VAAPI

Generated by Doxygen 1.8.16

Chapter 1

Visual-auditory API

1.1 Introduction

Scalar fields are used in many research areas, where computer simulations or experimental studies are involved, such as computational chemistry, medical data analysis and physical phenomena studies. Visualisation of scalar fields usually employs Volume Rendering techniques as for the computer systems the scalar fields data is often converted to data volumes stored in the texture memory. The visual analysis of scalar fields is not always straightforward, especially when a complex dynamic phenomenon is represented. In Volume Rendering the Multidimensional Transfer functions (TF) and consideration of more complex object-light interaction processes in the optical model is used to address the issues of visual analysis quality improvement and highlight features of interest.

However, the visual perception limitations not always can be overcome solely with the enhancement of the optical model. The visual system can be overloaded and perturbed due to fatigue. An additional introduction of auditory sensory stimuli to address a problem of visual analysis limitations is a well-known technique, called `sonification`.

This API targets the problems of the visual-auditory interactive study of the dynamic scalar fields using the concept of a heterogeneous object influencing various sensory stimuli. The API takes advantages of the similarities between light and sound propagation to suggest a novel procedure of a visual-auditory rendering based on the ray-casting procedure. It makes it possible to conduct a simultaneous visual analysis along with spatial positioning/measuring. The API suggests rendering and modelling of both surface and solid geometry with visual and auditory properties. However mainly it is concentrated on the Volume scene representation based on the HyperVolume (HV) model and take advantage of the Signed Distance Fields (SDF) for rendering.
The concept introduces three separate parts of the HV model for Volume Rendering that can be evaluated independently on demand to speed up the rendering process of complex dynamic volume objects and a general visual-auditory scene representation for interactive Volume Rendering.

For more details, on heterogeneous objects modelling and HyperVolume approach look at **Approach Overview** (p. 3) Section:

1.2 Optix engine and VTK

The API is based on Optix Engine and takes advantage of it's native rendering acceleration structures like BVH. The API syntacsis is mainly a "copy" of `Visualization Toolkit`. As well it much inherits the standard visualisation concepts reflected in VTK API, like visualisation pipeline, replacing it with notion of visual-auditory pipeline, unified on base of ray-casting procedure. Thus, API targets to provide high level access to geometric modelling and rendering, allowing the researcher or programmer to operate familiar visualisation concepts and terminology. At the same time it is very lightweight and fast as all core procedures are CUDA kernals running on NVIDIA GPU card. The API it mostly oriented to support geometry modelling on base of `FRep concept` and targets various aspects of multisensory interaction, like collision detection, interactive manipulations and etc.

For concept of basic use, similarities and differencec to VTK see **API basics** (p. 13) For FRep modelling example look **FRep tutorial** (p. 5)

1.3 TODO

- Haptic implementation
  - Collision detection
- Practical examples
  - Molecular haptics -Visual-auditory tutorials -Extending API (advanced tutorials)

Chapter 2

Approach overview

A scalar field is a mapping $f : x \to \Re, x \in \Re^n$, which associates any point in space with a scalar value. The conventional Volume Rendering scheme suggests that volume boundary (or scalar field domain) is defined as a bounding box and the ray/volume intersection is computed to define the ray's start and end points to solve the Volume Rendering equation with the ray-marching procedure. In the general case, the scalar field can be defined on an arbitrary domain and the object boundary can have a more complex representation and can change dynamically in time.

Let us consider the case of the molecular structure, that is a result of quantum simulation. The molecule is usually defined with electron density field and electrostatic potential fields. The entire structure can be treated as heterogeneous object, where first field defines the molecule interaction boundary $F(X)$, and the second field represents the physical property, namely the charge distribution $S_1(X)$. We take advantage of the Hyper Volume (HV) model to define the initial structure and the visual-auditory pipeline, where initial data is represented with HV model: $o(t) = (G(t), A_1(t)) : (F(X|t), S_1(X|t))$

,where t - is time parameter as in general case the researcher has to deal with dynamic molecules.

Via application of functional mapping procedures, that define optic and auditory transfer function, we receive a HV representation: $m(t) = (G(t), A_o(t), A_s(t)) : (F(X|t), S_o(X|t), S_s(X|t))$

, that represents an abstract heterogeneous object with visual and auditory properties.

2.1 Rendering optical model and geometry

Optix engine, optical materials

2.2 Auditory rendering

Optix engine, auditory materials

Chapter 3

Tutorial: FRep basics

This tutorial demonstrates basics of geometry creation according to `FRep modelling approach`. At the same time tutorial demonstrates the basics of API use and main concept that is much simillar to `Visualisation Toolkit`. The core of the VTK is a visualisation pipeline of data, that starts from data source and ends up with image rendered on screen. The basic classes of VTK API are:

- Sources
- Filters
- Mappers
- Actors
- Renderers
- Interactors
- Windows

The Visual-auditory API implements basic Mappers, Actors, Renderers and Windows that are intended to support ray-casting of mainly Signed Distance Fields, althought triangulated geometry is also supported and can be integrated within pipeline. The scalar field data is provided by file Readers that eighter output structured data or directly create Optix Buffers and TextureSamplers on GPU.

The GUI procedures are implemented on base of `GLFW` and `ImGUI` libraries. A custom GUI can be easily implemented on base of those libraries and integrated to API pipeline.

3.1 FRep primitive creation and rendering

Below is simple program, creating SDF box primitive. Simillar to VTK API, the Camera, Renderer, GLFW Window and Interactor to process interactive manipulations with scene are created. For syntacsis comparison with simple VTK primitive creation look `wiki`:

```
#include "optixWindow.h"
#include "optixSDFPrimitives.h"
int main(int argc, char *argv[])
{
  int  windowWidth  = 512;
  int  windowHeight = 512;

  PinholeCamera pinholeCamera; //creates basic camera
  contextManager m;
  m.Update();//creates context
  optixRenderer ren;
  ren.SetValid(m.GetValid());
  ren.SetContext(m.GetOutput());//m returns the context
  ren.SetOpticalDims(windowWidth, windowWidth);
  ren.SetCamera(&pinholeCamera);
  //set not dynamic
  ren.SetDynamic(false);
  ren.SetAuditory(false);

  //window procedure
  GLFW_Window optixWindowProc;

  optixWindowProc.SetDim(windowWidth, windowHeight);
  optixWindowProc.SetRenderer(&ren);
  optixWindowProc.SetContext(m.GetOutput());
  optixWindowProc.SetCamera(&pinholeCamera);
  RenderWindowInteractor iren; //TODO always check that basic still works
  iren.SetWindow(&optixWindowProc);

  std::cout « "DONE WITH WINDOW" « std::endl;

  if (iren.SetUp()) //update of WindowProc is successful, we can start scene creation with predefined
       rendering parameters
  {
      std::cout « "START MAIN LOOP" « std::endl;
      //Scene creation
      try
      {


          optixSDFBox sdf;
          sdf.SetContext(m.GetOutput());
          sdf.SetCenter1(optix::make_float3(0));
          sdf.SetDims(optix::make_float3(0.3,1.0,3.0));
          sdf.Update();
          std::cout « "BOX created" « std::endl;
          SDFMaterial mSdf;
          mSdf.SetContext(m.GetOutput());
          mSdf.Update();
          std::cout « "MATERIAL created" « std::endl;
          optixMapper map21;
          map21.SetContext(m.GetOutput());
          map21.SetInput(sdf.GetOutput());
          map21.AddMaterial(mSdf.GetOutput(), mSdf.GetType());
          map21.Update();
          optixSdfActor acSdf1;
          acSdf1.SetContext(m.GetOutput());
          acSdf1.AddMapper(&map21);
          acSdf1.Update();
          ren.AddActor(&acSdf1);
      }
      catch (optix::Exception& e)
      {
          std::cerr « e.getErrorString() « std::endl;
      }
      std::cout « "DONE WITH SCENE" « std::endl;
      //Main loop
      iren.Start();
      std::cout « "MAIN LOOP STARTED" « std::endl;
  }


  return 0;
}
```

The output of the program is:

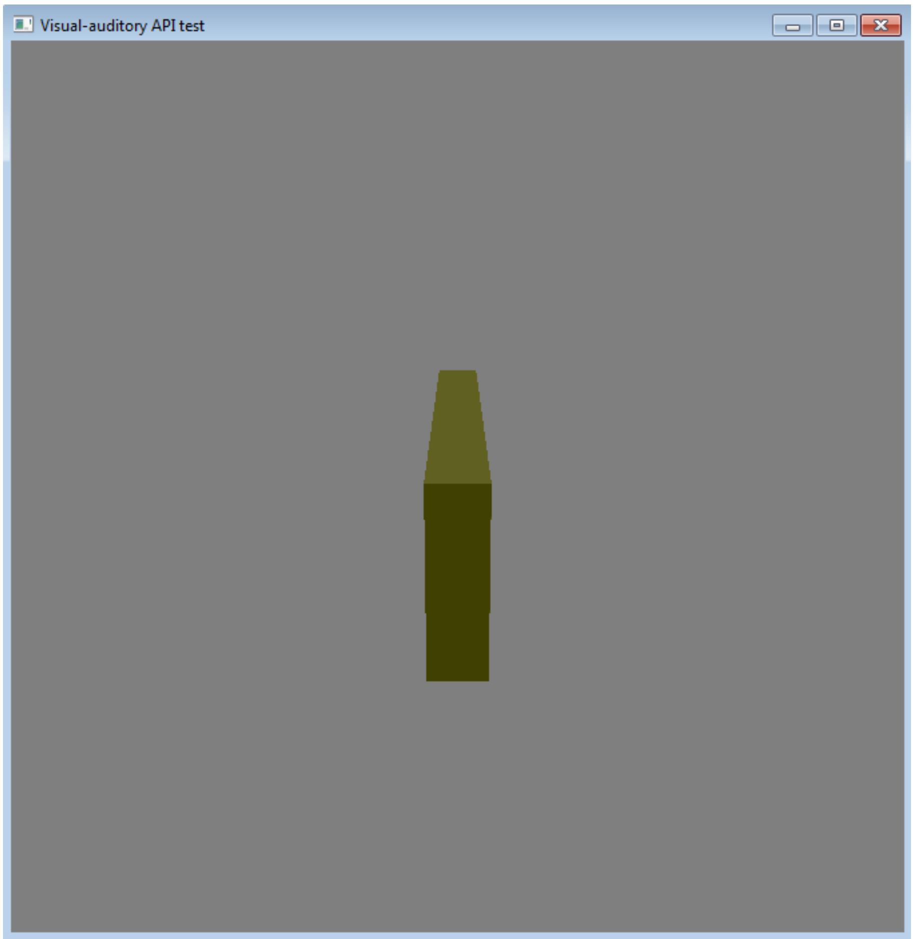


Figure 3.1 Rendered SDF box

A more complex example is below:

```
#include "optixWindow.h"
#include "optixSDFPrimitives.h"
#include "optixSDFOperations.h"
int main(int argc, char *argv[])
{
  int  windowWidth  = 512;
  int  windowHeight = 512;

  PinholeCamera pinholeCamera; //creates basic camera
  contextManager m;
  m.Update();//creates context
  optixRenderer ren;
  ren.SetValid(m.GetValid());
  ren.SetContext(m.GetOutput());
  ren.SetOpticalDims(windowWidth, windowWidth);
  ren.SetCamera(&pinholeCamera);
  //set not dynamic
  ren.SetDynamic(false);
  ren.SetAuditory(false);

  //window procedure
  GLFW_Window optixWindowProc;

  optixWindowProc.SetDim(windowWidth, windowHeight);
  optixWindowProc.SetRenderer(&ren);
  optixWindowProc.SetContext(m.GetOutput()); //returns context
  optixWindowProc.SetCamera(&pinholeCamera);
  RenderWindowInteractor iren; //TODO always check that basic still works
  iren.SetWindow(&optixWindowProc);

  std::cout « "DONE WITH WINDOW" « std::endl;

  if (iren.SetUp()) //update of WindowProc
  {
      std::cout « "START MAIN LOOP" « std::endl;
      //Scene creation
      try
      {


          optixSDFBox sdf;
          sdf.SetContext(m.GetOutput());
          sdf.SetCenter1(optix::make_float3(1.0));
          sdf.SetDims(optix::make_float3(0.3));
          sdf.Update();
          const int nums = 10;
          SDFRoundingOp round[nums];
          round[0].SetContext(m.GetOutput());
          round[0].AddOpperand(&sdf);
          round[0].SetKoeff(0.01);
          round[0].Update();
          for (int i = 1; i < nums; i++)
          {
              round[i].SetContext(m.GetOutput());
              round[i].AddOpperand(round[i - 1].GetOutputSdfObject());// &sdf);
              round[i].SetKoeff(0.01*i);
              round[i].Update();
          }
          optixSDFTorus sdfT;
          sdfT.SetContext(m.GetOutput());
          sdfT.SetCenter1(optix::make_float3(0.0));
          sdfT.SetRadius1(optix::make_float2(0.4, 0.1));
          sdfT.Update();
          SDFElongateOp el;
          el.SetContext(m.GetOutput());
          el.AddOpperand(&sdfT);
          el.SetHKoeff(optix::make_float3(0.0, 1.0, 2.1));
          el.Update();
          SDFBlendUnionOp opBlend;
          opBlend.SetContext(m.GetOutput());
          opBlend.AddOpperand1(&sdf);
          opBlend.AddOpperand2(el.GetOutputSdfObject());
          opBlend.SetKoeff(0.3);
          opBlend.Update();

          SDFMaterial mSdf;
          mSdf.SetContext(m.GetOutput());
          mSdf.Update();
          optixMapper map21;
          map21.SetContext(m.GetOutput());
          map21.SetInput(sdf.GetOutput());
          map21.AddMaterial(mSdf.GetOutput(), mSdf.GetType());
          map21.Update();
          optixSdfActor acSdf1;
          acSdf1.SetContext(m.GetOutput());
          acSdf1.AddMapper(&map21);
          acSdf1.Update();
          ren.AddActor(&acSdf1);
      }
      catch (optix::Exception& e)
      {
          std::cerr « e.getErrorString() « std::endl;
      }
      std::cout « "DONE WITH SCENE" « std::endl;
      //Main loop
      iren.Start();
      std::cout « "MAIN LOOP STARTED" « std::endl;
  }


  return 0;
}
```

The output of the program is :

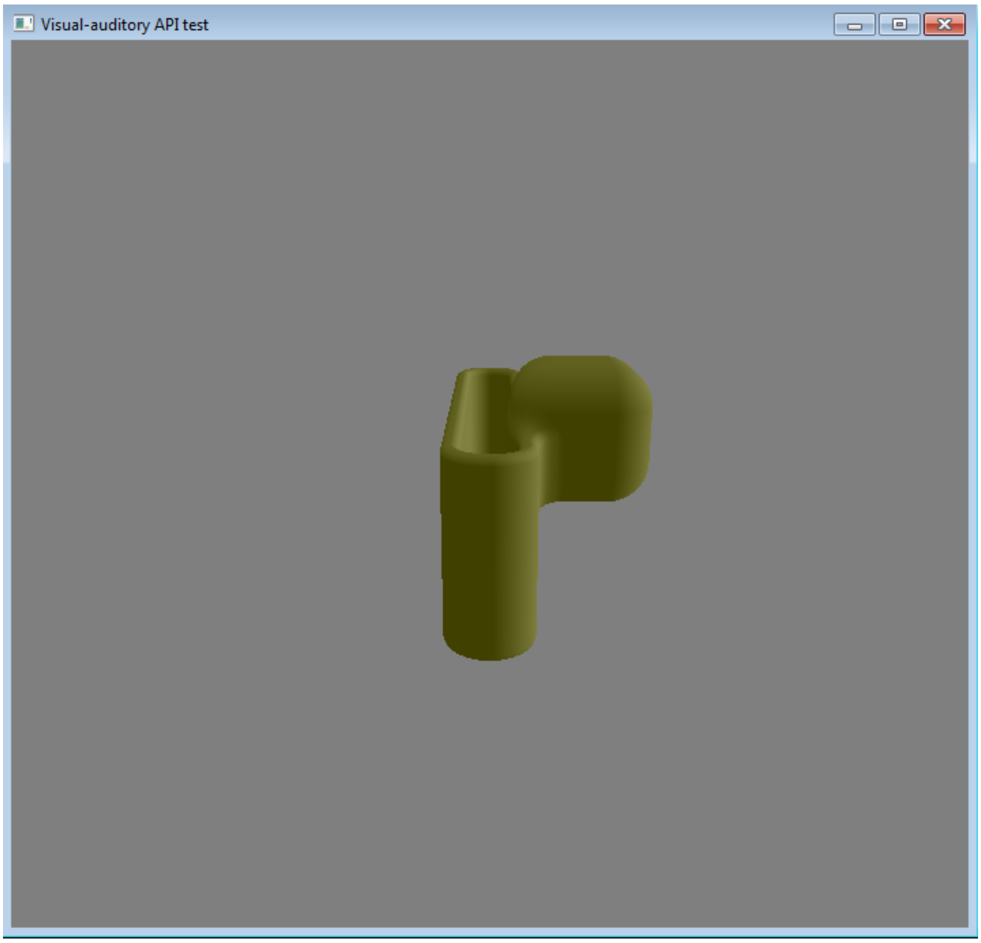


Figure 3.2 Rendered FRep modelled geometry

Chapter 4

FRep tutorial

Below is a basic example of creation of the FRep tree

Chapter 5

Main logic and similarities with VTK API

The basics of API use and main concept that is much simillar to `Visualisation Toolkit`. The core of the VTK is a visualisation pipeline of data, that starts from data source and ends up with image rendered on screen. The basic classes of VTK API are:

- Sources
- Filters
- Mappers
- Actors
- Renderers
- Interactors
- Windows

The Visual-auditory API implements basic Mappers, Actors, Renderers and Windows that are intended to support ray-casting of mainly Signed Distance Fields, althought triangulated geometry is also supported and can be integrated within pipeline. Similarly to VTK the pipeline is constructed through "input-output" concept. The scalar field data is provided by file Readers that eighter output structured data or directly create Optix Buffers and TextureSamplers on GPU. The inputs and outputs are usually Optix objects, like optix::Geometry, optix::Material, optix::GeometryInstance and etc.. As all this objects are created through optix::Context, the reference to it is given to each class. For context creation a ContextManager class is responsible.

