the adapter populat the scene. Modified field must be set if the scene is changed. displayInSeparateFrame is currently ignored.

Definition at line 88 of file DefOrgViewerAdapterBase.h.

The documentation for this struct was generated from the following file:

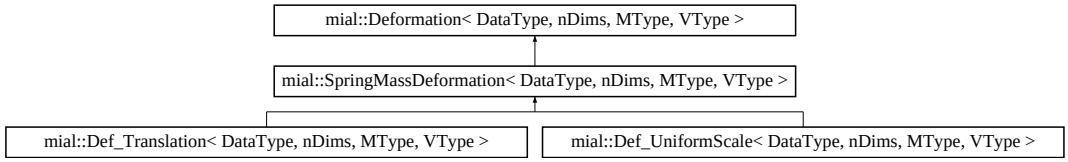
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBase.h

5.84 mial::SpringMassDeformation< DataType, nDims, MType, VType > Class Template Reference

This class extends the basic deformation class with specific functionality for spring mass systems.

`#include <SpringMassDeformation.h>`

Inheritance diagram for mial::SpringMassDeformation< DataType, nDims, MType, VType >::



Public Types

- typedef MType MatrixType

  Public typedef for the internal matrix type.

- typedef VType VectorType

  Public typedef for the internal vector type.

Classes

- struct DefArgSet

  A customized argument set.

5.84.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> class mial::SpringMassDeformation< DataType, nDims, MType, VType >

This class extends the basic deformation class with specific functionality for spring mass systems.

All spring-mass deformations should inherit from this class.

Parameters:

DataType the type of container

nDims the dimensionality of the deformation

MType The matrix type used

VType The vector type used

Definition at line 21 of file SpringMassDeformation.h.

The documentation for this class was generated from the following file:

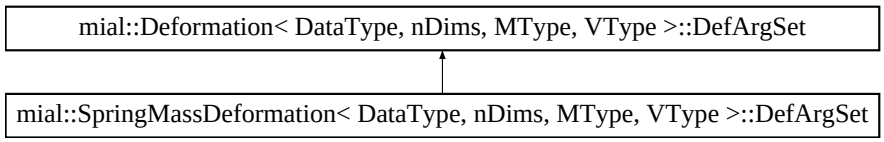
- C:/cmcintos/defOrgs/source/physical/abc/SpringMassDeformation.h

5.85 mial::SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet Struct Reference

A customized argument set.

`#include <SpringMassDeformation.h>`

Inheritance diagram for mial::SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet::



Public Attributes

- MatrixType * nodes
  The nodes of the physics layer.
- MatrixType * nodesV
  The velocities of the physics layer.
- MatrixType * nodesF
  The forces of the physics layer.
- MatrixType * nodesFDef
  The deformation forces of the physics layer.
- VectorType * springsRest
  The spring rest lengths of the physics layer.
- MatrixType * springsNodes
  The spring node connections of the physics layer.
- VectorType * springLengths
  The spring lengths of the physics layer.

5.85.1 Detailed Description

template<class DataType, int nDims, class MType = vnl_matrix<DataType>, class VType = vnl_vector<DataType>> struct mial::SpringMassDeformation< DataType, nDims, MType, VType >::DefArgSet

A customized argument set.

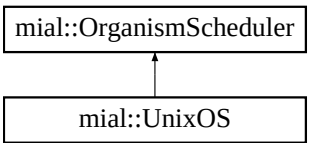
Definition at line 28 of file SpringMassDeformation.h.

The documentation for this struct was generated from the following file:

- C:/cmcintos/defOrgs/source/physical/abc/SpringMassDeformation.h

5.86 mial::UnixOS Class Reference

Inheritance diagram for mial::UnixOS::



5.86.1 Detailed Description

Definition at line 13 of file UnixOS.h.

