Aladdin 3d

Generated by Doxygen 1.9.6

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 documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 aladdin_3d::BoundingBox Struct Reference

A bounding box struct.

`#include <BoundingBox.h>`

Public Attributes

- glm::vec3 **min**
- glm::vec3 **max**

4.1.1 Detailed Description

A bounding box struct.

This Struct represents the bounding box of an object.

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

- Sources/Structs/BoundingBox/BoundingBox.h

4.2 aladdin_3d::Camera Class Reference

Implements a camera class.

`#include <Camera.h>`

Public Member Functions

- Camera (glm::vec3 position, glm::vec3 direction, float fov, float near, float far, int width, int height)

  Constructs a camera instance.
- glm::mat4 getCameraMatrix ()

  Get the camera matrix.
- glm::vec3 getDirection ()

  Get the camera direction.
- glm::vec3 getPosition ()

  Get the camera position.
- glm::mat4 getProjection ()

  Get the projection matrix.
- glm::vec3 getUp ()

  Get the camera up vector.
- glm::mat4 getView ()

  Get the view matrix.
- void moveBack ()

  Move tha camera backwards.
- void moveDown ()

  Move tha camera down.
- void moveFront ()

  Move the camera forward.
- void moveLeft ()

  Move the camera: to the left, to the left.
- void moveRight ()

  Move the camera to the right.
- void moveUp ()

  Move tha camera up.
- void rotateDown ()

  Rotate the camera down.
- void rotateLeft ()

  Rotate the camera left.
- void rotateRight ()

  Rotate the camera right.
- void rotateUp ()

  Rotate the camera up.
- void update ()

  Calculate the camera matrix from the parameters.

4.2.1 Detailed Description

Implements a camera class.

Implements a camera class to handle the POV of OpenGL as well as the interactions with the window.

Author

Borja García Quiroga garcaqub@tcd.ie

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Camera()

```
aladdin_3d::Camera::Camera (
            glm::vec3 position,
            glm::vec3 direction,
            float fov,
            float near,
            float far,
            int width,
            int height )
```

Constructs a camera instance.

Constructs a camera instance and sets its initial values.

Parameters

position	The camera coordinates.
direction	Where the camera is looking.
fov	Camera field of view.
near	Minimum distance rendered.
far	maximum distance rendered.
width	The camera width.
height	The camera height.

4.2.3 Member Function Documentation

4.2.3.1 getCameraMatrix()

```
glm::mat4 aladdin_3d::Camera::getCameraMatrix ( )
```

Get the camera matrix.

Get the camera matrix.

4.2.3.2 getDirection()

```
glm::vec3 aladdin_3d::Camera::getDirection ( )
```

Get the camera direction.

Get the camera direction.

4.2.3.3 getPosition()

```
glm::vec3 aladdin_3d::Camera::getPosition ( )
```

Get the camera position.

Get the camera position.

4.2.3.4 getProjection()

```
glm::mat4 aladdin_3d::Camera::getProjection ( )
```

Get the projection matrix.

Get the projection matrix corresponding to this camera.

4.2.3.5 getUp()

```
glm::vec3 aladdin_3d::Camera::getUp ( )
```

Get the camera up vector.

Get the camera up vector.

4.2.3.6 getView()

```
glm::mat4 aladdin_3d::Camera::getView ( )
```

Get the view matrix.

Get the view matrix corresponding to this camera.

4.2.3.7 moveBack()

```
void aladdin_3d::Camera::moveBack ( )
```

Move tha camera backwards.

Move tha camera backwards.

4.2.3.8 moveDown()

```
void aladdin_3d::Camera::moveDown ( )
```

Move tha camera down.

Move tha camera down.

4.2.3.9 moveFront()

```
void aladdin_3d::Camera::moveFront ( )
```

Move the camera forward.

Move the camera forward.

4.2.3.10 moveLeft()

```
void aladdin_3d::Camera::moveLeft ( )
```

Move the camera: to the left, to the left.

Move the camera: to the left, to the left.

4.2.3.11 moveRight()

```
void aladdin_3d::Camera::moveRight ( )
```

Move the camera to the right.

Move the camera to the right.

4.2.3.12 moveUp()

```
void aladdin_3d::Camera::moveUp ( )
```

Move tha camera up.

Move tha camera up.

4.2.3.13 rotateDown()

```
void aladdin_3d::Camera::rotateDown ( )
```

Rotate the camera down.

Rotate the camera down.

4.2.3.14 rotateLeft()

```
void aladdin_3d::Camera::rotateLeft ( )
```

Rotate the camera left.

Rotate the camera left.

4.2.3.15 rotateRight()

```
void aladdin_3d::Camera::rotateRight ( )
```

Rotate the camera right.

Rotate the camera right.

4.2.3.16 rotateUp()

```
void aladdin_3d::Camera::rotateUp ( )
```

Rotate the camera up.

Rotate the camera up.

4.2.3.17 update()

```
void aladdin_3d::Camera::update ( )
```

Calculate the camera matrix from the parameters.

Calculates the camera matrix from the parameters.

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

- Sources/Classes/Camera/Camera.h
- Sources/Classes/Camera/Camera.cpp

4.3 aladdin_3d::EBO Class Reference

Implementation of a EBO class.

```
#include <EBO.h>
```

Public Member Functions

- EBO (const std::vector< GLuint > &indices)
  Constructs a Elements Buffer Object.
- void bind ()
  Binds the EBO.
- void remove ()
  Removes the EBO.
- void unbind ()
  Unbinds the EBO.

4.3.1 Detailed Description

Implementation of a EBO class.

Implementation of a EBO class that will allow us to bind it to the OpenGL pipe, destroy it or deactivate it.

Author

Borja García Quiroga garcaqub@tcd.ie

4.3.2 Constructor & Destructor Documentation

4.3.2.1 EBO()

```
aladdin_3d::EBO::EBO (
            const std::vector< GLuint > & indices )
```

Constructs a Elements Buffer Object.

Constructs a Elements Buffer Object and links its vertices.

Parameters

indices	Indices that will be linked.

4.3.3 Member Function Documentation

4.3.3.1 bind()

```
void aladdin_3d::EBO::bind ( )
```

Binds the EBO.

Binds the EBO in the GL pipe.

4.3.3.2 remove()

```
void aladdin_3d::EBO::remove ( )
```

Removes the EBO.

Removes the EBO from OpenGL.

4.3.3.3 unbind()

```
void aladdin_3d::EBO::unbind ( )
```

Unbinds the EBO.

Unbinds the EBO in the GL pipe.

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

- Sources/Classes/EBO/EBO.h
- Sources/Classes/EBO/EBO.cpp

4.4 aladdin_3d::Geometry Class Reference

Implementation of a Geometry class.

```
#include <Geometry.h>
```

Public Member Functions

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

  Initializes the Geometry.
- std::vector< GLuint > getIndices ()

  Get the indices of the geometry.
- std::vector< Texture > getTextures ()

  Get the textures.
- VAO getVAO ()

  Get the VAO.
- std::vector< Vertex > getVertices ()

  Get the vertices of the geometry.
- void draw (Shader &shader, Camera &camera)

  Draws the Geometry.
- BoundingBox getBoundingBox ()

  Gets the bounding box.
- void resetTransforms ()

  Reset.
- void rotate (float x, float y, float z, float angle)

  Add a translation matrix to the model.
- void scale (float x, float y, float z)

  Add a translation matrix to the model.
- void translate (float x, float y, float z)

  Add a translation matrix to the model.

4.4.1 Detailed Description

Implementation of a Geometry class.

Implementation of a Geometry class that will allow us to handle the geometric part of the objects in the VBOs.

Author

Borja Garcuiroga garcaqub@tcd.ie

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Geometry()

```
aladdin_3d::Geometry::Geometry (
            const std::vector< Vertex > & vertices,
            const std::vector< GLuint > & indices,
            const std::vector< Texture > & textures )
```

Initializes the Geometry.

Initializes the geometry and stores it.

Parameters

vertices	Vertices of the object.
indices	Indices of the vertices.
textures	Textures in connection with this geometry.

