Vender

Generated by Doxygen 1.10.0

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:

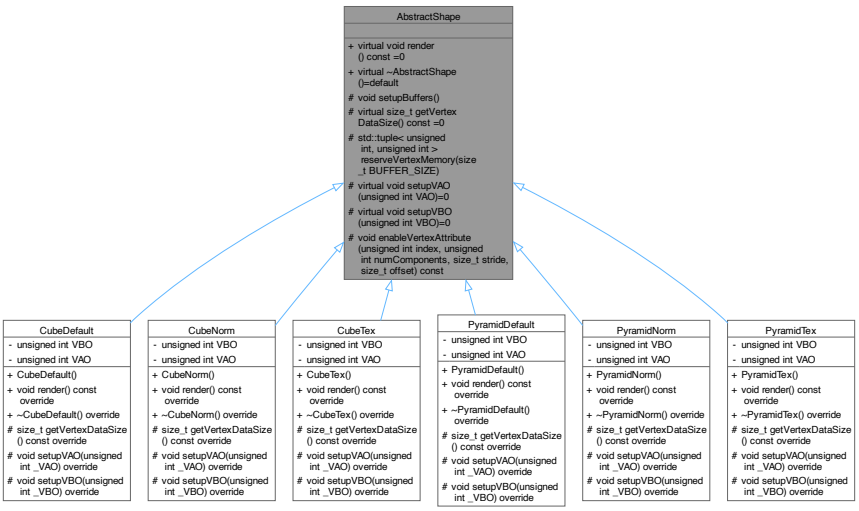
Chapter 4

Class Documentation

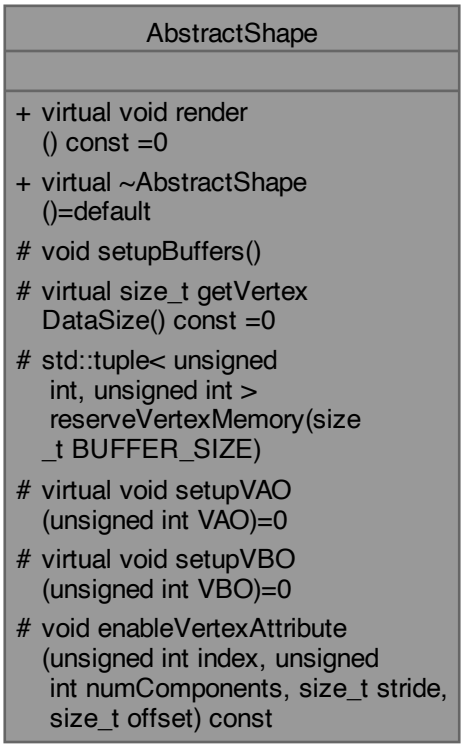
4.1 AbstractShape Class Reference

`#include <shape.h>`

Inheritance diagram for AbstractShape:



Collaboration diagram for AbstractShape:



Public Member Functions

- virtual void render () const =0
- virtual ~AbstractShape ()=default

Protected Member Functions

- void setupBuffers ()
- virtual size_t getVertexDataSize () const =0
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- virtual void setupVAO (unsigned int VAO)=0
- virtual void setupVBO (unsigned int VBO)=0
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

4.1.1 Detailed Description

Definition at line 4 of file shape.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ~AbstractShape()

```
virtual AbstractShape::~AbstractShape ( )  [virtual], [default]
```

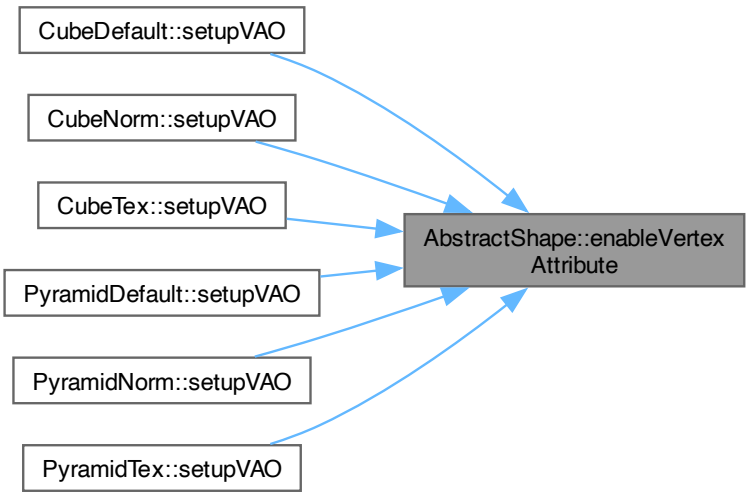
4.1.3 Member Function Documentation

4.1.3.1 enableVertexAttribute()

```
void AbstractShape::enableVertexAttribute (
            unsigned int index,
            unsigned int numComponents,
            size_t stride,
            size_t offset ) const  [inline], [protected]
```

Definition at line 35 of file shape.h.

Here is the caller graph for this function:

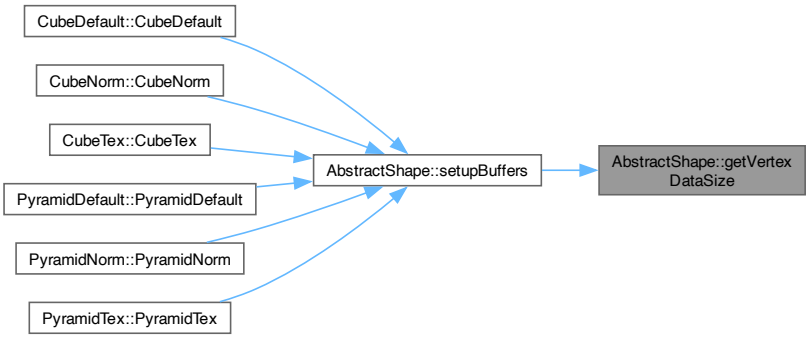


4.1.3.2 getVertexDataSize()

```
virtual size_t AbstractShape::getVertexDataSize ( ) const  [protected], [pure virtual]
```

Implemented in CubeDefault, CubeNorm, CubeTex, PyramidDefault, PyramidNorm, and PyramidTex.

Here is the caller graph for this function:



4.1.3.3 render()

```
virtual void AbstractShape::render ( ) const  [pure virtual]
```

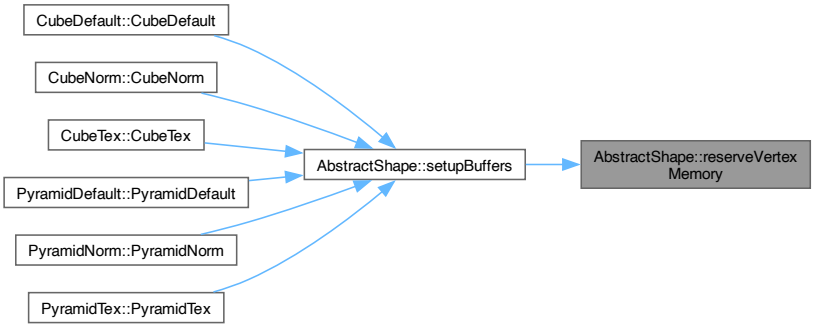
Implemented in CubeDefault, CubeNorm, CubeTex, PyramidDefault, PyramidNorm, and PyramidTex.

4.1.3.4 reserveVertexMemory()

```
std::tuple< unsigned int, unsigned int > AbstractShape::reserveVertexMemory (
            size_t BUFFER_SIZE )  [inline], [protected]
```

Definition at line 21 of file shape.h.

Here is the caller graph for this function:

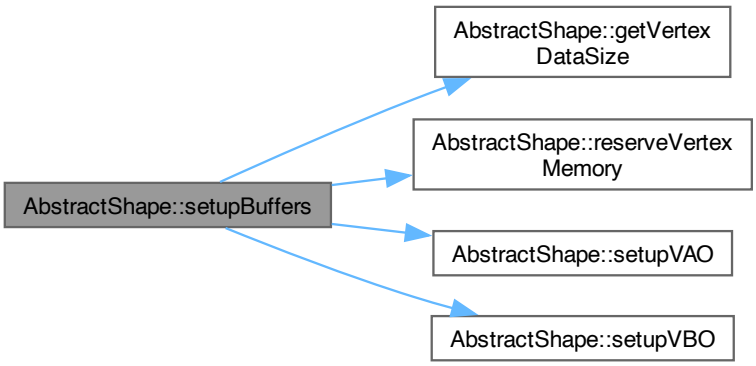


4.1.3.5 setupBuffers()

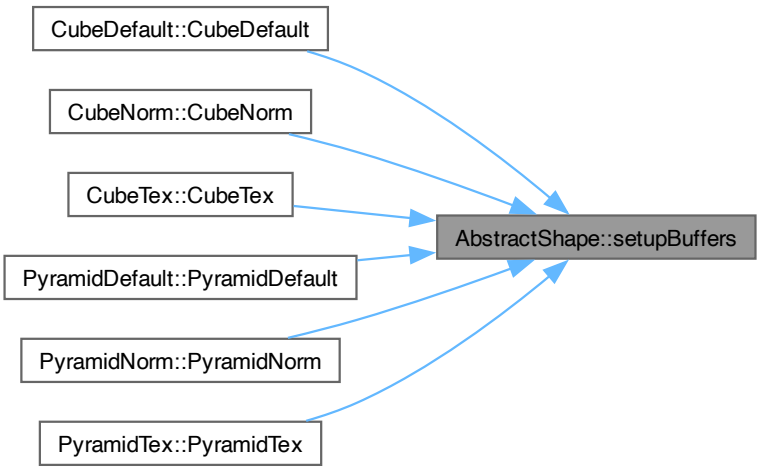
```
void AbstractShape::setupBuffers ( )  [inline], [protected]
```

Definition at line 11 of file shape.h.

