YAGE

1.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

YAGE (Yet Another Game Engine)

This is a C++ based 3D game engine targeted for Windows.

Requirements

- Microsoft Visual Studio 2015 (sorry, will port to gcc later)

Getting Started

- Clone the repository
- Open `YAGE.sln` file with visual studio
- Install missing dependancies:
  - `Tools > Nuget Package Manager > Package Manager Console`
  - Type this into the the console: `Update-Package -Reinstall`
- Edit `main.cpp` and play with the game code
- Run the default startup project `F10`

Documentation

If you are viewing this on Github, you can find our documentation here: `http://harrygogonis.github.io/YAGE`

Screenshots

Viewport Controls

Control	Action
W	Moves the camera forward.
S	Moves the camera backward.
A	Moves the camera left.
D	Moves the camera right.
E	Moves the camera up.
Q	Moves the camera down.
Mouse Scroll DOWN	Zooms the camera out (raises FOV).
Mouse Scroll UP	Zooms the camera in (lowers FOV).

3rd Party Libraries

- ASSIMP
- SOIL
- Bullet
- Freeglut
- OpenGL
- GLM

Features

- Lighting system (Ambient, Point, Directional, Spotlight)
- Physics system (using bullet physics)
- Diffuse mapping
- Normal mapping
- Specular mapping
- Import scenes via .fbx, .obj, and more
- Game object builder
- Dynamic shadows
- Particle system

Limitations

- Significant frame drop after ~500k polygons
- Shadows only supported w/ one directional light

Future Features

- Animation
- Text rendering

Helpful resources

- in2gpu OpenGL Tutorial
- OpenGL Documentation
- Red Book
- OpenGL-Tutorial
- OGLDev

Authors

- Harry Gogonis
- Dylan Richardson

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

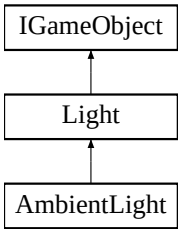
Class Documentation

4.1 AmbientLight Class Reference

An ambient light. Uniformly lights up everything.

`#include <Light.h>`

Inheritance diagram for AmbientLight:



Public Member Functions

- **AmbientLight** (glm::vec3 **color**, float strength=1.0)
- virtual void **Draw** (GLuint) override final

  Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Additional Inherited Members

4.1.1 Detailed Description

An ambient light. Uniformly lights up everything.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AmbientLight::AmbientLight (glm::vec3 *color,* float *strength* = `1.0`)

4.1.3 Member Function Documentation

4.1.3.1 void AmbientLight::Draw (GLuint *program*) `[final],[override],[virtual]`

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Parameters

program	The shader program.

Reimplemented from **Light** (p. 22).

4.2 Camera Class Reference

The game camera implemented as a Singleton. Primary use is to generate Model-View-Projection matrices.

`#include <Camera.h>`

Public Member Functions

- **Camera** (**Camera** const &)=delete
- void **operator=** (**Camera** const &)=delete

Static Public Member Functions

- static **Camera** & **GetInstance** ()
- static glm::mat4 **GetProjectionMatrix** ()

  Gets the camera projection matrix.
- static glm::mat4 **GetViewMatrix** ()

  Gets the camera view matrix.
- static glm::vec3 **GetEyeDirection** ()

  Gets the direction the camera is looking at.
- static void **resizeWindow** (float width, float height)
- static void **ComputeMatrices** ()

Static Public Attributes

- static float **fov** = 45.0f

  The field of view.
- static float **aspect**

  The aspect ratio.
- static float **zNear**
- static float **zFar**

4.2.1 Detailed Description

The game camera implemented as a Singleton. Primary use is to generate Model-View-Projection matrices.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Camera::Camera (Camera const &) [delete]

4.2.3 Member Function Documentation

4.2.3.1 void Camera::ComputeMatrices ( ) [static]

4.2.3.2 static glm::vec3 Camera::GetEyeDirection ( ) [static]

Gets the direction the camera is looking at.

Returns

The eye direction.

4.2.3.3 static Camera& Camera::GetInstance ( ) [inline], [static]

4.2.3.4 static glm::mat4 Camera::GetProjectionMatrix ( ) [static]

Gets the camera projection matrix.

Returns

The projection matrix.

4.2.3.5 static glm::mat4 Camera::GetViewMatrix ( ) [static]

Gets the camera view matrix.

Returns

The view matrix.

4.2.3.6 void Camera::operator= (Camera const &) [delete]

4.2.3.7 void Camera::resizeWindow (float *width,* float *height*) [static]

4.2.4 Member Data Documentation

4.2.4.1 float Camera::aspect [static]

The aspect ratio.

4.2.4.2 float Camera::fov = 45.0f [static]

The field of view.

4.2.4.3 float Camera::zFar [static]

4.2.4.4 float Camera::zNear [static]

4.3 ContextInfo Struct Reference

`#include <ContextInfo.h>`

Public Member Functions

- **ContextInfo** ()
- **ContextInfo** (int **major_version**, int **minor_version**, bool **core**)
- void **operator=** (const **ContextInfo** &info)

Public Attributes

- int **major_version**
- int **minor_version**
- bool **core**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 ContextInfo::ContextInfo ( ) [inline]

4.3.1.2 ContextInfo::ContextInfo (int *major_version,* int *minor_version,* bool *core*) [inline]

4.3.2 Member Function Documentation

4.3.2.1 void ContextInfo::operator= (const ContextInfo & *info*) [inline]

4.3.3 Member Data Documentation

4.3.3.1 bool ContextInfo::core

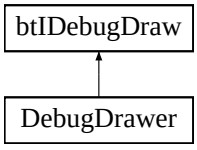
4.3.3.2 int ContextInfo::major_version

4.3.3.3 int ContextInfo::minor_version

4.4 DebugDrawer Class Reference

`#include <DebugDrawer.h>`

Inheritance diagram for DebugDrawer:



Public Member Functions

- **DebugDrawer** ()
- ∼**DebugDrawer** ()
- void **drawLine** (const btVector3 &from, const btVector3 &to, const btVector3 &color) override
- void **drawContactPoint** (const btVector3 &PointOnB, const btVector3 &normalOnB, btScalar distance, int lifeTime, const btVector3 &color) override
- void **reportErrorWarning** (const char ∗warningString) override
- void **draw3dText** (const btVector3 &location, const char ∗textString) override
- void **setDebugMode** (int debugMode) override
- int **getDebugMode** () const override
- void **clearBuffers** ()
- void **render** ()

4.4.1 Constructor & Destructor Documentation

4.4.1.1 DebugDrawer::DebugDrawer ( )

4.4.1.2 DebugDrawer::∼DebugDrawer ( )

4.4.2 Member Function Documentation

4.4.2.1 void DebugDrawer::clearBuffers ( )

4.4.2.2 void DebugDrawer::draw3dText (const btVector3 & *location,* const char ∗ *textString*) [override]

4.4.2.3 void DebugDrawer::drawContactPoint (const btVector3 & *PointOnB,* const btVector3 & *normalOnB,* btScalar *distance,* int *lifeTime,* const btVector3 & *color*) [override]

4.4.2.4 void DebugDrawer::drawLine (const btVector3 & *from,* const btVector3 & *to,* const btVector3 & *color*) [override]

4.4.2.5 int DebugDrawer::getDebugMode ( ) const [override]

4.4.2.6 void DebugDrawer::render ( )

4.4.2.7 void DebugDrawer::reportErrorWarning (const char ∗ *warningString*) [override]

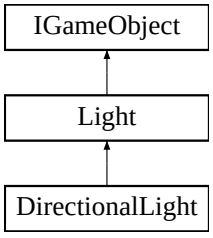
4.4.2.8 void DebugDrawer::setDebugMode (int *debugMode*) [override]

4.5 DirectionalLight Class Reference

A directional light. Comes from "infinity" and lights up anything in the direction it is coming from.