5.1 Data readers and SDF geometry creation

In the example below, the Reader that reads molecular data described in `XYZ file format` and transfers it to "ball" like representation, when atoms represented as spheres (balls). This is current version of `spacefill molecule representation`. For info on `molecule representation types`.

```
contextManager m;
m.Update();//creates context
optixXYZReader read;  //reads data
read.Setfile("molecule.xyz");
read.Update();
optixSDFMolBalls mol;  //
mol.SetContext(m.GetOutput());
mol.SetCenter(read.GetOutput1()); //centers, data stored in std::vector
mol.SetTypes(read.GetOutput2()); //set types of atoms, data stored in std::vector
mol.SetMaterialType(0);
mol.Update();
```

The other example is reading dynamic scalar data, that represents . Gamess files. The reader performs signed distance transform of data on base of `Image Processing Toolkit`. It generates optix Buffers and Textures Samplers. The optixSDFTexture gets Texture Samplers describing initial and final states of molecule dynamic electron density field and forms dynamic SDF geometry object.

```
contextManager m;
m.Update();//creates context
optixSDFTextureReader<float> readSDFTex1;
readSDFTex1.SetContext(m.GetOutput());
readSDFTex1.SetSize(139, 150, 160); //data dimensions
readSDFTex1.SetThreshold(0.1); //isolevel of scalar field
readSDFTex1.Setfile("ed1.txt");
readSDFTex1.Update();
optixSDFTextureReader<float> readSDFTex2;
readSDFTex2.SetContext(m.GetOutput());
readSDFTex2.SetSize(138, 150, 160);
readSDFTex2.SetThreshold(0.1);
readSDFTex2.Setfile("ed2.txt");
readSDFTex2.Update();
optixSDFTexture tex; //dynamic SDF object creation
tex.SetContext(m.GetOutput());
tex.SetTexture(readSDFTex1.GetTexture(), readSDFTex1.GetParam());
tex.SetTexture(readSDFTex2.GetTexture(), readSDFTex1.GetParam());
tex.Update();
```

5.2 Mapper, Actor, Renderer

Mapper, Actor and Renderer classes are much simillar to VTK in concept and syntacsis. The Mapper applies a material to the objects (in current implementation with optical or auditory properties). The Mapper is can activate optical or auditory material, depending on type of rendering procedure: rendering optical or auditory properties. For this, in current implementation the Mapper receives optix::Material from Material class (material.GetOutput() procedure) and description of it's properties in , material.GetType() procedure.

The Renderer is responsible for rendering visual and auditory properties. Whether optical or auditory properties should be rendered, the Renderer operates its actors to activate the necessary visual or auditory model for all created pipelines of data.

Below the code demonstrating the above concepts and syntacsis of Mapper, Actor and Renderer classes basic usage.

```
SDFMaterial mSdf;
mSdf.SetContext(m.GetOutput());
mSdf.Update();
optixMapper map21;
map21.SetContext(m.GetOutput());
map21.SetInput(sdf.GetOutput());
map21.AddMaterial(mSdf.GetOutput(), mSdf.GetType());
map21.Update();
optixSdfActor acSdf1;
acSdf1.SetContext(m.GetOutput());
acSdf1.AddMapper(&map21);
acSdf1.Update();
ren.AddActor(&acSdf1);
....
ren.LaunchAuditoryContext(); //compute auditory ray-tracing
...
ren.LaunchOpticContext();// compute optical ray-tracing
```

5.3 GUI and interaction

The GUI procedures are implemented on base of `GLFW` and `ImGUI` libraries. A custom GUI can be easily implemented on base of those libraries and integrated to API pipeline.

The API provides a basic Renderer, GLFW window and Interactor:

```
contextManager m;
m.Update();//creates context
PinholeCamera pinholeCamera;  //basic camera
optixRenderer ren;
ren.SetValid(m.GetValid());
ren.SetContext(m.GetOutput());
ren.SetOpticalDims(windowWidth, windowWidth);
ren.SetCamera(&pinholeCamera);
//set not dynamic
ren.SetDynamic(false);
ren.SetAuditory(false);
//window procedure
GLFW_Window optixWindowProc;
optixWindowProc.SetDim(windowWidth, windowHeight);
optixWindowProc.SetRenderer(&ren);
optixWindowProc.SetContext(m.GetOutput()); //returns context
optixWindowProc.SetCamera(&pinholeCamera);
RenderWindowInteractor iren; //TODO always check that basic still works
iren.SetWindow(&optixWindowProc);
```

Chapter 6

Namespace Index

6.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

Chapter 8

Class Index

8.1 Class List

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

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

Chapter 10

Namespace Documentation

10.1 stkSound Namespace Reference

Classes

- struct **soundDesc**

Functions

- int **InitAL** (char ∗∗∗argv, int ∗argc)
- void **CloseAL** (void)
- void **ApplySin** (ALfloat ∗data, ALdouble g, ALuint srate, ALuint freq)
- void **CreateWave** (StkFloat ∗ptr, int srate2, int duration, ALuint &buffer)
- void **InitEnv** ()
- void **ExitEnv** ()
- StkFrames **GenerateNoise** (double time, double srate2, float seed)
- void **WaitforSound** (float scale)
- void **PlaySelSound** (ALuint source)
- void **MoveSounds** (std::vector< **stkSound::soundDesc** > &descSs, std::vector< float > &xpos, int scale)
- void **RotateTest** (int scale)
- **soundDesc** **PlayBuffer** (StkFloat ∗ptr, int time, int size, int j, optix::float2 xpos)
- void **RotateAngle** (double angle, ALuint &source)
- void **ApplyADSR** (stk::StkFrames &frames, int releaseCount)
- StkFrames **GeneratePlucked** (int time, double srate2, double freq, double amplitude)
- **soundDesc** **PlayBuffer** (stk::StkFloat ∗ptr, int time, int size, int j, optix::float2 xpos)

10.1.1 Function Documentation

10.1.1.1 ApplyADSR()

```
void stkSound::ApplyADSR (
            stk::StkFrames & frames,
            int releaseCount )
```

10.1.1.2 ApplySin()

```
void stkSound::ApplySin (
            ALfloat * data,
            ALdouble g,
            ALuint srate,
            ALuint freq )
```

10.1.1.3 CloseAL()

```
void stkSound::CloseAL (
            void  )
```

10.1.1.4 CreateWave()

```
void stkSound::CreateWave (
            StkFloat * ptr,
            int srate2,
            int duration,
            ALuint & buffer )
```

10.1.1.5 ExitEnv()

```
void stkSound::ExitEnv ( )
```

10.1.1.6 GenerateNoise()

```
stk::StkFrames stkSound::GenerateNoise (
            double time,
            double srate2,
            float seed )
```

10.1.1.7 GeneratePlucked()

```
stk::StkFrames stkSound::GeneratePlucked (
            int time,
            double srate2,
            double freq,
            double amplitude )
```

10.1.1.8 InitAL()

```
int stkSound::InitAL (
            char *** argv,
            int * argc )
```

10.1.1.9 InitEnv()

```
void stkSound::InitEnv ( )
```

10.1.1.10 MoveSounds()

```
void stkSound::MoveSounds (
            std::vector< stkSound::soundDesc > & descSs,
            std::vector< float > & xpos,
            int scale )
```

10.1.1.11 PlayBuffer() [1/2]

```
 soundDesc stkSound::PlayBuffer (
            stk::StkFloat * ptr,
            int time,
            int size,
            int j,
            optix::float2 xpos )
```

10.1.1.12 PlayBuffer() [2/2]

```
 soundDesc stkSound::PlayBuffer (
            StkFloat * ptr,
            int time,
            int size,
            int j,
            optix::float2 xpos )
```

10.1.1.13 PlaySelSound()

```
void stkSound::PlaySelSound (
            ALuint source )
```

10.1.1.14 RotateAngle()

```
void stkSound::RotateAngle (
            double angle,
            ALuint & source )
```

10.1.1.15 RotateTest()

```
void stkSound::RotateTest (
            int scale )
```

10.1.1.16 WaitforSound()

```
void stkSound::WaitforSound (
            float scale )
```

Chapter 11

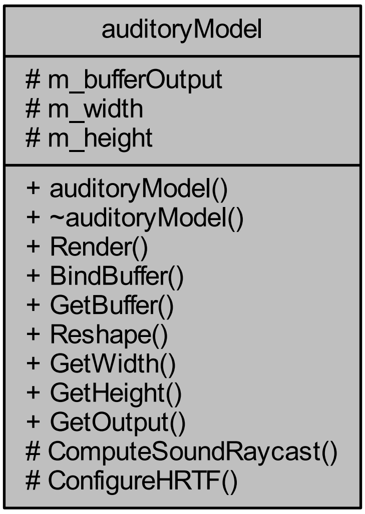
Class Documentation

11.1 auditoryModel Class Reference

Architecture much inspired by VTK Renderer(`http://vtk.org`) implementation The classes are mainly responsible for openGL/openAL interop.

```
#include <optixAudioOptic.h>
```

Collaboration diagram for auditoryModel:



Public Member Functions

- **auditoryModel** ()
- ∼**auditoryModel** ()

- void **Render** ()
- void **BindBuffer** (optix::Context context)
- void **GetBuffer** ()
- void **Reshape** (int width, int height)
- int **GetWidth** ()
- int **GetHeight** ()
- optix::Buffer **GetOutput** ()

Protected Member Functions

- void **ComputeSoundRaycast** (float distNorm, float distComp)
- void **ConfigureHRTF** ()

Protected Attributes

- optix::Buffer **m_bufferOutput**
- int **m_width**
- int **m_height**

11.1.1 Detailed Description

Architecture much inspired by VTK Renderer(`http://vtk.org`) implementation The classes are mainly responsible for openGL/openAL interop.

11.1.2 Constructor & Destructor Documentation

11.1.2.1 auditoryModel()

```
auditoryModel::auditoryModel ( ) [inline]
```

11.1.2.2 ∼auditoryModel()

```
auditoryModel::∼auditoryModel ( ) [inline]
```

11.1.3 Member Function Documentation

11.1.3.1 BindBuffer()

```
void auditoryModel::BindBuffer (
            optix::Context context )
```

11.1.3.2 ComputeSoundRaycast()

```
void auditoryModel::ComputeSoundRaycast (
            float distNorm,
            float distComp )  [protected]
```

11.1.3.3 ConfigureHRTF()

```
void auditoryModel::ConfigureHRTF ( )  [protected]
```

11.1.3.4 GetBuffer()

```
void auditoryModel::GetBuffer ( )
```

11.1.3.5 GetHeight()

```
int auditoryModel::GetHeight ( )  [inline]
```

11.1.3.6 GetOutput()

```
optix::Buffer auditoryModel::GetOutput ( )  [inline]
```

11.1.3.7 GetWidth()

```
int auditoryModel::GetWidth ( )  [inline]
```

11.1.3.8 Render()

```
void auditoryModel::Render ( )  [inline]
```

11.1.3.9 Reshape()

```
void auditoryModel::Reshape (
             int width,
             int height )
```

11.1.4 Member Data Documentation

11.1.4.1 m_bufferOutput

```
optix::Buffer auditoryModel::m_bufferOutput  [protected]
```

11.1.4.2 m_height

```
int auditoryModel::m_height  [protected]
```

11.1.4.3 m_width

```
int auditoryModel::m_width  [protected]
```

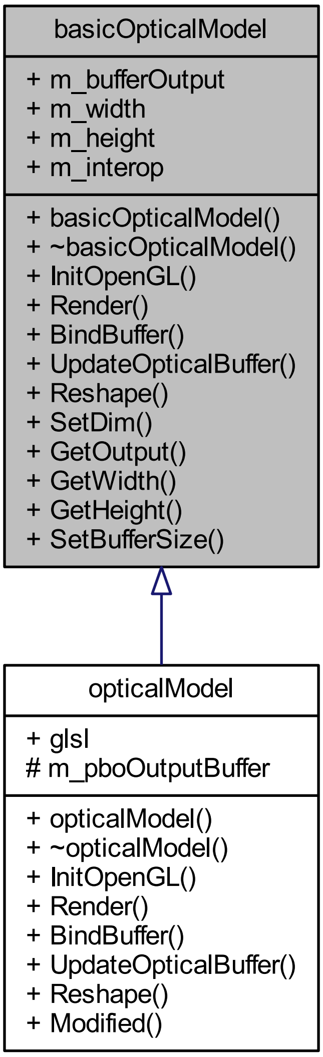
The documentation for this class was generated from the following files:

- **optixAudioOptic.h**
- **optixAudioOptic.cpp**

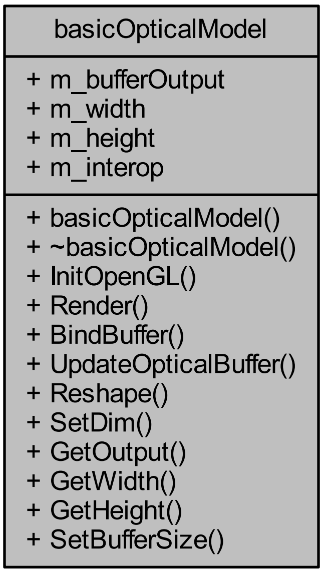
11.2 basicOpticalModel Class Reference

```
#include <optixAudioOptic.h>
```

Inheritance diagram for basicOpticalModel:



Collaboration diagram for basicOpticalModel:



Public Member Functions

- **basicOpticalModel** ()
- ∼**basicOpticalModel** ()
- virtual void **InitOpenGL** ()
- virtual void **Render** ()
- virtual void **BindBuffer** (optix::Context context)
- virtual void **UpdateOpticalBuffer** ()
- virtual void **Reshape** (int width, int height)
- void **SetDim** (int width, int height)
- optix::Buffer **GetOutput** ()
- int **GetWidth** ()
- int **GetHeight** ()
- void **SetBufferSize** ()

Public Attributes

- optix::Buffer **m_bufferOutput**
- int **m_width**
- int **m_height**
- bool **m_interop**

11.2.1 Constructor & Destructor Documentation

11.2.1.1 basicOpticalModel()

```
basicOpticalModel::basicOpticalModel ( ) [inline]
```

11.2.1.2 ∼basicOpticalModel()

```
basicOpticalModel::∼basicOpticalModel ( ) [inline]
```

11.2.2 Member Function Documentation

11.2.2.1 BindBuffer()

```
virtual void basicOpticalModel::BindBuffer (
             optix::Context context ) [inline], [virtual]
```

Reimplemented in **opticalModel** (p. 44).

11.2.2.2 GetHeight()

```
int basicOpticalModel::GetHeight ( ) [inline]
```

11.2.2.3 GetOutput()

```
optix::Buffer basicOpticalModel::GetOutput ( ) [inline]
```

11.2.2.4 GetWidth()

```
int basicOpticalModel::GetWidth ( ) [inline]
```

11.2.2.5 InitOpenGL()

```
virtual void basicOpticalModel::InitOpenGL ( ) [inline], [virtual]
```

Reimplemented in **opticalModel** (p. 44).

11.2.2.6 Render()

```
virtual void basicOpticalModel::Render ( )  [inline], [virtual]
```

Reimplemented in **opticalModel** (p. 44).

11.2.2.7 Reshape()

```
virtual void basicOpticalModel::Reshape (
             int width,
             int height )  [inline], [virtual]
```

Reimplemented in **opticalModel** (p. 45).

11.2.2.8 SetBufferSize()

```
void basicOpticalModel::SetBufferSize ( )  [inline]
```

11.2.2.9 SetDim()

```
void basicOpticalModel::SetDim (
             int width,
             int height )  [inline]
```

11.2.2.10 UpdateOpticalBuffer()

```
virtual void basicOpticalModel::UpdateOpticalBuffer ( )  [inline], [virtual]
```

Reimplemented in **opticalModel** (p. 45).