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/source/organismScheduler/UnixOS.h

5.87 VistVTKCellsClass Class Reference

Public Types

- typedef itk::CellInterface< DefOrgViewer::MeshType::PixelType, DefOrgViewer::MeshType::CellTraits > CellInterfaceType
- typedef itk::TriangleCell< CellInterfaceType > floatTriangleCell
- typedef itk::QuadrilateralCell< CellInterfaceType > floatQuadrilateralCell
- typedef itk::CellInterface< DefOrgViewerAdapter::MeshType::PixelType, DefOrgViewerAdapter::MeshType::CellTraits > CellInterfaceType
- typedef itk::TriangleCell< CellInterfaceType > floatTriangleCell
- typedef itk::QuadrilateralCell< CellInterfaceType > floatQuadrilateralCell

Public Member Functions

- void SetCellArray (vtkCellArray *a)
- void SetCellCounter (int *i)
- void SetTypeArray (int *i)
- void Visit (unsigned long, floatTriangleCell *t)
- void Visit (unsigned long, floatQuadrilateralCell *t)
- void SetCellArray (vtkCellArray *a)
- void SetCellCounter (int *i)
- void SetTypeArray (int *i)
- void Visit (unsigned long, floatTriangleCell *t)
- void Visit (unsigned long, floatQuadrilateralCell *t)

5.87.1 Detailed Description

Definition at line 210 of file DefOrgViewer.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/DefOrgViewer.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapter.h

5.88 mial::VistVTKCellsClass< MESHTYPE > Class Template Reference

`#include <DefOrgViewerAdapterBaseTemplated.h>`

Public Types

- typedef itk::CellInterface< typename MESHTYPE::PixelType, typename MESHTYPE::CellTraits > CellInterfaceType
- typedef itk::TriangleCell< CellInterfaceType > floatTriangleCell
- typedef itk::QuadrilateralCell< CellInterfaceType > floatQuadrilateralCell

Public Member Functions

- void SetCellArray (vtkCellArray *a)
- void SetCellCounter (int *i)
- void SetTypeArray (int *i)
- void Visit (unsigned long, floatTriangleCell *t)
- void Visit (unsigned long, floatQuadrilateralCell *t)

5.88.1 Detailed Description

template<class MESHTYPE> class mial::VistVTKCellsClass< MESHTYPE >

Internal class for MeshToUnstructuredGrid. Do not use

Definition at line 86 of file DefOrgViewerAdapterBaseTemplated.h.

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/DefOrgViewerAdapterBaseTemplated.h

5.89 vtkDefOrgViewerWithKW Class Reference

`#include <vtkDefOrgViewerWithKW.h>`

Public Member Functions

- int Run (int argc, char *argv[ ])
- void InitializeState (vtkKWWindow *win)
- virtual void SetSliceFromScaleCallback (double value)
- virtual void SetSliceCallback (int slice)
- virtual int GetSliceCallback ()
- virtual int GetSliceMinCallback ()
- virtual int GetSliceMaxCallback ()
- virtual void SetSliceOrientationToXYCallback ()
- virtual void SetSliceOrientationToXZCallback ()
- virtual void SetSliceOrientationToYZCallback ()
- virtual void WindowLevelPresetApplyCallback (int id)
- virtual void WindowLevelPresetAddCallback ()
- virtual void WindowLevelPresetUpdateCallback (int id)
- virtual void WindowLevelPresetHasChangedCallback (int id)
- virtual void LoadImageDialogCallback ()
- virtual void LoadScheduleDialogCallback ()
- virtual void LoadMeshDialogCallback ()
- virtual void LoadDefOrgDialogCallback ()
- virtual void InitOrganismButtonCallback ()
- virtual void SimulateOrganismButtonCallback ()
- virtual void StepOrganismButtonCallback ()
- virtual void StepOrganism ()
- virtual void SendOrganismMessageCallback ()
- virtual void UpdateOrganismTextOutputCallback ()
- virtual void GenerateDefOrgDialogCallback ()
- virtual void StopSimulation ()
- virtual void SetVolumeRenderingCallback (int id)
- virtual void SetVolumeRenderingControlBoxCallback (char *val)
- virtual void SetSOColorCallback (double h, double s, double v)
- virtual void LayerComboCallback (int id, const char *choice)
- virtual void ForcePrimaryRender ()
- virtual int NoOp ()
- virtual void NoOp2 (int i)
- DefOrgViewerAdapterBase * GetCurrDefOrg ()