`#include <Light.h>`

Inheritance diagram for DirectionalLight:



Public Member Functions

- **DirectionalLight** (glm::vec3 **color**, glm::vec3 **position**, glm::vec3 **halfVector**)
- virtual void **Draw** (GLuint) override final

  Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.
- virtual void **DrawShadow** (GLuint) override final

  Sends shadow information. Each subclass must implement this functionality!

Additional Inherited Members

4.5.1 Detailed Description

A directional light. Comes from "infinity" and lights up anything in the direction it is coming from.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DirectionalLight::DirectionalLight (glm::vec3 *color,* glm::vec3 *position,* glm::vec3 *halfVector*)

4.5.3 Member Function Documentation

4.5.3.1 void DirectionalLight::Draw (GLuint *program*) [final], [override], [virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Parameters

program	The shader program.

Reimplemented from **Light** (p. 22).

4.5.3.2 void DirectionalLight::DrawShadow (GLuint) [final], [override], [virtual]

Sends shadow information. Each subclass must implement this functionality!

Parameters

program	The shader program.

Reimplemented from **Light** (p. 23).

4.6 DVertexFormat Struct Reference

`#include <DebugDrawer.h>`

Public Member Functions

- **DVertexFormat** (const glm::vec3 &inPos, const glm::vec3 &inColor)

Public Attributes

- glm::vec3 **position**
- glm::vec3 **color**

4.6.1 Constructor & Destructor Documentation

4.6.1.1 DVertexFormat::DVertexFormat (const glm::vec3 & *inPos,* const glm::vec3 & *inColor*) [inline]

4.6.2 Member Data Documentation

4.6.2.1 glm::vec3 DVertexFormat::color

4.6.2.2 glm::vec3 DVertexFormat::position

4.7 FrameBufferInfo Struct Reference

`#include <FrameBufferInfo.h>`

Public Member Functions

- **FrameBufferInfo** ()
- **FrameBufferInfo** (bool color, bool depth, bool stencil, bool **msaa**)

Public Attributes

- unsigned int **flags**
- bool **msaa**

4.7.1 Constructor & Destructor Documentation

4.7.1.1 FrameBufferInfo::FrameBufferInfo ( ) [inline]

4.7.1.2 FrameBufferInfo::FrameBufferInfo (bool *color,* bool *depth,* bool *stencil,* bool *msaa*) [inline]

4.7.2 Member Data Documentation

4.7.2.1 unsigned int FrameBufferInfo::flags

4.7.2.2 bool FrameBufferInfo::msaa

4.8 GameObjectsBuilder Class Reference

`#include <GameObjectsBuilder.h>`

Public Member Functions

- **GameObjectsBuilder** ()
- ∼**GameObjectsBuilder** ()
- **GameObjectsBuilder** & **addModel** (const std::string modelPath, const std::string texturePath=std::string())
- **GameObjectsBuilder** & **copyModel** ()
- **GameObjectsBuilder** & **setPosition** (glm::vec3)
- **GameObjectsBuilder** & **setScale** (float)
- **GameObjectsBuilder** & **setRotation** (float angleX, float angleY, float angleZ)
- **GameObjectsBuilder** & **setDiffuse** (const std::string &)
- **GameObjectsBuilder** & **setNormal** (const std::string &)
- **GameObjectsBuilder** & **setSpecular** (const std::string &)
- **GameObjectsBuilder** & **addRigidBody** (float mass)
- **GameObjectsBuilder** & **lockUpright** ()
- **GameObjectsBuilder** & **addParticleSystem** (const std::string &texturePath)
- **GameObjectsBuilder** & **setParticleCount** (int)
- **GameObjectsBuilder** & **setParticleSpawnRate** (int)
- **GameObjectsBuilder** & **setParticleLife** (float)
- **GameObjectsBuilder** & **addLight** (GameObjectType type)
- **GameObjectsBuilder** & **setColor** (glm::vec3 color)
- **GameObjectsBuilder** & **setHalfVector** (glm::vec3 halfVector)
- **GameObjectsBuilder** & **setCastsShadows** (bool)
- **GameObjectsBuilder** & **setStrength** (float strength)
- **GameObjectsBuilder** & **setAttenuation** (float constant, float linear, float quadratic)
- **GameObjectsBuilder** & **setConeDirection** (glm::vec3 coneDirection)
- **GameObjectsBuilder** & **setSpotCutoff** (float cutoff)
- **GameObjectsBuilder** & **setSpotExponent** (float exponent)
- **Models_Manager** ∗ **getResult** () const

4.8.1 Constructor & Destructor Documentation

4.8.1.1 GameObjectsBuilder::GameObjectsBuilder ( )

4.8.1.2 GameObjectsBuilder::∼GameObjectsBuilder ( )

4.8.2 Member Function Documentation

4.8.2.1 GameObjectsBuilder & GameObjectsBuilder::addLight (GameObjectType *type*)

4.8.2.2 GameObjectsBuilder & GameObjectsBuilder::addModel (const std::string *modelPath,* const std::string *texturePath* = `std::string()`)

4.8.2.3 GameObjectsBuilder & GameObjectsBuilder::addParticleSystem (const std::string & *texturePath*)

4.8.2.4 GameObjectsBuilder & GameObjectsBuilder::addRigidBody (float *mass*)

4.8.2.5 GameObjectsBuilder & GameObjectsBuilder::copyModel ( )

4.8.2.6 Models_Manager ∗ GameObjectsBuilder::getResult ( ) const

4.8.2.7 GameObjectsBuilder & GameObjectsBuilder::lockUpright ( )

4.8.2.8 GameObjectsBuilder & GameObjectsBuilder::setAttenuation (float *constant,* float *linear,* float *quadratic*)

4.8.2.9 GameObjectsBuilder & GameObjectsBuilder::setCastsShadows (bool *castsShadow*)

4.8.2.10 GameObjectsBuilder & GameObjectsBuilder::setColor (glm::vec3 *color*)

4.8.2.11 GameObjectsBuilder & GameObjectsBuilder::setConeDirection (glm::vec3 *coneDirection*)

4.8.2.12 GameObjectsBuilder & GameObjectsBuilder::setDiffuse (const std::string & *path*)

4.8.2.13 GameObjectsBuilder & GameObjectsBuilder::setHalfVector (glm::vec3 *halfVector*)

4.8.2.14 GameObjectsBuilder & GameObjectsBuilder::setNormal (const std::string & *path*)

4.8.2.15 GameObjectsBuilder & GameObjectsBuilder::setParticleCount (int *n*)

4.8.2.16 GameObjectsBuilder & GameObjectsBuilder::setParticleLife (float *seconds*)

4.8.2.17 GameObjectsBuilder & GameObjectsBuilder::setParticleSpawnRate (int *rate*)

4.8.2.18 GameObjectsBuilder & GameObjectsBuilder::setPosition (glm::vec3 *pos*)

4.8.2.19 GameObjectsBuilder & GameObjectsBuilder::setRotation (float *angleX,* float *angleY,* float *angleZ*)

4.8.2.20 GameObjectsBuilder & GameObjectsBuilder::setScale (float *scale*)

4.8.2.21 GameObjectsBuilder & GameObjectsBuilder::setSpecular (const std::string & *path*)

4.8.2.22 GameObjectsBuilder & GameObjectsBuilder::setSpotCutoff (float *cutoff*)

4.8.2.23 GameObjectsBuilder & GameObjectsBuilder::setSpotExponent (float *exponent*)

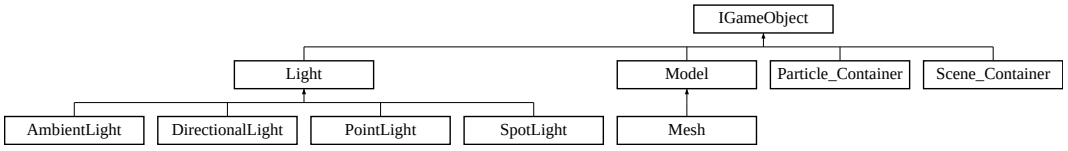
4.8.2.24 GameObjectsBuilder & GameObjectsBuilder::setStrength (float *strength*)

4.9 IGameObject Class Reference

A Game Object. All objects that are to be rendered will inherit this class.