4.4.3 Member Function Documentation

4.4.3.1 draw()

```
void aladdin_3d::Geometry::draw (
            Shader & shader,
            Camera & camera )
```

Draws the Geometry.

Displays the Geometry in OpenGL.

4.4.3.2 getBoundingBox()

```
BoundingBox aladdin_3d::Geometry::getBoundingBox ( )
```

Gets the bounding box.

Gets the bounding box of the geometry.

Returns

The bounding box struct.

4.4.3.3 getIndices()

```
std::vector< GLuint > aladdin_3d::Geometry::getIndices ( )
```

Get the indices of the geometry.

Get the indices of the geometry.

4.4.3.4 getTextures()

```
std::vector< Texture > aladdin_3d::Geometry::getTextures ( )
```

Get the textures.

Get the textures.

4.4.3.5 getVAO()

```
VAO aladdin_3d::Geometry::getVAO ( )
```

Get the VAO.

Get the VAO.

4.4.3.6 getVertices()

```
std::vector< Vertex > aladdin_3d::Geometry::getVertices ( )
```

Get the vertices of the geometry.

Get the vertices of the geometry.

4.4.3.7 resetTransforms()

```
void aladdin_3d::Geometry::resetTransforms ( )
```

Reset.

Add a translation matrix to the model.

4.4.3.8 rotate()

```
void aladdin_3d::Geometry::rotate (
            float x,
            float y,
            float z,
            float angle )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

x	The x rotation.
y	The y rotation.
z	The z rotation.
angle	The angle to rotate.

4.4.3.9 scale()

```
void aladdin_3d::Geometry::scale (
            float x,
            float y,
            float z )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

x	The x scale.
y	The y scale.
z	The z scale.

4.4.3.10 translate()

```
void aladdin_3d::Geometry::translate (
            float x,
            float y,
            float z )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

x	The x translation.
y	The y translation.
z	The z translation.

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

- Sources/Classes/Geometry/Geometry.h
- Sources/Classes/Geometry/Geometry.cpp

4.5 aladdin_3d::Light Class Reference

Implementation of a Light class.

`#include <Light.h>`

Public Member Functions

- Light (glm::vec3 light_pos, glm::vec4 light_color)
  Constructs a Light from its components.
- glm::vec4 getColor ()
  Get the color of the light.
- glm::vec3 getPosition ()
  Get the position of the light.

4.5.1 Detailed Description

Implementation of a Light class.

Implementation of a Light class that will allow us light the scenes up.

Author

Borja Garcuiroga garcaqub@tcd.ie

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Light()

```
aladdin_3d::Light::Light (
            glm::vec3 light_pos,
            glm::vec4 light_color )
```

Constructs a Light from its components.

Constructs a Light from its components.

Parameters

light_pos	The position of the light.
light_color	The light color.

4.5.3 Member Function Documentation

4.5.3.1 getColor()

```
glm::vec4 aladdin_3d::Light::getColor ( )
```

Get the color of the light.

Get the color of the light.

4.5.3.2 getPosition()

```
glm::vec3 aladdin_3d::Light::getPosition ( )
```

Get the position of the light.

Get the position of the light.

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

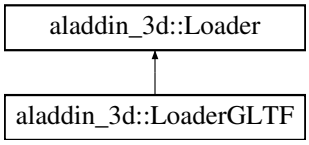
- Sources/Classes/Light/Light.h
- Sources/Classes/Light/Light.cpp

4.6 aladdin_3d::Loader Class Reference

Implements a Loader class.

```
#include <Loader.h>
```

Inheritance diagram for aladdin_3d::Loader:



Public Types

- enum LoaderTypes { **GLTF** }

  Declares the types of Loaders available.

Public Member Functions

- Loader (const char *filename)
  Build a loader instance.
- virtual void getGeometries (std::vector< Geometry > *geoms, std::vector< glm::mat4 > *matrices)=0
  Get the geometries from the loaded model.
- virtual void loadModel ()=0
  Loads the data from the file.

Static Public Member Functions

- static std::string readFileContents (const char *filename)
  Gets the content of a file as a string.

Protected Attributes

- std::vector< Geometry > **geometries**
- const char * **filename**
  The Geometries loaded by the model loader.
- std::vector< glm::mat4 > **transform_matrixes**
  Name of the file containing the model.

4.6.1 Detailed Description

Implements a Loader class.

Implements a loader class that will allow us to load models.

Author

Borja Garcuiroga garcaqub@tcd.ie

4.6.2 Member Enumeration Documentation

4.6.2.1 LoaderTypes

`enum aladdin_3d::Loader::LoaderTypes`

Declares the types of Loaders available.

Declares the types of Loaders available.

4.6.3 Constructor & Destructor Documentation

4.6.3.1 Loader()

```
aladdin_3d::Loader::Loader (
            const char * filename )
```

Build a loader instance.

Build a loader instance.

4.6.4 Member Function Documentation

4.6.4.1 getGeometries()

```
virtual void aladdin_3d::Loader::getGeometries (
            std::vector< Geometry > * geoms,
            std::vector< glm::mat4 > * matrices )  [pure virtual]
```

Get the geometries from the loaded model.

Get the geometries from the loaded model.

Parameters

geoms	Outputs the geometries returned.
matrices	Outputs the transformation matrices.

Implemented in aladdin_3d::LoaderGLTF.

4.6.4.2 loadModel()

```
virtual void aladdin_3d::Loader::loadModel ( )  [pure virtual]
```

Loads the data from the file.

Loads the data from the file.

Implemented in aladdin_3d::LoaderGLTF.

4.6.4.3 readFileContents()

```
std::string aladdin_3d::Loader::readFileContents (
            const char * filename )  [static]
```

Gets the content of a file as a string.

Gets the contents of a file, given its filename, and returns it as a string.

Parameters

filename	The name of the file to be read.
file_contents	An output string containing the contents of the file.

Exceptions

121-1001	Could not read file.

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

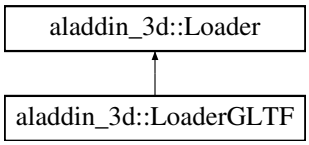
- Sources/Classes/Loader/Loader.h
- Sources/Classes/Loader/Loader.cpp

4.7 aladdin_3d::LoaderGLTF Class Reference

Implements a GLTF Loader class.

`#include <LoaderGLTF.h>`

Inheritance diagram for aladdin_3d::LoaderGLTF:



Public Member Functions

- LoaderGLTF (const char *filename)

  Build a loader instance.
- void getGeometries (std::vector< Geometry > *geoms, std::vector< glm::mat4 > *matrices)

  Get the geometries from the loaded model.
- void loadModel ()

  Loads the data from the file.

Public Member Functions inherited from aladdin_3d::Loader

- Loader (const char *filename)

  Build a loader instance.
- virtual void getGeometries (std::vector< Geometry > *geoms, std::vector< glm::mat4 > *matrices)=0

  Get the geometries from the loaded model.
- virtual void loadModel ()=0

  Loads the data from the file.

Additional Inherited Members

Public Types inherited from aladdin_3d::Loader

- enum LoaderTypes { **GLTF** }

  Declares the types of Loaders available.

Static Public Member Functions inherited from aladdin_3d::Loader

- static std::string readFileContents (const char *filename)

  Gets the content of a file as a string.

Protected Attributes inherited from aladdin_3d::Loader

- std::vector< Geometry > **geometries**
- const char * **filename**

  The Geometries loaded by the model loader.
- std::vector< glm::mat4 > **transform_matrixes**

  Name of the file containing the model.

4.7.1 Detailed Description

Implements a GLTF Loader class.

Implements a loader class that will allow us to load GLTF models.

Author

Borja Garcuiroga garcaqub@tcd.ie

4.7.2 Constructor & Destructor Documentation

4.7.2.1 LoaderGLTF()

```
aladdin_3d::LoaderGLTF::LoaderGLTF (
            const char * filename )
```

Build a loader instance.

Build a loader instance.

4.7.3 Member Function Documentation

4.7.3.1 getGeometries()