Here is the call graph for this function:



Here is the caller graph for this function:

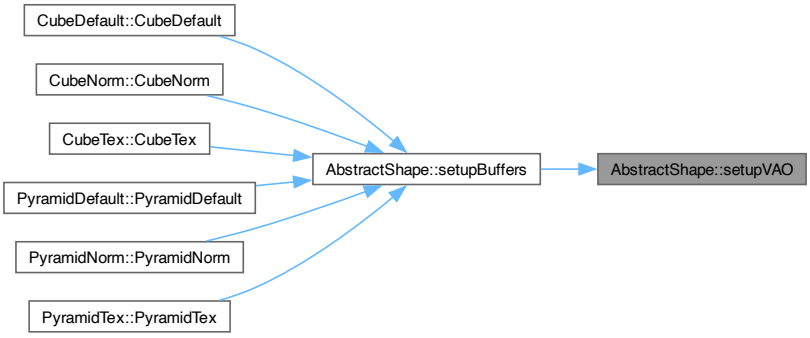


4.1.3.6 setupVAO()

```
virtual void AbstractShape::setupVAO (
            unsigned int VAO )  [protected], [pure virtual]
```

Implemented in CubeDefault, CubeNorm, CubeTex, PyramidDefault, PyramidNorm, and PyramidTex.

Here is the caller graph for this function:

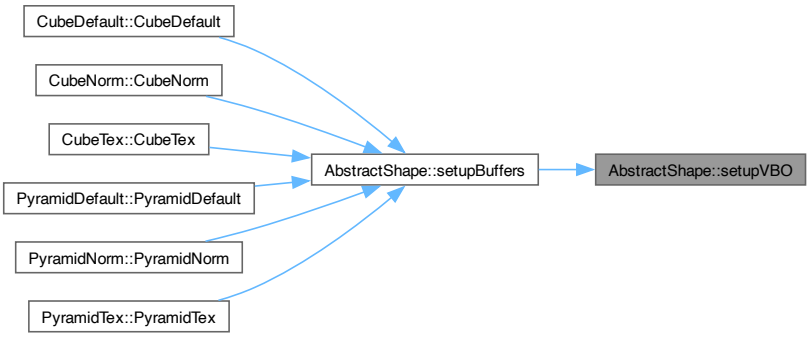


4.1.3.7 setupVBO()

```
virtual void AbstractShape::setupVBO (
            unsigned int VBO )  [protected], [pure virtual]
```

Implemented in CubeDefault, CubeNorm, CubeTex, PyramidDefault, PyramidNorm, and PyramidTex.

Here is the caller graph for this function:



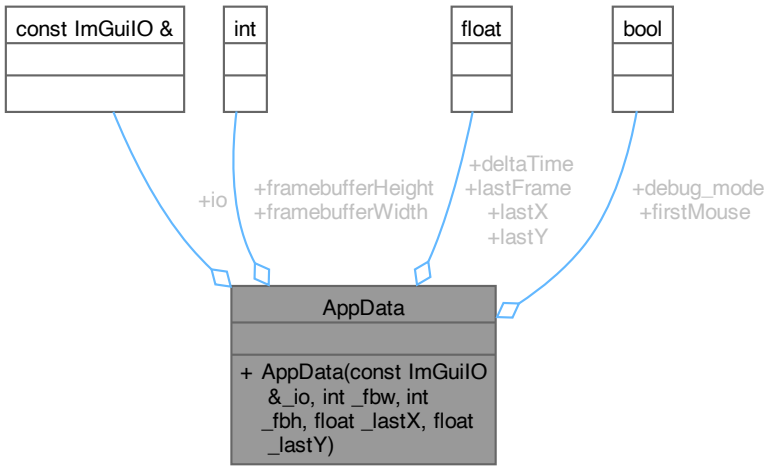
The documentation for this class was generated from the following file:

- vender/render/models/objects/shape.h

4.2 AppData Struct Reference

`#include <appdata.h>`

Collaboration diagram for AppData:



Public Member Functions

- AppData (const ImGuiIO &_io, int _fbw, int _fbh, float _lastX, float _lastY)

Public Attributes

- const ImGuiIO & io
- int framebufferWidth
- int framebufferHeight
- float lastX
- float lastY
- float deltaTime = 0.0f
- float lastFrame = 0.0f
- bool firstMouse = true
- bool debug_mode = false

4.2.1 Detailed Description

Definition at line 7 of file appdata.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AppData()

```
AppData::AppData (
            const ImGuiIO & _io,
            int _fbw,
            int _fbh,
            float _lastX,
            float _lastY )  [inline]
```

Definition at line 18 of file appdata.h.

4.2.3 Member Data Documentation

4.2.3.1 debug_mode

```
bool AppData::debug_mode = false
```

Definition at line 17 of file appdata.h.

4.2.3.2 deltaTime

```
float AppData::deltaTime = 0.0f
```

Definition at line 14 of file appdata.h.

4.2.3.3 firstMouse

```
bool AppData::firstMouse = true
```

Definition at line 16 of file appdata.h.

4.2.3.4 framebufferHeight

```
int AppData::framebufferHeight
```

Definition at line 11 of file appdata.h.

4.2.3.5 framebufferWidth

```
int AppData::framebufferWidth
```

Definition at line 10 of file appdata.h.

4.2.3.6 io

```
const ImGuiIO& AppData::io
```

Definition at line 9 of file appdata.h.

4.2.3.7 lastFrame

```
float AppData::lastFrame = 0.0f
```

Definition at line 15 of file appdata.h.

4.2.3.8 lastX

```
float AppData::lastX
```

Definition at line 12 of file appdata.h.

4.2.3.9 lastY

```
float AppData::lastY
```

Definition at line 13 of file appdata.h.

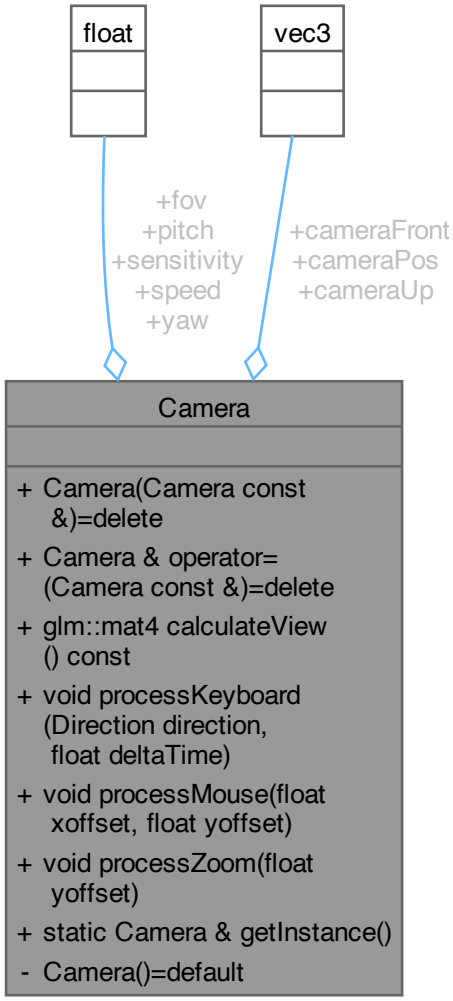
The documentation for this struct was generated from the following file:

- vender/render/appdata/appdata.h

4.3 Camera Class Reference

```
#include <camera.h>
```

Collaboration diagram for Camera:



Public Member Functions

- Camera (Camera const &)=delete
- Camera & operator= (Camera const &)=delete
- glm::mat4 calculateView () const
- void processKeyboard (Direction direction, float deltaTime)
- void processMouse (float xoffset, float yoffset)
- void processZoom (float yoffset)

Static Public Member Functions

- static Camera & getInstance ()

Public Attributes

- float yaw = -90.0f
- float pitch = 0.0f
- float fov = 45.0f
- float speed = 2.5f
- float sensitivity = 0.1f
- glm::vec3 cameraPos = glm::vec3(0.0f, 0.0f, 3.0f)
- glm::vec3 cameraFront = glm::vec3(0.0f, 0.0f, -1.0f)
- glm::vec3 cameraUp = glm::vec3(0.0f, 1.0f, 0.0f)

Private Member Functions

- Camera ()=default

4.3.1 Detailed Description

Definition at line 14 of file camera.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Camera() [1/2]

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

4.3.2.2 Camera() [2/2]

```
Camera::Camera ( )  [private], [default]
```

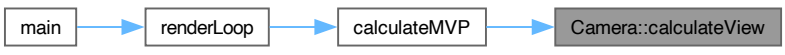
4.3.3 Member Function Documentation

4.3.3.1 calculateView()

```
glm::mat4 Camera::calculateView ( ) const
```

Definition at line 3 of file camera.cpp.

