TurtleSDK

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

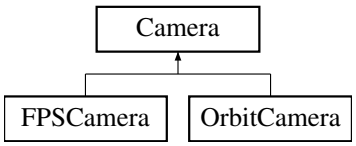
Class Documentation

4.1 Camera Class Reference

Camera used to navigate in a scene.

`#include <camera.h>`

Inheritance diagram for Camera:



Public Member Functions

- virtual void reset ()=0

  Reset the camera values.
- virtual void process_mouse_move (GLFWwindow ∗window, double xpos, double ypos)
- virtual void process_mouse_action (GLFWwindow ∗window, int button, int action, int mods)
- virtual void process_scroll (GLFWwindow ∗window, double xoffset, double yoffset)
- virtual void process_key (GLFWwindow ∗, int, int, int, int)
- virtual glm::mat4 view () const =0
- virtual glm::mat4 projection () const
- virtual glm::vec3 up () const =0

Public Attributes

- glm::vec3 pos
- float fov = 0

Protected Attributes

- glm::vec3 _initialPos = {0, 0, 3}
- float _lastPosX
- float _lastPosY
- bool _firstMove = true

4.1.1 Detailed Description

Camera used to navigate in a scene.

4.1.2 Member Function Documentation

4.1.2.1 process_key()

```
void Camera::process_key (
            GLFWwindow * ,
            int ,
            int ,
            int ,
            int  )  [virtual]
```

Reimplemented in FPSCamera.

4.1.2.2 process_mouse_action()

```
void Camera::process_mouse_action (
            GLFWwindow * window,
            int button,
            int action,
            int mods )  [virtual]
```

Reimplemented in OrbitCamera.

4.1.2.3 process_mouse_move()

```
void Camera::process_mouse_move (
            GLFWwindow * window,
            double xpos,
            double ypos )  [virtual]
```

Reimplemented in FPSCamera, and OrbitCamera.

4.1.2.4 process_scroll()

```
void Camera::process_scroll (
            GLFWwindow * window,
            double xoffset,
            double yoffset )  [virtual]
```

Reimplemented in FPSCamera, and OrbitCamera.

4.1.2.5 projection()

```
glm::mat4 Camera::projection ( ) const  [virtual]
```

4.1.2.6 reset()

```
virtual void Camera::reset ( )  [pure virtual]
```

Reset the camera values.

Implemented in FPSCamera, and OrbitCamera.

4.1.2.7 up()

```
virtual glm::vec3 Camera::up ( ) const  [pure virtual]
```

Implemented in FPSCamera, and OrbitCamera.

4.1.2.8 view()

```
virtual glm::mat4 Camera::view ( ) const  [pure virtual]
```

Implemented in FPSCamera, and OrbitCamera.

4.1.3 Member Data Documentation

4.1.3.1 _firstMove

```
bool Camera::_firstMove = true  [protected]
```

4.1.3.2 _initialPos

```
glm::vec3 Camera::_initialPos = {0, 0, 3}  [protected]
```

4.1.3.3 _lastPosX

```
float Camera::_lastPosX  [protected]
```

4.1.3.4 _lastPosY

```
float Camera::_lastPosY  [protected]
```

4.1.3.5 fov

```
float Camera::fov = 0
```

4.1.3.6 pos

```
glm::vec3 Camera::pos
```

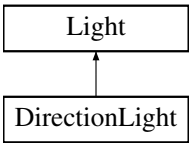
The documentation for this class was generated from the following files:

- src/camera.h
- src/camera.cpp

4.2 DirectionLight Class Reference

```
#include <light.h>
```

Inheritance diagram for DirectionLight:



Public Member Functions

- virtual void setUniforms (const Shader &shader, const std::string &uname) const
- virtual void ui ()

Public Attributes

- glm::vec3 direction_ = {-10.f, -10.f, -10.f}

4.2.1 Member Function Documentation

4.2.1.1 setUniforms()

```
void DirectionLight::setUniforms (
            const Shader & shader,
            const std::string & uname ) const  [virtual]
```

Reimplemented from Light.

4.2.1.2 ui()

```
void DirectionLight::ui ( )  [virtual]
```

Reimplemented from Light.