```
void aladdin_3d::LoaderGLTF::getGeometries (
            std::vector< Geometry > * geoms,
            std::vector< glm::mat4 > * matrices )  [virtual]
```

Get the geometries from the loaded model.

Get the geometries from the loaded model.

Parameters

geoms	Outputs the geometries returned.
matrices	Outputs the transformation matrices.

Implements aladdin_3d::Loader.

4.7.3.2 loadModel()

```
void aladdin_3d::LoaderGLTF::loadModel ( )  [virtual]
```

Loads the data from the file.

Loads the data from the file.

Implements aladdin_3d::Loader.

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

- Sources/Classes/LoaderGLTF/LoaderGLTF.h
- Sources/Classes/LoaderGLTF/LoaderGLTF.cpp

4.8 aladdin_3d::Object Class Reference

Public Member Functions

- Object (const char *filename, const char *filetype)
  Loads a model in the gltf format.
- Object (std::vector< Geometry > geometries)
  Loads the object from specified geometries.
- void draw (Shader &shader, Camera &camera)
  Draws this object.
- BoundingBox getBoundingBox ()
  Gets the bounding box.
- std::vector< Geometry > getGeometries ()
  Get the geometries of the object.
- std::vector< glm::mat4 > getGeometryMatrices ()
  Get the matrices of the geometries.
- void resetTransforms ()
  Reset.
- void rotate (float x, float y, float z, float angle)
  Add a translation matrix to the model.
- void rotate (int num, float x, float y, float z, float angle)
  Add a translation matrix to the model.
- void scale (float x, float y, float z)
  Add a translation matrix to the model.
- void scale (int num, float x, float y, float z)
  Add a translation matrix to the model.
- void translate (float x, float y, float z)
  Add a translation matrix to the model.
- void translate (int num, float x, float y, float z)
  Add a translation matrix to the model.

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Object() [1/2]

```
aladdin_3d::Object::Object (
            const char * filename,
            const char * filetype )
```

Loads a model in the gltf format.

Loads in a GLTF model from a fileand stores tha information in 'data', 'JSON', and 'file'.

Parameters

filename	The name of the model file.
filetype	The type of the model file.

4.8.1.2 Object() [2/2]

```
aladdin_3d::Object::Object (
            std::vector< Geometry > geometries )
```

Loads the object from specified geometries.

Loads the object from specified geometries.

Parameters

geometries	The geometries that will be part of the object.

4.8.2 Member Function Documentation

4.8.2.1 draw()

```
void aladdin_3d::Object::draw (
            aladdin_3d::Shader & shader,
            aladdin_3d::Camera & camera )
```

Draws this object.

Draws this object.

4.8.2.2 getBoundingBox()

```
BoundingBox aladdin_3d::Object::getBoundingBox ( )
```

Gets the bounding box.

Gets the bounding box of all the geometries.

Returns

The bounding box struct.

4.8.2.3 getGeometries()

```
std::vector< Geometry > aladdin_3d::Object::getGeometries ( )
```

Get the geometries of the object.

Get the geometries of the object.

4.8.2.4 getGeometryMatrices()

```
std::vector< glm::mat4 > aladdin_3d::Object::getGeometryMatrices ( )
```

Get the matrices of the geometries.

Get the matrices of the geometries.

4.8.2.5 resetTransforms()

```
void aladdin_3d::Object::resetTransforms ( )
```

Reset.

Add a translation matrix to the model.

4.8.2.6 rotate() [1/2]

```
void aladdin_3d::Object::rotate (
            float x,
            float y,
            float z,
            float angle )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

x	The x rotation.
y	The y rotation.
z	The z rotation.
angle	The angle to rotate.

4.8.2.7 rotate() [2/2]

```
void aladdin_3d::Object::rotate (
            int num,
            float x,
            float y,
            float z,
            float angle )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

num	The geometry index this will apply to.
x	The x rotation.
y	The y rotation.
z	The z rotation.
angle	The angle to rotate.

4.8.2.8 scale() [1/2]

```
void aladdin_3d::Object::scale (
            float x,
            float y,
            float z )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

x	The x scale.
y	The y scale.
z	The z scale.

4.8.2.9 scale() [2/2]

```
void aladdin_3d::Object::scale (
            int num,
            float x,
            float y,
            float z )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

num	The geometry index this will apply to.
x	The x scale.
y	The y scale.
z	The z scale.

4.8.2.10 translate() [1/2]

```
void aladdin_3d::Object::translate (
            float x,
            float y,
            float z )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

x	The x translation.
y	The y translation.
z	The z translation.

4.8.2.11 translate() [2/2]

```
void aladdin_3d::Object::translate (
            int num,
            float x,
            float y,
            float z )
```

Add a translation matrix to the model.

Add a translation matrix to the model.

Parameters

num	The geometry index this will apply to.
x	The x translation.
y	The y translation.
z	The z translation.

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

- Sources/Classes/Object/Object.h
- Sources/Classes/Object/Object.cpp

4.9 aladdin_3d::Shader Class Reference

Implementation of a Shader class.

`#include <Shader.h>`

Public Member Functions

- Shader ()

  Construct the shader instance.
- Shader (const char *vertex_filename, const char *fragment_filename)

  Construct the shader instance.
- unsigned int getProgramID ()

  Returns the program ID.
- void activate ()

  Activate this shader program.
- void passBool (const std::string &name, bool value)

  Pass a given bool to the shaders.
- void passCamera (Camera camera)

  Pass the camera matrix and camera position to the shader.
- void passLight (Light light)

  Pass a light to the shader.
- void passInt (const std::string &name, int value)

  Pass a given integer to the shaders.
- void passFloat (const std::string &name, float value)

  Pass a given float to the shaders.
- void passTexture (Texture texture)

  Pass a texture to the shader.
- void remove ()

  Remove the shader from OpenGL.

4.9.1 Detailed Description

Implementation of a Shader class.

Implementation of a Shader class to handle loading, activation and errors in vertex and fragment shaders.

Author

Borja Garcuiroga garcaqub@tcd.ie

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Shader() [1/2]

```
aladdin_3d::Shader::Shader ( )
```

Construct the shader instance.

Construct the shader instance by passing no files.

4.9.2.2 Shader() [2/2]

```
aladdin_3d::Shader::Shader (
            const char * vertex_filename,
            const char * fragment_filename )
```

Construct the shader instance.

Construct the shader instance by passing the shaders' files.

Parameters

vertex_filename	Vertex shader filename.
fragment_filename	Fragment shader filename.

4.9.3 Member Function Documentation

4.9.3.1 activate()

```
void aladdin_3d::Shader::activate ( )
```

Activate this shader program.

Activate this shader program and start using it.

4.9.3.2 getProgramID()

```
unsigned int aladdin_3d::Shader::getProgramID ( )
```

Returns the program ID.

Returns the program ID for this Shader program.

Returns

The program ID as an unsigned integer.

4.9.3.3 passBool()

```
void aladdin_3d::Shader::passBool (
            const std::string & name,
            bool value )
```

Pass a given bool to the shaders.

Pass a given bool variable to the shader program.

Parameters

name	The name that the variable will receive within the shaders.
value	The bool to be passed to the program.

4.9.3.4 passCamera()

```
void aladdin_3d::Shader::passCamera (
            Camera camera )
```

Pass the camera matrix and camera position to the shader.

Pass the camera matrix and camera position to the shader.

Parameters

camera	The camera.

4.9.3.5 passFloat()

```
void aladdin_3d::Shader::passFloat (
```

```
            const std::string & name,
            float value )
```

Pass a given float to the shaders.

Pass a given float variable to the shader program.

Parameters

name	The name that the variable will receive within the shaders.
value	The float to be passed to the program.

4.9.3.6 passInt()

```
void aladdin_3d::Shader::passInt (
            const std::string & name,
            int value )
```

Pass a given integer to the shaders.

Pass a given integer variable to the shader program.

Parameters

name	The name that the variable will receive within the shaders.
value	The int to be passed to the program.

4.9.3.7 passLight()

```
void aladdin_3d::Shader::passLight (
            Light light )
```

Pass a light to the shader.