Public Attributes

- bool m_updateOrganismResults

Protected Member Functions

- vtkDefOrgViewerWithKW ()
- void SetupImageViewerPipeline ()
- void UpdateImageViewers (bool flushAll=false)
- void CreateImageDataFrame (int imagePageCookie, vtkKWWindow ∗win)
- void CreateVolumeRenderingFrame (int volumeRenderingPageCookie, vtkKWWindow ∗win)
- void CreateOrganismDataFrame (int organismPageCookie, vtkKWWindow ∗win)
- void CreateSpatialObjectFrame (int polyPageCookie, vtkKWWindow ∗win)
- void AddActorsFromSOViewers (bool flushAll=false)
- void RemoveActorsFromSOViewers (bool flushAll=false)
- void SetScenesToSOViewers (bool flushAll=false)
- void RenderSOViewers (bool flushAll=false)
- virtual void UpdateSliceScale ()
- ∼vtkDefOrgViewerWithKW ()

Protected Attributes

- vtkKWScale ∗ SliceScale
- vtkKWWindowLevelPresetSelector ∗ WindowLevelPresetSelector
- vtkKWRenderWidget ∗ RenderWidget
- vtkKWLoadSaveButton ∗ LoadImageButton
- vtkKWLoadSaveButton ∗ LoadScheduleButton
- vtkKWLoadSaveButton ∗ LoadMeshButton
- vtkKWLoadSaveButton ∗ LoadDefOrgButton
- vtkKWPushButton ∗ InitOrganismButton
- vtkKWPushButton ∗ SimulateOrganismButton
- vtkKWPushButton ∗ StepOrganismButton
- vtkKWPushButton ∗ SendOrganismMessageButton
- vtkKWTextWithScrollbars ∗ OrganismOutputTextbox
- vtkKWTextWithScrollbars ∗ OrganismInputTextbox
- vtkKWVolumePropertyWidget ∗ VRVolumePropertyWidget
- vtkVolume ∗∗ RenderVolume
- vtkKWSurfaceMaterialPropertyWidget ∗ SOMatieralPropertyWidget
- vtkKWHSVColorSelector ∗ SOColorWidget
- vtkKWWindow ∗ MainWindow
- vtkKWTopLevel ∗∗ viewerWindows
- vtkKWFrameWithLabel ∗ OrganismTextIOFrame
- vtkBoxWidget ∗ MeshBoxWidget
- vtkKWCheckButton ∗ VolumeRenderingCheckButton
- int ∗ VolumeRenderToggles
- vtkMultiThreader ∗ m_threader
- int m_currThreadID
- bool m_organismRunning

5.89.1 Detailed Description

Core viewer class for handling UI interfaces

Definition at line 44 of file vtkDefOrgViewerWithKW.h.

5.89.2 Member Function Documentation

5.89.2.1 void vtkDefOrgViewerWithKW::ForcePrimaryRender () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 295 of file vtkDefOrgViewerWithKW.cxx.

References RenderWidget.

5.89.2.2 void vtkDefOrgViewerWithKW::GenerateDefOrgDialogCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1103 of file vtkDefOrgViewerWithKW.cxx.

References vtkGenerateDefOrgDialog::Create(), vtkGenerateDefOrgDialog::Invoke(), and MainWindow.

5.89.2.3 int vtkDefOrgViewerWithKW::GetSliceCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 320 of file vtkDefOrgViewerWithKW.cxx.

5.89.2.4 int vtkDefOrgViewerWithKW::GetSliceMaxCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 338 of file vtkDefOrgViewerWithKW.cxx.

5.89.2.5 int vtkDefOrgViewerWithKW::GetSliceMinCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 329 of file vtkDefOrgViewerWithKW.cxx.

5.89.2.6 void vtkDefOrgViewerWithKW::InitOrganismButtonCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 978 of file vtkDefOrgViewerWithKW.cxx.

References GetCurrDefOrg(), InitOrganismButton, mial::DefOrgViewerAdapterBase::m_PropertyBag, mial::DefOrgViewerAdapterBase::SetupOrganism(), SimulateOrganismButton, StepOrganismButton, and UpdateOrganismTextOutputCallback().