4.2.2 Member Data Documentation

4.2.2.1 direction_

```
glm::vec3 DirectionLight::direction_ = {-10.f, -10.f, -10.f}
```

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

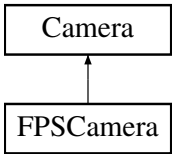
- src/light.h
- src/light.cpp

4.3 FPSCamera Class Reference

Camera used to navigate in a scene.

`#include <fpsCamera.h>`

Inheritance diagram for FPSCamera:



Public Member Functions

- FPSCamera ()
  Create a default fps camera.
- void reset ()
  Reset the camera values.
- void process_mouse_move (GLFWwindow ∗window, double xpos, double ypos)
- void process_scroll (GLFWwindow ∗window, double xoffset, double yoffset)
- void process_key (GLFWwindow ∗, int, int, int, int)
- glm::mat4 view () const
- glm::vec3 up () const

Public Attributes

- glm::vec3 camFront
- glm::vec3 camUp

Additional Inherited Members

4.3.1 Detailed Description

Camera used to navigate in a scene.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 FPSCamera()

`FPSCamera::FPSCamera ( )`

Create a default fps camera.

4.3.3 Member Function Documentation

4.3.3.1 process_key()

```
void FPSCamera::process_key (
            GLFWwindow * window,
            int key,
            int ,
            int action,
            int moods )  [virtual]
```

Reimplemented from Camera.

4.3.3.2 process_mouse_move()

```
void FPSCamera::process_mouse_move (
            GLFWwindow * window,
            double xpos,
            double ypos )  [virtual]
```

Reimplemented from Camera.

4.3.3.3 process_scroll()

```
void FPSCamera::process_scroll (
            GLFWwindow * window,
            double xoffset,
            double yoffset )  [virtual]
```

Reimplemented from Camera.

4.3.3.4 reset()

```
void FPSCamera::reset ( )  [virtual]
```

Reset the camera values.

Implements Camera.

4.3.3.5 up()

```
glm::vec3 FPSCamera::up ( ) const  [virtual]
```

Implements Camera.

4.3.3.6 view()

```
glm::mat4 FPSCamera::view ( ) const  [virtual]
```

Implements Camera.

4.3.4 Member Data Documentation

4.3.4.1 camFront

```
glm::vec3 FPSCamera::camFront
```

4.3.4.2 camUp

```
glm::vec3 FPSCamera::camUp
```

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

- src/fpsCamera.h
- src/fpsCamera.cpp

4.4 Grid Class Reference

A point grid.

```
#include <grid.h>
```

Public Member Functions

- float size () const
  Get the size.
- unsigned int slicing () const
  Get the slicing.

Public Attributes

- std::vector< glm::vec2 > points
  Points of the grid.
- std::vector< unsigned int > indices
  Indices of the grid.

4.4.1 Detailed Description

A point grid.

4.4.2 Member Function Documentation

4.4.2.1 size()

```
float Grid::size ( ) const  [inline]
```

Get the size.

Returns

4.4.2.2 slicing()

```
unsigned int Grid::slicing ( ) const  [inline]
```

Get the slicing.

Returns

4.4.3 Member Data Documentation

4.4.3.1 indices

```
std::vector<unsigned int> Grid::indices
```

Indices of the grid.

4.4.3.2 points

```
std::vector<glm::vec2> Grid::points
```

Points of the grid.

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

- src/grid.h

4.5 GridGenerator Class Reference

Grid generator.

```
#include <grid.h>
```

Static Public Member Functions

- static Grid flatGrid (float, unsigned int)
  Generate a flat grid.

4.5.1 Detailed Description

Grid generator.

4.5.2 Member Function Documentation

4.5.2.1 flatGrid()

```
Grid GridGenerator::flatGrid (
            float pSize,
            unsigned int pSlicing )  [static]
```

Generate a flat grid.