Pass a light to the shader.

Parameters

light	the light that will be passed to tha shader.

4.9.3.8 passTexture()

```
void aladdin_3d::Shader::passTexture (
            Texture texture )
```

Pass a texture to the shader.

Pass a texture to the shader.

Parameters

texture	The texture itself.

4.9.3.9 remove()

```
void aladdin_3d::Shader::remove ( )
```

Remove the shader from OpenGL.

Remove the shader from OpenGL.

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

- Sources/Classes/Shader/Shader.h
- Sources/Classes/Shader/Shader.cpp

4.10 aladdin_3d::Texture Class Reference

Implements a texture class to handle object textures.

```
#include <Texture.h>
```

Public Member Functions

- Texture (const char *image, const char *type, GLuint slot)
  Creates a texture from an image.
- GLuint getID ()
  Get the ID of the texture.
- GLuint getSlot ()
  Get the slot of the texture.
- int getWidth ()
  Gets the width of the image.
- int getHeight ()
  Get the height of the image.
- int getChannels ()
  Gets the number of channels of the texture.
- std::string getName ()
  Gets the texture name.
- void bind ()
  Binds the texture.
- void remove ()
  Removes the texture from OpenGL.
- void unbind ()
  Unbinds the texture.

4.10.1 Detailed Description

Implements a texture class to handle object textures.

Implements a texture object to handle textures and their content to use with the objects.

Author

Borja Garcuiroga garcaqub@tcd.ie

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Texture()

```
aladdin_3d::Texture::Texture (
            const char * image,
            const char * type,
            GLuint slot )
```

Creates a texture from an image.

Creates a textures and passes it to OpenGL.

Parameters

image	Image containing the texture.
type	Texture type.
slot	Texture slot.

4.10.3 Member Function Documentation

4.10.3.1 bind()

```
void aladdin_3d::Texture::bind ( )
```

Binds the texture.

Binds the texture.

4.10.3.2 getChannels()

```
int aladdin_3d::Texture::getChannels ( )
```

Gets the number of channels of the texture.

Gets the number of channels that the texture has.

Returns

The number of channels the texture has.

4.10.3.3 getHeight()

```
int aladdin_3d::Texture::getHeight ( )
```

Get the height of the image.

Gets the height of the image in pixels.

Returns

The height of the image in pixels.

4.10.3.4 getID()

```
GLuint aladdin_3d::Texture::getID ( )
```

Get the ID of the texture.

Get the ID of the texture.

Returns

The ID of the texture.

4.10.3.5 getName()

```
std::string aladdin_3d::Texture::getName ( )
```

Gets the texture name.

Gets the texture name as a char array.

Returns

A char string containing the name name of the texture.

4.10.3.6 getSlot()

```
GLuint aladdin_3d::Texture::getSlot ( )
```

Get the slot of the texture.

Get the slot of the texture.

Returns

The slot of the texture.

4.10.3.7 getWidth()

```
int aladdin_3d::Texture::getWidth ( )
```

Gets the width of the image.

Gets the width of the image in pixels.

Returns

The width of the image in pixels.

4.10.3.8 remove()

```
void aladdin_3d::Texture::remove ( )
```

Removes the texture from OpenGL.

Removes the texture from OpenGL.

4.10.3.9 unbind()

```
void aladdin_3d::Texture::unbind ( )
```

Unbinds the texture.

Unbinds the texture.

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

- Sources/Classes/Texture/Texture.h
- Sources/Classes/Texture/Texture.cpp

4.11 aladdin_3d::VAO Class Reference

Implementation of a VAO class.

`#include <VAO.h>`

Public Member Functions

- VAO ()

  Constructs a Vertex Array Object.
- void bind ()

  Binds the VBO.
- void link_attribute (VBO &vbo, GLuint layout, GLuint num_components, GLenum type, GLsizeiptr step, void *offset)

  Links a VBO attribute to the VAO.
- void remove ()

  Remove the VAO.
- void unbind ()

  Unbinds the VBO.

4.11.1 Detailed Description

Implementation of a VAO class.

Implementation of a VAO class that will allow us to bind it to the OpenGL pipe, destroy it or deactivate it.

Author

Borja García Quiroga garcaqub@tcd.ie

4.11.2 Constructor & Destructor Documentation

4.11.2.1 VAO()

`aladdin_3d::VAO::VAO ( )`

Constructs a Vertex Array Object.

Constructs a Vertex Array Object.

4.11.3 Member Function Documentation

4.11.3.1 bind()

```
void aladdin_3d::VAO::bind ( )
```

Binds the VBO.

Binds the VBO in the GL pipe.

4.11.3.2 link_attribute()

```
void aladdin_3d::VAO::link_attribute (
            VBO & vbo,
            GLuint layout,
            GLuint num_components,
            GLenum type,
            GLsizeiptr step,
            void * offset )
```

Links a VBO attribute to the VAO.

Links a VBO attribute such as color, UV, or others.

Parameters

vbo	The VBO to link the attribute to.
layout	The layout identifier that will be used in the shader.
num_components	The number of components that are in the list.
type	The type of data that we will be passing.
step	The amount of bytes we have to skip to find the next item.
offset	The amount of data we have to skip to find the first item.

4.11.3.3 remove()

```
void aladdin_3d::VAO::remove ( )
```

Remove the VAO.

Removes the VAO in GL.

4.11.3.4 unbind()

```
void aladdin_3d::VAO::unbind ( )
```

Unbinds the VBO.

Unbinds the VBO in the GL pipe.

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

- Sources/Classes/VAO/VAO.h
- Sources/Classes/VAO/VAO.cpp

4.12 aladdin_3d::VBO Class Reference

Implementation of a VBO class.

`#include <VBO.h>`

Public Member Functions

- VBO (const std::vector< Vertex > &vertices)
  Constructs a Vertex Buffer Object.
- void bind ()
  Binds the VBO.
- void remove ()
  Removes the VBO.
- void unbind ()
  Unbinds the VBO.

4.12.1 Detailed Description

Implementation of a VBO class.

Implementation of a VBO class that will allow us to bind it to the OpenGL pipe, destroy it or deactivate it.

Author

Borja García Quiroga garcaqub@tcd.ie

4.12.2 Constructor & Destructor Documentation

4.12.2.1 VBO()

```
aladdin_3d::VBO::VBO (
            const std::vector< Vertex > & vertices )
```

Constructs a Vertex Buffer Object.

Constructs a Vertex Buffer Object and links its vertices.

Parameters

vertices	Vertices that will be linked.

4.12.3 Member Function Documentation

4.12.3.1 bind()

```
void aladdin_3d::VBO::bind ( )
```

Binds the VBO.

Binds the VBO in the GL pipe.

4.12.3.2 remove()

```
void aladdin_3d::VBO::remove ( )
```

Removes the VBO.

Removes the VBO from OpenGL.

4.12.3.3 unbind()

```
void aladdin_3d::VBO::unbind ( )
```

Unbinds the VBO.

Unbinds the VBO in the GL pipe.

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

- Sources/Classes/VBO/VBO.h
- Sources/Classes/VBO/VBO.cpp

4.13 aladdin_3d::Vertex Struct Reference

A geometry vertex.

```
#include <Vertex.h>
```

Public Attributes

- glm::vec3 **position**
- glm::vec3 **normal**

  3D coordinates of the vertex.
- glm::vec3 **color**

  Normal vector of the vertex.
- glm::vec2 **uv**

  Color of the vertex in RGB.

4.13.1 Detailed Description

A geometry vertex.

This Struct represents a vertex of a given geometry and all its attributes.

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

- Sources/Structs/Vertex/Vertex.h

Chapter 5

File Documentation

5.1 Sources/Classes/Camera/Camera.cpp File Reference

Camera class implementation file.

```
#include "Camera.h"
#include <iostream>
#include "glew/glew.h"
#include "glm/glm.hpp"
#include "glm/gtc/matrix_transform.hpp"
#include "glm/gtc/type_ptr.hpp"
#include <glm/gtx/string_cast.hpp>
```

5.1.1 Detailed Description

Camera class implementation file.