`#include <IGameObject.h>`

Inheritance diagram for IGameObject:



Public Member Functions

- virtual ~**IGameObject** ()

  Destructor that deletes the unique shader program and shadow program belonging to the object.
- virtual void **Draw** (GLuint shader)=0

  Draws via the given shader.
- virtual void **DrawShadow** (GLuint shader)=0

  Draw shadows via the given shader.
- virtual void **Update** ()=0

  Updates this object.
- virtual void **Destroy** ()=0

  Destroys this object.
- virtual void **SetProgram** (GLuint shaderName)

  Sets the main shader program corresponding to this object.
- virtual void **SetShadowProgram** (GLuint shaderName)

  Sets the shadow pass shader program corresponding to this object.
- virtual GLuint **GetProgram** ()

  Gets the object's main shader program.
- virtual GLuint **GetShadowProgram** ()

  Gets the object's shadow program.
- virtual void **SetTexture** (const std::string &textureName, const TextureType &textureType, const GLuint &texture)=0

  Adds a texture to an object.
- virtual const GLuint **GetTexture** (const std::string &textureName) const =0

  Gets a texture.

Protected Attributes

- GLuint **program** = 0
- GLuint **shadowProgram** = 0

4.9.1 Detailed Description

A Game Object. All objects that are to be rendered will inherit this class.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 IGameObject::~IGameObject ( ) [inline], [virtual]

Destructor that deletes the unique shader program and shadow program belonging to the object.

4.9.3 Member Function Documentation

4.9.3.1 void IGameObject::Destroy ( ) [pure virtual]

Destroys this object.

Implemented in **Light** (p. 22), **Scene_Container** (p. 40), **Model** (p. 30), and **Particle_Container** (p. 35).

4.9.3.2 void IGameObject::Draw (GLuint *shader*) [pure virtual]

Draws via the given shader.

Parameters

shader	The shader.

Implemented in **SpotLight** (p. 47), **PointLight** (p. 38), **AmbientLight** (p. 7), **DirectionalLight** (p. 12), **Light** (p. 22), **Scene_Container** (p. 40), **Mesh** (p. 28), and **Particle_Container** (p. 35).

4.9.3.3 void IGameObject::DrawShadow (GLuint *shader*) [pure virtual]

Draw shadows via the given shader.

Implemented in **DirectionalLight** (p. 12), **Light** (p. 23), **Scene_Container** (p. 40), **Mesh** (p. 29), and **Particle_Container** (p. 36).

4.9.3.4 GLuint IGameObject::GetProgram ( ) [inline], [virtual]

Gets the object's main shader program.

Returns

The main shader program.

4.9.3.5 GLuint IGameObject::GetShadowProgram ( ) [inline], [virtual]

Gets the object's shadow program.

Returns

The shadow program.

4.9.3.6 const GLuint IGameObject::GetTexture (const std::string & *textureName*) const [pure virtual]

Gets a texture.

Parameters

textureName	The name of the texture.

Returns

The texture.

Implemented in **Light** (p. 23), **Scene_Container** (p. 40), **Model** (p. 30), and **Particle_Container** (p. 36).

4.9.3.7 void IGameObject::SetProgram (GLuint *shaderName*) [inline], [virtual]

Sets the main shader program corresponding to this object.

Author

Harry

Date

4/14/2016

Parameters

shaderName	Name of the shader.

Reimplemented in **Scene_Container** (p. 41), and **Model** (p. 31).

4.9.3.8 void IGameObject::SetShadowProgram (GLuint *shaderName*) [inline], [virtual]

Sets the shadow pass shader program corresponding to this object.

Parameters

shaderName	Name of the shader.

Reimplemented in **Scene_Container** (p. 41), and **Model** (p. 31).

4.9.3.9 void IGameObject::SetTexture (const std::string & *textureName,* const TextureType & *textureType,* const GLuint & *texture*) [pure virtual]

Adds a texture to an object.

Parameters

textureName	Name of the texture.
textureType	Type of the texture.
texture	The texture id.

Implemented in **Light** (p. 24), **Scene_Container** (p. 42), **Model** (p. 32), and **Particle_Container** (p. 36).

4.9.3.10 void IGameObject::Update ( ) [pure virtual]

Updates this object.

Implemented in **Light** (p. 24), **Scene_Container** (p. 42), **Mesh** (p. 29), and **Particle_Container** (p. 37).

4.9.4 Member Data Documentation

4.9.4.1 GLuint IGameObject::program = 0 [protected]

4.9.4.2 GLuint IGameObject::shadowProgram = 0 [protected]

4.10 Init_GLEW Class Reference

Helper class to initalize GLEW.

`#include <Init_GLEW.h>`

Public Member Functions

- **Init_GLEW** ()
- ~**Init_GLEW** ()

Static Public Member Functions

- static void **Init** ()

4.10.1 Detailed Description

Helper class to initalize GLEW.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Init_GLEW::Init_GLEW ( )

4.10.2.2 Init_GLEW::~Init_GLEW ( )

4.10.3 Member Function Documentation

4.10.3.1 void Init_GLEW::Init ( ) [static]

4.11 Init_GLUT Class Reference

Helper class to initalize GLUT and setup listener callbacks.

`#include <Init_GLUT.h>`

Public Member Functions

- void **enterFullscreen** ()
- void **exitFullscreen** ()

Static Public Member Functions

- static void **init** (const **WindowInfo** &, const **ContextInfo** &, const **FrameBufferInfo** &)
  Initalizes GLUT with a window.
- static void **run** ()
- static void **close** ()
- static void **printOpenGLInfo** (const **WindowInfo** &window, const **ContextInfo** &context)
- static void **setListener** (**Scene_Manager** *&iListener)

4.11.1 Detailed Description

Helper class to initalize GLUT and setup listener callbacks.

4.11.2 Member Function Documentation

4.11.2.1 void Init_GLUT::close ( ) [static]

4.11.2.2 void Init_GLUT::enterFullscreen ( )

4.11.2.3 void Init_GLUT::exitFullscreen ( )

4.11.2.4 static void Init_GLUT::init (const WindowInfo & *windowInfo,* const ContextInfo & *contextInfo,* const FrameBufferInfo & *frameBufferInfo*) [static]

Initalizes GLUT with a window.

4.11.2.5 void Init_GLUT::printOpenGLInfo (const WindowInfo & *window,* const ContextInfo & *context*) [static]

4.11.2.6 void Init_GLUT::run ( ) [static]

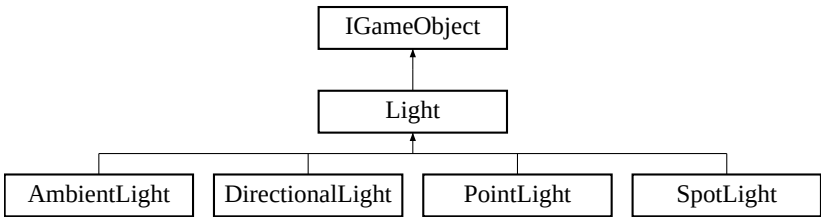
4.11.2.7 void Init_GLUT::setListener (Scene_Manager *& *iListener*) [static]

4.12 Light Class Reference

A light is a special class of Game Object that mostly just sends uniform data to the shaders. Lights are fundamental to the shading pipeline! Without a light, nothing will show up in the scene.