5.89.2.7 void vtkDefOrgViewerWithKW::LoadDefOrgDialogCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1123 of file vtkDefOrgViewerWithKW.cxx.

References CreateOrganismDataFrame(), CreateSpatialObjectFrame(), CreateVolumeRenderingFrame(), InitializeState(), mial::vtkDefOrgViewerWithKWState::isDefOrgSet(), MainWindow, and RenderWidget.

5.89.2.8 void vtkDefOrgViewerWithKW::LoadImageDialogCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1059 of file vtkDefOrgViewerWithKW.cxx.

References CreateImageDataFrame(), GetCurrDefOrg(), LoadImageButton, mial::DefOrgViewerAdapterBase::m_OutputImages, MainWindow, mial::DefOrgViewerAdapterBase::PopulateVtkImage(), RenderWidget, SetupImageViewerPipeline(), SimulateOrganismButton, StepOrganismButton, StopSimulation(), and UpdateImageViewers().

5.89.2.9 void vtkDefOrgViewerWithKW::LoadMeshDialogCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1152 of file vtkDefOrgViewerWithKW.cxx.

References AddActorsFromSOViewers(), GetCurrDefOrg(), LoadMeshButton, mial::DefOrgViewerAdapterBase::MaxNumberOfOutputItkSpatialObjects(), mial::DefOrgViewerAdapterBase::PopulateItkScene(), RemoveActorsFromSOViewers(), RenderSOViewers(), SetScenesToSOViewers(), mial::vtkDefOrgViewerWithKWState::SetSOViewer(), SimulateOrganismButton, StepOrganismButton, and StopSimulation().

5.89.2.10 void vtkDefOrgViewerWithKW::LoadScheduleDialogCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1088 of file vtkDefOrgViewerWithKW.cxx.

References GetCurrDefOrg(), LoadScheduleButton, SimulateOrganismButton, StepOrganismButton, and StopSimulation().

5.89.2.11 int vtkDefOrgViewerWithKW::NoOp () [virtual]

Internal callback. It is public for TCL wrapping. Do not use. Hack for animation widget to record deformation instead

Definition at line 829 of file vtkDefOrgViewerWithKW.cxx.

5.89.2.12 void vtkDefOrgViewerWithKW::NoOp2 (int *i*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use Hack for animation widget to record deformation instead

Definition at line 833 of file vtkDefOrgViewerWithKW.cxx.

References StepOrganismButtonCallback().

5.89.2.13 void vtkDefOrgViewerWithKW::SendOrganismMessageCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 918 of file vtkDefOrgViewerWithKW.cxx.

References GetCurrDefOrg(), OrganismInputTextbox, and mial::DefOrgViewerAdapterBase::theDefOrgTextInput.

5.89.2.14 void vtkDefOrgViewerWithKW::SetSliceCallback (int *slice*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 311 of file vtkDefOrgViewerWithKW.cxx.

References RenderWidget.

5.89.2.15 void vtkDefOrgViewerWithKW::SetSliceFromScaleCallback (double *value*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 301 of file vtkDefOrgViewerWithKW.cxx.

References RenderWidget.

5.89.2.16 void vtkDefOrgViewerWithKW::SetSliceOrientationToXYCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 357 of file vtkDefOrgViewerWithKW.cxx.

References UpdateSliceScale().

5.89.2.17 void vtkDefOrgViewerWithKW::SetSliceOrientationToXZCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 366 of file vtkDefOrgViewerWithKW.cxx.

References UpdateSliceScale().

5.89.2.18 void vtkDefOrgViewerWithKW::SetSliceOrientationToYZCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 375 of file vtkDefOrgViewerWithKW.cxx.

References UpdateSliceScale().

5.89.2.19 void vtkDefOrgViewerWithKW::SetSOColorCallback (double *h*, double *s*, double *v*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 434 of file vtkDefOrgViewerWithKW.cxx.

References mial::vtkDefOrgViewerWithKWState::GetCurrentSOViewer(), mial::vtkDefOrgViewerWithKWState::GetCurrentSOViewerRenderWidget(), and SOMatieralPropertyWidget.

5.89.2.20 void vtkDefOrgViewerWithKW::SetVolumeRenderingCallback (int *id*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1301 of file vtkDefOrgViewerWithKW.cxx.