Parameters

float	The size
unsigned	int The number of slice on each axis

Returns

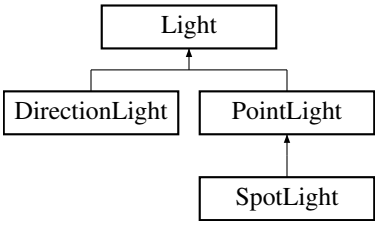
The documentation for this class was generated from the following files:

- src/grid.h
- src/grid.cpp

4.6 Light Class Reference

`#include <light.h>`

Inheritance diagram for Light:



Public Member Functions

- Light (const glm::vec3 &pambient, const glm::vec3 &pdiffuse, const glm::vec3 &pspecular)
- Light ()
- virtual void setUniforms (const Shader &shader, const std::string &uname) const
- virtual void ui ()

Public Attributes

- glm::vec3 ambient_
- glm::vec3 diffuse_
- glm::vec3 specular_

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Light() [1/2]

```
Light::Light (
            const glm::vec3 & pambient,
            const glm::vec3 & pdiffuse,
            const glm::vec3 & pspecular )
```

4.6.1.2 Light() [2/2]

```
Light::Light ( )
```

4.6.2 Member Function Documentation

4.6.2.1 setUniforms()

```
void Light::setUniforms (
            const Shader & shader,
            const std::string & uname ) const  [virtual]
```

Reimplemented in SpotLight, PointLight, and DirectionLight.

4.6.2.2 ui()

```
void Light::ui ( )  [virtual]
```

Reimplemented in SpotLight, PointLight, and DirectionLight.

4.6.3 Member Data Documentation

4.6.3.1 ambient_

```
glm::vec3 Light::ambient_
```

4.6.3.2 diffuse_

```
glm::vec3 Light::diffuse_
```

4.6.3.3 specular_

```
glm::vec3 Light::specular_
```

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

- src/light.h
- src/light.cpp

4.7 Mesh Class Reference

Mesh wrapper.

`#include <mesh.h>`

Public Member Functions

- Mesh (const std::vector< Vertex > &vertices, const std::vector< GLuint > &indices, const std::vector< Texture > &textures)
- void draw (const Shader &shader, const GLenum &mode=-1) const

  Draw the model.
- void updateDataBuffer ()

  Update the data buffer.

Static Public Member Functions

- template<class T >
  static void addBuffer (GLuint &idLocation, GLenum bufType, const std::vector< T > &data)

  Generate and allocate a buffer in the VAO.

4.7.1 Detailed Description

Mesh wrapper.

Keep a link on the VBOs and VAO of a model.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Mesh()

```
Mesh::Mesh (
            const std::vector< Vertex > & vertices,
            const std::vector< GLuint > & indices,
            const std::vector< Texture > & textures )
```

4.7.3 Member Function Documentation

4.7.3.1 addBuffer()

```
template<class T >
void Mesh::addBuffer (
            GLuint & idLocation,
            GLenum bufType,
            const std::vector< T > & data )  [static]
```

Generate and allocate a buffer in the VAO.

Parameters

idLocation	Where to store the id
bufType	The buffer type
data	The points to copy in the buffer

4.7.3.2 draw()

```
void Mesh::draw (
            const Shader & shader,
            const GLenum & mode = -1 ) const
```

Draw the model.

4.7.3.3 updateDataBuffer()

```
void Mesh::updateDataBuffer ( )
```

Update the data buffer.

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

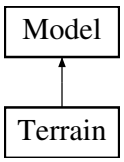
- src/mesh.h
- src/mesh.cpp

4.8 Model Class Reference

A set of one or more mesh.

`#include <model.h>`

Inheritance diagram for Model:



Public Member Functions

- Model (const std::string &path)

  Create a model from a given path.
- virtual void draw (const Shader &shader)

  Draw the model's meshes.
- virtual void ui ()

Protected Member Functions

- Model ()
- void loadModel (const std::string &path)

  Load the given model.

Protected Attributes

- std::vector< Mesh > meshes

  Meshes included in the model.

4.8.1 Detailed Description

A set of one or more mesh.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Model() [1/2]

```
Model::Model (
            const std::string & path )
```

Create a model from a given path.

Parameters

path	Path to the object (obj)

4.8.2.2 Model() [2/2]

```
Model::Model ( )  [protected]
```

4.8.3 Member Function Documentation

4.8.3.1 draw()

```
void Model::draw (
            const Shader & shader )  [virtual]
```

Draw the model's meshes.

Parameters

shader	Shader used to draw

Reimplemented in Terrain.

4.8.3.2 loadModel()

```
void Model::loadModel (
            const std::string & path )  [protected]
```

Load the given model.

Parameters

path	Path to the model

4.8.3.3 ui()

```
void Model::ui ( )  [virtual]
```

Reimplemented in Terrain.