Here is the caller graph for this function:

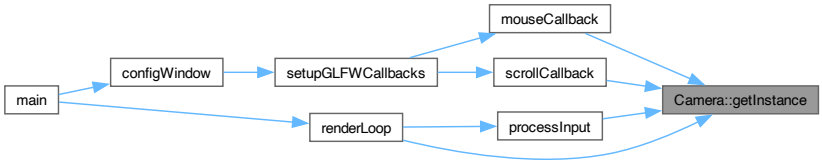


4.3.3.2 getInstance()

```
static Camera & Camera::getInstance ( )  [inline], [static]
```

Definition at line 27 of file camera.h.

Here is the caller graph for this function:



4.3.3.3 operator=()

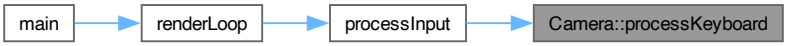
```
Camera & Camera::operator= (
            Camera const &  )  [delete]
```

4.3.3.4 processKeyboard()

```
void Camera::processKeyboard (
            Direction direction,
            float deltaTime )
```

Definition at line 8 of file camera.cpp.

Here is the caller graph for this function:



4.3.3.5 processMouse()

```
void Camera::processMouse (
            float xoffset,
            float yoffset )
```

Definition at line 22 of file camera.cpp.

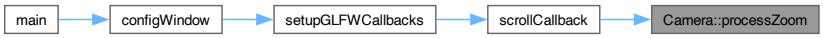
Here is the caller graph for this function:

4.3.3.6 processZoom()

```
void Camera::processZoom (
            float yoffset )
```

Definition at line 43 of file camera.cpp.

Here is the caller graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 cameraFront

```
glm::vec3 Camera::cameraFront = glm::vec3(0.0f, 0.0f, -1.0f)
```

Definition at line 24 of file camera.h.

4.3.4.2 cameraPos

```
glm::vec3 Camera::cameraPos = glm::vec3(0.0f, 0.0f, 3.0f)
```

Definition at line 23 of file camera.h.

4.3.4.3 cameraUp

```
glm::vec3 Camera::cameraUp = glm::vec3(0.0f, 1.0f, 0.0f)
```

Definition at line 25 of file camera.h.

4.3.4.4 fov

```
float Camera::fov = 45.0f
```

Definition at line 19 of file camera.h.

4.3.4.5 pitch

```
float Camera::pitch = 0.0f
```

Definition at line 18 of file camera.h.

4.3.4.6 sensitivity

```
float Camera::sensitivity = 0.1f
```

Definition at line 21 of file camera.h.

4.3.4.7 speed

```
float Camera::speed = 2.5f
```

Definition at line 20 of file camera.h.

4.3.4.8 yaw

```
float Camera::yaw = -90.0f
```

Definition at line 17 of file camera.h.

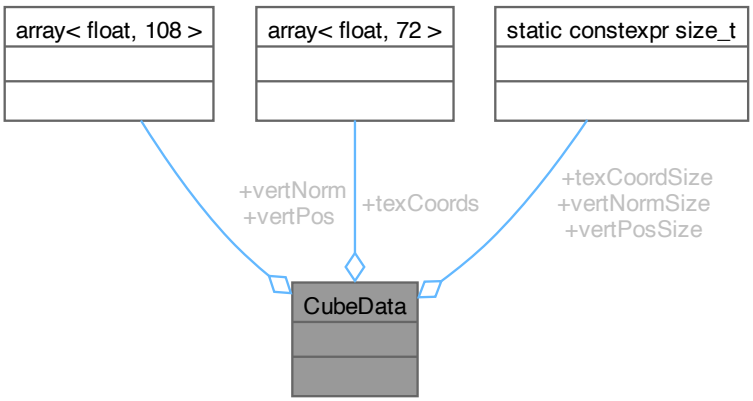
The documentation for this class was generated from the following files:

- vender/render/camera/camera.h
- vender/render/camera/camera.cpp

4.4 CubeData Struct Reference

```
#include <cube_data.h>
```

Collaboration diagram for CubeData:



Static Public Attributes

- static constexpr std::array< float, 108 > vertPos
- static constexpr std::array< float, 108 > vertNorm
- static constexpr std::array< float, 72 > texCoords
- static constexpr size_t vertPosSize = sizeof(vertPos)
- static constexpr size_t vertNormSize = sizeof(vertNorm)
- static constexpr size_t texCoordSize = sizeof(texCoords)

4.4.1 Detailed Description

Definition at line 5 of file cube_data.h.

4.4.2 Member Data Documentation

4.4.2.1 texCoords

```
constexpr std::array<float, 72> CubeData::texCoords  [static], [constexpr]
```

Definition at line 93 of file cube_data.h.

4.4.2.2 texCoordSize

```
constexpr size_t CubeData::texCoordSize = sizeof(texCoords)  [static], [constexpr]
```

Definition at line 138 of file cube_data.h.

4.4.2.3 vertNorm

```
constexpr std::array<float, 108> CubeData::vertNorm  [static], [constexpr]
```

Definition at line 50 of file cube_data.h.

4.4.2.4 vertNormSize

```
constexpr size_t CubeData::vertNormSize = sizeof(vertNorm)  [static], [constexpr]
```

Definition at line 137 of file cube_data.h.

4.4.2.5 vertPos

```
constexpr std::array<float, 108> CubeData::vertPos  [static], [constexpr]
```

Definition at line 7 of file cube_data.h.

4.4.2.6 vertPosSize

```
constexpr size_t CubeData::vertPosSize = sizeof(vertPos)  [static], [constexpr]
```

Definition at line 136 of file cube_data.h.

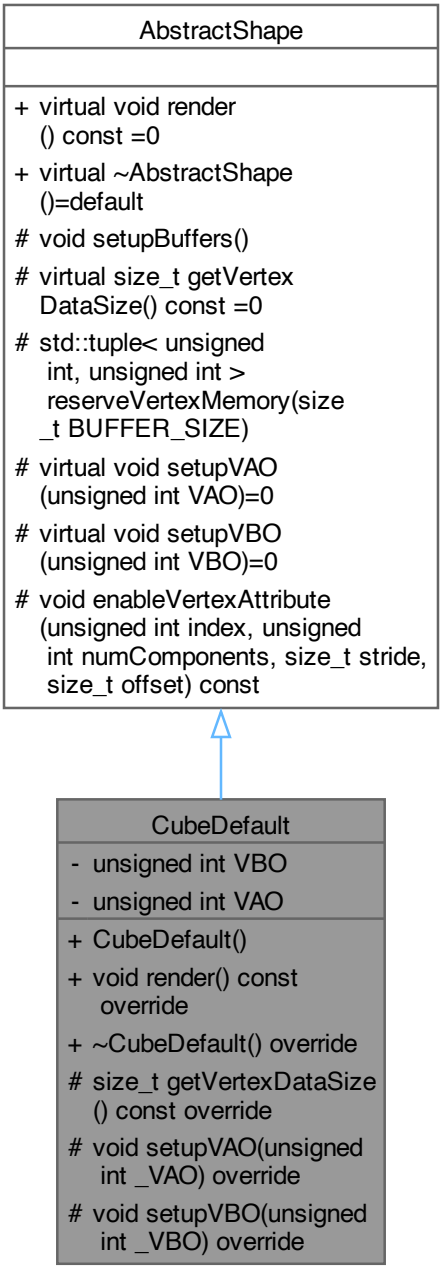
The documentation for this struct was generated from the following file:

- vender/render/models/objects/cube/cube_data.h

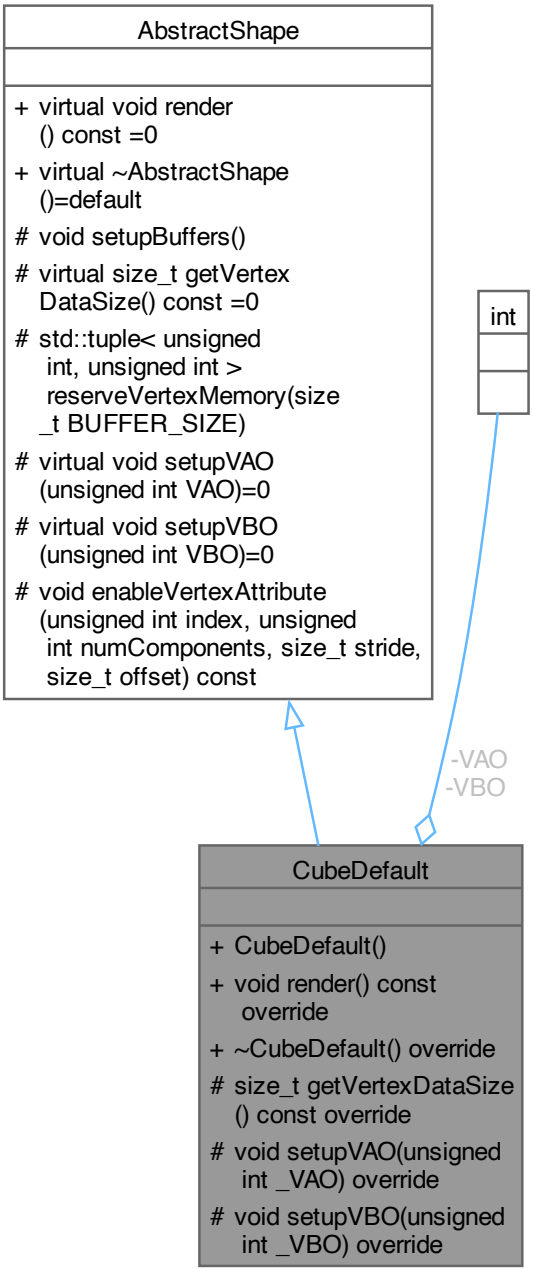
4.5 CubeDefault Class Reference

```
#include <cube.h>
```