References GetCurrDefOrg(), mial::vtkDefOrgViewerWithKWState::GetImageViewerRenderWidget(), RenderVolume, and VolumeRenderToggles.

5.89.2.21 void vtkDefOrgViewerWithKW::SetVolumeRenderingControlBoxCallback (char * *val*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1279 of file vtkDefOrgViewerWithKW.cxx.

References GetCurrDefOrg(), mial::DefOrgViewerAdapterBase::MaxNumberOfOutputImages(), viewerWindows, VolumeRenderingCheckButton, and VolumeRenderToggles.

5.89.2.22 void vtkDefOrgViewerWithKW::SimulateOrganismButtonCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 1016 of file vtkDefOrgViewerWithKW.cxx.

References m_organismRunning, SimulateOrganismButton, StepOrganism(), StepOrganismButton, and StopSimulation().

5.89.2.23 void vtkDefOrgViewerWithKW::StepOrganism () [virtual]

Internal callback. It is public for TCL wrapping. Do not use. Call run on the organism asynchronously

Definition at line 945 of file vtkDefOrgViewerWithKW.cxx.

References AddActorsFromSOViewers(), m_currThreadID, m_threader, m_updateOrganismResults, RemoveActorsFromSOViewers(), RenderSOViewers(), SetScenesToSOViewers(), UpdateImageViewers(), and UpdateOrganismTextOutputCallback().

Referenced by SimulateOrganismButtonCallback().

5.89.2.24 void vtkDefOrgViewerWithKW::StepOrganismButtonCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use. Call run on the organism synchronously.

Definition at line 966 of file vtkDefOrgViewerWithKW.cxx.

References AddActorsFromSOViewers(), GetCurrDefOrg(), RemoveActorsFromSOViewers(), RenderSOViewers(), SetScenesToSOViewers(), UpdateImageViewers(), mial::DefOrgViewerAdapterBase::UpdateOrganism(), and UpdateOrganismTextOutputCallback().

Referenced by NoOp2().

5.89.2.25 void vtkDefOrgViewerWithKW::StopSimulation () [virtual]

Internal callback. It is public for TCL wrapping. Do not use. Helper method. Also see SimulateOrganismButtonCallback

Definition at line 1005 of file vtkDefOrgViewerWithKW.cxx.

References m_currThreadID, m_organismRunning, m_threader, m_updateOrganismResults, and SimulateOrganismButton.

Referenced by InitializeState(), LoadImageDialogCallback(), LoadMeshDialogCallback(), LoadScheduleDialogCallback(), and SimulateOrganismButtonCallback().

5.89.2.26 void vtkDefOrgViewerWithKW::UpdateOrganismTextOutputCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 924 of file vtkDefOrgViewerWithKW.cxx.

References GetCurrDefOrg(), OrganismOutputTextbox, OrganismTextIOFrame, and mial::DefOrgViewerAdapterBase::theDefOrgTextOutput.

Referenced by InitOrganismButtonCallback(), StepOrganism(), and StepOrganismButtonCallback().

5.89.2.27 void vtkDefOrgViewerWithKW::WindowLevelPresetAddCallback () [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 399 of file vtkDefOrgViewerWithKW.cxx.

5.89.2.28 void vtkDefOrgViewerWithKW::WindowLevelPresetApplyCallback (int *id*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 384 of file vtkDefOrgViewerWithKW.cxx.

References mial::vtkDefOrgViewerWithKWState::GetCurrentImageViewer().

5.89.2.29 void vtkDefOrgViewerWithKW::WindowLevelPresetHasChangedCallback (int *id*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 422 of file vtkDefOrgViewerWithKW.cxx.

Referenced by WindowLevelPresetUpdateCallback().

5.89.2.30 void vtkDefOrgViewerWithKW::WindowLevelPresetUpdateCallback (int *id*) [virtual]

Internal callback. It is public for TCL wrapping. Do not use

Definition at line 408 of file vtkDefOrgViewerWithKW.cxx.