4.8.4 Member Data Documentation

4.8.4.1 meshes

```
std::vector<Mesh> Model::meshes  [protected]
```

Meshes included in the model.

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

- src/model.h
- src/model.cpp

4.9 Object Class Reference

Wrapper of a model Can be positioned in the spaces Usefulle when creating instances, avoid to duplicate the model.

```
#include <object.h>
```

Public Member Functions

- Object ()

  Create an empty object. Not used in the application yet.
- Object (std::shared_ptr< Model >)

  Create a world object.
- void draw (const Shader &shader)

  Draw the model's meshes.
- void ui ()

4.9.1 Detailed Description

Wrapper of a model Can be positioned in the spaces Usefulle when creating instances, avoid to duplicate the model.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Object() [1/2]

```
Object::Object ( )
```

Create an empty object. Not used in the application yet.

4.9.2.2 Object() [2/2]

```
Object::Object (
            std::shared_ptr< Model > pModel )
```

Create a world object.

Parameters

The	model linked to the object

4.9.3 Member Function Documentation

4.9.3.1 draw()

```
void Object::draw (
            const Shader & shader )
```

Draw the model's meshes.

Parameters

shader	Shader used to draw

4.9.3.2 ui()

```
void Object::ui ( )
```

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

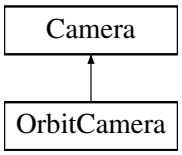
- src/object.h
- src/object.cpp

4.10 OrbitCamera Class Reference

Camera used to rotate around a round-shaped object.

```
#include <orbitCamera.h>
```

Inheritance diagram for OrbitCamera:



Public Member Functions

- OrbitCamera ()

  Create a camera with a default position and target.
- void reset ()

  Reset the camera target and position.
- void process_mouse_move (GLFWwindow ∗window, double xpos, double ypos)
- void process_mouse_action (GLFWwindow ∗window, int button, int action, int mods)
- void process_scroll (GLFWwindow ∗window, double xoffset, double yoffset)
- glm::mat4 view () const
- glm::vec3 up () const

Public Attributes

- glm::vec3 target

Additional Inherited Members

4.10.1 Detailed Description

Camera used to rotate around a round-shaped object.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 OrbitCamera()

```
OrbitCamera::OrbitCamera ( )
```

Create a camera with a default position and target.

4.10.3 Member Function Documentation

4.10.3.1 process_mouse_action()

```
void OrbitCamera::process_mouse_action (
            GLFWwindow * window,
            int button,
            int action,
            int mods )  [virtual]
```

Reimplemented from Camera.

4.10.3.2 process_mouse_move()

```
void OrbitCamera::process_mouse_move (
            GLFWwindow * window,
            double xpos,
            double ypos )  [virtual]
```

Reimplemented from Camera.

4.10.3.3 process_scroll()

```
void OrbitCamera::process_scroll (
            GLFWwindow * window,
            double xoffset,
            double yoffset )  [virtual]
```

Reimplemented from Camera.

4.10.3.4 reset()

```
void OrbitCamera::reset ( )  [virtual]
```

Reset the camera target and position.

Implements Camera.

4.10.3.5 up()

```
glm::vec3 OrbitCamera::up ( ) const  [virtual]
```

Implements Camera.

4.10.3.6 view()

```
glm::mat4 OrbitCamera::view ( ) const  [virtual]
```

Implements Camera.

4.10.4 Member Data Documentation

4.10.4.1 target

```
glm::vec3 OrbitCamera::target
```

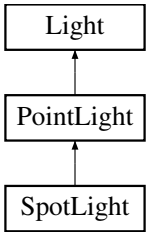
The documentation for this class was generated from the following files:

- src/orbitCamera.h
- src/orbitCamera.cpp

4.11 PointLight Class Reference

`#include <light.h>`

Inheritance diagram for PointLight:



Public Member Functions

- virtual void setUniforms (const Shader &shader, const std::string &uname) const
- virtual void ui ()

Public Attributes

- glm::vec3 position_ = {5.f, 5.f, 5.f}
- float constant_ = 1
- float linear_ = 0.09
- float quadratic_ = 0.032

4.11.1 Member Function Documentation

4.11.1.1 setUniforms()

```
void PointLight::setUniforms (
            const Shader & shader,
            const std::string & uname ) const  [virtual]
```

Reimplemented from Light.