`#include <Light.h>`

Inheritance diagram for Light:



Classes

- struct **LightUniformLocations**

  *Holds the various uniform location ID's of the **Light** (p. 20).*

Public Member Functions

- **Light** ()
- ~**Light** ()
- virtual void **Draw** (GLuint) override

  Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.
- virtual void **DrawShadow** (GLuint) override

  Sends shadow information. Each subclass must implement this functionality!
- virtual void **Update** () override

  *Updates the light. This currently does nothing! **Draw()** (p. 22) does update the light.*
- virtual void **Destroy** () override

  Destroys the light. This currently only turns off the enabled flag!
- virtual void **SetTexture** (const std::string &, const TextureType &, const GLuint &) override

  Adds a texture to an object.
- virtual const GLuint **GetTexture** (const std::string &) const override

  Gets a texture.
- bool **EnableShadows** ()

  Enables shadow generation for this light.
- bool **DisableShadows** ()

  Disables shadow generation for this light.
- void **SetAttenuation** (float constant=0.0f, float linear=1.0f, float quadratic=0.0f)

  Sets the attenutation parameters. See this wiki article for more information about attenuation: `https://developer.valvesoftware.com/wiki/Constant-Linear-Quadratic_Falloff`.

Public Attributes

- glm::vec3 **ambient**

  light's contribution to ambient light.
- glm::vec3 **color**

  The color/intensity of the light.
- glm::vec3 **position**

  location of light if isLocal is true. For directional light, it is the direction
- glm::vec3 **halfVector**

  ?? direction of headlights for directional light ??
- glm::vec3 **coneDirection**

  spotlight cone direction.
- float **spotCosCutoff**

  how wide the spotlight is [0-1].
- float **spotExponent**

  control light fall-off in the spotlight.
- float **constantAttenuation**

  constant light fall-out
- float **linearAttenuation**

  linear light fall-off
- float **quadraticAttenuation**

  quadratic light fall-out
- bool **isEnabled**

  true if this object is enabled.

Protected Attributes

- GLuint **texture**
- unsigned int **_id**
- bool **castsShadow**
- int **type**
- struct **Light::LightUniformLocations ids**

4.12.1 Detailed Description

A light is a special class of Game Object that mostly just sends uniform data to the shaders. Lights are fundamental to the shading pipeline! Without a light, nothing will show up in the scene.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Light::Light ( )

4.12.2.2 Light::~Light ( ) [inline]

4.12.3 Member Function Documentation

4.12.3.1 void Light::Destroy ( ) [override], [virtual]

Destroys the light. This currently only turns off the enabled flag!

Implements **IGameObject** (p. 16).

4.12.3.2 bool Light::DisableShadows ( )

Disables shadow generation for this light.

Returns

true if it succeeds, false if it fails.

4.12.3.3 void Light::Draw (GLuint *program*) [override], [virtual]

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Parameters

program	The shader program.

Implements **IGameObject** (p. 16).

Reimplemented in **SpotLight** (p. 47), **PointLight** (p. 38), **AmbientLight** (p. 7), and **DirectionalLight** (p. 12).

4.12.3.4 void Light::DrawShadow (GLuint) [override], [virtual]

Sends shadow information. Each subclass must implement this functionality!

Parameters

program	The shader program.

Implements **IGameObject** (p. 17).

Reimplemented in **DirectionalLight** (p. 12).

4.12.3.5 bool Light::EnableShadows ( )

Enables shadow generation for this light.

Returns

true if it succeeds, false if it fails.

4.12.3.6 const GLuint Light::GetTexture (const std::string & *textureName*) const [override], [virtual]

Gets a texture.

Parameters

textureName	The name of the texture.

Returns

The texture.

Implements **IGameObject** (p. 17).

4.12.3.7 void Light::SetAttenuation (float *constant* = 0.0f, float *linear* = 1.0f, float *quadratic* = 0.0f)

Sets the attenutation parameters. See this wiki article for more information about attenuation: https://developer.valvesoftware.com/wiki/Constant-Linear-Quadratic_Falloff.

Author

Harry

Date

4/14/2016

Parameters

constant	The constant attenuation factor.
linear	The linear attenuation factor.
quadratic	The quadratic attenuation factor.

4.12.3.8 void Light::SetTexture (const std::string & *textureName,* const TextureType & *textureType,* const GLuint & *texture*) [override], [virtual]

Adds a texture to an object.

Parameters

textureName	Name of the texture.
textureType	Type of the texture.
texture	The texture id.

Implements **IGameObject** (p. 18).

4.12.3.9 void Light::Update ( ) [override], [virtual]

Updates the light. This currently does nothing! **Draw()** (p. 22) does update the light.

Implements **IGameObject** (p. 18).

4.12.4 Member Data Documentation

4.12.4.1 unsigned int Light::_id [protected]

4.12.4.2 glm::vec3 Light::ambient

light's contribution to ambient light.

4.12.4.3 bool Light::castsShadow [protected]

4.12.4.4 glm::vec3 Light::color

The color/intensity of the light.

4.12.4.5 glm::vec3 Light::coneDirection

spotlight cone direction.

4.12.4.6 float Light::constantAttenuation

constant light fall-out

4.12.4.7 glm::vec3 Light::halfVector

?? direction of headlights for directional light ??

4.12.4.8 struct Light::LightUniformLocations Light::ids [protected]

4.12.4.9 bool Light::isEnabled

true if this object is enabled.

4.12.4.10 float Light::linearAttenuation

linear light fall-off

4.12.4.11 glm::vec3 Light::position

location of light if isLocal is true. For directional light, it is the direction

4.12.4.12 float Light::quadraticAttenuation

quadratic light fall-out

4.12.4.13 float Light::spotCosCutoff

how wide the spotlight is [0-1].

4.12.4.14 float Light::spotExponent

control light fall-off in the spotlight.

4.12.4.15 GLuint Light::texture [protected]

4.12.4.16 int Light::type [protected]

4.13 Light::LightUniformLocations Struct Reference

Holds the various uniform location ID's of the **Light** (p. 20).

`#include <Light.h>`

Public Attributes

- GLuint **isEnabled**
- GLuint **type**
- GLuint **ambient**
- GLuint **color**
- GLuint **position**
- GLuint **halfVector**
- GLuint **coneDirection**
- GLuint **spotCosCutoff**
- GLuint **spotExponent**
- GLuint **constantAttenuation**
- GLuint **linearAttenuation**
- GLuint **quadraticAttenuation**

4.13.1 Detailed Description

Holds the various uniform location ID's of the **Light** (p. 20).

4.13.2 Member Data Documentation

4.13.2.1 GLuint Light::LightUniformLocations::ambient

4.13.2.2 GLuint Light::LightUniformLocations::color

4.13.2.3 GLuint Light::LightUniformLocations::coneDirection

4.13.2.4 GLuint Light::LightUniformLocations::constantAttenuation

4.13.2.5 GLuint Light::LightUniformLocations::halfVector

4.13.2.6 GLuint Light::LightUniformLocations::isEnabled

4.13.2.7 GLuint Light::LightUniformLocations::linearAttenuation

4.13.2.8 GLuint Light::LightUniformLocations::position

4.13.2.9 GLuint Light::LightUniformLocations::quadraticAttenuation

4.13.2.10 GLuint Light::LightUniformLocations::spotCosCutoff

4.13.2.11 GLuint Light::LightUniformLocations::spotExponent

4.13.2.12 GLuint Light::LightUniformLocations::type

4.14 Material Struct Reference

A material. XXX Not yet implemented!

`#include <Material.h>`

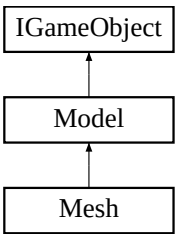
4.14.1 Detailed Description

A material. XXX Not yet implemented!

4.15 Mesh Class Reference

`#include <Mesh.h>`

Inheritance diagram for Mesh:



Public Member Functions

- **Mesh** (const aiMesh ∗, const aiMaterial ∗, **Transform** ∗)
- **Mesh** (const **Mesh** ∗, **Transform** ∗)
- ∼**Mesh** ()
- std::vector< **VertexFormat** > **GetVertices** () override final
- void **Create** ()
- void **DrawShadow** (GLuint) override final

  Draw shadows via the given shader.
- void **Draw** (GLuint) override final

  Draws via the given shader.
- void **Update** () override final

  Updates this object.
- std::vector< glm::vec3 > **GetPositionVertices** ()

Public Attributes

- float **shininess**
- float **strength**

Additional Inherited Members

4.15.1 Constructor & Destructor Documentation

4.15.1.1 Mesh::Mesh (const aiMesh * *ai_mesh,* const aiMaterial * *ai_mat,* Transform * *transform*)

4.15.1.2 Mesh::Mesh (const Mesh * *mesh,* Transform * *t*)

4.15.1.3 Mesh::~Mesh ( )

4.15.2 Member Function Documentation

4.15.2.1 void Mesh::Create ( )

4.15.2.2 void Mesh::Draw (GLuint *shader*) [final], [override], [virtual]

Draws via the given shader.