References WindowLevelPresetHasChangedCallback(), and WindowLevelPresetSelector.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKW.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKW.cxx

5.90 mial::vtkDefOrgViewerWithKWState Class Reference

`#include <vtkDefOrgViewerWithKWState.h>`

Public Types

- typedef sov::VTKRenderer3D::Pointer SOViewerPointer

Public Member Functions

- SOViewerPointer GetCurrentSOViewer ()
- vtkKWRenderWidget * GetCurrentSOViewerRenderWidget ()
- SOViewerPointer GetSOViewer (itkScenePointer itkScenePointerKey)
- vtkKWRenderWidget * GetSOViewerRenderWidget (itkScenePointer itkScenePointerKey)
- void SetSOViewerRenderWidget (itkScenePointer itkScenePointerKey, vtkKWRenderWidget *renderWidget)
- void SetSOViewer (itkScenePointer itkScenePointerKey, SOViewerPointer newSOViewerPointer)
- vtkImageViewer2 * GetCurrentImageViewer ()
- vtkKWRenderWidget * GetCurrentImageViewerRenderWidget ()
- vtkImageViewer2 * GetImageViewer (vtkImageImport *vtkImageImportPointerKey)
- vtkKWRenderWidget * GetImageViewerRenderWidget (vtkImageImport *vtkImageImportPointerKey)
- void SetImageViewerRenderWidget (vtkImageImport *vtkImageImportPointerKey, vtkKWRenderWidget *renderWidget)
- void SetImageViewer (vtkImageImport *vtkImageImportPointerKey, vtkImageViewer2 *newImageViewerPointer)
- bool isDefOrgSet ()
- vtkDefOrgViewerWithKWState ()
- ~vtkDefOrgViewerWithKWState ()

Public Attributes

- std::vector< vtkKWTopLevel * > m_OpenedWindows
- std::map< std::string, vtkKWScaleWithEntry * > m_DefOrgPropertyScaleMap
- std::map< itkScenePointer, SOViewerDescriptor > m_OutputSceneSOVMap
- std::map< vtkImageImport *, ImageViewerDescriptor > m_OutputImageViewerMap
- DefOrgViewerAdapterBase * theDefOrg
- int currSOViewerIndex
- int currImageViewerIndex
- DefOrgViewerAdapterDynamicLoader DynamicLoader

5.90.1 Detailed Description

Storing the state relating to the currently loaded organism. Created and destroyed as new organism is loaded

Definition at line 61 of file vtkDefOrgViewerWithKWState.h.

5.90.2 Constructor & Destructor Documentation

5.90.2.1 mial::vtkDefOrgViewerWithKWState::vtkDefOrgViewerWithKWState () [inline]

Constructor initializes currSOViewerIndex to 0

Definition at line 100 of file vtkDefOrgViewerWithKWState.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKWState.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKWState.cxx

5.91 vtkFlRenderWindowInteractor Class Reference

Public Member Functions

- vtkFlRenderWindowInteractor ()
- vtkFlRenderWindowInteractor (int x, int y, int w, int h, const char *l="")
- ~vtkFlRenderWindowInteractor (void)
- void Initialize ()
- void Enable ()
- void Disable ()
- void Start ()
- void SetRenderWindow (vtkRenderWindow *aren)
- void UpdateSize (int x, int y)
- int CreateTimer (int timertype)
- int DestroyTimer ()
- void OnTimer (void)
- void TerminateApp ()

Static Public Member Functions

- static vtkFlRenderWindowInteractor * New ()

Protected Member Functions

- void flush (void)
- void draw (void)
- void resize (int x, int y, int w, int h)
- int handle (int event)

5.91.1 Detailed Description

Definition at line 34 of file vtkFlRenderWindowInteractor.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/vtkFlRenderWindowInteractor.h
- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/vtkFlRenderWindowInteractor.cxx

5.92 vtkGenerateDefOrgDialog Class Reference

`#include <vtkGenerateDefOrgDialog.h>`

Public Member Functions

- void PrintSelf (ostream &os, vtkIndent indent)
- virtual void Create (vtkKWApplication *app)
- virtual void GenerateDefOrgCallBack ()
- virtual void SetTemplateDirectoryLocationCallBack ()
- virtual void GenerateFiles (const char *dirName, const char *className)
- virtual int Invoke ()

Protected Member Functions

- vtkGenerateDefOrgDialog ()
- ~vtkGenerateDefOrgDialog ()

Protected Attributes

- vtkKWEntryWithLabel * LabelClassName
- vtkKWLoadSaveButtonWithLabel * TemplateLocationButton
- vtkKWLoadSaveButtonWithLabel * DirectoryButton
- vtkKWPushButton * GenerateButton

5.92.1 Detailed Description

Constructing a user interface for user to generate DefOrg skelecton codes

Definition at line 15 of file vtkGenerateDefOrgDialog.h.