Reimplemented in SpotLight.

4.11.1.2 ui()

```
void PointLight::ui ( )  [virtual]
```

Reimplemented from Light.

Reimplemented in SpotLight.

4.11.2 Member Data Documentation

4.11.2.1 constant_

```
float PointLight::constant_ = 1
```

4.11.2.2 linear_

```
float PointLight::linear_ = 0.09
```

4.11.2.3 position_

```
glm::vec3 PointLight::position_ = {5.f, 5.f, 5.f}
```

4.11.2.4 quadratic_

```
float PointLight::quadratic_ = 0.032
```

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

- src/light.h
- src/light.cpp

4.12 Shader Class Reference

```
#include <shader.h>
```

Public Member Functions

- Shader (const std::string &vertSrcPath, const std::string &fragSrcPath, const std::string &geoSrcPath="")
  Create and compile a shader.
- Shader (std::string path, const bool &useGeo=false)
  Create and compile a shader with sources using same name but different extension.
- void use () const
  Use the shader with glUseProgram.
- void setBool (const std::string &name, bool value) const
- void setInt (const std::string &name, int value) const
- void setFloat (const std::string &name, float value) const
- void setVec2 (const std::string &name, const glm::vec2 &value) const
- void setVec2 (const std::string &name, float x, float y) const
- void setVec3 (const std::string &name, const glm::vec3 &value) const
- void setVec3 (const std::string &name, float x, float y, float z) const
- void setVec4 (const std::string &name, const glm::vec4 &value) const
- void setVec4 (const std::string &name, float x, float y, float z, float w)
- void setMat2 (const std::string &name, const glm::mat2 &mat) const
- void setMat3 (const std::string &name, const glm::mat3 &mat) const
- void setMat4 (const std::string &name, const glm::mat4 &mat) const

4.12.1 Constructor & Destructor Documentation

4.12.1.1 Shader() [1/2]

```
Shader::Shader (
            const std::string & vertSrcPath,
            const std::string & fragSrcPath,
            const std::string & geoSrcPath = "" )
```

Create and compile a shader.

Parameters

vertSrcPath	Path to the vertex shader source
fragSrcPath	Path to the fragment shader soruce
geoSrcPath	Path to the geometry shader source (default=null)

4.12.1.2 Shader() [2/2]

```
Shader::Shader (
            std::string path,
            const bool & useGeo = false )  [explicit]
```

Create and compile a shader with sources using same name but different extension.

Parameters

path	Path to the source without extension
useGeo	If true, use the geo shader

4.12.2 Member Function Documentation

4.12.2.1 setBool()

```
void Shader::setBool (
            const std::string & name,
            bool value ) const  [inline]
```

4.12.2.2 setFloat()

```
void Shader::setFloat (
            const std::string & name,
            float value ) const  [inline]
```

4.12.2.3 setInt()

```
void Shader::setInt (
            const std::string & name,
            int value ) const  [inline]
```

4.12.2.4 setMat2()

```
void Shader::setMat2 (
            const std::string & name,
            const glm::mat2 & mat ) const  [inline]
```

4.12.2.5 setMat3()

```
void Shader::setMat3 (
            const std::string & name,
            const glm::mat3 & mat ) const  [inline]
```

4.12.2.6 setMat4()

```
void Shader::setMat4 (
            const std::string & name,
            const glm::mat4 & mat ) const  [inline]
```

4.12.2.7 setVec2() [1/2]

```
void Shader::setVec2 (
            const std::string & name,
            const glm::vec2 & value ) const  [inline]
```

4.12.2.8 setVec2() [2/2]

```
void Shader::setVec2 (
            const std::string & name,
            float x,
            float y ) const  [inline]
```

4.12.2.9 setVec3() [1/2]

```
void Shader::setVec3 (
            const std::string & name,
            const glm::vec3 & value ) const  [inline]
```

4.12.2.10 setVec3() [2/2]

```
void Shader::setVec3 (
            const std::string & name,
            float x,
            float y,
            float z ) const  [inline]
```

4.12.2.11 setVec4() [1/2]

```
void Shader::setVec4 (
            const std::string & name,
            const glm::vec4 & value ) const  [inline]
```

4.12.2.12 setVec4() [2/2]

```
void Shader::setVec4 (
            const std::string & name,
            float x,
            float y,
            float z,
            float w )  [inline]
```

4.12.2.13 use()

```
void Shader::use ( ) const
```

Use the shader with glUseProgram.