Parameters

shader	The shader.

Implements **IGameObject** (p. 16).

4.15.2.3 void Mesh::DrawShadow (GLuint *shader*) [final], [override], [virtual]

Draw shadows via the given shader.

Implements **IGameObject** (p. 17).

4.15.2.4 std::vector< glm::vec3 > Mesh::GetPositionVertices ( )

4.15.2.5 std::vector< VertexFormat > Mesh::GetVertices ( ) [final], [override], [virtual]

Implements **Model** (p. 31).

4.15.2.6 void Mesh::Update ( ) [final], [override], [virtual]

Updates this object.

Implements **IGameObject** (p. 18).

4.15.3 Member Data Documentation

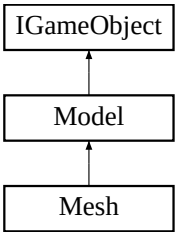
4.15.3.1 float Mesh::shininess

4.15.3.2 float Mesh::strength

4.16 Model Class Reference

`#include <Model.h>`

Inheritance diagram for Model:



Public Member Functions

- **Model** ()
- **Model** (**Transform** ∗t)
- virtual ∼**Model** ()
- void **SetProgram** (GLuint) override

  Sets the main shader program corresponding to this object.
- void **SetShadowProgram** (GLuint) override

  Sets the shadow pass shader program corresponding to this object.
- void **Destroy** () override

  Destroys this object.
- virtual std::vector< **VertexFormat** > **GetVertices** ()=0
- GLuint **GetVao** () const
- const std::vector< GLuint > & **GetVbos** () const
- const GLuint **GetTexture** (const std::string &) const override

  Gets a texture.
- void **SetTexture** (const std::string &textureName, const TextureType &textureType, const GLuint &texture) override

  Adds a texture to an object.

Protected Attributes

- **Transform** ∗ **transform**
- GLuint **vao**
- std::vector< GLuint > **vbos**
- std::vector< **Texture** > **textures**

4.16.1 Constructor & Destructor Documentation

4.16.1.1 Model::Model ( )

4.16.1.2 Model::Model (Transform ∗ *t*)

4.16.1.3 Model::∼Model ( ) [virtual]

4.16.2 Member Function Documentation

4.16.2.1 void Model::Destroy ( ) [override], [virtual]

Destroys this object.

Implements **IGameObject** (p. 16).

4.16.2.2 const GLuint Model::GetTexture (const std::string & *textureName*) const [override], [virtual]

Gets a texture.

Parameters

textureName	The name of the texture.

Returns

The texture.

Implements **IGameObject** (p. 17).

4.16.2.3 GLuint Model::GetVao ( ) const

4.16.2.4 const std::vector< GLuint > & Model::GetVbos ( ) const

4.16.2.5 virtual std::vector<VertexFormat> Model::GetVertices ( ) [pure virtual]

Implemented in **Mesh** (p. 29).

4.16.2.6 void Model::SetProgram (GLuint *shaderName*) [override], [virtual]

Sets the main shader program corresponding to this object.

Author

Harry

Date

4/14/2016

Parameters

shaderName	Name of the shader.

Reimplemented from **IGameObject** (p. 18).

4.16.2.7 void Model::SetShadowProgram (GLuint *shaderName*) [override], [virtual]

Sets the shadow pass shader program corresponding to this object.

Parameters

shaderName	Name of the shader.

Reimplemented from **IGameObject** (p. 18).

4.16.2.8 void Model::SetTexture (const std::string & *textureName,* const TextureType & *textureType,* const GLuint & *texture*) [override], [virtual]

Adds a texture to an object.

Parameters

textureName	Name of the texture.
textureType	Type of the texture.
texture	The texture id.

Implements **IGameObject** (p. 18).

4.16.3 Member Data Documentation

4.16.3.1 std::vector<Texture> Model::textures [protected]

4.16.3.2 Transform* Model::transform [protected]

4.16.3.3 GLuint Model::vao [protected]

4.16.3.4 std::vector<GLuint> Model::vbos [protected]

4.17 Models_Manager Class Reference

Manager for all models in a scene.

`#include <Models_Manager.h>`

Public Member Functions

- **Models_Manager** ()
- ~**Models_Manager** ()
- void **Draw** ()

  *The **Scene_Manager** (p. 42) will call this method. Notifies all the **Scene_Container** (p. 39), Particle_System, and **Light** (p. 20) to draw.*
- void **DrawShadows** ()

  *The **Scene_Manager** (p. 42) will call this method. Notifies all the **Scene_Container** (p. 39), Particle_System, and **Light** (p. 20) to draw shadows.*
- void **Update** ()

  *The **Scene_Manager** (p. 42) will call this method. Notifies all the **Scene_Container** (p. 39), Particle_System, and **Light** (p. 20) to **Draw()** (p. 34).*
- void **addLight** (**Light** *light)

  Adds a light to the light list.
- **Scene_Container** * **CreateModel** (const std::string &modelPath, const **Transform** &transform, const std::string &texturePath="", const TextureType type=Texture_Diffuse)

  Creates a model and adds it to the model list.
- **Scene_Container** * **CreateModel** (const **Scene_Container** *&otherModel, const **Transform** &transform)

  Copies a model and adds it to the model list.
- **Particle_Container** * **CreateParticleSystem** (**Transform** transform, const std::string &texturePath)

  Creates particle system and adds it to the particle list.

4.17.1 Detailed Description

Manager for all models in a scene.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Models_Manager::Models_Manager ( )

4.17.2.2 Models_Manager::~Models_Manager ( )

4.17.3 Member Function Documentation

4.17.3.1 void Models_Manager::addLight (Light * *light*)

Adds a light to the light list.

Parameters

light	If non-null, the light.

4.17.3.2 Scene_Container * Models_Manager::CreateModel (const std::string & *modelPath,* const Transform & *transform,* const std::string & *texturePath* = " ", const TextureType *type* = `Texture_Diffuse`)

Creates a model and adds it to the model list.

Parameters

modelPath	Full pathname to the object file.
transform	The transform.
texturePath	Full pathname to the texture file.
type	The texture type.

Returns

null if it fails, else the new model.

4.17.3.3 Scene_Container * Models_Manager::CreateModel (const Scene_Container *& *otherModel,* const Transform & *transform*)

Copies a model and adds it to the model list.

Parameters

otherModel	The other model.
transform	The transform.

Returns

null if it fails, else the new model.

4.17.3.4 Particle_Container * Models_Manager::CreateParticleSystem (Transform *t,* const std::string & *texturePath*)

Creates particle system and adds it to the particle list.

Parameters

transform	The **Transform** (p. 47)
texturePath	Full pathname to the texture file.

Returns

null if it fails, else the new particle system.

4.17.3.5 void Models_Manager::Draw ( )

The **Scene_Manager** (p. 42) will call this method. Notifies all the **Scene_Container** (p. 39), Particle_System, and **Light** (p. 20) to draw.

4.17.3.6 void Models_Manager::DrawShadows ( )

The **Scene_Manager** (p. 42) will call this method. Notifies all the **Scene_Container** (p. 39), Particle_System, and **Light** (p. 20) to draw shadows.

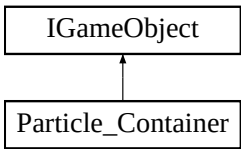
4.17.3.7 void Models_Manager::Update ( )

The **Scene_Manager** (p. 42) will call this method. Notifies all the **Scene_Container** (p. 39), Particle_System, and **Light** (p. 20) to **Draw()** (p. 34).