Version

1.0.0 (2022-11-23)

Date

2022-11-23

Author

Borja Garc♦a Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.2 Sources/Classes/Camera/Camera.h File Reference

Camera class header file.

```
#include "glm/glm.hpp"
```

Classes

- class aladdin_3d::Camera

  Implements a camera class.

5.2.1 Detailed Description

Camera class header file.

Version

1.0.0 (2022-11-23)

Date

2022-11-23

Author

Borja Garc◆a Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.3 Camera.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASSES_CAMERA_H_
00017 #define ALADDIN_3D_CLASSES_CAMERA_H_
00018
00019 #include "glm/glm.hpp"
00020
00021 namespace aladdin_3d {
00022
00031     class Camera {
00032
00033         public:
00034
00048             Camera(glm::vec3 position, glm::vec3 direction, float fov, float near, float far, int
      width, int height);
00049
00055             glm::mat4 getCameraMatrix();
00056
00062             glm::vec3 getDirection();
00063
00069             glm::vec3 getPosition();
```

```
00070
00076             glm::mat4 getProjection();
00077
00083             glm::vec3 getUp();
00084
00090             glm::mat4 getView();
00091
00097             void moveBack();
00098
00104             void moveDown();
00105
00111             void moveFront();
00112
00118             void moveLeft();
00119
00125             void moveRight();
00126
00132             void moveUp();
00133
00139             void rotateDown();
00140
00146             void rotateLeft();
00147
00153             void rotateRight();
00154
00160             void rotateUp();
00161
00167             void update();
00168
00169         private:
00170
00171             glm::mat4 camera_matrix;
00172             glm::vec3 direction;
00173             float far;
00174             float fov;
00175             float near;
00176             glm::vec3 position;
00177             glm::vec3 up;
00178             int window_height;
00179             int window_width;
00180             glm::mat4 view;
00181             glm::mat4 projection;
00182
00183             const float speed = 0.25f;
00184             const float horizontal_rotation = 3.0f;
00185             const float vertical_rotation = 0.1f;
00186
00187     };
00188
00189 }  // namespace aladdin_3d
00190
00191 #endif
```

5.4 Sources/Classes/EBO/EBO.cpp File Reference

EBO class implementation file.

```
#include "EBO.h"
#include <vector>
#include "glew/glew.h"
```

5.4.1 Detailed Description

EBO class implementation file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja García Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.5 Sources/Classes/EBO/EBO.h File Reference

EBO class header file.

```
#include <vector>
#include "glew/glew.h"
```

Classes

- class aladdin_3d::EBO

  Implementation of a EBO class.

5.5.1 Detailed Description

EBO class header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja García Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.6 EBO.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASS_EBO_H_
00017 #define ALADDIN_3D_CLASS_EBO_H_
00018
00019 #include <vector>
00020
00021 #include "glew/glew.h"
00022
00023 namespace aladdin_3d {
00024
00033     class EBO {
00034
00035         public:
00036
00044             EBO(const std::vector<GLuint> &indices);
00045
00051             void bind();
00052
00058             void remove();
00059
00065             void unbind();
00066
00067         private:
00068
00069             GLuint ID; // GL ID of the EBO.
00070
00071     };
00072
00073 }  // namespace aladdin_3d
00074
00075 #endif
```

5.7 Sources/Classes/Geometry/Geometry.cpp File Reference

Geometry class implementation file.

```
#include "Geometry.h"
#include <vector>
#include <stdexcept>
#include "glew/glew.h"
#include "glm/glm.hpp"
#include "glm/gtc/type_ptr.hpp"
#include "Classes/Camera/Camera.h"
#include "Classes/EBO/EBO.h"
#include "Classes/Shader/Shader.h"
#include "Classes/Texture/Texture.h"
#include "Classes/VAO/VAO.h"
#include "Structs/Vertex/Vertex.h"
#include "Structs/BoundingBox/BoundingBox.h"
```

5.7.1 Detailed Description

Geometry class implementation file.

Version

1.0.0 (2022-11-26)

Date

2022-11-26

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.8 Sources/Classes/Geometry/Geometry.h File Reference

Geometry class header file.

```
#include <vector>
#include "glew/glew.h"
#include "glm/glm.hpp"
#include "glm/gtc/type_ptr.hpp"
#include "Classes/Camera/Camera.h"
#include "Classes/Shader/Shader.h"
#include "Classes/Texture/Texture.h"
#include "Classes/EBO/EBO.h"
#include "Classes/VBO/VBO.h"
#include "Classes/VAO/VAO.h"
#include "Structs/Vertex/Vertex.h"
#include "Structs/BoundingBox/BoundingBox.h"
```

Classes

- class aladdin_3d::Geometry

  Implementation of a Geometry class.

5.8.1 Detailed Description

Geometry class header file.

Version

1.0.0 (2022-11-26)

Date

2022-11-26

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.9 Geometry.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASSES_GEOMETRY_H_
00017 #define ALADDIN_3D_CLASSES_GEOMETRY_H_
00018
00019 #include <vector>
00020
00021 #include "glew/glew.h"
00022 #include "glm/glm.hpp"
00023 #include "glm/gtc/type_ptr.hpp"
00024
00025 #include "Classes/Camera/Camera.h"
00026 #include "Classes/Shader/Shader.h"
00027 #include "Classes/Texture/Texture.h"
00028 #include "Classes/EBO/EBO.h"
00029 #include "Classes/VBO/VBO.h"
00030 #include "Classes/VAO/VAO.h"
00031 #include "Structs/Vertex/Vertex.h"
00032 #include "Structs/BoundingBox/BoundingBox.h"
00033
00034 namespace aladdin_3d {
00035
00044     class Geometry {
00045
00046         public:
00047
00057             Geometry(const std::vector<Vertex> &vertices, const std::vector<GLuint> &indices, const
      std::vector<Texture> &textures);
00058
00064             std::vector<GLuint> getIndices();
00065
00071             std::vector<Texture> getTextures();
00072
00078             VAO getVAO();
00079
00085             std::vector<Vertex> getVertices();
00086
00092             void draw(Shader &shader, Camera &camera);
00093
00101             BoundingBox getBoundingBox();
00102
00108             void resetTransforms();
00109
00120             void rotate(float x, float y, float z, float angle);
00121
00131             void scale(float x, float y, float z);
00132
00142             void translate(float x, float y, float z);
00143
00144         private:
00145
00151             void updateNormalMatrix();
00152
00153             std::vector<GLuint> indices;
00154             std::vector<Texture> textures;
00155             VAO vao;
00156             std::vector<Vertex> vertices;
00157             glm::mat4 transforms = glm::mat4(1.0f);
00158
00159     };
00160
00161 }  // namespace aladdin_3d
00162
00163 #endif
```

5.10 Sources/Classes/Light/Light.cpp File Reference

Light class implementation file.

```
#include "Light.h"
#include "glew/glew.h"
```

5.10.1 Detailed Description

Light class implementation file.

Version

1.0.0 (2022-10-27)

Date

2022-10-27

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.11 Sources/Classes/Light/Light.h File Reference

Light class header file.

```
#include "glew/glew.h"
#include "glm/glm.hpp"
```

Classes

- class aladdin_3d::Light

  Implementation of a Light class.

5.11.1 Detailed Description

Light class header file.

Version

1.0.0 (2022-10-27)

Date

2022-10-27

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.12 Light.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASSES_LIGHT_H_
00017 #define ALADDIN_3D_CLASSES_LIGHT_H_
00018
00019 #include "glew/glew.h"
00020 #include "glm/glm.hpp"
00021
00022 namespace aladdin_3d {
00023
00031     class Light {
00032
00033     public:
00034
00043         Light(glm::vec3 light_pos, glm::vec4 light_color);
00044
00050         glm::vec4 getColor();
00051
00057         glm::vec3 getPosition();
00058
00059     private:
00060
00061         glm::vec3 position;
00062         glm::vec4 color;
00063
00064     };
00065
00066 }  // namespace aladdin_3d
00067
00068 #endif
```

5.13 Sources/Classes/Loader/Loader.cpp File Reference

Loader class implementation file.

```
#include "Loader.h"
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>
```

5.13.1 Detailed Description

Loader class implementation file.