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

- src/shader.h
- src/shader.cpp

4.13 SkyBox Class Reference

A Cubemap.

```
#include <skybox.h>
```

Public Member Functions

- SkyBox (const std::string &, const std::string &)

  Create a skybox by specifying the file path and the extension ex: world, .png, will use world_bk.png, ...
- void draw (Shader &sh)

  Draw the Skybox.

Protected Attributes

- GLuint vao
- GLuint cubemap

4.13.1 Detailed Description

A Cubemap.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 SkyBox()

```
SkyBox::SkyBox (
            const std::string & base,
            const std::string & extension )
```

Create a skybox by specifying the file path and the extension ex: world, .png, will use world_bk.png, ...

4.13.3 Member Function Documentation

4.13.3.1 draw()

```
void SkyBox::draw (
            Shader & sh )
```

Draw the Skybox.

4.13.4 Member Data Documentation

4.13.4.1 cubemap

```
GLuint SkyBox::cubemap  [protected]
```

4.13.4.2 vao

```
GLuint SkyBox::vao  [protected]
```

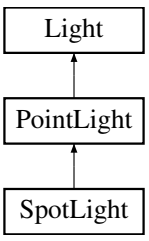
The documentation for this class was generated from the following files:

- src/skybox.h
- src/skybox.cpp

4.14 SpotLight Class Reference

```
#include <light.h>
```

Inheritance diagram for SpotLight:



Public Member Functions

- virtual void setUniforms (const Shader &shader, const std::string &uname) const
- virtual void ui ()

Public Attributes

- float cutOff_ = 12.5f
- float outerCutOff_ = 20.f
- glm::vec3 direction_ = {-10.f, -10.f, -10.f}

4.14.1 Member Function Documentation

4.14.1.1 setUniforms()

```
void SpotLight::setUniforms (
            const Shader & shader,
            const std::string & uname ) const  [virtual]
```

Reimplemented from PointLight.

4.14.1.2 ui()

```
void SpotLight::ui ( )  [virtual]
```

Reimplemented from PointLight.

4.14.2 Member Data Documentation

4.14.2.1 cutOff_

```
float SpotLight::cutOff_ = 12.5f
```

4.14.2.2 direction_

```
glm::vec3 SpotLight::direction_ = {-10.f, -10.f, -10.f}
```

4.14.2.3 outerCutOff_

```
float SpotLight::outerCutOff_ = 20.f
```

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

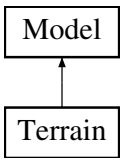
- src/light.h
- src/light.cpp

4.15 Terrain Class Reference

A Terrain.

```
#include <terrain.h>
```

Inheritance diagram for Terrain:



Public Member Functions

- Terrain (const Grid &)

  Generate a terrain from a grid.
- void draw (const Shader &shader)

  Draw the terrain This override set the model mat to nothing.
- void randomize ()

  Randomize the height of the terrain using a perlin noise.
- void ui ()

Static Public Member Functions

- static void randomize (std::vector< Vertex > &pPoints, unsigned int pSlicing)

  Randomize a given set of point.
- static void calculateNormals (std::vector< Vertex > &, const std::vector< unsigned int > &)

  Calculate normals for the given set of points and given set of triangls.

Additional Inherited Members

4.15.1 Detailed Description

A Terrain.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Terrain()

```
Terrain::Terrain (
            const Grid & pGrid )
```

Generate a terrain from a grid.

4.15.3 Member Function Documentation

4.15.3.1 calculateNormals()

```
void Terrain::calculateNormals (
            std::vector< Vertex > & pPoints,
            const std::vector< unsigned int > & pTriangles )  [static]
```

Calculate normals for the given set of points and given set of triangls.

Parameters

Points	
Triangle	indices

4.15.3.2 draw()

```
void Terrain::draw (
            const Shader & shader )  [virtual]
```

Draw the terrain This override set the model mat to nothing.

Reimplemented from Model.

4.15.3.3 randomize() [1/2]

```
void Terrain::randomize ( )
```

Randomize the height of the terrain using a perlin noise.