11.2.3 Member Data Documentation

11.2.3.1 m_bufferOutput

```
optix::Buffer basicOpticalModel::m_bufferOutput
```

11.2.3.2 m_height

```
int basicOpticalModel::m_height
```

11.2.3.3 m_interop

```
bool basicOpticalModel::m_interop
```

11.2.3.4 m_width

```
int basicOpticalModel::m_width
```

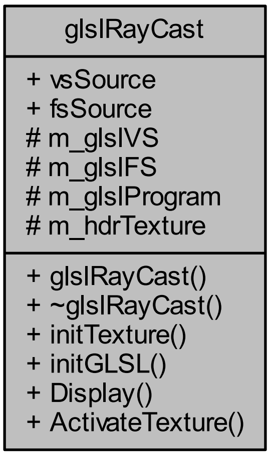
The documentation for this class was generated from the following file:

- **optixAudioOptic.h**

11.3 glslRayCast Class Reference

```
#include <optixAudioOptic.h>
```

Collaboration diagram for glslRayCast:



Public Member Functions

- **glslRayCast** ()
- ∼**glslRayCast** ()
- void **initTexture** ()
- void **initGLSL** ()
- void **Display** ()
- void **ActivateTexture** ()

Public Attributes

- std::string **vsSource**
- std::string **fsSource**

Protected Attributes

- GLuint **m_glslVS**
- GLuint **m_glslFS**
- GLuint **m_glslProgram**
- GLuint **m_hdrTexture**

11.3.1 Constructor & Destructor Documentation

11.3.1.1 glslRayCast()

```
glslRayCast::glslRayCast ( )
```

11.3.1.2 ∼glslRayCast()

```
glslRayCast::∼glslRayCast ( )  [inline]
```

11.3.2 Member Function Documentation

11.3.2.1 ActivateTexture()

```
void glslRayCast::ActivateTexture ( )  [inline]
```

11.3.2.2 Display()

```
void glslRayCast::Display ( )
```

11.3.2.3 initGLSL()

```
void glslRayCast::initGLSL ( )
```

11.3.2.4 initTexture()

```
void glslRayCast::initTexture ( )
```

11.3.3 Member Data Documentation

11.3.3.1 fsSource

```
std::string glslRayCast::fsSource
```

11.3.3.2 m_glslFS

```
GLuint glslRayCast::m_glslFS  [protected]
```

11.3.3.3 m_glslProgram

```
GLuint glslRayCast::m_glslProgram  [protected]
```

11.3.3.4 m_glslVS

```
GLuint glslRayCast::m_glslVS  [protected]
```

11.3.3.5 m_hdrTexture

```
GLuint glslRayCast::m_hdrTexture  [protected]
```

11.3.3.6 vsSource

```
std::string glslRayCast::vsSource
```

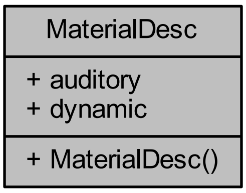
The documentation for this class was generated from the following files:

- **optixAudioOptic.h**
- **optixAudioOptic.cpp**

11.4 MaterialDesc Struct Reference

```
#include <optixAbstractMaterial.h>
```

Collaboration diagram for MaterialDesc:



Public Member Functions

- **MaterialDesc** (bool au, bool dy)

Public Attributes

- bool **auditory**
- bool **dynamic**

11.4.1 Constructor & Destructor Documentation

11.4.1.1 MaterialDesc()

```
MaterialDesc::MaterialDesc (
            bool au,
            bool dy ) [inline]
```

11.4.2 Member Data Documentation

11.4.2.1 auditory

```
bool MaterialDesc::auditory
```

11.4.2.2 dynamic

```
bool MaterialDesc::dynamic
```

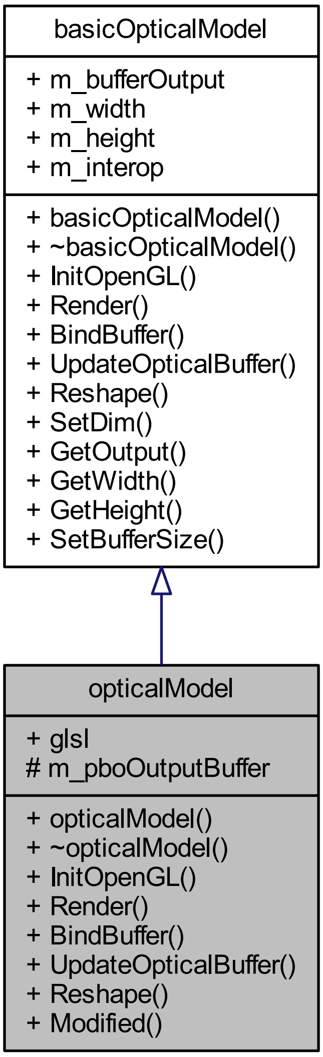
The documentation for this struct was generated from the following file:

- **optixAbstractMaterial.h**

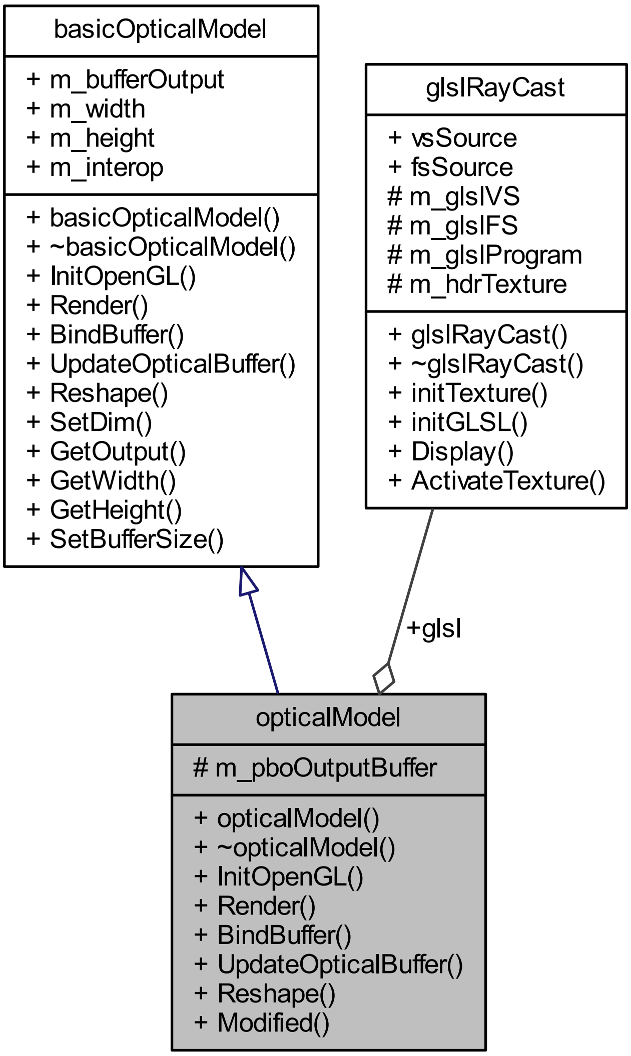
11.5 opticalModel Class Reference

```
#include <optixAudioOptic.h>
```

Inheritance diagram for opticalModel:



Collaboration diagram for opticalModel:



Public Member Functions

- **opticalModel** ()
- ∼**opticalModel** ()
- virtual void **InitOpenGL** ()
- virtual void **Render** ()
- virtual void **BindBuffer** (optix::Context context)
- virtual void **UpdateOpticalBuffer** ()
- virtual void **Reshape** (int width, int height)
- void **Modified** ()

Public Attributes

- **glslRayCast glsl**

Protected Attributes

- GLuint **m_pboOutputBuffer**

11.5.1 Constructor & Destructor Documentation

11.5.1.1 opticalModel()

```
opticalModel::opticalModel ( )  [inline]
```

11.5.1.2 ∼opticalModel()

```
opticalModel::∼opticalModel ( )  [inline]
```

11.5.2 Member Function Documentation

11.5.2.1 BindBuffer()

```
void opticalModel::BindBuffer (
            optix::Context context )  [virtual]
```

Reimplemented from **basicOpticalModel** (p. 35).

11.5.2.2 InitOpenGL()

```
void opticalModel::InitOpenGL ( )  [virtual]
```

Reimplemented from **basicOpticalModel** (p. 35).

11.5.2.3 Modified()

```
void opticalModel::Modified ( )  [inline]
```

11.5.2.4 Render()

```
virtual void opticalModel::Render ( )  [inline], [virtual]
```

Reimplemented from **basicOpticalModel** (p. 35).

11.5.2.5 Reshape()

```
void opticalModel::Reshape (
             int width,
             int height )  [virtual]
```

Reimplemented from **basicOpticalModel** (p. 36).

11.5.2.6 UpdateOpticalBuffer()

```
void opticalModel::UpdateOpticalBuffer ( )  [virtual]
```

Reimplemented from **basicOpticalModel** (p. 36).

11.5.3 Member Data Documentation

11.5.3.1 glsl

```
glslRayCast opticalModel::glsl
```

11.5.3.2 m_pboOutputBuffer

```
GLuint opticalModel::m_pboOutputBuffer  [protected]
```

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

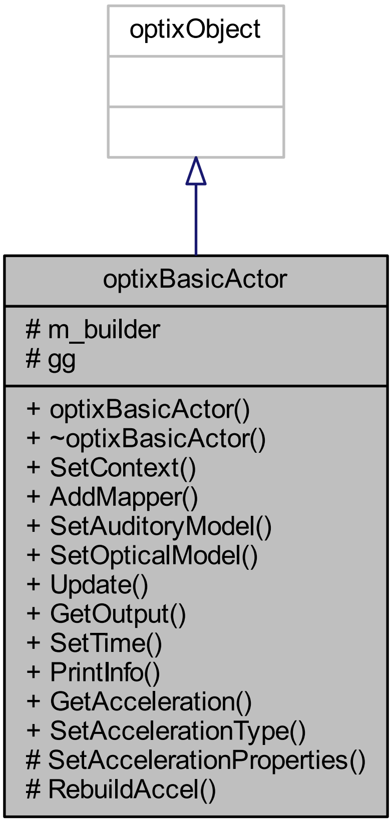
- **optixAudioOptic.h**
- **optixAudioOptic.cpp**

11.6 optixBasicActor Class Reference

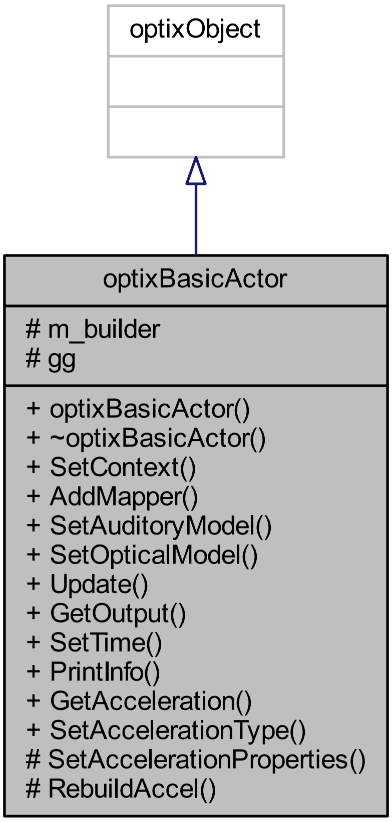
basic abstract actor class

```
#include <optixBasicActor.h>
```

Inheritance diagram for optixBasicActor:



Collaboration diagram for optixBasicActor:



Public Member Functions

- **optixBasicActor** ()
- ∼**optixBasicActor** ()
- virtual void **SetContext** (optix::Context &context)
- void **AddMapper** (**optixMapper** ∗map)
- void **SetAuditoryModel** ()
- void **SetOpticalModel** ()
- void **Update** ()
- optix::GeometryGroup **GetOutput** ()
- void **SetTime** (float time)
- void **PrintInfo** ()
- optix::Acceleration **GetAcceleration** ()
- void **SetAccelerationType** (std::string type)

Protected Member Functions

- virtual void **SetAccelerationProperties** ()
- virtual void **RebuildAccel** ()

Protected Attributes

- std::string **m_builder**
- optix::GeometryGroup **gg**

11.6.1 Detailed Description

basic abstract actor class

The class creates a basic Actor for Optix use. Architecture much inspired by VTK (`http://vtk.org`) implementation

11.6.2 Constructor & Destructor Documentation

11.6.2.1 optixBasicActor()

```
optixBasicActor::optixBasicActor ( )  [inline]
```

11.6.2.2 ∼optixBasicActor()

```
optixBasicActor::∼optixBasicActor ( )  [inline]
```

11.6.3 Member Function Documentation

11.6.3.1 AddMapper()

```
void optixBasicActor::AddMapper (
              optixMapper * map )  [inline]
```

11.6.3.2 GetAcceleration()

```
optix::Acceleration optixBasicActor::GetAcceleration ( )  [inline]
```

11.6.3.3 GetOutput()

```
optix::GeometryGroup optixBasicActor::GetOutput ( )  [inline]
```

11.6.3.4 PrintInfo()

```
void optixBasicActor::PrintInfo ( )  [inline]
```

11.6.3.5 RebuildAccel()

```
virtual void optixBasicActor::RebuildAccel ( )  [inline], [protected], [virtual]
```

11.6.3.6 SetAccelerationProperties()

```
virtual void optixBasicActor::SetAccelerationProperties ( )  [inline], [protected], [virtual]
```

11.6.3.7 SetAccelerationType()

```
void optixBasicActor::SetAccelerationType (
             std::string type )  [inline]
```

11.6.3.8 SetAuditoryModel()

```
void optixBasicActor::SetAuditoryModel ( )  [inline]
```

11.6.3.9 SetContext()

```
virtual void optixBasicActor::SetContext (
             optix::Context & context )  [inline], [virtual]
```

11.6.3.10 SetOpticalModel()

```
void optixBasicActor::SetOpticalModel ( )  [inline]
```

11.6.3.11 SetTime()

```
void optixBasicActor::SetTime (
             float time )  [inline]
```

11.6.3.12 Update()

```
void optixBasicActor::Update ( )
```

11.6.4 Member Data Documentation

11.6.4.1 gg

```
optix::GeometryGroup optixBasicActor::gg  [protected]
```

11.6.4.2 m_builder

```
std::string optixBasicActor::m_builder  [protected]
```

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

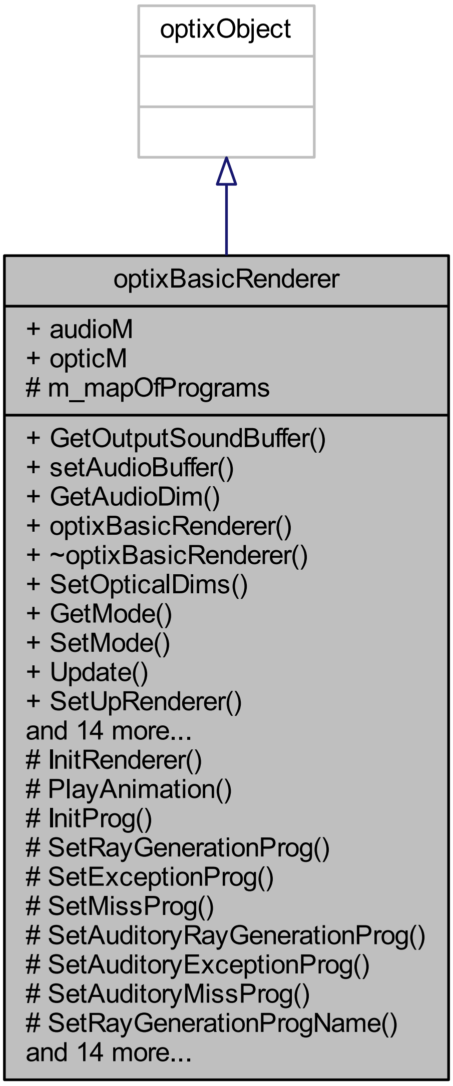
- **optixBasicActor.h**
- **optixBasicActor.cpp**

11.7 optixBasicRenderer Class Reference

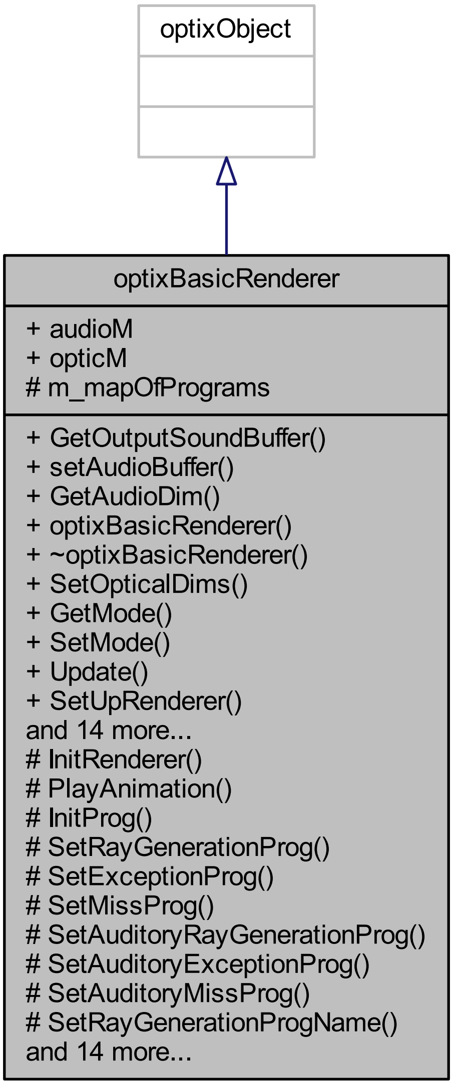
Architecture much inspired by VTK Renderer(`http://vtk.org`) implementation Abstract visual-auditory renderer.

```
#include <optixBasicRenderer.h>
```

Inheritance diagram for optixBasicRenderer:



Collaboration diagram for optixBasicRenderer:



Public Types

- enum { **OPTIC_RAYCASTING**, **AUDITORY_RAYCASTING** }
- enum **RenderModes** { **INTERACTIVE_CAMERA**, **COMPUTE_SOUND**, **PLAY_ANIMATION** }

Public Member Functions

- optix::Buffer **GetOutputSoundBuffer** ()
- void **setAudioBuffer** ()
- void **GetAudioDim** (int &w, int &h)
- **optixBasicRenderer** ()

- ∼**optixBasicRenderer** ()
- void **SetOpticalDims** (int width, int height)
- **RenderModes** **GetMode** ()
- void **SetMode** (**RenderModes** m)
- virtual void **Update** ()
- void **SetUpRenderer** ()
- void **SetDynamic** (bool type)
- void **SetAuditory** (bool type)
- bool **isDynamic** ()
- bool **isAuditory** ()
- void **SetTime** (float time)
- float **GetTime** ()
- void **AddActor** (**optixBasicActor** ∗act)
- **optixBasicActor** ∗ **GetActor** (int i)
- int **GetNumOfActors** ()
- void **LaunchOpticContext** ()
- void **LaunchAuditoryContext** ()
- virtual void **Reshape** (int width, int height)
- virtual bool **Render** ()
- virtual void **Display** ()

Public Attributes

- std::unique_ptr< **auditoryModel** > **audioM**
- std::unique_ptr< **basicOpticalModel** > **opticM**

Protected Member Functions

- virtual void **InitRenderer** ()
- bool **PlayAnimation** (float play_time)
- void **InitProg** (std::string prog, std::string file, std::string name)
- void **SetRayGenerationProg** ()
- void **SetExceptionProg** ()
- void **SetMissProg** ()
- void **SetAuditoryRayGenerationProg** ()
- void **SetAuditoryExceptionProg** ()
- void **SetAuditoryMissProg** ()
- void **SetRayGenerationProgName** (std::string name)
- void **SetExceptionProgName** (std::string name)
- void **SetMissProgName** (std::string name)
- void **SetAuditoryRayGenerationProgName** (std::string name)
- void **SetAuditoryExceptionProgName** (std::string name)
- void **SetAuditoryMissProgName** (std::string name)
- std::string **GetRayGenerationProgName** ()
- std::string **GetExceptionProgName** ()
- std::string **GetMissProgName** ()
- std::string **GetAuditoryRayGenerationProgName** ()
- std::string **GetAuditoryExceptionProgName** ()
- std::string **GetAuditoryMissProgName** ()
- void **InitAcceleration** ()
- void **SetPrograms** ()
- void **InitializePrograms** ()

Protected Attributes

- std::map< std::string, optix::Program > **m_mapOfPrograms**

11.7.1 Detailed Description

Architecture much inspired by VTK Renderer(`http://vtk.org`) implementation Abstract visual-auditory renderer.