4.18 Particle_Container Class Reference

`#include <Particle_Container.h>`

Inheritance diagram for Particle_Container:



Public Member Functions

- **Particle_Container** (**Transform** t, const std::string &texturePath, bool enable=true, int max_particles=10000, int particle_rate=1000, float particle_max_life=5)
- ~**Particle_Container** ()
- void **Draw** ()
- void **Draw** (GLuint) override

  Draws via the given shader.
- void **DrawShadow** ()
- void **DrawShadow** (GLuint) override

  Draw shadows via the given shader.
- void **Update** () override

  Updates this object.
- void **Destroy** () override

  Destroys this object.
- void **SetTexture** (const std::string &, const TextureType &, const GLuint &) override

  Adds a texture to an object.
- const GLuint **GetTexture** (const std::string &) const override

  Gets a texture.
- void **setMaxParticles** (int)
- void **setSpawnRate** (int)
- void **setMaxLife** (float)

Public Attributes

- **Transform transform**

Additional Inherited Members

4.18.1 Constructor & Destructor Documentation

4.18.1.1 Particle_Container::Particle_Container (Transform *t,* const std::string & *texturePath,* bool *enable* = `true`, int *max_particles* = `10000`, int *particle_rate* = `1000`, float *particle_max_life* = `5`)

4.18.1.2 Particle_Container::~Particle_Container ( )

4.18.2 Member Function Documentation

4.18.2.1 void Particle_Container::Destroy ( ) `[override], [virtual]`

Destroys this object.

Implements **IGameObject** (p. 16).

4.18.2.2 void Particle_Container::Draw ( )

4.18.2.3 void Particle_Container::Draw (GLuint *shader*) `[override], [virtual]`

Draws via the given shader.

Parameters

shader	The shader.

Implements **IGameObject** (p. 16).

4.18.2.4 void Particle_Container::DrawShadow ( ) [inline]

4.18.2.5 void Particle_Container::DrawShadow (GLuint *shader*) [inline], [override], [virtual]

Draw shadows via the given shader.

Implements **IGameObject** (p. 17).

4.18.2.6 const GLuint Particle_Container::GetTexture (const std::string & *textureName*) const [override], [virtual]

Gets a texture.

Parameters

textureName	The name of the texture.

Returns

The texture.

Implements **IGameObject** (p. 17).

4.18.2.7 void Particle_Container::setMaxLife (float *life*)

4.18.2.8 void Particle_Container::setMaxParticles (int *n*)

4.18.2.9 void Particle_Container::setSpawnRate (int *rate*)

4.18.2.10 void Particle_Container::SetTexture (const std::string & *textureName,* const TextureType & *textureType,* const GLuint & *texture*) [override], [virtual]

Adds a texture to an object.

Parameters

textureName	Name of the texture.
textureType	Type of the texture.
texture	The texture id.

Implements **IGameObject** (p. 18).

4.18.2.11 void Particle_Container::Update ( ) [override], [virtual]

Updates this object.

Implements **IGameObject** (p. 18).

4.18.3 Member Data Documentation

4.18.3.1 Transform Particle_Container::transform

4.19 Physics_Manager Class Reference

`#include <Physics_Manager.h>`

Public Member Functions

- ~**Physics_Manager** ()
- void **Step** ()
- void **DrawDebug** ()
- void **AddRigidBody** (btRigidBody *)
- void **AddConstraint** (btTypedConstraint *)

Static Public Member Functions

- static **Physics_Manager** * **GetInstance** ()

4.19.1 Constructor & Destructor Documentation

4.19.1.1 Physics_Manager::~Physics_Manager ( )

4.19.2 Member Function Documentation

4.19.2.1 void Physics_Manager::AddConstraint (btTypedConstraint * *constraint*)

4.19.2.2 void Physics_Manager::AddRigidBody (btRigidBody * *body*)

4.19.2.3 void Physics_Manager::DrawDebug ( )

4.19.2.4 Physics_Manager * Physics_Manager::GetInstance ( ) [static]

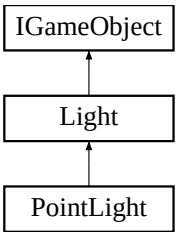
4.19.2.5 void Physics_Manager::Step ( )

4.20 PointLight Class Reference

A point light.

`#include <Light.h>`

Inheritance diagram for PointLight:



Public Member Functions

- **PointLight** (glm::vec3 **color**, glm::vec3 **position**, float **constantAttenuation**=0.0f, float linear↩ Attenuation=1.0f, float quadraticAttenuation=0.0f)
- virtual void **Draw** (GLuint) override final

  Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Additional Inherited Members

4.20.1 Detailed Description

A point light.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 PointLight::PointLight (glm::vec3 *color,* glm::vec3 *position,* float *constantAttenuation* = `0.0f`, float *linearAttenuation* = `1.0f`, float *quadraticAttenuation* = `0.0f`)

4.20.3 Member Function Documentation

4.20.3.1 void PointLight::Draw (GLuint *program*) `[final]`, `[override]`, `[virtual]`

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Parameters

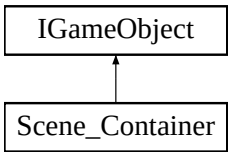
program	The shader program.

Reimplemented from **Light** (p. 22).

4.21 Scene_Container Class Reference

`#include <Scene_Container.h>`

Inheritance diagram for Scene_Container:



Public Member Functions

- **Scene_Container** (const std::string &, **Transform**)
- **Scene_Container** (const **Scene_Container** ∗&, **Transform**)
- ∼**Scene_Container** ()
- void **Draw** ()
- void **Draw** (GLuint) override

  Draws via the given shader.
- void **DrawShadow** ()
- void **DrawShadow** (GLuint) override

  Draw shadows via the given shader.
- void **Update** () override

  Updates this object.
- void **Destroy** () override

  Destroys this object.
- void **SetProgram** (GLuint) override

  Sets the main shader program corresponding to this object.
- void **SetShadowProgram** (GLuint) override

  Sets the shadow pass shader program corresponding to this object.
- btRigidBody ∗ **getRigidBody** ()
- virtual void **InitRigidBody** (btScalar mass)
- const std::vector< **Mesh** ∗ > **GetMeshes** () const
- void **SetTexture** (const std::string &, const TextureType &, const GLuint &) override

  Adds a texture to an object.
- const GLuint **GetTexture** (const std::string &) const override

  Gets a texture.

Public Attributes

- **Transform transform**

Additional Inherited Members

4.21.1 Constructor & Destructor Documentation

4.21.1.1 Scene_Container::Scene_Container (const std::string & *path,* Transform *transform*)

4.21.1.2 Scene_Container::Scene_Container (const Scene_Container *& *other,* Transform *t*)

4.21.1.3 Scene_Container::~Scene_Container ( )

4.21.2 Member Function Documentation

4.21.2.1 void Scene_Container::Destroy ( ) [override], [virtual]

Destroys this object.

Implements **IGameObject** (p. 16).

4.21.2.2 void Scene_Container::Draw ( )

4.21.2.3 void Scene_Container::Draw (GLuint *shader*) [override], [virtual]

Draws via the given shader.

Parameters

shader	The shader.

Implements **IGameObject** (p. 16).

4.21.2.4 void Scene_Container::DrawShadow ( )

4.21.2.5 void Scene_Container::DrawShadow (GLuint *shader*) [override], [virtual]

Draw shadows via the given shader.

Implements **IGameObject** (p. 17).

4.21.2.6 const std::vector< Mesh * > Scene_Container::GetMeshes ( ) const

4.21.2.7 btRigidBody * Scene_Container::getRigidBody ( )

4.21.2.8 const GLuint Scene_Container::GetTexture (const std::string & *textureName*) const [override], [virtual]

Gets a texture.

Parameters

textureName	The name of the texture.