Inheritance diagram for CubeDefault:



Collaboration diagram for CubeDefault:



Public Member Functions

- CubeDefault ()
- void render () const override
- ~CubeDefault () override

Public Member Functions inherited from AbstractShape

- virtual ~AbstractShape ()=default

Protected Member Functions

- size_t getVertexDataSize () const override
- void setupVAO (unsigned int _VAO) override
- void setupVBO (unsigned int _VBO) override

Protected Member Functions inherited from AbstractShape

- void setupBuffers ()
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

Private Attributes

- unsigned int VBO
- unsigned int VAO

4.5.1 Detailed Description

Definition at line 9 of file cube.h.

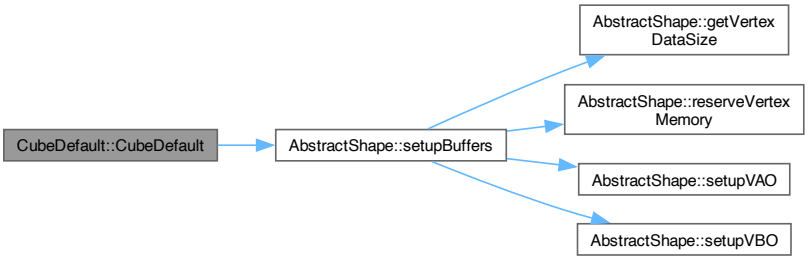
4.5.2 Constructor & Destructor Documentation

4.5.2.1 CubeDefault()

```
CubeDefault::CubeDefault ( )  [inline]
```

Definition at line 12 of file cube.h.

Here is the call graph for this function:



4.5.2.2 ~CubeDefault()

```
CubeDefault::~CubeDefault ( )  [inline], [override]
```

Definition at line 22 of file cube.h.

4.5.3 Member Function Documentation

4.5.3.1 getVertexDataSize()

```
size_t CubeDefault::getVertexDataSize ( ) const  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 29 of file cube.h.

4.5.3.2 render()

```
void CubeDefault::render ( ) const  [inline], [override], [virtual]
```

Implements AbstractShape.

Definition at line 17 of file cube.h.

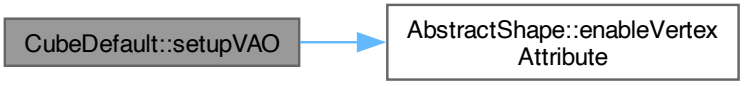
4.5.3.3 setupVAO()

```
void CubeDefault::setupVAO (
            unsigned int _VAO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 34 of file cube.h.

Here is the call graph for this function:



4.5.3.4 setupVBO()

```
void CubeDefault::setupVBO (
            unsigned int _VBO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 41 of file cube.h.

4.5.4 Member Data Documentation

4.5.4.1 VAO

```
unsigned int CubeDefault::VAO  [private]
```

Definition at line 49 of file cube.h.

4.5.4.2 VBO

```
unsigned int CubeDefault::VBO  [private]
```

Definition at line 48 of file cube.h.

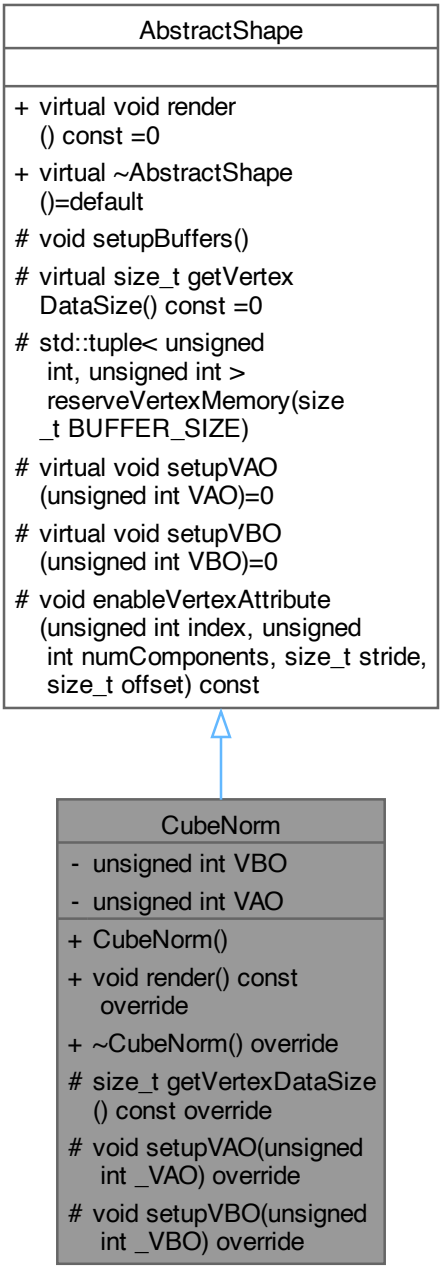
The documentation for this class was generated from the following file:

- vender/render/models/objects/cube/cube.h

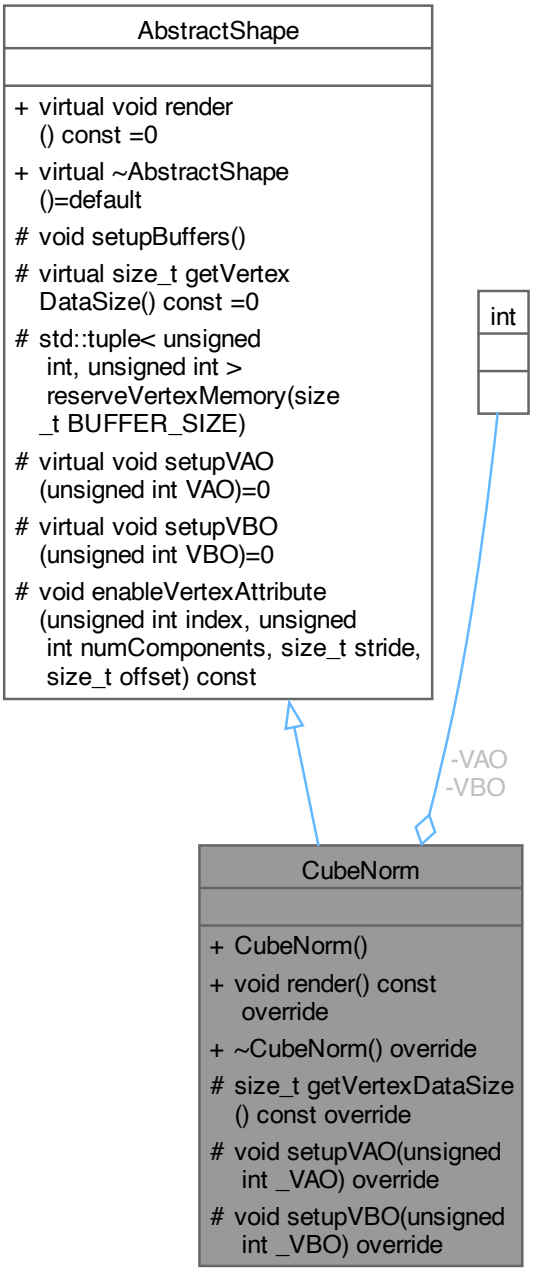
4.6 CubeNorm Class Reference

```
#include <cube.h>
```

Inheritance diagram for CubeNorm:



Collaboration diagram for CubeNorm:



Public Member Functions

- CubeNorm ()
- void render () const override
- ~CubeNorm () override

Public Member Functions inherited from AbstractShape

- virtual ~AbstractShape ()=default

Protected Member Functions

- size_t getVertexDataSize () const override
- void setupVAO (unsigned int _VAO) override
- void setupVBO (unsigned int _VBO) override

Protected Member Functions inherited from AbstractShape

- void setupBuffers ()
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

Private Attributes

- unsigned int VBO
- unsigned int VAO

4.6.1 Detailed Description

Definition at line 52 of file cube.h.

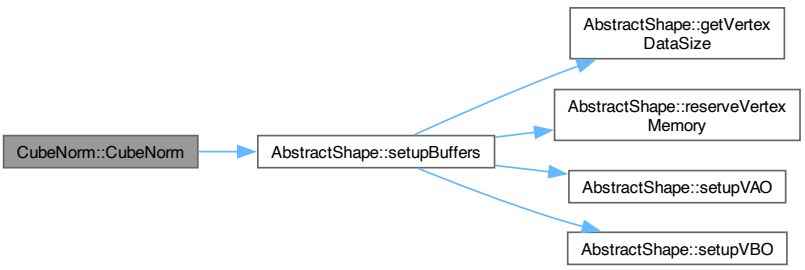
4.6.2 Constructor & Destructor Documentation

4.6.2.1 CubeNorm()

```
CubeNorm::CubeNorm ( )  [inline]
```

Definition at line 55 of file cube.h.

Here is the call graph for this function:



4.6.2.2 ~CubeNorm()

```
CubeNorm::~CubeNorm ( )  [inline], [override]
```

Definition at line 66 of file cube.h.

4.6.3 Member Function Documentation

4.6.3.1 getVertexDataSize()

```
size_t CubeNorm::getVertexDataSize ( ) const  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 73 of file cube.h.