11.7.2 Member Enumeration Documentation

11.7.2.1 anonymous enum

```
anonymous enum
```

Enumerator

OPTIC_RAYCASTING	
AUDITORY_RAYCASTING	

11.7.2.2 RenderModes

```
enum optixBasicRenderer::RenderModes
```

Enumerator

INTERACTIVE_CAMERA	
COMPUTE_SOUND	
PLAY_ANIMATION	

11.7.3 Constructor & Destructor Documentation

11.7.3.1 optixBasicRenderer()

```
optixBasicRenderer::optixBasicRenderer ( )  [inline]
```

11.7.3.2 ∼optixBasicRenderer()

```
optixBasicRenderer::∼optixBasicRenderer ( )  [inline]
```

11.7.4 Member Function Documentation

11.7.4.1 AddActor()

```
void optixBasicRenderer::AddActor (
              optixBasicActor * act )
```

11.7.4.2 Display()

```
virtual void optixBasicRenderer::Display ( )  [inline], [virtual]
```

11.7.4.3 GetActor()

```
 optixBasicActor* optixBasicRenderer::GetActor (
             int i )  [inline]
```

11.7.4.4 GetAudioDim()

```
void optixBasicRenderer::GetAudioDim (
             int & w,
             int & h )  [inline]
```

11.7.4.5 GetAuditoryExceptionProgName()

```
std::string optixBasicRenderer::GetAuditoryExceptionProgName ( )  [inline], [protected]
```

11.7.4.6 GetAuditoryMissProgName()

```
std::string optixBasicRenderer::GetAuditoryMissProgName ( )  [inline], [protected]
```

11.7.4.7 GetAuditoryRayGenerationProgName()

```
std::string optixBasicRenderer::GetAuditoryRayGenerationProgName ( )  [inline], [protected]
```

11.7.4.8 GetExceptionProgName()

```
std::string optixBasicRenderer::GetExceptionProgName ( )  [inline], [protected]
```

11.7.4.9 GetMissProgName()

```
std::string optixBasicRenderer::GetMissProgName ( )  [inline], [protected]
```

11.7.4.10 GetMode()

```
RenderModes optixBasicRenderer::GetMode ( )  [inline]
```

11.7.4.11 GetNumOfActors()

```
int optixBasicRenderer::GetNumOfActors ( )  [inline]
```

11.7.4.12 GetOutputSoundBuffer()

```
optix::Buffer optixBasicRenderer::GetOutputSoundBuffer ( )  [inline]
```

11.7.4.13 GetRayGenerationProgName()

```
std::string optixBasicRenderer::GetRayGenerationProgName ( )  [inline], [protected]
```

11.7.4.14 GetTime()

```
float optixBasicRenderer::GetTime ( )  [inline]
```

11.7.4.15 InitAcceleration()

```
void optixBasicRenderer::InitAcceleration ( )  [protected]
```

11.7.4.16 InitializePrograms()

```
void optixBasicRenderer::InitializePrograms ( )  [protected]
```

11.7.4.17 InitProg()

```
void optixBasicRenderer::InitProg (
            std::string prog,
            std::string file,
            std::string name )  [protected]
```

11.7.4.18 InitRenderer()

```
virtual void optixBasicRenderer::InitRenderer ( )  [inline], [protected], [virtual]
```

11.7.4.19 isAuditory()

```
bool optixBasicRenderer::isAuditory ( )  [inline]
```

11.7.4.20 isDynamic()

```
bool optixBasicRenderer::isDynamic ( )  [inline]
```

11.7.4.21 LaunchAuditoryContext()

```
void optixBasicRenderer::LaunchAuditoryContext ( )
```

11.7.4.22 LaunchOpticContext()

```
void optixBasicRenderer::LaunchOpticContext ( )
```

11.7.4.23 PlayAnimation()

```
bool optixBasicRenderer::PlayAnimation (
            float play_time )  [protected]
```

11.7.4.24 Render()

```
virtual bool optixBasicRenderer::Render ( )  [inline], [virtual]
```

11.7.4.25 Reshape()

```
virtual void optixBasicRenderer::Reshape (
            int width,
            int height )  [inline], [virtual]
```

11.7.4.26 setAudioBuffer()

```
void optixBasicRenderer::setAudioBuffer ( )  [inline]
```

11.7.4.27 SetAuditory()

```
void optixBasicRenderer::SetAuditory (
            bool type )  [inline]
```

11.7.4.28 SetAuditoryExceptionProg()

```
void optixBasicRenderer::SetAuditoryExceptionProg ( )  [protected]
```

11.7.4.29 SetAuditoryExceptionProgName()

```
void optixBasicRenderer::SetAuditoryExceptionProgName (
            std::string name )  [inline], [protected]
```

11.7.4.30 SetAuditoryMissProg()

```
void optixBasicRenderer::SetAuditoryMissProg ( )  [protected]
```

11.7.4.31 SetAuditoryMissProgName()

```
void optixBasicRenderer::SetAuditoryMissProgName (
            std::string name )  [inline], [protected]
```

11.7.4.32 SetAuditoryRayGenerationProg()

```
void optixBasicRenderer::SetAuditoryRayGenerationProg ( )  [protected]
```

11.7.4.33 SetAuditoryRayGenerationProgName()

```
void optixBasicRenderer::SetAuditoryRayGenerationProgName (
            std::string name )  [inline], [protected]
```

11.7.4.34 SetDynamic()

```
void optixBasicRenderer::SetDynamic (
            bool type )  [inline]
```

11.7.4.35 SetExceptionProg()

```
void optixBasicRenderer::SetExceptionProg ( )  [protected]
```

11.7.4.36 SetExceptionProgName()

```
void optixBasicRenderer::SetExceptionProgName (
             std::string name )  [inline], [protected]
```

11.7.4.37 SetMissProg()

```
void optixBasicRenderer::SetMissProg ( )  [protected]
```

11.7.4.38 SetMissProgName()

```
void optixBasicRenderer::SetMissProgName (
             std::string name )  [inline], [protected]
```

11.7.4.39 SetMode()

```
void optixBasicRenderer::SetMode (
              RenderModes m )  [inline]
```

11.7.4.40 SetOpticalDims()

```
void optixBasicRenderer::SetOpticalDims (
             int width,
             int height )  [inline]
```

11.7.4.41 SetPrograms()

```
void optixBasicRenderer::SetPrograms ( )  [protected]
```

11.7.4.42 SetRayGenerationProg()

```
void optixBasicRenderer::SetRayGenerationProg ( )  [protected]
```

11.7.4.43 SetRayGenerationProgName()

```
void optixBasicRenderer::SetRayGenerationProgName (
             std::string name )  [inline], [protected]
```

11.7.4.44 SetTime()

```
void optixBasicRenderer::SetTime (
             float time )  [inline]
```

11.7.4.45 SetUpRenderer()

```
void optixBasicRenderer::SetUpRenderer ( )
```

11.7.4.46 Update()

```
virtual void optixBasicRenderer::Update ( )  [inline], [virtual]
```

11.7.5 Member Data Documentation

11.7.5.1 audioM

```
std::unique_ptr< auditoryModel> optixBasicRenderer::audioM
```

11.7.5.2 m_mapOfPrograms

```
std::map<std::string, optix::Program> optixBasicRenderer::m_mapOfPrograms  [protected]
```

11.7.5.3 opticM

```
std::unique_ptr< basicOpticalModel> optixBasicRenderer::opticM
```

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

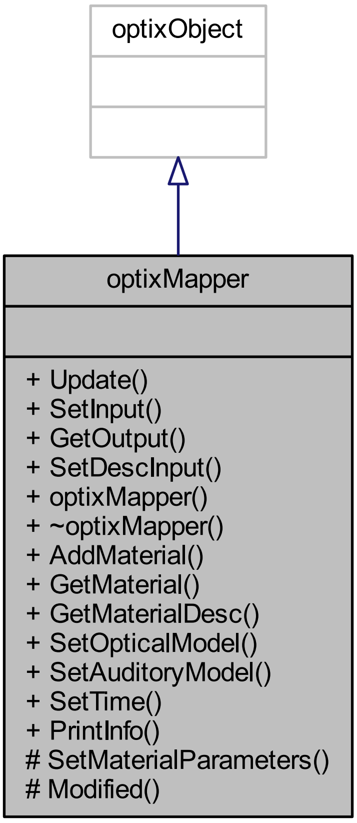
- **optixBasicRenderer.h**
- **optixBasicRenderer.cpp**

11.8 optixMapper Class Reference

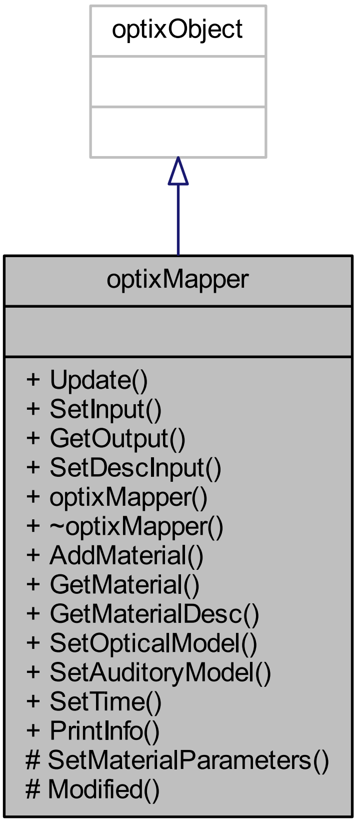
A basic mapper that suggest application of one visual and one auditory material to one geometry instance.

```
#include <optixAbstractMaterial.h>
```

Inheritance diagram for optixMapper:



Collaboration diagram for optixMapper:



Public Member Functions

- virtual void **Update** ()
- void **SetInput** (optix::Geometry g)
- optix::GeometryInstance **GetOutput** ()
- void **SetDescInput** (sdfGeo ∗g)
- **optixMapper** ()
- ∼**optixMapper** ()
- void **AddMaterial** (optix::Material m, **MaterialDesc** desc)
- optix::Material **GetMaterial** (int i)
- **MaterialDesc** **GetMaterialDesc** (int i)
- void **SetOpticalModel** ()
- void **SetAuditoryModel** ()
- void **SetTime** (float time)
- void **PrintInfo** ()

Protected Member Functions

- virtual void **SetMaterialParameters** (int i)
- virtual void **Modified** ()

11.8.1 Detailed Description

A basic mapper that suggest application of one visual and one auditory material to one geometry instance.

The codes inspired by vtkMapper class. Returns optix::GeometryInstance with assigned material. Each material suggests definition of optical model , that treats geometry defined object as a sufrace or takes geometry and hetegoreneously defined attributes and maps them to optical and/or auditory properties for more details see ..heterogeneous objects modelling

The class takes optix geometry as input, assigns materials and returns optix GeometryInstance. Class also handles switching between optical and auditory material to perform visual and auditory rendering accordingly

11.8.2 Constructor & Destructor Documentation

11.8.2.1 optixMapper()

```
optixMapper::optixMapper ( )  [inline]
```

11.8.2.2 ∼optixMapper()

```
optixMapper::∼optixMapper ( )  [inline]
```

11.8.3 Member Function Documentation

11.8.3.1 AddMaterial()

```
void optixMapper::AddMaterial (
             optix::Material m,
              MaterialDesc desc )
```

11.8.3.2 GetMaterial()

```
optix::Material optixMapper::GetMaterial (
             int i )
```

11.8.3.3 GetMaterialDesc()

```
 MaterialDesc optixMapper::GetMaterialDesc (
             int i )  [inline]
```

11.8.3.4 GetOutput()

```
optix::GeometryInstance optixMapper::GetOutput ( )  [inline]
```

11.8.3.5 Modified()

```
virtual void optixMapper::Modified ( )  [inline], [protected], [virtual]
```

11.8.3.6 PrintInfo()

```
void optixMapper::PrintInfo ( )  [inline]
```

11.8.3.7 SetAuditoryModel()

```
void optixMapper::SetAuditoryModel ( )
```

11.8.3.8 SetDescInput()

```
void optixMapper::SetDescInput (
             sdfGeo * g )  [inline]
```

11.8.3.9 SetInput()

```
void optixMapper::SetInput (
             optix::Geometry g )  [inline]
```

11.8.3.10 SetMaterialParameters()

```
virtual void optixMapper::SetMaterialParameters (
             int i )  [inline], [protected], [virtual]
```

11.8.3.11 SetOpticalModel()

```
void optixMapper::SetOpticalModel ( )
```

11.8.3.12 SetTime()

```
void optixMapper::SetTime (
             float time )  [inline]
```

11.8.3.13 Update()

```
void optixMapper::Update ( )  [virtual]
```

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

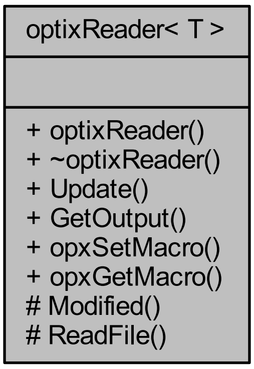
- **optixAbstractMaterial.h**
- **optixAbstractMaterial.cpp**

11.9 optixReader< T > Class Template Reference

The class creates a basic triangular plane for Optix use. Architecture much inspired by VTK (`http://vtk.org`) implementation.

`#include <optixReader.h>`

Collaboration diagram for optixReader< T >:



Public Member Functions

- **optixReader** ()
- ∼**optixReader** ()
- virtual void **Update** ()
- virtual std::shared_ptr< T > **GetOutput** ()
- **opxSetMacro** (file, std::string)
- **opxGetMacro** (file, std::string)

Protected Member Functions

- virtual void **Modified** ()
- virtual void **ReadFile** ()

11.9.1 Detailed Description

template<class T>
class optixReader< T >

The class creates a basic triangular plane for Optix use. Architecture much inspired by VTK (`http://vtk.org`) implementation.

11.9.2 Constructor & Destructor Documentation

11.9.2.1 optixReader()

```
template<class T >
optixReader< T >:: optixReader ( ) [inline]
```

11.9.2.2 ∼optixReader()

```
template<class T >
optixReader< T >::∼ optixReader ( ) [inline]
```

11.9.3 Member Function Documentation

11.9.3.1 GetOutput()

```
template<class T >
virtual std::shared_ptr<T>  optixReader< T >::GetOutput ( )  [inline], [virtual]
```

11.9.3.2 Modified()

```
template<class T >
virtual void  optixReader< T >::Modified ( )  [inline], [protected], [virtual]
```

11.9.3.3 opxGetMacro()

```
template<class T >
optixReader< T >::opxGetMacro (
             file ,
             std::string  )
```

11.9.3.4 opxSetMacro()

```
template<class T >
optixReader< T >::opxSetMacro (
             file ,
             std::string  )
```

11.9.3.5 ReadFile()

```
template<class T >
virtual void  optixReader< T >::ReadFile ( )  [inline], [protected], [virtual]
```

11.9.3.6 Update()

```
template<class T >
virtual void  optixReader< T >::Update ( )  [inline], [virtual]
```

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

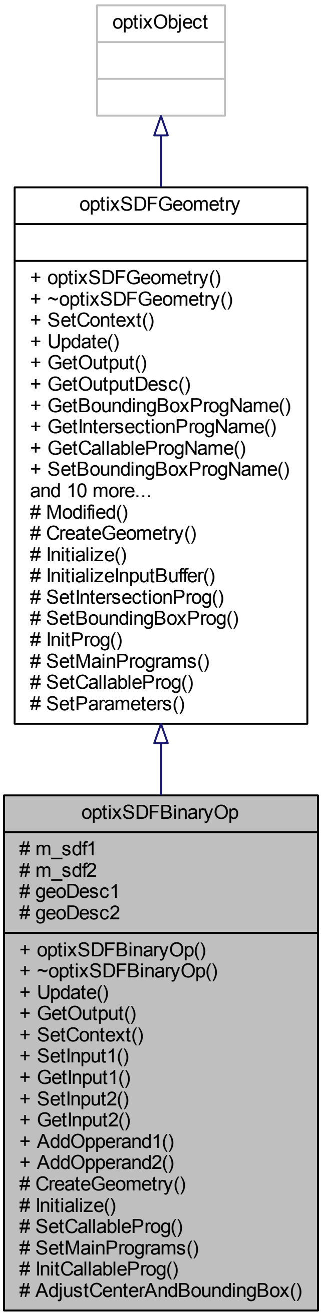
- **optixReader.h**

11.10 optixSDFBinaryOp Class Reference

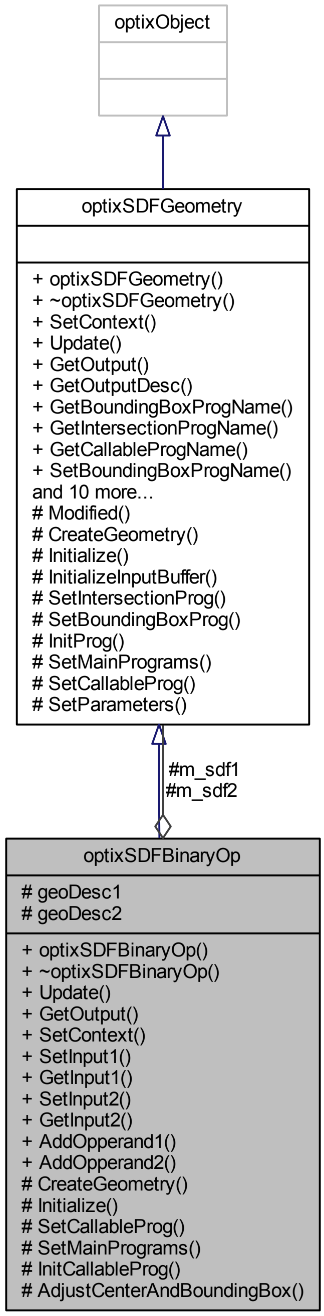
Binary operation.

```
#include <optixSDFBasicOperations.h>
```

Inheritance diagram for optixSDFBinaryOp:



Collaboration diagram for optixSDFBinaryOp:



Public Member Functions

- **optixSDFBinaryOp** ()
- ∼**optixSDFBinaryOp** ()
- virtual void **Update** ()
- virtual optix::Geometry **GetOutput** ()
- virtual void **SetContext** (optix::Context &context)

- virtual void **SetInput1** (optix::Geometry geo1)
- optix::Geometry **GetInput1** ()
- virtual void **SetInput2** (optix::Geometry geo1)
- optix::Geometry **GetInput2** ()
- void **AddOpperand1** (**optixSDFGeometry** ∗sdf)
- void **AddOpperand2** (**optixSDFGeometry** ∗sdf)

Protected Member Functions

- virtual void **CreateGeometry** ()
- virtual void **Initialize** ()
- virtual void **SetCallableProg** ()
- virtual void **SetMainPrograms** ()
- virtual void **InitCallableProg** ()
- virtual void **AdjustCenterAndBoundingBox** ()

Protected Attributes

- **optixSDFGeometry** ∗ **m_sdf1**
- **optixSDFGeometry** ∗ **m_sdf2**
- sdfGeo ∗ **geoDesc1**
- sdfGeo ∗ **geoDesc2**

11.10.1 Detailed Description

Binary operation.