Version

1.0.0 (2022-11-27)

Date

2022-11-27

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

Version

1.0.0 (2022-11-27)

Date

2022-11-27

Author

Borja Garcuiroga garcaqub@tcd.ie

The code in this class has been partially based on the OpenGL Tutorials code. The auxiliary functions have been grabbed from the repository below and belong to Victor Gordan.

See also

https://github.com/VictorGordan/opengl-tutorials

Copyright (c) Borja Garcuiroga, All Rights Reserved.

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.14 Sources/Classes/Loader/Loader.h File Reference

Loader class header file.

```
#include <string>
#include <vector>
#include "Classes/Geometry/Geometry.h"
```

Classes

- class aladdin_3d::Loader

  Implements a Loader class.

5.14.1 Detailed Description

Loader class header file.

Version

1.0.0 (2022-11-27)

Date

2022-11-27

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.15 Loader.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASSES_LOADER_H_
00017 #define ALADDIN_3D_CLASSES_LOADER_H_
00018
00019 #include <string>
00020 #include <vector>
00021
00022 #include "Classes/Geometry/Geometry.h"
00023
00024 namespace aladdin_3d {
00025
00033     class Loader {
00034
00035         public:
00036
00042             enum LoaderTypes { GLTF };
00043
00049             Loader(const char *filename);
00050
00059             virtual void getGeometries(std::vector<Geometry> *geoms, std::vector<glm::mat4> *matrices)
      = 0;
00060
00066             virtual void loadModel() = 0;
00067
00078             static std::string readFileContents(const char* filename);
00079
00080         protected:
00081
00082             std::vector<Geometry> geometries;
00083             const char *filename;
00084             std::vector<glm::mat4> transform_matrixes;
00085
00086     };
00087
00088 }
00089
00090 #endif
```

5.16 LoaderGLTF.h

```
00001
00016 #ifndef ALADDIN_3D_CLASSES_LOADER_GLTF_H_
00017 #define ALADDIN_3D_CLASSES_LOADER_GLTF_H_
00018
00019 #include "Classes/Loader/Loader.h"
00020
00021 #include <vector>
00022
00023 #include "glm/glm.hpp"
00024 #include "json/json.h"
00025
00026 namespace aladdin_3d {
00027
00035     class LoaderGLTF : public Loader {
00036
00037         public:
00038
00044             LoaderGLTF(const char* filename);
00045
00054             void getGeometries(std::vector<Geometry> *geoms, std::vector<glm::mat4> *matrices);
00055
00061             void loadModel();
00062
00063         private:
00064
00072             void loadGeometry(unsigned int indMesh);
00073
00082             void recursiveGetNode(unsigned int nextNode, glm::mat4 matrix = glm::mat4(1.0f));
00083
00084             // Interprets the binary data into floats, indices, and textures
00085             std::vector<float> getFloats(nlohmann::json accessor);
00086             std::vector<GLuint> getIndices(nlohmann::json accessor);
00087             std::vector<Texture> getTextures();
00088
00089             // Assembles all the floats into vertices
00090             std::vector<Vertex> assembleVertices(std::vector<glm::vec3> positions,
      std::vector<glm::vec3> normals, std::vector<glm::vec2> texUVs);
00091
00092             // Helps with the assembly from above by grouping floats
00093             std::vector<glm::vec2> groupFloatsVec2(std::vector<float> floatVec);
00094             std::vector<glm::vec3> groupFloatsVec3(std::vector<float> floatVec);
00095             std::vector<glm::vec4> groupFloatsVec4(std::vector<float> floatVec);
00096
00097             std::vector<unsigned char> bin_data;
00098             nlohmann::json json_file;
00099
00100     };
00101
00102 }
00103
00104 #endif
00105
```

5.17 Sources/Classes/Object/Object.cpp File Reference

Object class implementation file.

```
#include "Object.h"
#include <cassert>
#include <iostream>
#include "json/json.h"
#include "Classes/Loader/Loader.h"
#include "Classes/LoaderGLTF/LoaderGLTF.h"
#include "Structs/Vertex/Vertex.h"
#include "Structs/BoundingBox/BoundingBox.h"
```

5.17.1 Detailed Description

Object class implementation file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.18 Sources/Classes/Object/Object.h File Reference

Object class header file.

```
#include <vector>
#include "json/json.h"
#include "Classes/Geometry/Geometry.h"
#include "Classes/Loader/Loader.h"
#include "Structs/BoundingBox/BoundingBox.h"
```

Classes

- class aladdin_3d::Object

5.18.1 Detailed Description

Object class header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.19 Object.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASSES_OBJECT_H_
00017 #define ALADDIN_3D_CLASSES_OBJECT_H_
00018
00019 #include <vector>
00020
00021 #include "json/json.h"
00022
00023 #include "Classes/Geometry/Geometry.h"
00024 #include "Classes/Loader/Loader.h"
00025 #include "Structs/BoundingBox/BoundingBox.h"
00026
00027 namespace aladdin_3d {
00028
00029     class Object {
00030
00031         public:
00032
00041             Object(const char* filename, const char *filetype);
00042
00050             Object(std::vector<Geometry> geometries);
00051
00057             void draw(Shader &shader, Camera &camera);
00058
00066             BoundingBox getBoundingBox();
00067
00073             std::vector<Geometry> getGeometries();
00074
00080             std::vector<glm::mat4> getGeometryMatrices();
00081
00087             void resetTransforms();
00088
00099             void rotate(float x, float y, float z, float angle);
00100
00112             void rotate(int num, float x, float y, float z, float angle);
00113
00123             void scale(float x, float y, float z);
00124
00135             void scale(int num, float x, float y, float z);
00136
00146             void translate(float x, float y, float z);
00147
00158             void translate(int num, float x, float y, float z);
00159
00160         private:
00161
00162             // All the geometries and transformations
00163             std::vector<Geometry> geoms;
00164             std::vector<glm::mat4> matrices_geoms;
00165
00166     };
00167
00168 }
00169
00170 #endif
```

5.20 Sources/Classes/Shader/Shader.cpp File Reference

Shader class implementation file.

```
#include "Shader.h"
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>
#include "glew/glew.h"
#include "glm/glm.hpp"
#include "glm/gtc/type_ptr.hpp"
#include <glm/gtx/string_cast.hpp>
#include "Classes/Camera/Camera.h"
#include "Classes/Light/Light.h"
#include "Classes/Texture/Texture.h"
```

5.20.1 Detailed Description

Shader class implementation file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.21 Sources/Classes/Shader/Shader.h File Reference

Shader class header file.

```
#include <string>
#include "glm/glm.hpp"
#include "Classes/Camera/Camera.h"
#include "Classes/Light/Light.h"
#include "Classes/Texture/Texture.h"
```

Classes

- class aladdin_3d::Shader

  Implementation of a Shader class.

5.21.1 Detailed Description

Shader class header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.22 Shader.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_SHADER_H_
00017 #define ALADDIN_3D_SHADER_H_
00018
00019 #include <string>
00020
00021 #include "glm/glm.hpp"
00022
00023 #include "Classes/Camera/Camera.h"
00024 #include "Classes/Light/Light.h"
00025 #include "Classes/Texture/Texture.h"
00026
00027 namespace aladdin_3d {
00028
00037     class Shader {
00038
00039     public:
00040
00046         Shader();
00047
00056         Shader(const char* vertex_filename, const char* fragment_filename);
00057
00065         unsigned int getProgramID();
00066
00072         void activate();
00073
00082         void passBool(const std::string& name, bool value);
00083
00091         void passCamera(Camera camera);
00092
00100         void passLight(Light light);
00101
00110         void passInt(const std::string& name, int value);
00111
00120         void passFloat(const std::string& name, float value);
00121
00129         void passTexture(Texture texture);
00130
00136         void remove();
00137
00138     private:
00139
00151         static bool checkShader(unsigned int shader, std::string type, std::string* log_str);
00152
00163         static void readFileContents(const char* filename, std::string *file_contents);
00164
00165         Light* light;
00166         unsigned int programID;
00167
00168     };
00169
00170 }  // namespace aladdin_3d
00171
00172 #endif // !ALADDIN_3D_SHADER_H_
```

5.23 Sources/Classes/Texture/Texture.cpp File Reference

Texture class implementation file.