4.6.3.2 render()

```
void CubeNorm::render ( ) const  [inline], [override], [virtual]
```

Implements AbstractShape.

Definition at line 60 of file cube.h.

4.6.3.3 setupVAO()

```
void CubeNorm::setupVAO (
            unsigned int _VAO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 78 of file cube.h.

Here is the call graph for this function:

4.6.3.4 setupVBO()

```
void CubeNorm::setupVBO (
            unsigned int _VBO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 86 of file cube.h.

4.6.4 Member Data Documentation

4.6.4.1 VAO

```
unsigned int CubeNorm::VAO  [private]
```

Definition at line 95 of file cube.h.

4.6.4.2 VBO

```
unsigned int CubeNorm::VBO  [private]
```

Definition at line 94 of file cube.h.

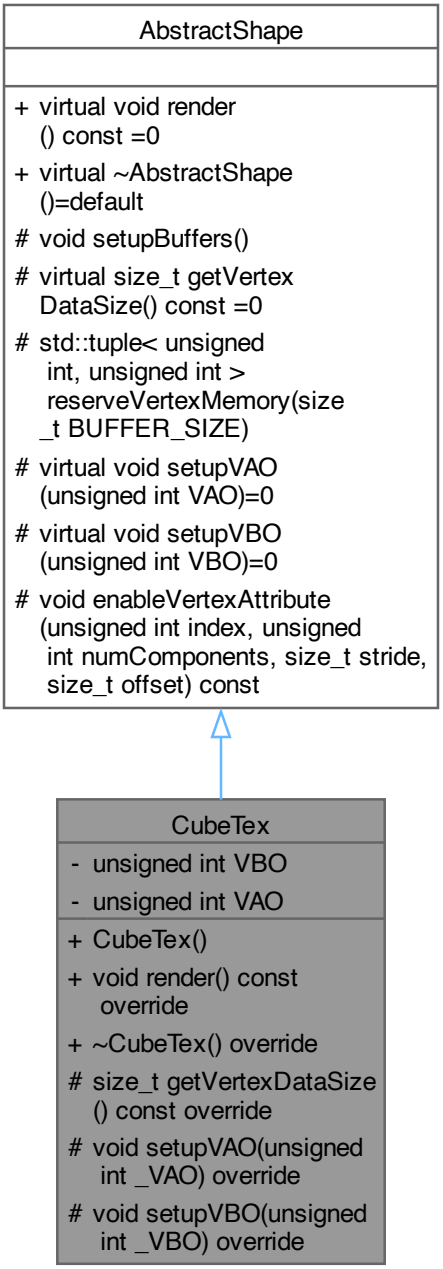
The documentation for this class was generated from the following file:

- vender/render/models/objects/cube/cube.h

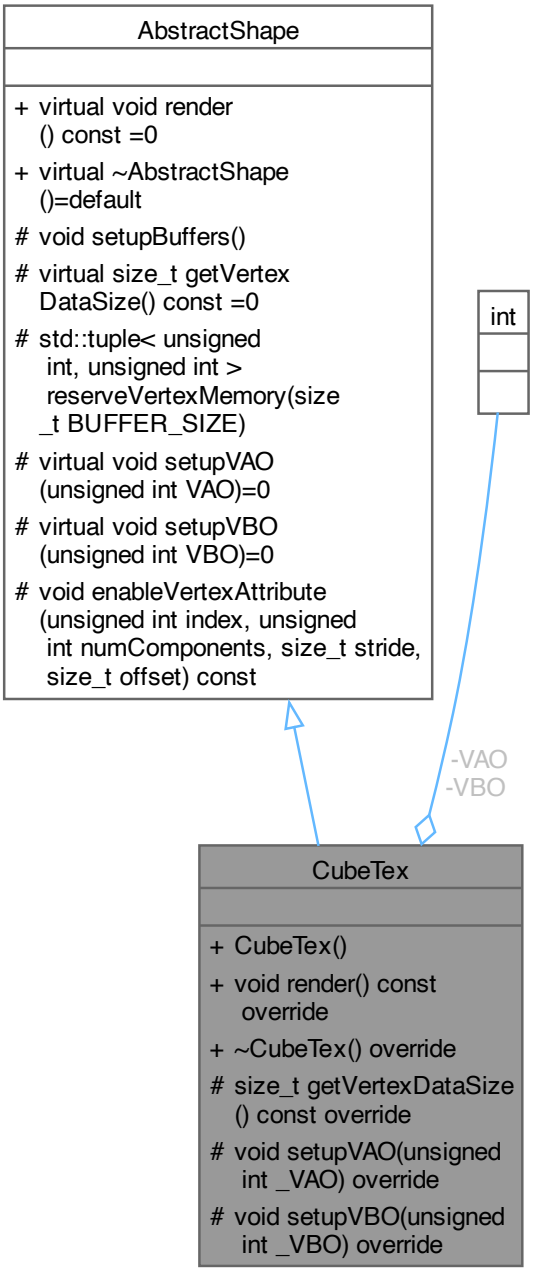
4.7 CubeTex Class Reference

```
#include <cube.h>
```

Inheritance diagram for CubeTex:



Collaboration diagram for CubeTex:



Public Member Functions

- CubeTex ()
- void render () const override
- ~CubeTex () override

Public Member Functions inherited from AbstractShape

- virtual ~AbstractShape ()=default

Protected Member Functions

- size_t getVertexDataSize () const override
- void setupVAO (unsigned int _VAO) override
- void setupVBO (unsigned int _VBO) override

Protected Member Functions inherited from AbstractShape

- void setupBuffers ()
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

Private Attributes

- unsigned int VBO
- unsigned int VAO

4.7.1 Detailed Description

Definition at line 98 of file cube.h.

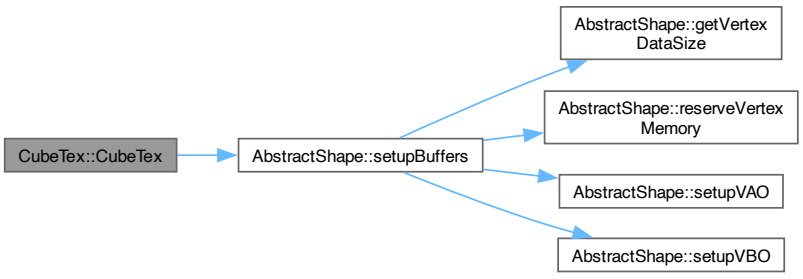
4.7.2 Constructor & Destructor Documentation

4.7.2.1 CubeTex()

```
CubeTex::CubeTex ( )  [inline]
```

Definition at line 101 of file cube.h.

Here is the call graph for this function:



4.7.2.2 ~CubeTex()

```
CubeTex::~CubeTex ( )  [inline], [override]
```

Definition at line 112 of file cube.h.

4.7.3 Member Function Documentation

4.7.3.1 getVertexDataSize()

```
size_t CubeTex::getVertexDataSize ( ) const  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 119 of file cube.h.

4.7.3.2 render()

```
void CubeTex::render ( ) const  [inline], [override], [virtual]
```

Implements AbstractShape.

Definition at line 106 of file cube.h.

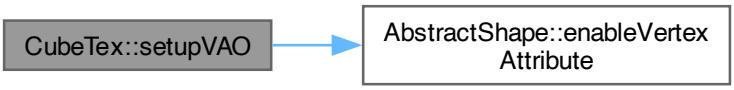
4.7.3.3 setupVAO()

```
void CubeTex::setupVAO (
            unsigned int _VAO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 124 of file cube.h.