Abstract class. Defines sdf Binary operation for FRep tree construction pipeline

11.10.2 Constructor & Destructor Documentation

11.10.2.1 optixSDFBinaryOp()

```
optixSDFBinaryOp::optixSDFBinaryOp ( ) [inline]
```

11.10.2.2 ∼optixSDFBinaryOp()

```
optixSDFBinaryOp::∼optixSDFBinaryOp ( ) [inline]
```

11.10.3 Member Function Documentation

11.10.3.1 AddOpperand1()

```
void optixSDFBinaryOp::AddOpperand1 (
              optixSDFGeometry * sdf )  [inline]
```

11.10.3.2 AddOpperand2()

```
void optixSDFBinaryOp::AddOpperand2 (
              optixSDFGeometry * sdf )  [inline]
```

11.10.3.3 AdjustCenterAndBoundingBox()

```
virtual void optixSDFBinaryOp::AdjustCenterAndBoundingBox ( )  [inline], [protected], [virtual]
```

11.10.3.4 CreateGeometry()

```
void optixSDFBinaryOp::CreateGeometry ( )  [protected], [virtual]
```

Reimplemented from **optixSDFGeometry** (p. 78).

11.10.3.5 GetInput1()

```
optix::Geometry optixSDFBinaryOp::GetInput1 ( )  [inline]
```

11.10.3.6 GetInput2()

```
optix::Geometry optixSDFBinaryOp::GetInput2 ( )  [inline]
```

11.10.3.7 GetOutput()

```
virtual optix::Geometry optixSDFBinaryOp::GetOutput ( )  [inline], [virtual]
```

Reimplemented from **optixSDFGeometry** (p. 79).

11.10.3.8 InitCallableProg()

```
virtual void optixSDFBinaryOp::InitCallableProg ( )  [inline], [protected], [virtual]
```

11.10.3.9 Initialize()

```
void optixSDFBinaryOp::Initialize ( )  [protected], [virtual]
```

TODO: potentialy read file with points here

Reimplemented from **optixSDFGeometry** (p. 79).

11.10.3.10 SetCallableProg()

```
void optixSDFBinaryOp::SetCallableProg ( )  [protected], [virtual]
```

Is called for setting callable prog

Reimplemented from **optixSDFGeometry** (p. 80).

11.10.3.11 SetContext()

```
void optixSDFBinaryOp::SetContext (
            optix::Context & context )  [virtual]
```

Sets optix context that will store all generated optix geometry

Reimplemented from **optixSDFGeometry** (p. 81).

11.10.3.12 SetInput1()

```
virtual void optixSDFBinaryOp::SetInput1 (
            optix::Geometry geo1 )  [inline], [virtual]
```

11.10.3.13 SetInput2()

```
virtual void optixSDFBinaryOp::SetInput2 (
            optix::Geometry geo1 )  [inline], [virtual]
```

11.10.3.14 SetMainPrograms()

```
void optixSDFBinaryOp::SetMainPrograms ( )  [protected], [virtual]
```

sets all main programs

Reimplemented from **optixSDFGeometry** (p. 81).

11.10.3.15 Update()

```
virtual void optixSDFBinaryOp::Update ( )  [inline], [virtual]
```

Runs the process of computation of optix geometry and setting up all necessary cuda programs and geometry parameters

Reimplemented from **optixSDFGeometry** (p. 82).

11.10.4 Member Data Documentation

11.10.4.1 geoDesc1

```
sdfGeo* optixSDFBinaryOp::geoDesc1  [protected]
```

11.10.4.2 geoDesc2

```
sdfGeo* optixSDFBinaryOp::geoDesc2  [protected]
```

11.10.4.3 m_sdf1

```
optixSDFGeometry* optixSDFBinaryOp::m_sdf1  [protected]
```

11.10.4.4 m_sdf2

```
optixSDFGeometry* optixSDFBinaryOp::m_sdf2  [protected]
```

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

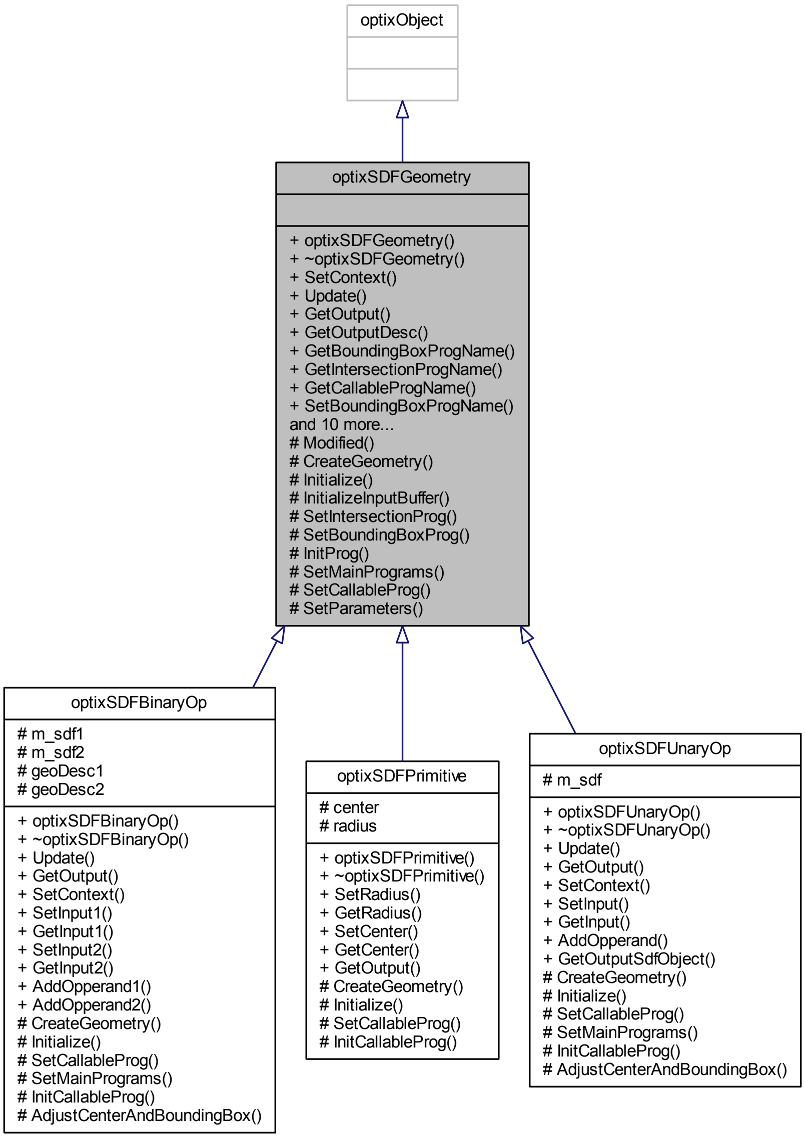
- **optixSDFBasicOperations.h**
- **optixSDFBasicOperations.cpp**

11.11 optixSDFGeometry Class Reference

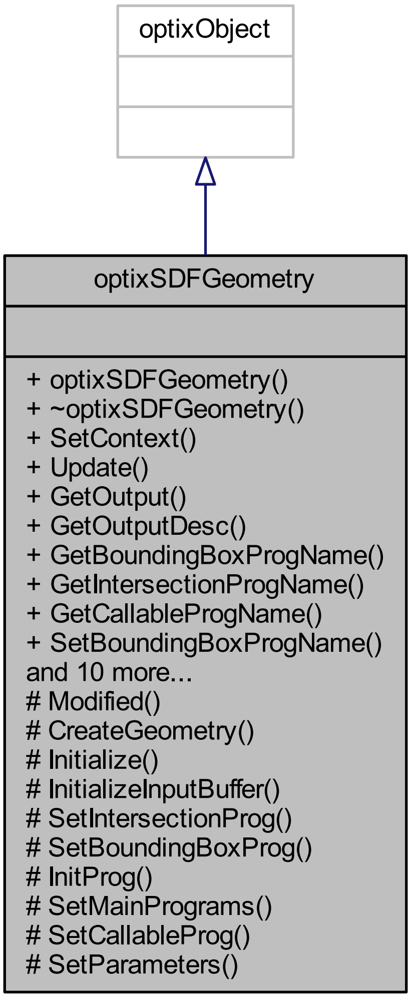
Abstract class for SDF Geometry. A basic class. All SDF geometry inherits it's functions.

```
#include <optixSDFGeometry.h>
```

Inheritance diagram for optixSDFGeometry:



Collaboration diagram for optixSDFGeometry:



Public Member Functions

- **optixSDFGeometry** ()
- ∼**optixSDFGeometry** ()
- virtual void **SetContext** (optix::Context &context)
- virtual void **Update** ()
- virtual optix::Geometry **GetOutput** ()
- virtual sdfGeo ∗ **GetOutputDesc** ()
- std::string **GetBoundingBoxProgName** ()
- std::string **GetIntersectionProgName** ()
- std::string **GetCallableProgName** ()
- void **SetBoundingBoxProgName** (std::string bb)
- void **SetIntersectionProgName** (std::string inter)
- void **SetCallableProgName** (std::string c)

- optix::Program **GetCallableProg** ()
- void **SetCallableProgManually** (optix::Program pr)
- void **SetMaterialType** (int type)
- int **GetMaterialType** ()
- void **SetDynamic** (bool d)
- bool **isDynamic** ()
- virtual void **SetTime** (float time)
- virtual float **GetTime** ()

Protected Member Functions

- virtual void **Modified** ()
- virtual void **CreateGeometry** ()
- virtual void **Initialize** ()
- template< class T >
  optix::Buffer **InitializeInputBuffer** (T Attributes, std::vector< T > attributes, optix::RTbuffermapflag mode)
- void **SetIntersectionProg** ()
- void **SetBoundingBoxProg** ()
- void **InitProg** (std::string prog, std::string file, std::string name)
- virtual void **SetMainPrograms** ()
- virtual void **SetCallableProg** ()
- virtual void **SetParameters** ()

11.11.1 Detailed Description

Abstract class for SDF Geometry. A basic class. All SDF geometry inherits it's functions.

All SDF geometry inherits it's functions, that includes:

1)SDF dynamic textures

2)FRep modelling primitives and operations that construct dynamically Construction Tree see `http://hyperfun.org` for details

3)Large objects, like molecular structures, that consist of lot's of objects and use BVH for rendering

...

may be something else

Architecture much inspired by VTK (`http://vtk.org`) implementation. Main operations similar to most of objects are much similar to VTK classes structure:

virtual void **SetContext(optix::Context &context)** (p. 81);

virtual void **Update()** (p. 82);

virtual optix::Geometry **GetOutput()** (p. 79)

virtual sdfGeo∗ **GetOutputDesc()** (p. 79)

See Tutorial for more info.

11.11.2 Constructor & Destructor Documentation

11.11.2.1 optixSDFGeometry()

```
optixSDFGeometry::optixSDFGeometry ( )
```

Constructor and destructor

11.11.2.2 ∼optixSDFGeometry()

```
optixSDFGeometry::∼optixSDFGeometry ( ) [inline]
```

11.11.3 Member Function Documentation

11.11.3.1 CreateGeometry()

```
virtual void optixSDFGeometry::CreateGeometry ( ) [inline], [protected], [virtual]
```

Reimplemented in **optixSDFBinaryOp** (p. 72), **optixSDFPrimitive** (p. 85), and **optixSDFUnaryOp** (p. 91).

11.11.3.2 GetBoundingBoxProgName()

```
std::string optixSDFGeometry::GetBoundingBoxProgName ( ) [inline]
```

Returns description, a more detailed info, that is optix geometry and description wheter it is dynamic, auditory and etc.

11.11.3.3 GetCallableProg()

```
optix::Program optixSDFGeometry::GetCallableProg ( )
```

11.11.3.4 GetCallableProgName()

```
std::string optixSDFGeometry::GetCallableProgName ( ) [inline]
```

11.11.3.5 GetIntersectionProgName()

```
std::string optixSDFGeometry::GetIntersectionProgName ( ) [inline]
```

11.11.3.6 GetMaterialType()

```
int optixSDFGeometry::GetMaterialType ( )
```

11.11.3.7 GetOutput()

```
virtual optix::Geometry optixSDFGeometry::GetOutput ( ) [inline], [virtual]
```

Reimplemented in **optixSDFBinaryOp** (p. 72), **optixSDFPrimitive** (p. 86), and **optixSDFUnaryOp** (p. 91).

11.11.3.8 GetOutputDesc()

```
virtual sdfGeo* optixSDFGeometry::GetOutputDesc ( ) [inline], [virtual]
```

Returns ready to use output, that is optix geometry

11.11.3.9 GetTime()

```
virtual float optixSDFGeometry::GetTime ( ) [inline], [virtual]
```

11.11.3.10 Initialize()

```
virtual void optixSDFGeometry::Initialize ( ) [inline], [protected], [virtual]
```

TODO: potentialy read file with points here

Reimplemented in **optixSDFBinaryOp** (p. 73), **optixSDFPrimitive** (p. 86), and **optixSDFUnaryOp** (p. 91).

11.11.3.11 InitializeInputBuffer()

```
template<class T >
optix::Buffer optixSDFGeometry::InitializeInputBuffer (
            T Attributes,
            std::vector< T > attributes,
            optix::RTbuffermapflag mode ) [protected]
```

read cuda procedure for SDF generation, the process is linked to name

11.11.3.12 InitProg()

```
void optixSDFGeometry::InitProg (
            std::string prog,
            std::string file,
            std::string name ) [protected]
```

11.11.3.13 isDynamic()

```
bool optixSDFGeometry::isDynamic ( ) [inline]
```

11.11.3.14 Modified()

```
virtual void optixSDFGeometry::Modified ( ) [inline], [protected], [virtual]
```

11.11.3.15 SetBoundingBoxProg()

```
void optixSDFGeometry::SetBoundingBoxProg ( ) [protected]
```

11.11.3.16 SetBoundingBoxProgName()

```
void optixSDFGeometry::SetBoundingBoxProgName (
            std::string bb ) [inline]
```

11.11.3.17 SetCallableProg()

```
virtual void optixSDFGeometry::SetCallableProg ( ) [inline], [protected], [virtual]
```

Is called for setting callable prog

Reimplemented in **optixSDFBinaryOp** (p. 73), **optixSDFPrimitive** (p. 86), and **optixSDFUnaryOp** (p. 92).

11.11.3.18 SetCallableProgManually()

```
void optixSDFGeometry::SetCallableProgManually (
            optix::Program pr )
```

11.11.3.19 SetCallableProgName()

```
void optixSDFGeometry::SetCallableProgName (
            std::string c )  [inline]
```

11.11.3.20 SetContext()

```
void optixSDFGeometry::SetContext (
            optix::Context & context )  [virtual]
```

Sets optix context that will store all generated optix geometry

Reimplemented in **optixSDFBinaryOp** (p. 73), and **optixSDFUnaryOp** (p. 92).

11.11.3.21 SetDynamic()

```
void optixSDFGeometry::SetDynamic (
            bool d )  [inline]
```

11.11.3.22 SetIntersectionProg()

```
void optixSDFGeometry::SetIntersectionProg ( )  [protected]
```

11.11.3.23 SetIntersectionProgName()

```
void optixSDFGeometry::SetIntersectionProgName (
            std::string inter )  [inline]
```

11.11.3.24 SetMainPrograms()

```
void optixSDFGeometry::SetMainPrograms ( )  [protected], [virtual]
```

sets all main programs

Reimplemented in **optixSDFBinaryOp** (p. 73), and **optixSDFUnaryOp** (p. 92).