```
#include "Texture.h"
#include <assert.h>
#include "glew/glew.h"
#include "stb/stb_image.h"
```

5.23.1 Detailed Description

Texture class implementation file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.24 Sources/Classes/Texture/Texture.h File Reference

Texture class header file.

```
#include <string>
#include "glew/glew.h"
```

Classes

- class aladdin_3d::Texture

  Implements a texture class to handle object textures.

5.24.1 Detailed Description

Texture class header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.25 Texture.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASS_TEXTURE_H_
00017 #define ALADDIN_3D_CLASS_TEXTURE_H_
00018
00019 #include <string>
00020
00021 #include "glew/glew.h"
00022
00023 namespace aladdin_3d {
00024
00033     class Texture {
00034
00035         public:
00036
00046             Texture(const char* image, const char* type, GLuint slot);
00047
00055             GLuint getID();
00056
00064             GLuint getSlot();
00065
00073             int getWidth();
00074
00082             int getHeight();
00083
00091             int getChannels();
00092
00100             std::string getName();
00101
00107             void bind();
00108
00114             void remove();
00115
00121             void unbind();
00122
00123         private:
00124
00125             GLuint ID;
00126             GLuint slot;
00127             int texture_width = 0;
00128             int texture_height = 0;
00129             int texture_channels = 0;
00130             std::string name;
00131
00132     };
00133
00134 }
00135
00136 #endif // !ALADDIN_3D_CLASS_TEXTURE_H_
```

5.26 Sources/Classes/VAO/VAO.cpp File Reference

VAO class implementation file.

```
#include "VAO.h"
#include "glew/glew.h"
#include "Classes/VBO/VBO.h"
```

5.26.1 Detailed Description

VAO class implementation file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja García Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.27 Sources/Classes/VAO/VAO.h File Reference

VAO class header file.

```
#include "glew/glew.h"
#include "Classes/VBO/VBO.h"
```

Classes

- class aladdin_3d::VAO

  Implementation of a VAO class.

5.27.1 Detailed Description

VAO class header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja García Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.28 VAO.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASS_VAO_H_
00017 #define ALADDIN_3D_CLASS_VAO_H_
00018
00019 #include "glew/glew.h"
00020
00021 #include "Classes/VBO/VBO.h"
00022
00023 namespace aladdin_3d {
00024
00033     class VAO {
00034
00035     public:
00036
00042         VAO();
00043
00049         void bind();
00050
00063         void link_attribute(VBO& vbo, GLuint layout, GLuint num_components, GLenum type,
00064             GLsizeiptr step, void* offset);
00065
00071         void remove();
00072
00078         void unbind();
00079
00080     private:
00081
00082         GLuint ID;
00083     };
00084
00085 }  // namespace aladdin_3d
00086
00087 #endif
```

5.29 Sources/Classes/VBO/VBO.cpp File Reference

VBO class implementation file.

```
#include "VBO.h"
#include <vector>
#include "glew/glew.h"
#include "Structs/Vertex/Vertex.h"
```

5.29.1 Detailed Description

VBO class implementation file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garc♦a Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.30 Sources/Classes/VBO/VBO.h File Reference

VBO class header file.

```
#include <vector>
#include "glew/glew.h"
#include "Structs/Vertex/Vertex.h"
```

Classes

- class aladdin_3d::VBO

  Implementation of a VBO class.

5.30.1 Detailed Description

VBO class header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja García Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.31 VBO.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_CLASS_VBO_H_
00017 #define ALADDIN_3D_CLASS_VBO_H_
00018
00019 #include <vector>
00020
00021 #include "glew/glew.h"
00022
00023 #include "Structs/Vertex/Vertex.h"
00024
00025 namespace aladdin_3d {
00026
00035     class VBO {
00036
00037     public:
00038
00046         VBO(const std::vector<Vertex> &vertices);
```

```
00047
00053         void bind();
00054
00060         void remove();
00061
00067         void unbind();
00068
00069     private:
00070
00071         GLuint ID; // GL ID of the VBO.
00072
00073     };
00074
00075 }  // namespace aladdin_3d
00076
00077 #endif
```

5.32 Sources/Main.cpp File Reference

Main aladdin 3d file.

```
#include "Main.h"
#include <math.h>
#include <algorithm>
#include <chrono>
#include <iostream>
#include <random>
#include <vector>
#include "glew/glew.h"
#include "freeglut/freeglut.h"
#include <glm/gtx/string_cast.hpp>
#include "Classes/Camera/Camera.h"
#include "Classes/Light/Light.h"
#include "Classes/Object/Object.h"
#include "Classes/Shader/Shader.h"
#include "Structs/BoundingBox/BoundingBox.h"
```

Functions

- void clean ()

  Clean everything to end the program.
- void createObstacles ()

  Create the obstacles.
- void createFloor ()
- void createLives ()

  Create the place where lives appear.
- void display ()

  Display the elements.
- void displayCharacters ()

  Display the characters.
- void handleSpecialEvents (int key, int x, int y)

  Handles the Freeglut events.
- void handleKeyEvents (unsigned char key, int x, int y)

  Handles the key events.
- float initBuildings (std::vector< aladdin_3d::Object > base_objects, std::vector< int > building_guide, float x_scale)

Init the buildings.

- void initElements ()

  Init the elements of the program.

- void initEnvironment (int argc, char **argv)

  Init the environment.

- int main (int argc, char **argv)

  Main function.

5.32.1 Detailed Description

Main aladdin 3d file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garc◆a Quiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.32.2 Function Documentation

5.32.2.1 clean()

```
void clean ( )
```

Clean everything to end the program.

Clean everything to end the program.

5.32.2.2 createFloor()

```
void createFloor ( )
```

This function generates the floor.

This function generates the floor for the scenario.

5.32.2.3 createLives()

```
void createLives ( )
```

Create the place where lives appear.

Create the place where lives appear.

5.32.2.4 createObstacles()

```
void createObstacles ( )
```

Create the obstacles.

Create the obstacles.

5.32.2.5 display()

```
void display ( )
```

Display the elements.

This function will be called in the main loop to display the elements.

5.32.2.6 displayCharacters()

```
void displayCharacters ( )
```

Display the characters.

Display the characters.

5.32.2.7 handleKeyEvents()

```
void handleKeyEvents (
            unsigned char key,
            int x,
            int y )
```

Handles the key events.

Handles the freeglut key events.

5.32.2.8 handleSpecialEvents()

```
void handleSpecialEvents (
            int key,
            int x,
            int y )
```

Handles the Freeglut events.

Handles the freeglut events.

5.32.2.9 initBuildings()

```
float initBuildings (
            std::vector< aladdin_3d::Object > base_objects,
            std::vector< int > building_guide,
            float x_scale )
```

Init the buildings.

Init the buildings.

5.32.2.10 initElements()

```
void initElements ( )
```

Init the elements of the program.

Initialize the objects, elements and all.

5.32.2.11 initEnvironment()

```
void initEnvironment (
            int argc,
            char ** argv )
```

Init the environment.

Initialize the OpenGL, Glew and Freeglut environments.

5.32.2.12 main()

```
int main (
            int argc,
            char ** argv )
```

Main function.

Main function.