Here is the call graph for this function:



4.7.3.4 setupVBO()

```
void CubeTex::setupVBO (
            unsigned int _VBO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 133 of file cube.h.

4.7.4 Member Data Documentation

4.7.4.1 VAO

```
unsigned int CubeTex::VAO  [private]
```

Definition at line 143 of file cube.h.

4.7.4.2 VBO

```
unsigned int CubeTex::VBO  [private]
```

Definition at line 142 of file cube.h.

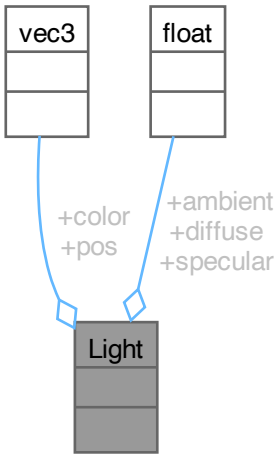
The documentation for this class was generated from the following file:

- vender/render/models/objects/cube/cube.h

4.8 Light Struct Reference

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

Collaboration diagram for Light:



Public Attributes

- glm::vec3 pos = glm::vec3(1.0f, 0.17f, 1.6f)
- glm::vec3 color = glm::vec3(1.0f, 1.0f, 1.0f)
- float ambient = 0.2f
- float diffuse = 0.5f
- float specular = 1.0f

4.8.1 Detailed Description

Definition at line 5 of file light.h.

4.8.2 Member Data Documentation

4.8.2.1 ambient

```
float Light::ambient = 0.2f
```

Definition at line 9 of file light.h.

4.8.2.2 color

```
glm::vec3 Light::color = glm::vec3(1.0f, 1.0f, 1.0f)
```

Definition at line 8 of file light.h.

4.8.2.3 diffuse

```
float Light::diffuse = 0.5f
```

Definition at line 10 of file light.h.

4.8.2.4 pos

```
glm::vec3 Light::pos = glm::vec3(1.0f, 0.17f, 1.6f)
```

Definition at line 7 of file light.h.

4.8.2.5 specular

```
float Light::specular = 1.0f
```

Definition at line 11 of file light.h.

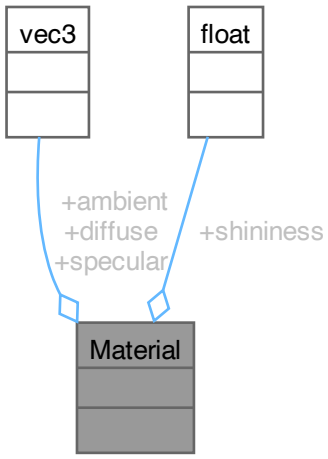
The documentation for this struct was generated from the following file:

- vender/render/models/lighting/light.h

4.9 Material Struct Reference

`#include <material.h>`

Collaboration diagram for Material:



Public Attributes

- glm::vec3 ambient = glm::vec3(1.0f, 0.5f, 0.5f)
- glm::vec3 diffuse = glm::vec3(1.0f, 0.5f, 0.5f)
- glm::vec3 specular = glm::vec3(0.5f, 0.5f, 0.5f)
- float shininess = 64.0f

4.9.1 Detailed Description

Definition at line 5 of file material.h.

4.9.2 Member Data Documentation

4.9.2.1 ambient

```
glm::vec3 Material::ambient = glm::vec3(1.0f, 0.5f, 0.5f)
```

Definition at line 7 of file material.h.

4.9.2.2 diffuse

```
glm::vec3 Material::diffuse = glm::vec3(1.0f, 0.5f, 0.5f)
```

Definition at line 8 of file material.h.

4.9.2.3 shininess

```
float Material::shininess = 64.0f
```

Definition at line 10 of file material.h.

4.9.2.4 specular

```
glm::vec3 Material::specular = glm::vec3(0.5f, 0.5f, 0.5f)
```

Definition at line 9 of file material.h.

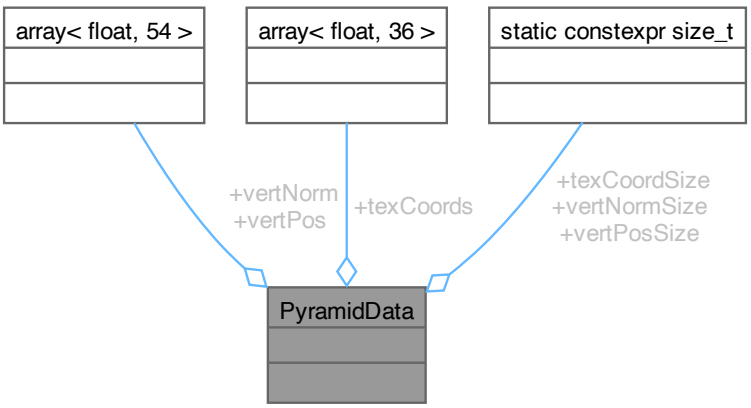
The documentation for this struct was generated from the following file:

- vender/assets/material.h

4.10 PyramidData Struct Reference

```
#include <pyramid_data.h>
```

Collaboration diagram for PyramidData:



Static Public Attributes

- static constexpr std::array< float, 54 > vertPos
- static constexpr std::array< float, 54 > vertNorm
- static constexpr std::array< float, 36 > texCoords
- static constexpr size_t vertPosSize = sizeof(vertPos)
- static constexpr size_t vertNormSize = sizeof(vertNorm)
- static constexpr size_t texCoordSize = sizeof(texCoords)

4.10.1 Detailed Description

Definition at line 5 of file pyramid_data.h.

4.10.2 Member Data Documentation

4.10.2.1 texCoords

```
constexpr std::array<float, 36> PyramidData::texCoords  [static], [constexpr]
```

Initial value:

```
= {
        0.5f, 1.0f,
        0.0f, 0.0f,
        1.0f, 0.0f,

        0.5f, 1.0f,
        0.0f, 0.0f,
        1.0f, 0.0f,

        0.5f, 1.0f,
        0.0f, 0.0f,
        1.0f, 0.0f,

        0.5f, 1.0f,
        0.0f, 0.0f,
        1.0f, 0.0f,

        0.0f, 1.0f,
        1.0f, 1.0f,
        1.0f, 0.0f,
        1.0f, 0.0f,
        0.0f, 0.0f,
        0.0f, 1.0f}
```

Definition at line 57 of file pyramid_data.h.

4.10.2.2 texCoordSize

```
constexpr size_t PyramidData::texCoordSize = sizeof(texCoords)  [static], [constexpr]
```

Definition at line 83 of file pyramid_data.h.

4.10.2.3 vertNorm

```
constexpr std::array<float, 54> PyramidData::vertNorm  [static], [constexpr]
```

Initial value:

```
= {
        0.0f, 0.71f, 0.71f,
        0.0f, 0.71f, 0.71f,
        0.0f, 0.71f, 0.71f,

        0.71f, 0.71f, 0.0f,
        0.71f, 0.71f, 0.0f,
        0.71f, 0.71f, 0.0f,

        0.0f, 0.71f, -0.71f,
        0.0f, 0.71f, -0.71f,
        0.0f, 0.71f, -0.71f,

        -0.71f, 0.71f, 0.0f,
        -0.71f, 0.71f, 0.0f,
        -0.71f, 0.71f, 0.0f,

        0.0f, -1.0f, 0.0f,
        0.0f, -1.0f, 0.0f,
        0.0f, -1.0f, 0.0f,

        0.0f, -1.0f, 0.0f,
        0.0f, -1.0f, 0.0f,
        0.0f, -1.0f, 0.0f}
```

Definition at line 32 of file pyramid_data.h.

4.10.2.4 vertNormSize

```
constexpr size_t PyramidData::vertNormSize = sizeof(vertNorm)  [static], [constexpr]
```

Definition at line 82 of file pyramid_data.h.

4.10.2.5 vertPos

```
constexpr std::array<float, 54> PyramidData::vertPos  [static], [constexpr]
```

Initial value:

```
= {
        0.0f, 0.5f, 0.0f,
        -0.5f, -0.5f, 0.5f,
        0.5f, -0.5f, 0.5f,

        0.0f, 0.5f, 0.0f,
        0.5f, -0.5f, 0.5f,
        0.5f, -0.5f, -0.5f,

        0.0f, 0.5f, 0.0f,
        0.5f, -0.5f, -0.5f,
        -0.5f, -0.5f, -0.5f,

        0.0f, 0.5f, 0.0f,
        -0.5f, -0.5f, -0.5f,
        -0.5f, -0.5f, 0.5f,

        -0.5f, -0.5f, -0.5f,
        0.5f, -0.5f, -0.5f,
        0.5f, -0.5f, 0.5f,

        0.5f, -0.5f, 0.5f,
        -0.5f, -0.5f, 0.5f,
        -0.5f, -0.5f, -0.5f}
```

Definition at line 7 of file pyramid_data.h.

4.10.2.6 vertPosSize