11.11.3.25 SetMaterialType()

```
void optixSDFGeometry::SetMaterialType (
             int type )
```

11.11.3.26 SetParameters()

```
virtual void optixSDFGeometry::SetParameters ( )  [inline], [protected], [virtual]
```

for setting custom cuda variables values

11.11.3.27 SetTime()

```
virtual void optixSDFGeometry::SetTime (
             float time )  [inline], [virtual]
```

11.11.3.28 Update()

```
void optixSDFGeometry::Update ( )  [virtual]
```

Runs the process of computation of optix geometry and setting up all necessary cuda programs and geometry parameters

Reimplemented in **optixSDFBinaryOp** (p. 74), and **optixSDFUnaryOp** (p. 92).

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

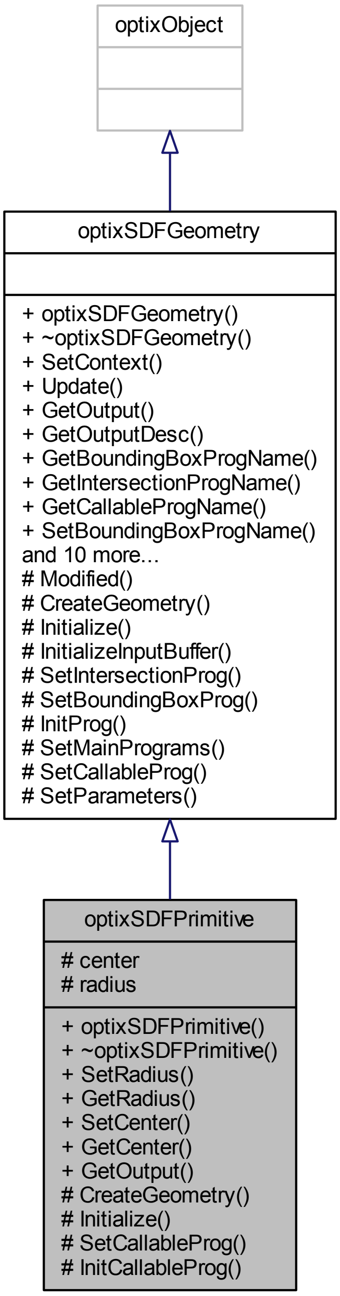
- **optixSDFGeometry.h**
- **optixSDFGeometry.cpp**

11.12 optixSDFPrimitive Class Reference

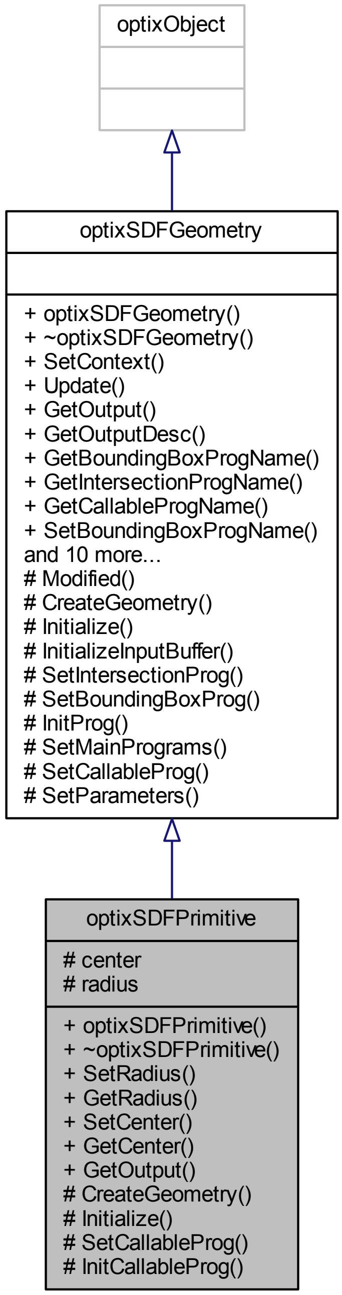
A basic class for all sdf primitives.

```
#include <optixSDFBasicPrimitives.h>
```

Inheritance diagram for optixSDFPrimitive:



Collaboration diagram for optixSDFPrimitive:



Public Member Functions

- **optixSDFPrimitive** ()
- ∼**optixSDFPrimitive** ()
- void **SetRadius** (optix::float3 rad)
- optix::float3 **GetRadius** ()
- void **SetCenter** (optix::float3 c)

- optix::float3 **GetCenter** ()
- virtual optix::Geometry **GetOutput** ()

Protected Member Functions

- virtual void **CreateGeometry** ()
- virtual void **Initialize** ()
- virtual void **SetCallableProg** ()
- virtual void **InitCallableProg** ()

Protected Attributes

- optix::float3 **center**
- optix::float3 **radius**

11.12.1 Detailed Description

A basic class for all sdf primitives.

The class creates a basic triangular plane for Optix use. Codes from Optix advanced samples. Architecture much inspired by VTK (`http://vtk.org`) implementation

11.12.2 Constructor & Destructor Documentation

11.12.2.1 optixSDFPrimitive()

```
optixSDFPrimitive::optixSDFPrimitive ( )  [inline]
```

11.12.2.2 ∼optixSDFPrimitive()

```
optixSDFPrimitive::∼optixSDFPrimitive ( )  [inline]
```

11.12.3 Member Function Documentation

11.12.3.1 CreateGeometry()

```
void optixSDFPrimitive::CreateGeometry ( )  [protected], [virtual]
```

Reimplemented from **optixSDFGeometry** (p. 78).

11.12.3.2 GetCenter()

```
optix::float3 optixSDFPrimitive::GetCenter ( )  [inline]
```

11.12.3.3 GetOutput()

```
virtual optix::Geometry optixSDFPrimitive::GetOutput ( )  [inline], [virtual]
```

Reimplemented from **optixSDFGeometry** (p. 79).

11.12.3.4 GetRadius()

```
optix::float3 optixSDFPrimitive::GetRadius ( )  [inline]
```

11.12.3.5 InitCallableProg()

```
virtual void optixSDFPrimitive::InitCallableProg ( )  [inline], [protected], [virtual]
```

11.12.3.6 Initialize()

```
void optixSDFPrimitive::Initialize ( )  [protected], [virtual]
```

TODO: potentialy read file with points here

Reimplemented from **optixSDFGeometry** (p. 79).

11.12.3.7 SetCallableProg()

```
void optixSDFPrimitive::SetCallableProg ( )  [protected], [virtual]
```

Is called for setting callable prog

Reimplemented from **optixSDFGeometry** (p. 80).

11.12.3.8 SetCenter()

```
void optixSDFPrimitive::SetCenter (
             optix::float3 c )  [inline]
```

11.12.3.9 SetRadius()

```
void optixSDFPrimitive::SetRadius (
             optix::float3 rad )  [inline]
```

11.12.4 Member Data Documentation

11.12.4.1 center

```
optix::float3 optixSDFPrimitive::center  [protected]
```

11.12.4.2 radius

```
optix::float3 optixSDFPrimitive::radius  [protected]
```

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

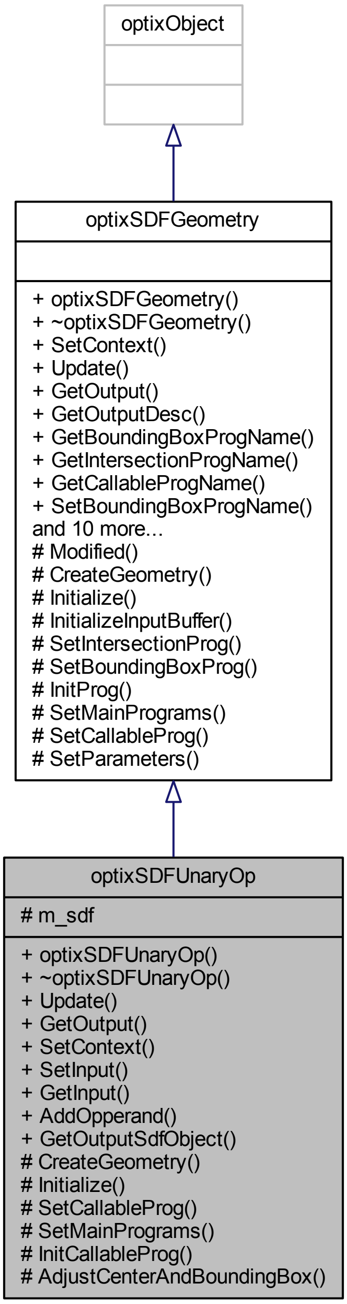
- **optixSDFBasicPrimitives.h**
- **optixSDFBasicPrimitives.cpp**

11.13 optixSDFUnaryOp Class Reference

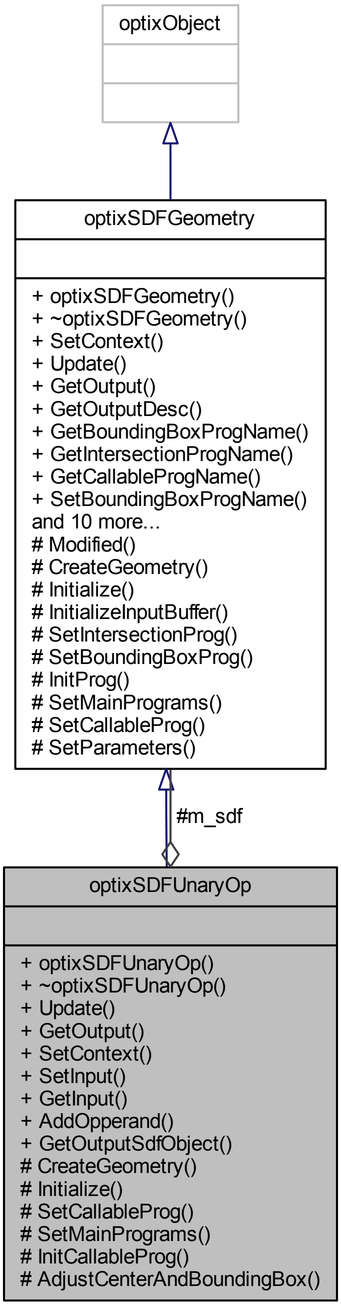
Unary operation.

```
#include <optixSDFBasicOperations.h>
```

Inheritance diagram for optixSDFUnaryOp:



Collaboration diagram for optixSDFUnaryOp:



Public Member Functions

- **optixSDFUnaryOp** ()
- ∼**optixSDFUnaryOp** ()
- virtual void **Update** ()
- virtual optix::Geometry **GetOutput** ()
- virtual void **SetContext** (optix::Context &context)

- virtual void **SetInput** (optix::Geometry geo1)
- optix::Geometry **GetInput** ()
- void **AddOpperand** (**optixSDFGeometry** ∗sdf)
- **optixSDFGeometry** ∗ **GetOutputSdfObject** ()

Protected Member Functions

- virtual void **CreateGeometry** ()
- virtual void **Initialize** ()
- virtual void **SetCallableProg** ()
- virtual void **SetMainPrograms** ()
- virtual void **InitCallableProg** ()
- virtual void **AdjustCenterAndBoundingBox** ()

Protected Attributes

- **optixSDFGeometry** ∗ **m_sdf**

11.13.1 Detailed Description

Unary operation.

Abstract class. Defines sdf unary operation for FRep tree construction pipeline

11.13.2 Constructor & Destructor Documentation

11.13.2.1 optixSDFUnaryOp()

```
optixSDFUnaryOp::optixSDFUnaryOp ( )  [inline]
```

11.13.2.2 ∼optixSDFUnaryOp()

```
optixSDFUnaryOp::∼optixSDFUnaryOp ( )  [inline]
```

11.13.3 Member Function Documentation

11.13.3.1 AddOpperand()

```
void optixSDFUnaryOp::AddOpperand (
              optixSDFGeometry * sdf )  [inline]
```

11.13.3.2 AdjustCenterAndBoundingBox()

```
virtual void optixSDFUnaryOp::AdjustCenterAndBoundingBox ( )  [inline], [protected], [virtual]
```

11.13.3.3 CreateGeometry()

```
void optixSDFUnaryOp::CreateGeometry ( )  [protected], [virtual]
```

Reimplemented from **optixSDFGeometry** (p. 78).

11.13.3.4 GetInput()

```
optix::Geometry optixSDFUnaryOp::GetInput ( )  [inline]
```

11.13.3.5 GetOutput()

```
virtual optix::Geometry optixSDFUnaryOp::GetOutput ( )  [inline], [virtual]
```

Reimplemented from **optixSDFGeometry** (p. 79).

11.13.3.6 GetOutputSdfObject()

```
optixSDFGeometry* optixSDFUnaryOp::GetOutputSdfObject ( )  [inline]
```

11.13.3.7 InitCallableProg()

```
virtual void optixSDFUnaryOp::InitCallableProg ( )  [inline], [protected], [virtual]
```

11.13.3.8 Initialize()

```
void optixSDFUnaryOp::Initialize ( )  [protected], [virtual]
```

TODO: potentialy read file with points here

Reimplemented from **optixSDFGeometry** (p. 79).

11.13.3.9 SetCallableProg()

```
void optixSDFUnaryOp::SetCallableProg ( )  [protected], [virtual]
```

Is called for setting callable prog

Reimplemented from **optixSDFGeometry** (p. 80).

11.13.3.10 SetContext()

```
void optixSDFUnaryOp::SetContext (
             optix::Context & context )  [virtual]
```

Sets optix context that will store all generated optix geometry

Reimplemented from **optixSDFGeometry** (p. 81).

11.13.3.11 SetInput()

```
virtual void optixSDFUnaryOp::SetInput (
             optix::Geometry geo1 )  [inline], [virtual]
```

11.13.3.12 SetMainPrograms()

```
void optixSDFUnaryOp::SetMainPrograms ( )  [protected], [virtual]
```

sets all main programs

Reimplemented from **optixSDFGeometry** (p. 81).

11.13.3.13 Update()

```
virtual void optixSDFUnaryOp::Update ( )  [inline], [virtual]
```

Runs the process of computation of optix geometry and setting up all necessary cuda programs and geometry parameters

Reimplemented from **optixSDFGeometry** (p. 82).

11.13.4 Member Data Documentation

11.13.4.1 m_sdf

```
optixSDFGeometry* optixSDFUnaryOp::m_sdf  [protected]
```

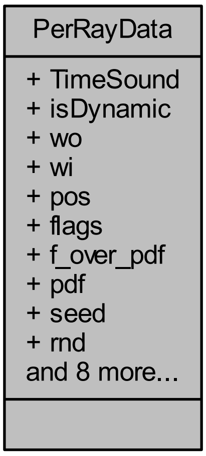
The documentation for this class was generated from the following files:

- **optixSDFBasicOperations.h**
- **optixSDFBasicOperations.cpp**

11.14 PerRayData Struct Reference

```
#include <per_ray_data.h>
```

Collaboration diagram for PerRayData:



Public Attributes

- float **TimeSound**
- int **isDynamic**
- optix::float3 **wo**
- optix::float3 **wi**
- optix::float3 **pos**
- int **flags**
- optix::float3 **f_over_pdf**
- float **pdf**
- unsigned int **seed**
- float **rnd**
- int **depth**
- optix::float3 **radiance**
- optix::float4 **result**
- int **cur_prim**
- bool **isSoundRay**
- int **numS**
- **auditoryPrim** **primitives**
- optix::float3 **Distances** [**MAX_PRIM_ALONG_RAY**]

11.14.1 Member Data Documentation

11.14.1.1 cur_prim

```
int PerRayData::cur_prim
```

11.14.1.2 depth

```
int PerRayData::depth
```

11.14.1.3 Distances

```
optix::float3 PerRayData::Distances[ MAX_PRIM_ALONG_RAY]
```

11.14.1.4 f_over_pdf

```
optix::float3 PerRayData::f_over_pdf
```

11.14.1.5 flags

```
int PerRayData::flags
```

11.14.1.6 isDynamic

```
int PerRayData::isDynamic
```

11.14.1.7 isSoundRay

```
bool PerRayData::isSoundRay
```

11.14.1.8 numS

```
int PerRayData::numS
```

11.14.1.9 pdf

```
float PerRayData::pdf
```

11.14.1.10 pos

```
optix::float3 PerRayData::pos
```

11.14.1.11 primitives

```
auditoryPrim PerRayData::primitives
```

11.14.1.12 radiance

```
optix::float3 PerRayData::radiance
```

11.14.1.13 result

```
optix::float4 PerRayData::result
```

11.14.1.14 rnd

```
float PerRayData::rnd
```

11.14.1.15 seed

```
unsigned int PerRayData::seed
```

11.14.1.16 TimeSound

```
float PerRayData::TimeSound
```

11.14.1.17 wi

```
optix::float3 PerRayData::wi
```

11.14.1.18 wo

```
optix::float3 PerRayData::wo
```

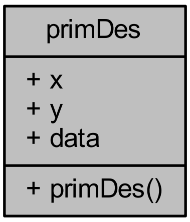
The documentation for this struct was generated from the following file:

- shaders/renderer/ **per_ray_data.h**

11.15 primDes Struct Reference

```
#include <per_ray_data.h>
```

Collaboration diagram for primDes:



Public Member Functions

- **primDes** (int xx, int yy, **auditoryPrim** data_t)

Public Attributes

- int **x**
- int **y**
- **auditoryPrim data**

11.15.1 Constructor & Destructor Documentation

11.15.1.1 primDes()

```
primDes::primDes (
            int xx,
            int yy,
             auditoryPrim data_t )  [inline]
```

11.15.2 Member Data Documentation

11.15.2.1 data

```
auditoryPrim primDes::data
```

11.15.2.2 x

```
int primDes::x
```

11.15.2.3 y

```
int primDes::y
```

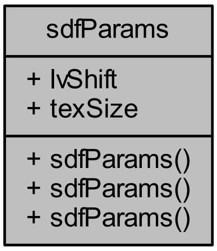
The documentation for this struct was generated from the following file:

- shaders/renderer/ **per_ray_data.h**

11.16 sdfParams Struct Reference

```
#include <renderTypes.h>
```

Collaboration diagram for sdfParams:



Public Member Functions

- **sdfParams** ()
- **sdfParams** (float l, int tex)
- **sdfParams** (**sdfParams** ∗param)

Public Attributes

- float **lvShift**
- int **texSize**

11.16.1 Constructor & Destructor Documentation

11.16.1.1 sdfParams() [1/3]

```
sdfParams::sdfParams ( )  [inline]
```

11.16.1.2 sdfParams() [2/3]

```
sdfParams::sdfParams (
             float l,
             int tex )  [inline]
```

11.16.1.3 sdfParams() [3/3]

```
sdfParams::sdfParams (
              sdfParams * param )  [inline]
```

11.16.2 Member Data Documentation

11.16.2.1 lvShift

```
float sdfParams::lvShift
```

11.16.2.2 texSize

```
int sdfParams::texSize
```

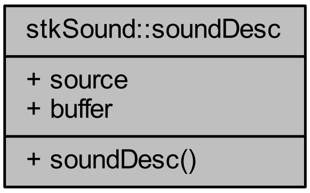
The documentation for this struct was generated from the following file:

- **renderTypes.h**

11.17 stkSound::soundDesc Struct Reference

```
#include <stkSound.h>
```

Collaboration diagram for stkSound::soundDesc:



Public Member Functions

- **soundDesc** (ALuint s, ALuint b)

Public Attributes

- ALuint **source**
- ALuint **buffer**

11.17.1 Constructor & Destructor Documentation

11.17.1.1 soundDesc()

```
stkSound::soundDesc::soundDesc (
            ALuint s,
            ALuint b )  [inline]
```

11.17.2 Member Data Documentation

11.17.2.1 buffer

```
ALuint stkSound::soundDesc::buffer
```

11.17.2.2 source

```
ALuint stkSound::soundDesc::source
```

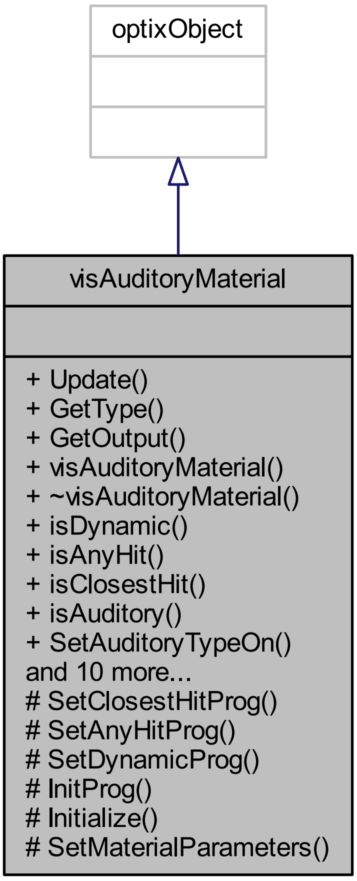
The documentation for this struct was generated from the following file:

- **stkSound.h**

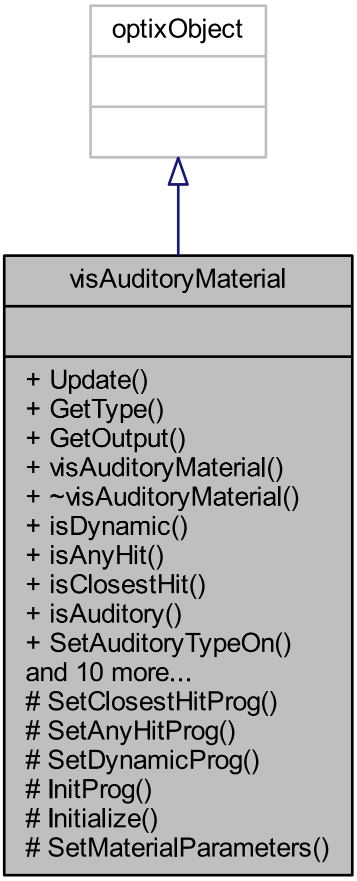
11.18 visAuditoryMaterial Class Reference

```
#include <optixAbstractMaterial.h>
```

Inheritance diagram for visAuditoryMaterial:



Collaboration diagram for visAuditoryMaterial:



Public Member Functions

- virtual void **Update** ()
- **MaterialDesc GetType** ()
- optix::Material **GetOutput** ()
- **visAuditoryMaterial** ()
- ∼**visAuditoryMaterial** ()
- bool **isDynamic** ()
- bool **isAnyHit** ()
- bool **isClosestHit** ()
- bool **isAuditory** ()
- void **SetAuditoryTypeOn** ()
- void **SetAuditoryTypeOff** ()
- void **SetClosestHit** (bool type)
- void **SetAnyHit** (bool type)
- std::string **GetClosestHitProgName** ()
- std::string **GetAnyHitProgName** ()
- std::string **GetDynamicProgName** ()

- void **SetClosestHitProgName** (std::string name)
- void **SetAnyHitProgName** (std::string name)
- void **SetDynamicProgName** (std::string name)
- std::string **GetProgName** (int i)

Protected Member Functions

- void **SetClosestHitProg** (optix::Material m)
- void **SetAnyHitProg** (optix::Material m)
- void **SetDynamicProg** (optix::Material m)
- void **InitProg** (std::string prog, std::string file, std::string name)
- virtual void **Initialize** ()
- virtual void **SetMaterialParameters** ()

11.18.1 Constructor & Destructor Documentation

11.18.1.1 visAuditoryMaterial()

```
visAuditoryMaterial::visAuditoryMaterial ( ) [inline]
```

11.18.1.2 ∼visAuditoryMaterial()

```
visAuditoryMaterial::∼visAuditoryMaterial ( ) [inline]
```

11.18.2 Member Function Documentation

11.18.2.1 GetAnyHitProgName()

```
std::string visAuditoryMaterial::GetAnyHitProgName ( ) [inline]
```

11.18.2.2 GetClosestHitProgName()

```
std::string visAuditoryMaterial::GetClosestHitProgName ( ) [inline]
```

11.18.2.3 GetDynamicProgName()

```
std::string visAuditoryMaterial::GetDynamicProgName ( )  [inline]
```

11.18.2.4 GetOutput()

```
optix::Material visAuditoryMaterial::GetOutput ( )  [inline]
```

11.18.2.5 GetProgName()

```
std::string visAuditoryMaterial::GetProgName (
             int i )  [inline]
```

11.18.2.6 GetType()

```
MaterialDesc visAuditoryMaterial::GetType ( )  [inline]
```

11.18.2.7 Initialize()

```
virtual void visAuditoryMaterial::Initialize ( )  [inline], [protected], [virtual]
```

11.18.2.8 InitProg()

```
void visAuditoryMaterial::InitProg (
             std::string prog,
             std::string file,
             std::string name )  [protected]
```

11.18.2.9 isAnyHit()

```
bool visAuditoryMaterial::isAnyHit ( )  [inline]
```

11.18.2.10 isAuditory()

```
bool visAuditoryMaterial::isAuditory ( )  [inline]
```

11.18.2.11 isClosestHit()

```
bool visAuditoryMaterial::isClosestHit ( )  [inline]
```

11.18.2.12 isDynamic()

```
bool visAuditoryMaterial::isDynamic ( )  [inline]
```

11.18.2.13 SetAnyHit()

```
void visAuditoryMaterial::SetAnyHit (
             bool type )  [inline]
```

11.18.2.14 SetAnyHitProg()

```
void visAuditoryMaterial::SetAnyHitProg (
             optix::Material m )  [protected]
```

11.18.2.15 SetAnyHitProgName()

```
void visAuditoryMaterial::SetAnyHitProgName (
             std::string name )  [inline]
```

11.18.2.16 SetAuditoryTypeOff()

```
void visAuditoryMaterial::SetAuditoryTypeOff ( )  [inline]
```

11.18.2.17 SetAuditoryTypeOn()

```
void visAuditoryMaterial::SetAuditoryTypeOn ( )  [inline]
```

11.18.2.18 SetClosestHit()

```
void visAuditoryMaterial::SetClosestHit (
             bool type )  [inline]
```

11.18.2.19 SetClosestHitProg()

```
void visAuditoryMaterial::SetClosestHitProg (
             optix::Material m )  [protected]
```

11.18.2.20 SetClosestHitProgName()

```
void visAuditoryMaterial::SetClosestHitProgName (
             std::string name )  [inline]
```

11.18.2.21 SetDynamicProg()

```
void visAuditoryMaterial::SetDynamicProg (
             optix::Material m )  [protected]
```

11.18.2.22 SetDynamicProgName()

```
void visAuditoryMaterial::SetDynamicProgName (
             std::string name )  [inline]
```

11.18.2.23 SetMaterialParameters()

```
virtual void visAuditoryMaterial::SetMaterialParameters ( )  [inline], [protected], [virtual]
```

11.18.2.24 Update()

```
void visAuditoryMaterial::Update ( ) [virtual]
```

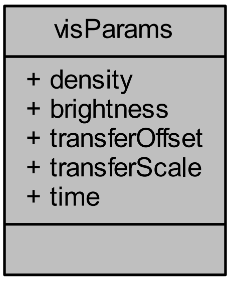
The documentation for this class was generated from the following files:

- **optixAbstractMaterial.h**
- **optixAbstractMaterial.cpp**

11.19 visParams Struct Reference

```
#include <renderTypes.h>
```

Collaboration diagram for visParams:



Public Attributes

- float **density**
- float **brightness**
- float **transferOffset**
- float **transferScale**
- float **time**

11.19.1 Member Data Documentation

11.19.1.1 brightness

```
float visParams::brightness
```

11.19.1.2 density

```
float visParams::density
```

11.19.1.3 time

```
float visParams::time
```

11.19.1.4 transferOffset

```
float visParams::transferOffset
```

11.19.1.5 transferScale

```
float visParams::transferScale
```

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

- **renderTypes.h**

Chapter 12

File Documentation

12.1 doc/approach.md File Reference

12.2 doc/freptutorial.md File Reference

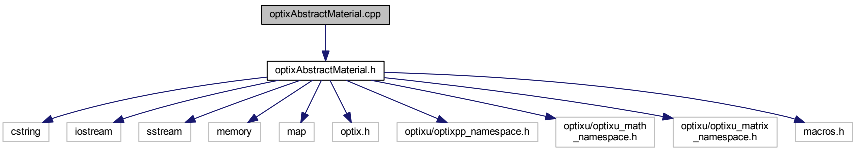
12.3 doc/freptutorial_test.md File Reference

12.4 doc/vtkstyle.md File Reference

12.5 optixAbstractMaterial.cpp File Reference

```
#include "optixAbstractMaterial.h"
```

Include dependency graph for optixAbstractMaterial.cpp:

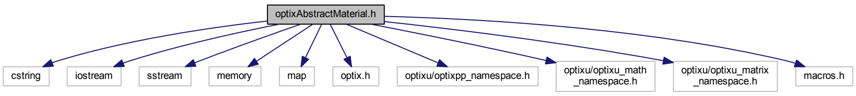


12.6 optixAbstractMaterial.h File Reference

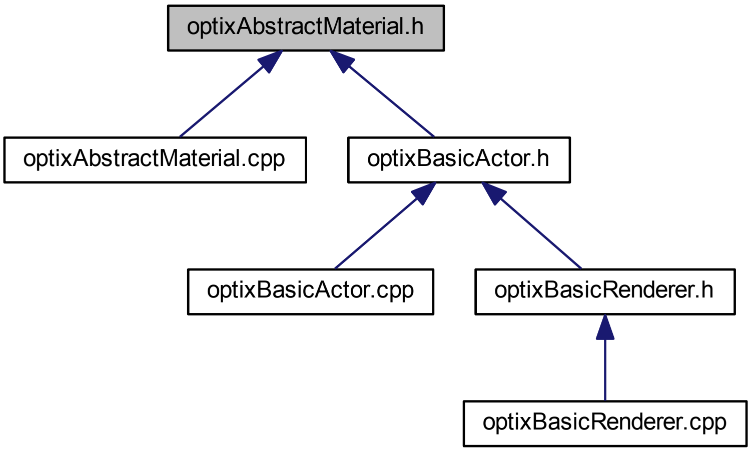
```
#include <cstring>
#include <iostream>
#include <sstream>
#include <memory>
#include <map>
#include <optix.h>
#include <optixu/optixpp_namespace.h>
```

```
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include "macros.h"
```

Include dependency graph for optixAbstractMaterial.h:


optixAbstractMaterial.h
cstring
iostream
sstream
memory
map
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math _namespace.h
optixu/optixu_matrix _namespace.h
macros.h

This graph shows which files directly or indirectly include this file:



Classes

- struct **MaterialDesc**
- class **visAuditoryMaterial**
- class **optixMapper**

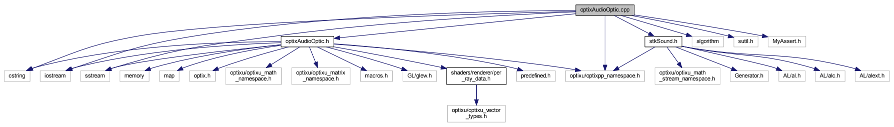
  A basic mapper that suggest application of one visual and one auditory material to one geometry instance.

12.7 optixAudioOptic.cpp File Reference

```
#include "optixAudioOptic.h"
#include <algorithm>
#include <cstring>
#include <iostream>
#include <sstream>
#include <sutil.h>
#include <optixu/optixpp_namespace.h>
#include "MyAssert.h"
```

```
#include "stkSound.h"
```

Include dependency graph for optixAudioOptic.cpp:



Macros

- #define **LOAD_PROC**(d, x, y) ((x) = (y)alcGetProcAddress((d), #x))

12.7.1 Macro Definition Documentation

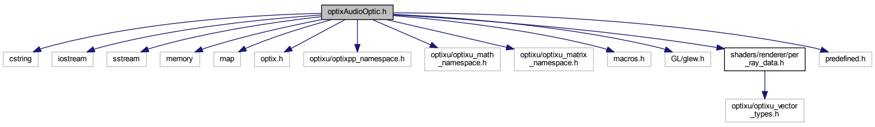
12.7.1.1 LOAD_PROC

```
#define LOAD_PROC(
            d,
            x,
            y ) ((x) = (y)alcGetProcAddress((d), #x))
```

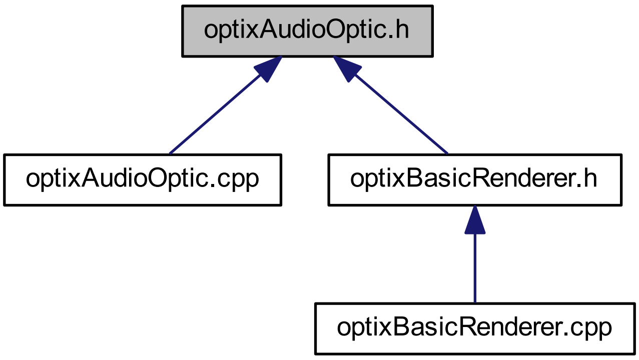
12.8 optixAudioOptic.h File Reference

```
#include <cstring>
#include <iostream>
#include <sstream>
#include <memory>
#include <map>
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include "macros.h"
#include <GL/glew.h>
#include "shaders/renderer/per_ray_data.h"
#include "predefined.h"
```

Include dependency graph for optixAudioOptic.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **auditoryModel**

  Architecture much inspired by VTK Renderer(`http://vtk.org`*) implementation The classes are mainly responsible for openGL/openAL interop.*
- class **basicOpticalModel**
- class **glslRayCast**
- class **opticalModel**

Macros

- #define **optixAudioOptic_h**

12.8.1 Macro Definition Documentation

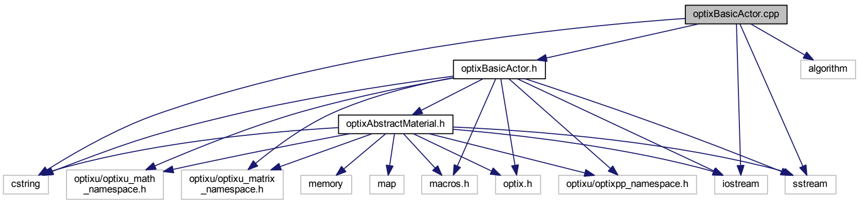
12.8.1.1 optixAudioOptic_h

```
#define optixAudioOptic_h
```

12.9 optixBasicActor.cpp File Reference

```
#include "optixBasicActor.h"
#include <algorithm>
#include <cstring>
#include <iostream>
#include <sstream>
```

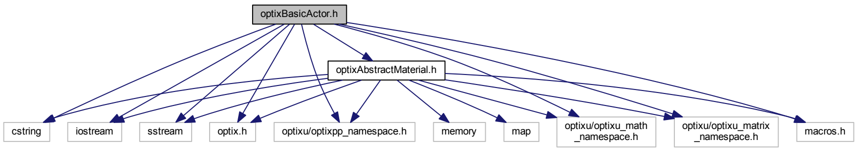
Include dependency graph for optixBasicActor.cpp:



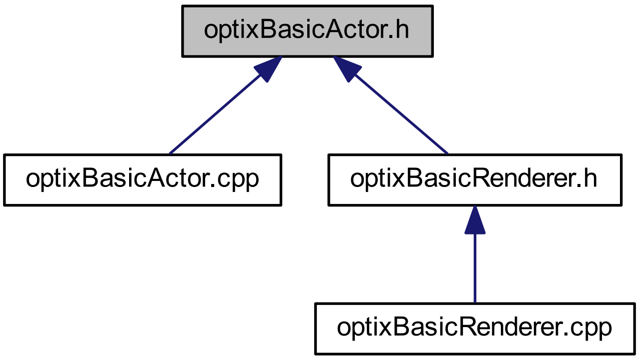
12.10 optixBasicActor.h File Reference

```
#include <cstring>
#include <iostream>
#include <sstream>
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include "macros.h"
#include "optixAbstractMaterial.h"
```