5.33 Sources/Main.h File Reference

Main header aladdin 3d file.

```
#include <vector>
#include <string>
#include <ctime>
#include "Classes/Camera/Camera.h"
#include "Classes/Object/Object.h"
#include "Classes/Shader/Shader.h"
```

Macros

- #define **WINDOW_WIDTH** 1000
- #define **WINDOW_HEIGHT** 800
- #define **GAME_NAME** "Aladdin 3D"

Functions

- const glm::vec4 **fog** (0.9, 0.7, 0.4, 1.0)

  This is just the gravity, in case we wanted another value.
- void clean ()

  Clean everything to end the program.
- void createObstacles ()

  Create the obstacles.
- void createLives ()

  Create the place where lives appear.
- void createFloor ()
- void display ()

  Display the elements.
- void displayCharacters ()

  Display the characters.
- void handleSpecialEvents (int key, int x, int y)

  Handles the Freeglut events.
- void handleKeyEvents (unsigned char key, int x, int y)

  Handles the key events.
- float initBuildings (std::vector< aladdin_3d::Object > base_objects, std::vector< int > building_guide, float x_scale)

  Init the buildings.
- void initElements ()

  Init the elements of the program.
- void initEnvironment (int argc, char **argv)

  Init the environment.
- int main (int argc, char **argv)

  Main function.

Variables

- std::vector< aladdin_3d::Camera > **cameras**
- unsigned int **current_camera** = 0

  Holds all the existing cameras.
- std::vector< aladdin_3d::Object > **characters**

  Current camera activated.
- std::vector< aladdin_3d::Object > **objects**

  Holds all the displayed characters.
- std::vector< unsigned int > **character_shader**

  Holds all the displayed objects.
- std::vector< unsigned int > **object_shader**

  Holds all the relationships between shaders and characters.
- std::vector< aladdin_3d::Shader > **shaders**

  Holds all the relationships between shaders and objects.

- int **window** = 0

  Holds all the initialized shanders.
- std::vector< float > **obstacles_positions**

  Window ID.
- std::vector< std::string > **obstacles_type**

  The positions of the obstacles in the game.
- double **internal_time** = 0

  The type of the obstacles in the game.
- double **time_start** = 0

  Time that will rule everything in the game.
- bool **is_paused** = true

  Time that will count as the beginning.
- float **jump_start** = -1.0f

  Control if the game is paused.
- int **lives** = 3

  The time point where the jump started.
- float **corridor_length** = 0

  Current lives of the player.
- double **last_hit** = -10

  Max length of the corridor.
- const float **velocity** = 5.0

  The moment when the character hit an obstacle the last time.
- const float **jump_velocity** = 4.0f

  The usual running speed of a person in m/s.
- const float **gravity** = -10.0f

  The initial velocity of the jump.

5.33.1 Detailed Description

Main header aladdin 3d file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.33.2 Function Documentation

5.33.2.1 clean()

```
void clean ( )
```

Clean everything to end the program.

Clean everything to end the program.

5.33.2.2 createFloor()

```
void createFloor ( )
```

This function generates the floor.

This function generates the floor for the scenario.

5.33.2.3 createLives()

```
void createLives ( )
```

Create the place where lives appear.

Create the place where lives appear.

5.33.2.4 createObstacles()

```
void createObstacles ( )
```

Create the obstacles.

Create the obstacles.

5.33.2.5 display()

```
void display ( )
```

Display the elements.

This function will be called in the main loop to display the elements.

5.33.2.6 displayCharacters()

```
void displayCharacters ( )
```

Display the characters.

Display the characters.

5.33.2.7 handleKeyEvents()

```
void handleKeyEvents (
            unsigned char key,
            int x,
            int y )
```

Handles the key events.

Handles the freeglut key events.

5.33.2.8 handleSpecialEvents()

```
void handleSpecialEvents (
            int key,
            int x,
            int y )
```

Handles the Freeglut events.

Handles the freeglut events.

5.33.2.9 initBuildings()

```
float initBuildings (
            std::vector< aladdin_3d::Object > base_objects,
            std::vector< int > building_guide,
            float x_scale )
```

Init the buildings.

Init the buildings.

5.33.2.10 initElements()

```
void initElements ( )
```

Init the elements of the program.

Initialize the objects, elements and all.

5.33.2.11 initEnvironment()

```
void initEnvironment (
            int argc,
            char ** argv )
```

Init the environment.

Initialize the OpenGL, Glew and Freeglut environments.

5.33.2.12 main()

```
int main (
            int argc,
            char ** argv )
```

Main function.

Main function.

5.34 Main.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_MAIN_H_
00017 #define ALADDIN_3D_MAIN_H_
00018
00019 #define WINDOW_WIDTH 1000
00020 #define WINDOW_HEIGHT 800
00021 #define GAME_NAME "Aladdin 3D"
00022
00023 #include <vector>
00024 #include <string>
00025 #include <ctime>
00026
00027 #include "Classes/Camera/Camera.h"
00028 #include "Classes/Object/Object.h"
00029 #include "Classes/Shader/Shader.h"
00030
00031 std::vector<aladdin_3d::Camera> cameras;
00032 unsigned int current_camera = 0;
00033 std::vector<aladdin_3d::Object> characters;
00034 std::vector<aladdin_3d::Object> objects;
00035 std::vector<unsigned int> character_shader;
00036 std::vector<unsigned int> object_shader;
00037 std::vector<aladdin_3d::Shader> shaders;
00038 int window = 0;
00039 std::vector<float> obstacles_positions;
00040 std::vector<std::string> obstacles_type;
00041 double internal_time = 0;
00042 double time_start = 0;
00043 bool is_paused = true;
00044 float jump_start = -1.0f;
00045 int lives = 3;
00046 float corridor_length = 0;
00047 double last_hit = -10;
00048
00049 const float velocity = 5.0;
00050 const float jump_velocity = 4.0f;
00051 const float gravity = -10.0f;
00052 const glm::vec4 fog(0.9, 0.7, 0.4, 1.0); // This is just the fog color.
00053
00059 void clean();
00060
00066 void createObstacles();
00067
00073 void createLives();
00074
00080 void createFloor();
```

```
00081
00087 void display();
00088
00094 void displayCharacters();
00095
00101 void handleSpecialEvents(int key, int x, int y);
00102
00108 void handleKeyEvents(unsigned char key, int x, int y);
00109
00115 float initBuildings(std::vector<aladdin_3d::Object> base_objects, std::vector<int> building_guide,
      float x_scale);
00116
00122 void initElements();
00123
00129 void initEnvironment(int argc, char** argv);
00130
00136 int main(int argc, char** argv);
00137
00138 #endif
```

5.35 Sources/Structs/BoundingBox/BoundingBox.h File Reference

BoundingBox struct header file.

```
#include "glm/glm.hpp"
```

Classes

- struct aladdin_3d::BoundingBox

  A bounding box struct.

5.35.1 Detailed Description

BoundingBox struct header file.

Version

1.0.0 (2023-01-02)

Date

2023-01-02

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.36 BoundingBox.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_STRUCT_BOUNDINGBOX_H_
00017 #define ALADDIN_3D_STRUCT_BOUNDINGBOX_H_
00018
00019 #include "glm/glm.hpp"
00020
00021 namespace aladdin_3d {
00022
00028     struct BoundingBox {
00029
00030         glm::vec3 min;  // Minimum vertex.
00031         glm::vec3 max; // Maximum vertex.
00032
00033     };
00034
00035 } // namespace aladdin_3d
00036
00037 #endif
```

5.37 Sources/Structs/Vertex/Vertex.h File Reference

Vertex struct header file.

```
#include "glm/glm.hpp"
```

Classes

- struct aladdin_3d::Vertex

  A geometry vertex.

5.37.1 Detailed Description

Vertex struct header file.

Version

1.0.0 (2022-10-21)

Date

2022-10-21

Author

Borja Garcuiroga garcaqub@tcd.ie

The information and material provided below was developed as partial requirements for the MSc in Computer Science at Trinity College Dublin, Ireland.

5.38 Vertex.h

Go to the documentation of this file.

```
00001
00016 #ifndef ALADDIN_3D_STRUCT_VERTEX_H_
00017 #define ALADDIN_3D_STRUCT_VERTEX_H_
00018
00019 #include "glm/glm.hpp"
00020
00021 namespace aladdin_3d {
00022
00028     struct Vertex {
00029
00030         glm::vec3 position;
00031         glm::vec3 normal;
00032         glm::vec3 color;
00033         glm::vec2 uv;
00034
00035     };
00036
00037 } // namespace aladdin_3d
00038
00039 #endif
```