4.15.3.4 randomize() [2/2]

```
void Terrain::randomize (
            std::vector< Vertex > & pPoints,
            unsigned int pSlicing )  [static]
```

Randomize a given set of point.

Parameters

pPoints	

4.15.3.5 ui()

```
void Terrain::ui ( )  [virtual]
```

Reimplemented from Model.

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

- src/terrain.h
- src/terrain.cpp

4.16 Texture Struct Reference

```
#include <mesh.h>
```

Public Attributes

- GLuint id
- std::string type

4.16.1 Member Data Documentation

4.16.1.1 id

```
GLuint Texture::id
```

4.16.1.2 type

```
std::string Texture::type
```

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

- src/mesh.h

4.17 Turtle Class Reference

Unique instance of the main turtle application. Group everything required to make turtle work.

```
#include <turtle.h>
```

Public Member Functions

- float getDeltaTime () const
- int getWinHeight () const
- int getWinWidth () const
- void init ()

  Init the application Should be called only once.
- void terminate ()

  Correctly terminate the application compounds. Should be called only once.
- void mainLoop ()

  Start the main loop which will draw frames.
- void displayFrame ()

  Display of frame.
- void displayLights ()

  Display world lights.
- void displayUi ()

  Display the ui. Called before each frame contents.

Static Public Member Functions

- static Turtle & getInstance ()

  Return the instance of the application Create a new one if there isn't any.

4.17.1 Detailed Description

Unique instance of the main turtle application. Group everything required to make turtle work.

4.17.2 Member Function Documentation

4.17.2.1 displayFrame()

```
void Turtle::displayFrame ( )
```

Display of frame.

4.17.2.2 displayLights()

```
void Turtle::displayLights ( )
```

Display world lights.

4.17.2.3 displayUi()

```
void Turtle::displayUi ( )
```

Display the ui. Called before each frame contents.

4.17.2.4 getDeltaTime()

```
float Turtle::getDeltaTime ( ) const
```

Returns the delta time of the current frame

4.17.2.5 getInstance()

```
Turtle & Turtle::getInstance ( )  [static]
```

Return the instance of the application Create a new one if there isn't any.

Returns

4.17.2.6 getWinHeight()

```
int Turtle::getWinHeight ( ) const
```

4.17.2.7 getWinWidth()

```
int Turtle::getWinWidth ( ) const
```

4.17.2.8 init()

```
void Turtle::init ( )
```

Init the application Should be called only once.

4.17.2.9 mainLoop()

```
void Turtle::mainLoop ( )
```

Start the main loop which will draw frames.

4.17.2.10 terminate()

```
void Turtle::terminate ( )
```

Correctly terminate the application compounds. Should be called only once.

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

- src/turtle.h
- src/turtle.cpp

4.18 Vertex Struct Reference

```
#include <mesh.h>
```

Public Attributes

- glm::vec3 Position
- glm::vec3 Normal
- glm::vec3 TexCoords

4.18.1 Member Data Documentation

4.18.1.1 Normal

```
glm::vec3 Vertex::Normal
```

4.18.1.2 Position

```
glm::vec3 Vertex::Position
```

4.18.1.3 TexCoords

```
glm::vec3 Vertex::TexCoords
```

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

- src/mesh.h

Chapter 5

File Documentation

5.1 src/camera.cpp File Reference

```
#include "camera.h"
#include "turtle.h"
```

5.2 src/camera.h File Reference

```
#include <glm/glm.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/rotate_vector.hpp>
#include <GL/gl3w.h>
#include <GLFW/glfw3.h>
```

Classes

- class Camera

  Camera used to navigate in a scene.

5.3 src/fpsCamera.cpp File Reference

```
#include "fpsCamera.h"
#include "turtle.h"
```

5.4 src/fpsCamera.h File Reference

```
#include "camera.h"
```

Classes

- class FPSCamera

  Camera used to navigate in a scene.

5.5 src/grid.cpp File Reference

```
#include "grid.h"
```

5.6 src/grid.h File Reference

```
#include <glm/vec2.hpp>
#include <vector>
```

Classes

- class Grid

  A point grid.
- class GridGenerator

  Grid generator.