Include dependency graph for optixBasicActor.h:


optixBasicActor.h
optixAbstractMaterial.h
cstring
iostream
sstream
optix.h
optixu/optixpp_namespace.h
memory
map
optixu/optixu_math _namespace.h
optixu/optixu_matrix _namespace.h
macros.h

This graph shows which files directly or indirectly include this file:



Classes

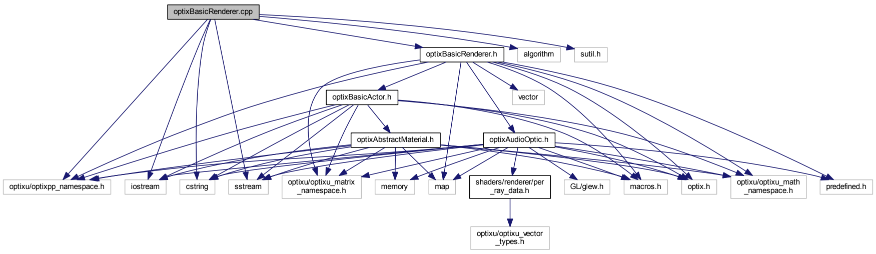
- class **optixBasicActor**

  basic abstract actor class

12.11 optixBasicRenderer.cpp File Reference

```
#include "optixBasicRenderer.h"
#include <algorithm>
#include <cstring>
#include <iostream>
#include <sstream>
#include <sutil.h>
#include <optixu/optixpp_namespace.h>
```

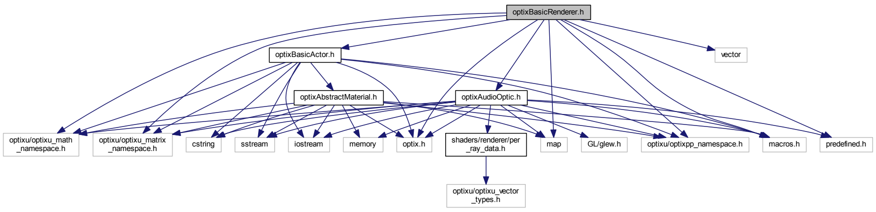
Include dependency graph for optixBasicRenderer.cpp:



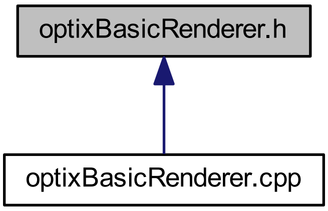
12.12 optixBasicRenderer.h File Reference

```
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include <vector>
#include <map>
#include "macros.h"
#include "predefined.h"
#include "optixAudioOptic.h"
#include "optixBasicActor.h"
```

Include dependency graph for optixBasicRenderer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **optixBasicRenderer**

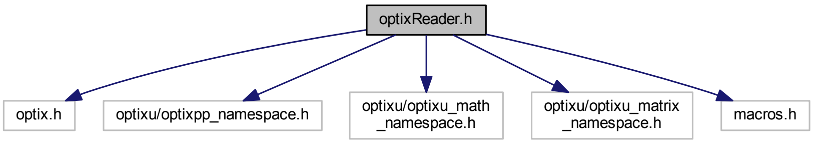
  Architecture much inspired by VTK Renderer(`http://vtk.org`*) implementation Abstract visual-auditory renderer.*

12.13 optixReader.h File Reference

```
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
```

```
#include <optixu/optixu_matrix_namespace.h>
#include "macros.h"
```

Include dependency graph for optixReader.h:


optixReader.h
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math_namespace.h
optixu/optixu_matrix_namespace.h
macros.h

Classes

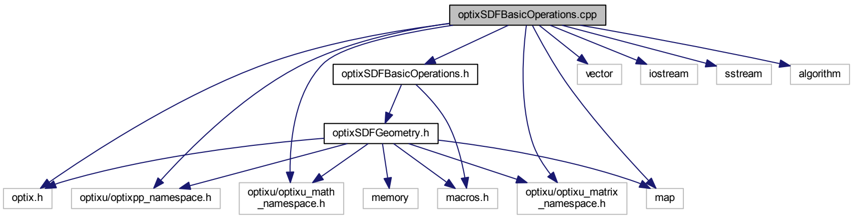
- class **optixReader**< **T** >

  The class creates a basic triangular plane for Optix use. Architecture much inspired by VTK (`http://vtk.org`) implementation.

12.14 optixSDFBasicOperations.cpp File Reference

```
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include <vector>
#include <map>
#include <iostream>
#include <sstream>
#include <algorithm>
#include "optixSDFBasicOperations.h"
```

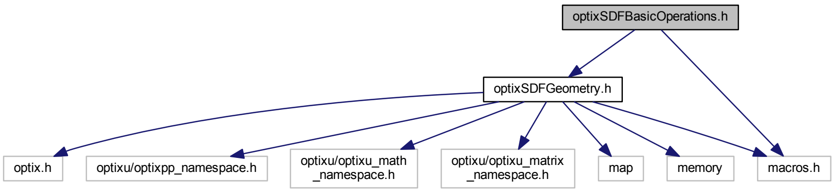
Include dependency graph for optixSDFBasicOperations.cpp:


optixSDFBasicOperations.cpp
optixSDFBasicOperations.h
vector
iostream
sstream
algorithm
optixSDFGeometry.h
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math_namespace.h
memory
macros.h
optixu/optixu_matrix_namespace.h
map

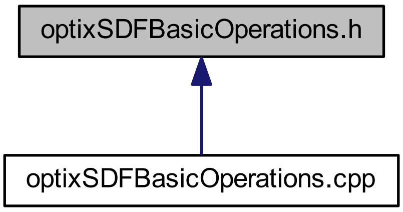
12.15 optixSDFBasicOperations.h File Reference

```
#include "optixSDFGeometry.h"
#include "macros.h"
```

Include dependency graph for optixSDFBasicOperations.h:


optixSDFBasicOperations.h
optixSDFGeometry.h
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math_namespace.h
optixu/optixu_matrix_namespace.h
map
memory
macros.h

This graph shows which files directly or indirectly include this file:



Classes

- class **optixSDFUnaryOp**

  Unary operation.
- class **optixSDFBinaryOp**

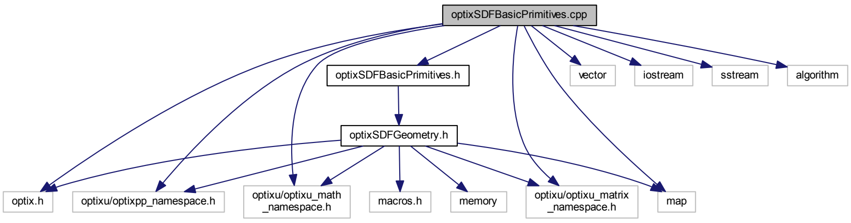
  Binary operation.

12.16 optixSDFBasicPrimitives.cpp File Reference

```
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include <vector>
#include <map>
#include <iostream>
#include <sstream>
#include <algorithm>
```

```
#include "optixSDFBasicPrimitives.h"
```

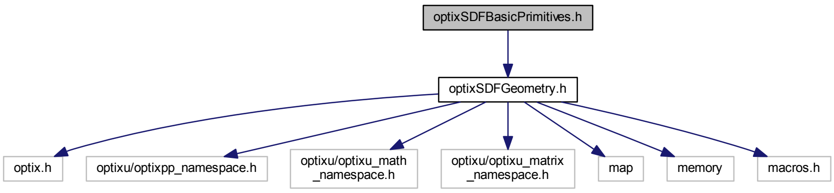
Include dependency graph for optixSDFBasicPrimitives.cpp:


optixSDFBasicPrimitives.cpp
optixSDFBasicPrimitives.h
vector
iostream
sstream
algorithm
optixSDFGeometry.h
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math_namespace.h
macros.h
memory
optixu/optixu_matrix_namespace.h
map

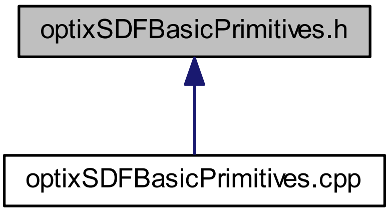
12.17 optixSDFBasicPrimitives.h File Reference

```
#include "optixSDFGeometry.h"
```

Include dependency graph for optixSDFBasicPrimitives.h:


optixSDFBasicPrimitives.h
optixSDFGeometry.h
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math_namespace.h
optixu/optixu_matrix_namespace.h
map
memory
macros.h

This graph shows which files directly or indirectly include this file:



Classes

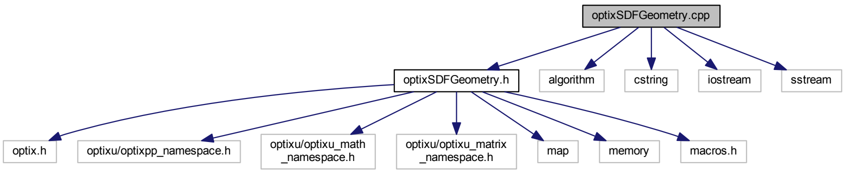
- class **optixSDFPrimitive**

    A basic class for all sdf primitives.

12.18 optixSDFGeometry.cpp File Reference

```
#include "optixSDFGeometry.h"
#include <algorithm>
#include <cstring>
#include <iostream>
#include <sstream>
```

Include dependency graph for optixSDFGeometry.cpp:



Functions

- template<class T >
  RTformat **getFormat** ()
- template<>
  RTformat **getFormat**< **int** > ()

12.18.1 Function Documentation

12.18.1.1 getFormat()

```
template<class T >
RTformat getFormat ( )
```

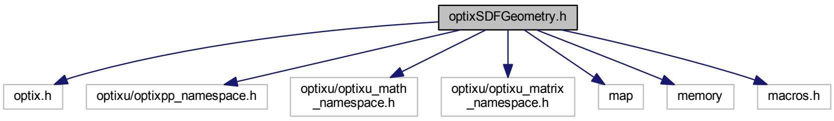
12.18.1.2 getFormat< int >()

```
template<>
RTformat getFormat< int > ( )
```

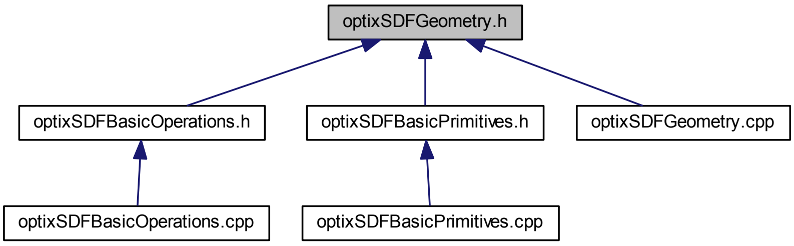
12.19 optixSDFGeometry.h File Reference

```
#include <optix.h>
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_namespace.h>
#include <optixu/optixu_matrix_namespace.h>
#include <map>
#include <memory>
#include "macros.h"
```

Include dependency graph for optixSDFGeometry.h:


optixSDFGeometry.h
optix.h
optixu/optixpp_namespace.h
optixu/optixu_math _namespace.h
optixu/optixu_matrix _namespace.h
map
memory
macros.h

This graph shows which files directly or indirectly include this file:


optixSDFGeometry.h
optixSDFBasicOperations.h
optixSDFBasicPrimitives.h
optixSDFGeometry.cpp
optixSDFBasicOperations.cpp
optixSDFBasicPrimitives.cpp

Classes

- class **optixSDFGeometry**

  Abstract class for SDF Geometry. A basic class. All SDF geometry inherits it's functions.

12.20 Readme.md File Reference

12.21 renderTypes.h File Reference

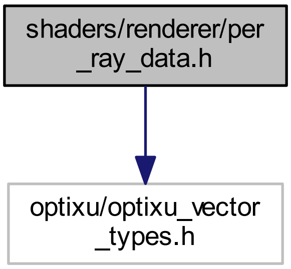
Classes

- struct **sdfParams**
- struct **visParams**

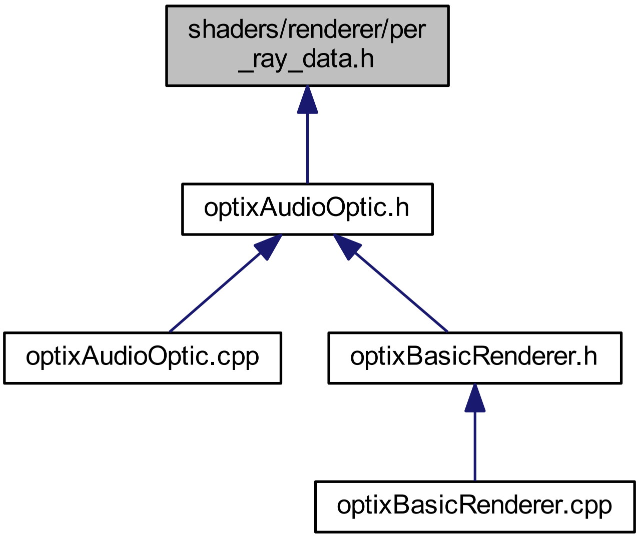
12.22 shaders/renderer/per_ray_data.h File Reference

```
#include <optixu/optixu_vector_types.h>
```

Include dependency graph for per_ray_data.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct **primDes**
- struct **PerRayData**

Macros

- #define **PER_RAY_DATA_H**

Typedefs

- typedef optix::float2 **auditoryPrim**[**MAX_PRIM_ALONG_RAY**]

Variables

- const int **MAX_PRIM_ALONG_RAY** = 10

12.22.1 Macro Definition Documentation

12.22.1.1 PER_RAY_DATA_H

```
#define PER_RAY_DATA_H
```

12.22.2 Typedef Documentation

12.22.2.1 auditoryPrim

```
typedef optix::float2 auditoryPrim[ MAX_PRIM_ALONG_RAY]
```

12.22.3 Variable Documentation

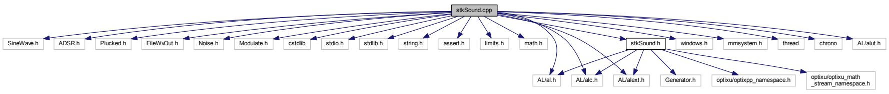
12.22.3.1 MAX_PRIM_ALONG_RAY

```
const int MAX_PRIM_ALONG_RAY = 10
```

12.23 stkSound.cpp File Reference

```
#include "SineWave.h"
#include "ADSR.h"
#include "Plucked.h"
#include "FileWvOut.h"
#include "Noise.h"
#include "Modulate.h"
#include <cstdlib>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <limits.h>
#include <math.h>
#include "AL/al.h"
#include "AL/alc.h"
#include "AL/alext.h"
#include <windows.h>
#include <mmsystem.h>
#include <thread>
#include <chrono>
#include "AL/alut.h"
#include "stkSound.h"
```

Include dependency graph for stkSound.cpp:



Namespaces

- **stkSound**

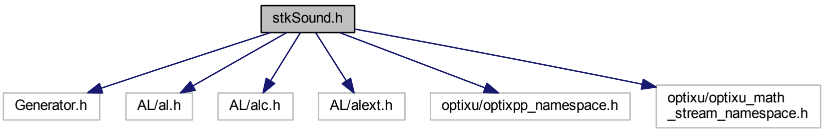
Functions

- int **stkSound::InitAL** (char ∗∗∗argv, int ∗argc)
- void **stkSound::CloseAL** (void)
- void **stkSound::ApplySin** (ALfloat ∗data, ALdouble g, ALuint srate, ALuint freq)
- void **stkSound::CreateWave** (StkFloat ∗ptr, int srate2, int duration, ALuint &buffer)
- void **stkSound::InitEnv** ()
- void **stkSound::ExitEnv** ()
- StkFrames **stkSound::GenerateNoise** (double time, double srate2, float seed)
- void **stkSound::WaitforSound** (float scale)
- void **stkSound::PlaySelSound** (ALuint source)
- void **stkSound::MoveSounds** (std::vector< **stkSound::soundDesc** > &descSs, std::vector< float > &xpos, int scale)
- void **stkSound::RotateTest** (int scale)
- soundDesc **stkSound::PlayBuffer** (StkFloat ∗ptr, int time, int size, int j, optix::float2 xpos)
- void **stkSound::RotateAngle** (double angle, ALuint &source)
- void **stkSound::ApplyADSR** (stk::StkFrames &frames, int releaseCount)
- StkFrames **stkSound::GeneratePlucked** (int time, double srate2, double freq, double amplitude)

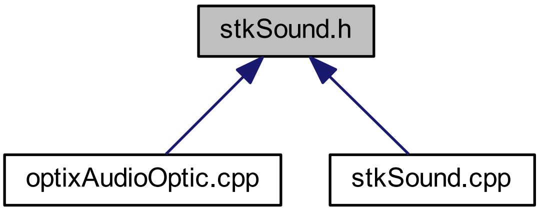
12.24 stkSound.h File Reference

```
#include "Generator.h"
#include "AL/al.h"
#include "AL/alc.h"
#include "AL/alext.h"
#include <optixu/optixpp_namespace.h>
#include <optixu/optixu_math_stream_namespace.h>
```

Include dependency graph for stkSound.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct **stkSound::soundDesc**

Namespaces

- **stkSound**

Functions

- StkFrames **stkSound::GeneratePlucked** (int time, double srate2, double freq, double amplitude)
- StkFrames **stkSound::GenerateNoise** (double time, double srate2, float seed)
- void **stkSound::ApplyADSR** (stk::StkFrames &frames, int releaseCount)
- soundDesc **stkSound::PlayBuffer** (stk::StkFloat ∗ptr, int time, int size, int j, optix::float2 xpos)
- void **stkSound::InitEnv** ()
- void **stkSound::ExitEnv** ()

- void **stkSound::RotateAngle** (double angle, ALuint &source)
- void **stkSound::PlaySelSound** (ALuint source)
- void **stkSound::MoveSounds** (std::vector< **stkSound::soundDesc** > &descSs, std::vector< float > &xpos, int scale)
- void **stkSound::WaitforSound** (float scale)
- void **stkSound::RotateTest** (int scale)

Index