```
constexpr size_t PyramidData::vertPosSize = sizeof(vertPos)  [static], [constexpr]
```

Definition at line 81 of file pyramid_data.h.

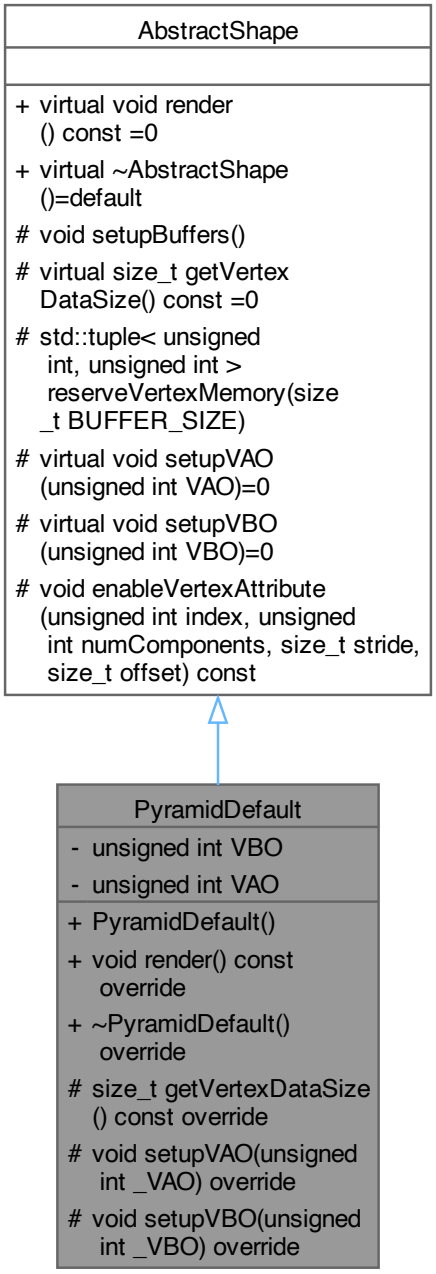
The documentation for this struct was generated from the following file:

- vender/render/models/objects/pyramid/pyramid_data.h

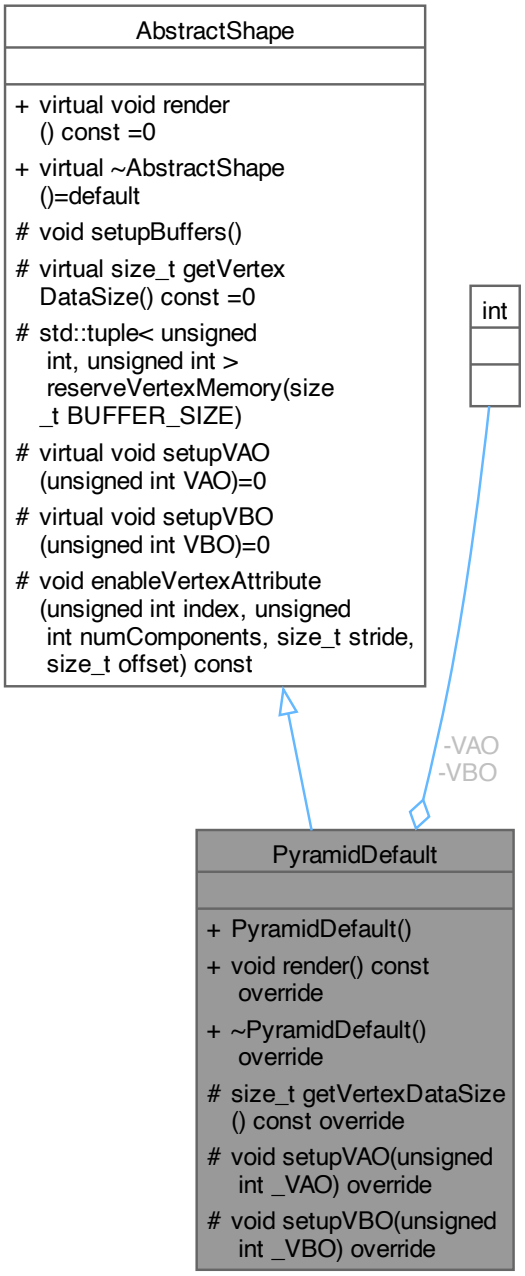
4.11 PyramidDefault Class Reference

```
#include <pyramid.h>
```

Inheritance diagram for PyramidDefault:



Collaboration diagram for PyramidDefault:



Public Member Functions

- PyramidDefault ()
- void render () const override
- ~PyramidDefault () override

Public Member Functions inherited from AbstractShape

- virtual ~AbstractShape ()=default

Protected Member Functions

- size_t getVertexDataSize () const override
- void setupVAO (unsigned int _VAO) override
- void setupVBO (unsigned int _VBO) override

Protected Member Functions inherited from AbstractShape

- void setupBuffers ()
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

Private Attributes

- unsigned int VBO
- unsigned int VAO

4.11.1 Detailed Description

Definition at line 9 of file pyramid.h.

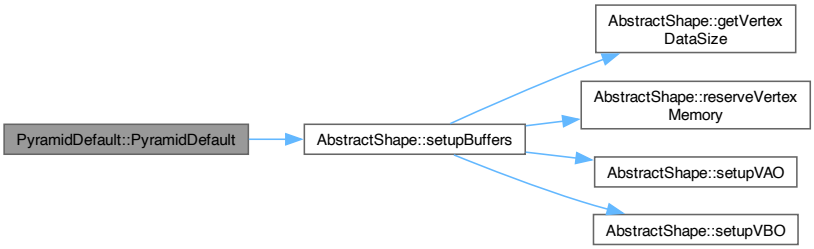
4.11.2 Constructor & Destructor Documentation

4.11.2.1 PyramidDefault()

```
PyramidDefault::PyramidDefault ( )  [inline]
```

Definition at line 12 of file pyramid.h.

Here is the call graph for this function:



4.11.2.2 ~PyramidDefault()

```
PyramidDefault::~PyramidDefault ( )  [inline], [override]
```

Definition at line 23 of file pyramid.h.

4.11.3 Member Function Documentation

4.11.3.1 getVertexDataSize()

```
size_t PyramidDefault::getVertexDataSize ( ) const  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 30 of file pyramid.h.

4.11.3.2 render()

```
void PyramidDefault::render ( ) const  [inline], [override], [virtual]
```

Implements AbstractShape.

Definition at line 17 of file pyramid.h.

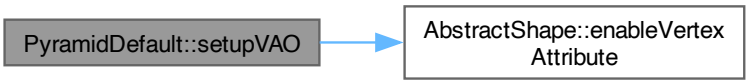
4.11.3.3 setupVAO()

```
void PyramidDefault::setupVAO (
            unsigned int _VAO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 35 of file pyramid.h.

Here is the call graph for this function:



4.11.3.4 setupVBO()

```
void PyramidDefault::setupVBO (
            unsigned int _VBO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 42 of file pyramid.h.

4.11.4 Member Data Documentation

4.11.4.1 VAO

```
unsigned int PyramidDefault::VAO  [private]
```

Definition at line 50 of file pyramid.h.

4.11.4.2 VBO

```
unsigned int PyramidDefault::VBO  [private]
```

Definition at line 49 of file pyramid.h.

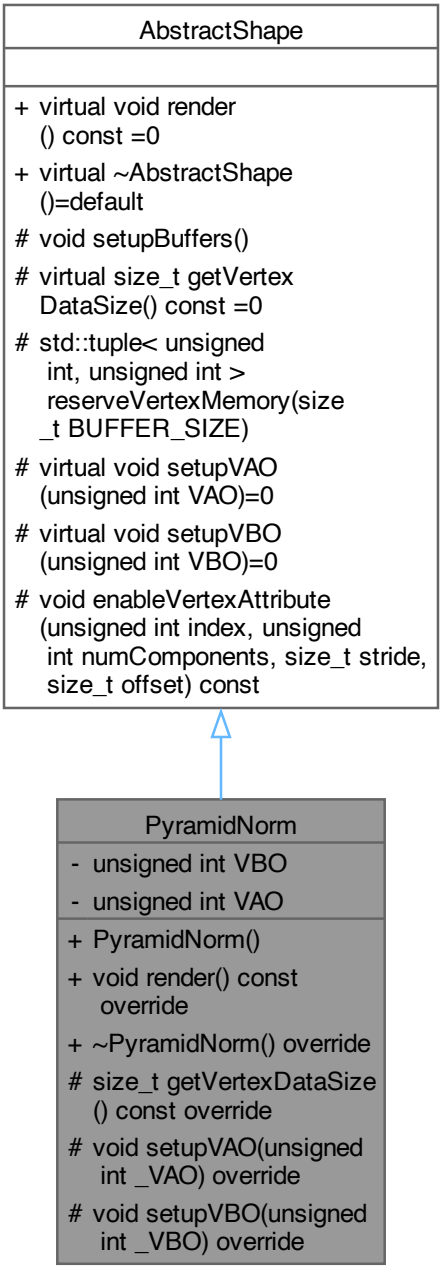
The documentation for this class was generated from the following file:

- vender/render/models/objects/pyramid/pyramid.h

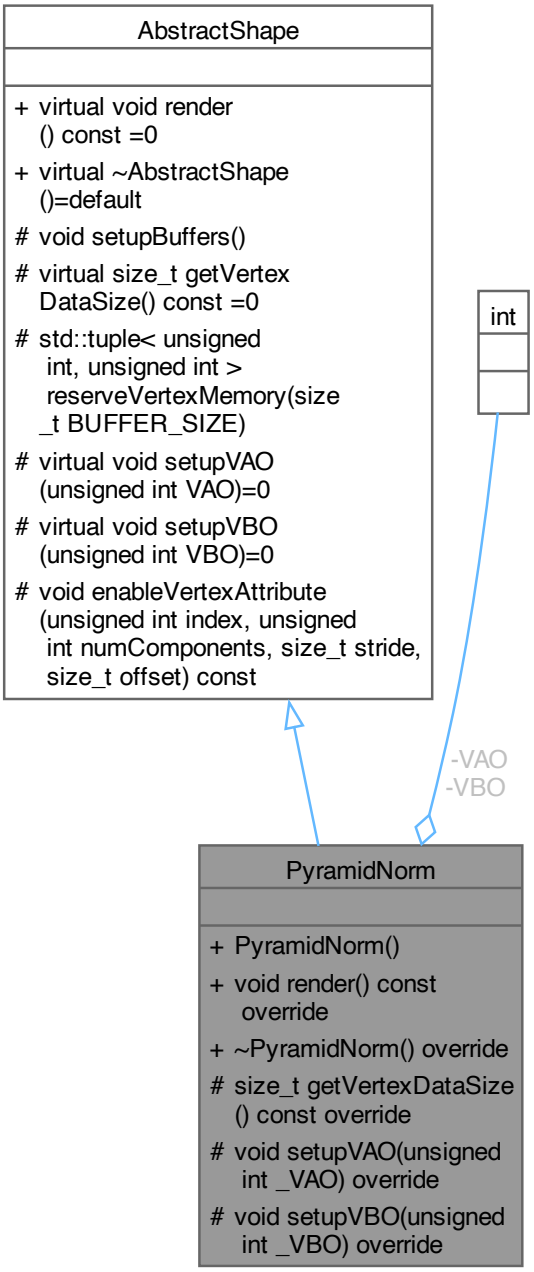
4.12 PyramidNorm Class Reference

```
#include <pyramid.h>
```