5.7 src/light.cpp File Reference

```
#include <imgui.h>
#include <glm/gtc/type_ptr.hpp>
#include "light.h"
```

5.8 src/light.h File Reference

```
#include <string>
#include <glm/glm.hpp>
#include "shader.h"
```

Classes

- class Light
- class DirectionLight
- class PointLight
- class SpotLight

5.9 src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include "stb_image.h"
#include "turtle.h"
#include <assimp/Importer.hpp>
#include <assimp/scene.h>
#include <assimp/material.h>
#include <assimp/postprocess.h>
```

Macros

- #define STB_IMAGE_IMPLEMENTATION

Functions

- int main ()

5.9.1 Macro Definition Documentation

5.9.1.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

5.9.2 Function Documentation

5.9.2.1 main()

```
int main ( )
```

5.10 src/mesh.cpp File Reference

```
#include "mesh.h"
#include <stdexcept>
#include <iostream>
```

5.11 src/mesh.h File Reference

```
#include "shader.h"
#include <vector>
#include <assimp/Importer.hpp>
#include <GL/gl3w.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
```

Classes

- struct Vertex
- struct Texture
- class Mesh

  Mesh wrapper.

5.12 src/model.cpp File Reference

```
#include "model.h"
#include <list>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <imgui.h>
#include "stb_image.h"
```

Functions

- GLuint textureFromFile (const char *path, const std::string &directory, bool gamma)

5.12.1 Function Documentation

5.12.1.1 textureFromFile()

```
GLuint textureFromFile (
            const char * path,
            const std::string & directory,
            bool gamma )
```

5.13 src/model.h File Reference

```
#include "shader.h"
#include "mesh.h"
#include <string>
#include <vector>
#include <assimp/Importer.hpp>
#include <assimp/scene.h>
#include <assimp/material.h>
#include <assimp/postprocess.h>
```

Classes

- class Model

  A set of one or more mesh.

Functions

- GLuint textureFromFile (const char *path, const std::string &directory, bool gamma=false)

5.13.1 Function Documentation

5.13.1.1 textureFromFile()

```
GLuint textureFromFile (
            const char * path,
            const std::string & directory,
            bool gamma = false )
```

5.14 src/object.cpp File Reference

```
#include "object.h"
#include <glm/gtc/matrix_transform.hpp>
#include <imgui.h>
```

5.15 src/object.h File Reference

```
#include <memory>
#include <glm/gtc/type_ptr.hpp>
#include "model.h"
#include "shader.h"
```

Classes

- class Object

  Wrapper of a model Can be positioned in the spaces Usefulle when creating instances, avoid to duplicate the model.

5.16 src/orbitCamera.cpp File Reference

```
#include "orbitCamera.h"
#include "turtle.h"
```

5.17 src/orbitCamera.h File Reference

```
#include "camera.h"
```

Classes

- class OrbitCamera

  Camera used to rotate around a round-shaped object.

5.18 src/shader.cpp File Reference

```
#include "shader.h"
#include <fstream>
```

5.19 src/shader.h File Reference

```
#include <string>
#include <GL/gl3w.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
```

Classes

- class Shader

5.20 src/skybox.cpp File Reference

```
#include "skybox.h"
#include "mesh.h"
#include <vector>
#include "stb_image.h"
```

5.21 src/skybox.h File Reference

```
#include "shader.h"
#include <string>
#include <vector>
```

Classes

- class SkyBox
  A Cubemap.

5.22 src/terrain.cpp File Reference

```
#include "terrain.h"
#include "PerlinNoise.h"
#include <imgui.h>
```

5.23 src/terrain.h File Reference

```
#include "grid.h"
#include "model.h"
```

Classes

- class Terrain
  A Terrain.

5.24 src/turtle.cpp File Reference

```
#include <cassert>
#include <iostream>
#include "turtle.h"
#include "grid.h"
#include "orbitCamera.h"
#include "fpsCamera.h"
#include "terrain.h"
```

5.25 src/turtle.h File Reference

```
#include <imgui.h>
#include "imgui_impl_glfw_gl3.h"
#include <GL/gl3w.h>
#include <GLFW/glfw3.h>
#include <vector>
#include <map>
#include <string>
#include <memory>
#include "shader.h"
#include "model.h"
#include "object.h"
#include "terrain.h"
#include "camera.h"
#include "light.h"
#include "skybox.h"
```

Classes

- class Turtle

  Unique instance of the main turtle application. Group everything required to make turtle work.

Index