Returns

The texture.

Implements **IGameObject** (p. 17).

4.21.2.9 void Scene_Container::InitRigidBody (btScalar *mass*) [virtual]

Scene_Container (p. 39) owns this method because I, Dylan, have made an executive decision to make lights bound to the laws of physics impossible; I see no point for it (except for flashlights maybe). Our lights will be static, so the initialization of rigid bodies can be the sole responsibility of the **Scene_Container** (p. 39), because it knows about all of its own vertices.

TODO: Support more than just Convex Hull Shape for collision detection; Triangle **Mesh** (p. 27) Shape might be more appropriate for static game objects.

4.21.2.10 void Scene_Container::SetProgram (GLuint *shaderName*) [override], [virtual]

Sets the main shader program corresponding to this object.

Author

Harry

Date

4/14/2016

Parameters

shaderName	Name of the shader.

Reimplemented from **IGameObject** (p. 18).

4.21.2.11 void Scene_Container::SetShadowProgram (GLuint *shaderName*) [override], [virtual]

Sets the shadow pass shader program corresponding to this object.

Parameters

shaderName	Name of the shader.

Reimplemented from **IGameObject** (p. 18).

4.21.2.12 void Scene_Container::SetTexture (const std::string & *textureName,* const TextureType & *textureType,* const GLuint & *texture*) [override], [virtual]

Adds a texture to an object.

Parameters

textureName	Name of the texture.
textureType	Type of the texture.
texture	The texture id.

Implements **IGameObject** (p. 18).

4.21.2.13 void Scene_Container::Update ( ) [override], [virtual]

Updates this object.

Implements **IGameObject** (p. 18).

4.21.3 Member Data Documentation

4.21.3.1 Transform Scene_Container::transform

4.22 Scene_Manager Class Reference

`#include <Scene_Manager.h>`

Public Member Functions

- **Scene_Manager** (std::string)
- ~**Scene_Manager** ()
- void **SetupScene** (const **GameObjectsBuilder** &)
- void **UpdatePass** () const
- void **ShadowPass** () const
- void **RenderPass** () const
- void **notifyBeginFrame** ()
- void **notifyDisplayFrame** ()
- void **notifyEndFrame** ()
- void **notifyReshape** (int width, int height, int p_width, int p_height)

Static Public Member Functions

- static int **GetDeltaTime** ()
- static float **GetFPS** ()

4.22.1 Constructor & Destructor Documentation

4.22.1.1 Scene_Manager::Scene_Manager (std::string *scene_name*)

4.22.1.2 Scene_Manager::~Scene_Manager ( )

4.22.2 Member Function Documentation

4.22.2.1 int Scene_Manager::GetDeltaTime ( ) [static]

4.22.2.2 float Scene_Manager::GetFPS ( ) [static]

4.22.2.3 void Scene_Manager::notifyBeginFrame ( )

4.22.2.4 void Scene_Manager::notifyDisplayFrame ( )

4.22.2.5 void Scene_Manager::notifyEndFrame ( )

4.22.2.6 void Scene_Manager::notifyReshape (int *width,* int *height,* int *p_width,* int *p_height*)

4.22.2.7 void Scene_Manager::RenderPass ( ) const

4.22.2.8 void Scene_Manager::SetupScene (const **GameObjectsBuilder** & *gob*)

4.22.2.9 void Scene_Manager::ShadowPass ( ) const

4.22.2.10 void Scene_Manager::UpdatePass ( ) const

4.23 Shader_Factory Class Reference

`#include <Shader_Factory.h>`

Public Member Functions

- ~**Shader_Factory** (void)
- const GLuint & **CreateProgram** (const std::string &shaderName, const std::string &VertexShaderFilename, const std::string &FragmentShaderFilename)
- const void **SetTextureShader** (**IGameObject** &model)
- const GLuint & **CreateDebugProgram** ()

Static Public Member Functions

- static **Shader_Factory** * **GetInstance** ()

4.23.1 Constructor & Destructor Documentation

4.23.1.1 Shader_Factory::~Shader_Factory (void)

4.23.2 Member Function Documentation

4.23.2.1 const GLuint & Shader_Factory::CreateDebugProgram ( )

4.23.2.2 const GLuint & Shader_Factory::CreateProgram (const std::string & *shaderName,* const std::string & *VertexShaderFilename,* const std::string & *FragmentShaderFilename*)

4.23.2.3 Shader_Factory * Shader_Factory::GetInstance ( ) [static]

4.23.2.4 const void Shader_Factory::SetTextureShader (IGameObject & *model*)

4.24 Shadow_Manager Class Reference

Manager for shadows implemented as a Singleton.

`#include <Shadow_Manager.h>`

Public Member Functions

- GLuint **GetShadowMap** () const
  Gets shadow map texture.
- glm::mat4 **GetDepthMatrix** () const
  Gets the 4x4 depth matrix.
- void **SetDepthMatrix** (glm::mat4 DepthMatrix)
  Sets the depth matrix.
- void **BindForWriting** () const
  Bind the depth t exture for writing.
- void **Unbind** () const
  Unbinds the depth texture.

Static Public Member Functions

- static **Shadow_Manager** * **GetInstance** ()
  Gets the instance.

Static Public Attributes

- static const int **DEPTH_TEXTURE_SIZE** = 1024
  Size of the depth texture in pixels.

4.24.1 Detailed Description

Manager for shadows implemented as a Singleton.

Author

Harry Gogonis

4.24.2 Member Function Documentation

4.24.2.1 void Shadow_Manager::BindForWriting ( ) const

Bind the depth t exture for writing.

4.24.2.2 glm::mat4 Shadow_Manager::GetDepthMatrix ( ) const

Gets the 4x4 depth matrix.

Returns

The depth matrix.

4.24.2.3 static Shadow_Manager * Shadow_Manager::GetInstance ( ) [static]

Gets the instance.

Author

Harry

Returns

The Singleton instance

4.24.2.4 GLuint Shadow_Manager::GetShadowMap ( ) const

Gets shadow map texture.

Returns

The shadow map.

4.24.2.5 void Shadow_Manager::SetDepthMatrix (glm::mat4 *DepthMatrix*)

Sets the depth matrix.

Parameters

DepthMatrix	The depth matrix.

4.24.2.6 void Shadow_Manager::Unbind ( ) const

Unbinds the depth texture.

4.24.3 Member Data Documentation

4.24.3.1 const int Shadow_Manager::DEPTH_TEXTURE_SIZE = 1024 [static]

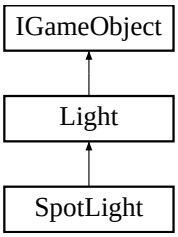
Size of the depth texture in pixels.

4.25 SpotLight Class Reference

A spot light.

`#include <Light.h>`

Inheritance diagram for SpotLight:



Public Member Functions

- **SpotLight** (glm::vec3 **color**, glm::vec3 **position**, glm::vec3 **coneDirection**, float **spotCosCutoff**=0.↩ 99f, float spotExponent=0.0f, float constantAttenuation=0.0f, float linearAttenuation=1.0f, float quadratic↩ Attenuation=0.0f)
- virtual void **Draw** (GLuint) override final

  Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Additional Inherited Members

4.25.1 Detailed Description

A spot light.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 SpotLight::SpotLight (glm::vec3 *color,* glm::vec3 *position,* glm::vec3 *coneDirection,* float *spotCosCutoff* = `0.99f`, float *spotExponent* = `0.0f`, float *constantAttenuation* = `0.0f`, float *linearAttenuation* = `1.0f`, float *quadraticAttenuation* = `0.0f`)

4.25.3 Member Function Documentation

4.25.3.1 void SpotLight::Draw (GLuint *program*) `[final],[override],[virtual]`

Sets up the light uniform location id's from the program. Each subclass must then send their corresponding params to the shader.

Parameters

program	The shader program.

Reimplemented from **Light** (p. 22).

4.26 Texture Struct Reference

A texture. This struct is currently not used!