The documentation for this class was generated from the following files:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkGenerateDefOrgDialog.h
- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkGenerateDefOrgDialog.cxx

5.93 vtkImageImport Class Reference

Public Types

- typedef void(∗) UpdateInformationCallbackType (void ∗)
- typedef int(∗) PipelineModifiedCallbackType (void ∗)
- typedef int ∗(∗) WholeExtentCallbackType (void ∗)
- typedef double ∗(∗) SpacingCallbackType (void ∗)
- typedef double ∗(∗) OriginCallbackType (void ∗)
- typedef const char ∗(∗) ScalarTypeCallbackType (void ∗)
- typedef int(∗) NumberOfComponentsCallbackType (void ∗)
- typedef void(∗) PropagateUpdateExtentCallbackType (void ∗, int ∗)
- typedef void(∗) UpdateDataCallbackType (void ∗)
- typedef int ∗(∗) DataExtentCallbackType (void ∗)
- typedef void ∗(∗) BufferPointerCallbackType (void ∗)

Public Member Functions

- void ∗ GetImportVoidPointer ()
- void SetDataScalarTypeToDouble ()
- void SetDataScalarTypeToFloat ()
- void SetDataScalarTypeToInt ()
- void SetDataScalarTypeToShort ()
- void SetDataScalarTypeToUnsignedShort ()
- void SetDataScalarTypeToUnsignedChar ()
- const char ∗ GetDataScalarTypeAsString ()
- void SetDataExtentToWholeExtent ()

Protected Attributes

- void ∗ ImportVoidPointer
- int SaveUserArray
- int NumberOfScalarComponents
- int DataScalarType
- int WholeExtent [6]
- int DataExtent [6]
- double DataSpacing [3]
- double DataOrigin [3]
- void ∗ CallbackUserData
- UpdateInformationCallbackType UpdateInformationCallback
- PipelineModifiedCallbackType PipelineModifiedCallback
- WholeExtentCallbackType WholeExtentCallback
- SpacingCallbackType SpacingCallback
- OriginCallbackType OriginCallback
- ScalarTypeCallbackType ScalarTypeCallback
- NumberOfComponentsCallbackType NumberOfComponentsCallback

- PropagateUpdateExtentCallbackType PropagateUpdateExtentCallback
- UpdateDataCallbackType UpdateDataCallback
- DataExtentCallbackType DataExtentCallback
- BufferPointerCallbackType BufferPointerCallback

5.93.1 Detailed Description

Definition at line 34 of file vtkImageImport.h.

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewer/Source/vtkImageImport.h

5.94 vtkMyCallback Class Reference

Public Member Functions

- vtkMyCallback ()
- virtual void SetApplicationInstance (vtkDefOrgViewerWithKW *instance)
- virtual void Execute (vtkObject *caller, unsigned long eventId, void *)

Static Public Member Functions

- static vtkMyCallback * New ()

5.94.1 Detailed Description

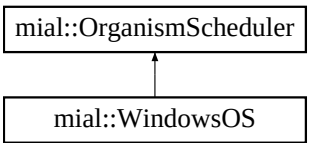
Definition at line 82 of file vtkDefOrgViewerWithKW.cxx.

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/examples/DefOrgViewerWithKW/Source/vtkDefOrgViewerWithKW.cxx

5.95 mial::WindowsOS Class Reference

Inheritance diagram for mial::WindowsOS::



5.95.1 Detailed Description

Definition at line 11 of file WindowsOS.h.

The documentation for this class was generated from the following file:

- C:/cmcintos/defOrgs/source/organismScheduler/WindowsOS.h