`#include <Texture.h>`

Public Attributes

- GLuint **id**
- std::string **name**
- TextureType **type**

4.26.1 Detailed Description

A texture. This struct is currently not used!

4.26.2 Member Data Documentation

4.26.2.1 GLuint Texture::id

4.26.2.2 std::string Texture::name

4.26.2.3 TextureType Texture::type

4.27 Transform Class Reference

Transform (p. 47) composed of Scale, Rotation (as a quaternion), and Translation. Transformation is applied in the order: Scale -> Rotate -> Translate.

`#include <Transform.h>`

Public Member Functions

- **Transform** ()
- **Transform** (glm::vec3 **position**, float **scale**, glm::quat **rotation**)
- void **SetScale** (float **scale**)

  Sets a scale.
- void **SetPosition** (const glm::vec3 &**position**)

  Sets a position.
- void **SetRotation** (float angle, glm::vec3 axis)

  Sets a rotation.
- void **IncRotation** (float angle, glm::vec3 axis)

  Increment rotation.
- void **RotateX** (float angle)

  Rotate along x axis.
- void **RotateY** (float angle)

  Rotate along y axis.
- void **RotateZ** (float angle)

  Rotate along z axis.
- glm::mat4 **getTranslationMatrix** () const

  Gets translation matrix.
- glm::mat4 **getScaleMatrix** () const

  Gets scale matrix.
- glm::mat4 **getRotationMatrix** () const

  Gets rotation matrix.
- glm::mat4 **getTransformMatrix** () const

  Gets transformation matrix. Transformation applied in the order: Scale -> Rotate -> Translate.
- **Transform** & **operator=** (const btTransform &trans)

  Assignment operator.
- ∼**Transform** ()

Public Attributes

- glm::vec3 **position**
- glm::quat **rotation**
- float **scale**

4.27.1 Detailed Description

Transform (p. 47) composed of Scale, Rotation (as a quaternion), and Translation. Transformation is applied in the order: Scale -> Rotate -> Translate.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 Transform::Transform ( )

4.27.2.2 Transform::Transform (glm::vec3 *position,* float *scale,* glm::quat *rotation*)

4.27.2.3 Transform::∼Transform ( )

4.27.3 Member Function Documentation

4.27.3.1 glm::mat4 Transform::getRotationMatrix ( ) const

Gets rotation matrix.

Returns

The rotation matrix.

4.27.3.2 glm::mat4 Transform::getScaleMatrix ( ) const

Gets scale matrix.

Returns

The scale matrix.

4.27.3.3 glm::mat4 Transform::getTransformMatrix ( ) const

Gets transformation matrix. Transformation applied in the order: Scale -> Rotate -> Translate.

Returns

The transform matrix.

4.27.3.4 glm::mat4 Transform::getTranslationMatrix ( ) const

Gets translation matrix.

Returns

The translation matrix.

4.27.3.5 void Transform::IncRotation (float *angle,* glm::vec3 *axis*)

Increment rotation.

Parameters

angle	The angle in degrees.
axis	The axis.

4.27.3.6 Transform & Transform::operator= (const btTransform & *trans*)

Assignment operator.

Parameters

trans	The transform from ASSIMP

Returns

A shallow copy of this object.

4.27.3.7 void Transform::RotateX (float *angle*)

Rotate along x axis.

Parameters

angle	The angle in degrees.

4.27.3.8 void Transform::RotateY (float *angle*)

Rotate along y axis.

Parameters

angle	The angle in degrees.

4.27.3.9 void Transform::RotateZ (float *angle*)

Rotate along z axis.

Parameters

angle	The angle in degrees.

4.27.3.10 void Transform::SetPosition (const glm::vec3 & *position*)

Sets a position.

Parameters

position	The position.

4.27.3.11 void Transform::SetRotation (float *angle,* glm::vec3 *axis*)

Sets a rotation.

Parameters

angle	The angle in degrees.
axis	The axis.

4.27.3.12 void Transform::SetScale (float *scale*)

Sets a scale.

Parameters

scale	The scale.

4.27.4 Member Data Documentation

4.27.4.1 glm::vec3 Transform::position

4.27.4.2 glm::quat Transform::rotation

4.27.4.3 float Transform::scale

4.28 VboIndexer Class Reference

`#include <VboIndexer.h>`

Static Public Member Functions

- static void **indexVBO** (std::vector< glm::vec3 > &in_vertices, std::vector< glm::vec3 > &in_colors, std::vector< unsigned short > &out_indices, std::vector< **DVertexFormat** > &out_formats)

4.28.1 Member Function Documentation

4.28.1.1 void VboIndexer::indexVBO (std::vector< glm::vec3 > & *in_vertices,* std::vector< glm::vec3 > & *in_colors,* std::vector< unsigned short > & *out_indices,* std::vector< **DVertexFormat** > & *out_formats*) [static]

4.29 VertexFormat Struct Reference

The vertex buffer format that is sent directly to the shader.

`#include <VertexFormat.h>`

Public Member Functions

- **VertexFormat** (const glm::vec3 &inPos, const glm::vec2 &inUV, const glm::vec3 &inNormal, const glm::vec3 &inTangent, const glm::vec3 &inBitangent)
- bool **operator**< (const **VertexFormat** that) const

Public Attributes

- glm::vec3 **position**
- glm::vec2 **uv**
- glm::vec3 **normal**
- glm::vec3 **tangent**
- glm::vec3 **bitangent**

Friends

- std::ostream & **operator**<< (std::ostream &os, const **VertexFormat** &v)

4.29.1 Detailed Description

The vertex buffer format that is sent directly to the shader.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 VertexFormat::VertexFormat (const glm::vec3 & *inPos,* const glm::vec2 & *inUV,* const glm::vec3 & *inNormal,* const glm::vec3 & *inTangent,* const glm::vec3 & *inBitangent*) [inline]

4.29.3 Member Function Documentation

4.29.3.1 bool VertexFormat::operator< (const **VertexFormat** *that*) const [inline]

4.29.4 Friends And Related Function Documentation

4.29.4.1 std::ostream& operator<< (std::ostream & *os,* const VertexFormat & *v*) [friend]

4.29.5 Member Data Documentation

4.29.5.1 glm::vec3 VertexFormat::bitangent

4.29.5.2 glm::vec3 VertexFormat::normal

4.29.5.3 glm::vec3 VertexFormat::position

4.29.5.4 glm::vec3 VertexFormat::tangent

4.29.5.5 glm::vec2 VertexFormat::uv

4.30 WindowInfo Struct Reference

Information about the game window.

`#include <WindowInfo.h>`

Public Member Functions

- **WindowInfo** ()
- **WindowInfo** (std::string **name**, int **position_x**, int **position_y**, int **width**, int **height**, bool **isReshapable**)
- **WindowInfo** (const **WindowInfo** &windowInfo)
- void **operator=** (const **WindowInfo** &windowInfo)

Public Attributes

- std::string **name**
- int **width**
- int **height**
- int **position_x**
- int **position_y**
- bool **isReshapable**

4.30.1 Detailed Description

Information about the game window.

Author

Harry

Date

4/14/2016

4.30.2 Constructor & Destructor Documentation

4.30.2.1 WindowInfo::WindowInfo ( ) [inline]

4.30.2.2 WindowInfo::WindowInfo (std::string *name,* int *position_x,* int *position_y,* int *width,* int *height,* bool *isReshapable*) [inline]

4.30.2.3 WindowInfo::WindowInfo (const WindowInfo & *windowInfo*) [inline]

4.30.3 Member Function Documentation

4.30.3.1 void WindowInfo::operator= (const WindowInfo & *windowInfo*) [inline]

4.30.4 Member Data Documentation

4.30.4.1 int WindowInfo::height

4.30.4.2 bool WindowInfo::isReshapable

4.30.4.3 std::string WindowInfo::name

4.30.4.4 int WindowInfo::position_x

4.30.4.5 int WindowInfo::position_y

4.30.4.6 int WindowInfo::width

Index