Inheritance diagram for PyramidNorm:



Collaboration diagram for PyramidNorm:



Public Member Functions

- PyramidNorm ()
- void render () const override
- ~PyramidNorm () override

Public Member Functions inherited from AbstractShape

- virtual ~AbstractShape ()=default

Protected Member Functions

- size_t getVertexDataSize () const override
- void setupVAO (unsigned int _VAO) override
- void setupVBO (unsigned int _VBO) override

Protected Member Functions inherited from AbstractShape

- void setupBuffers ()
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

Private Attributes

- unsigned int VBO
- unsigned int VAO

4.12.1 Detailed Description

Definition at line 53 of file pyramid.h.

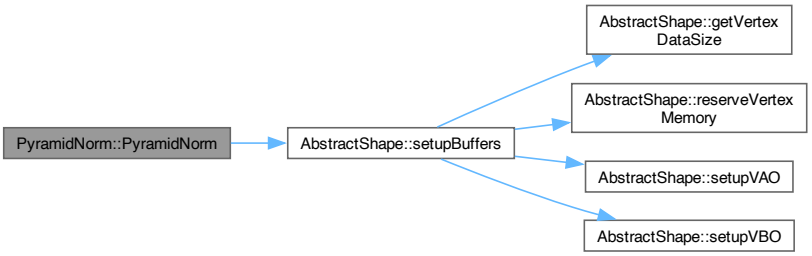
4.12.2 Constructor & Destructor Documentation

4.12.2.1 PyramidNorm()

```
PyramidNorm::PyramidNorm ( )  [inline]
```

Definition at line 56 of file pyramid.h.

Here is the call graph for this function:



4.12.2.2 ~PyramidNorm()

```
PyramidNorm::~PyramidNorm ( )  [inline], [override]
```

Definition at line 67 of file pyramid.h.

4.12.3 Member Function Documentation

4.12.3.1 getVertexDataSize()

```
size_t PyramidNorm::getVertexDataSize ( ) const  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 74 of file pyramid.h.

4.12.3.2 render()

```
void PyramidNorm::render ( ) const  [inline], [override], [virtual]
```

Implements AbstractShape.

Definition at line 61 of file pyramid.h.

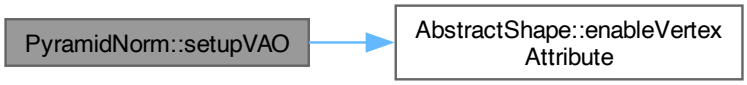
4.12.3.3 setupVAO()

```
void PyramidNorm::setupVAO (
            unsigned int _VAO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 79 of file pyramid.h.

Here is the call graph for this function:



4.12.3.4 setupVBO()

```
void PyramidNorm::setupVBO (
            unsigned int _VBO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 87 of file pyramid.h.

4.12.4 Member Data Documentation

4.12.4.1 VAO

```
unsigned int PyramidNorm::VAO  [private]
```

Definition at line 96 of file pyramid.h.

4.12.4.2 VBO

```
unsigned int PyramidNorm::VBO  [private]
```

Definition at line 95 of file pyramid.h.

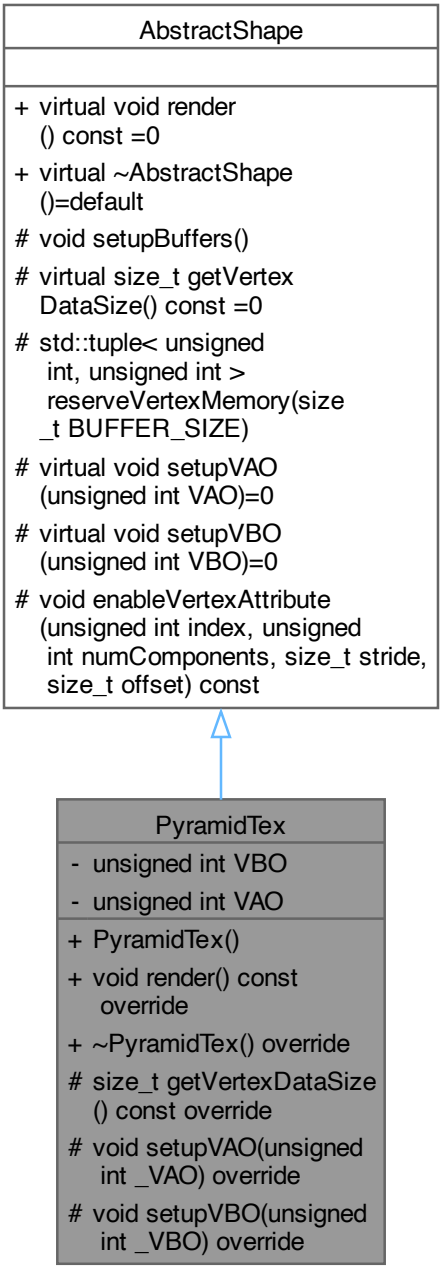
The documentation for this class was generated from the following file:

- vender/render/models/objects/pyramid/pyramid.h

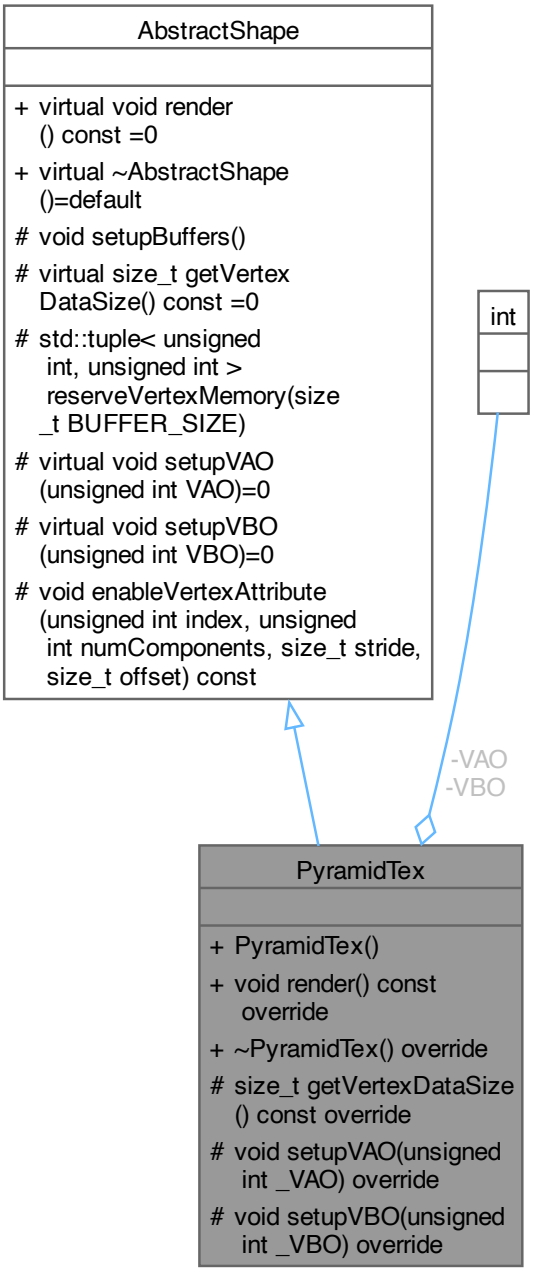
4.13 PyramidTex Class Reference

```
#include <pyramid.h>
```

Inheritance diagram for PyramidTex:



Collaboration diagram for PyramidTex:



Public Member Functions

- PyramidTex ()
- void render () const override
- ~PyramidTex () override

Public Member Functions inherited from AbstractShape

- virtual ~AbstractShape ()=default

Protected Member Functions

- size_t getVertexDataSize () const override
- void setupVAO (unsigned int _VAO) override
- void setupVBO (unsigned int _VBO) override

Protected Member Functions inherited from AbstractShape

- void setupBuffers ()
- std::tuple< unsigned int, unsigned int > reserveVertexMemory (size_t BUFFER_SIZE)
- void enableVertexAttribute (unsigned int index, unsigned int numComponents, size_t stride, size_t offset) const

Private Attributes

- unsigned int VBO
- unsigned int VAO

4.13.1 Detailed Description

Definition at line 99 of file pyramid.h.

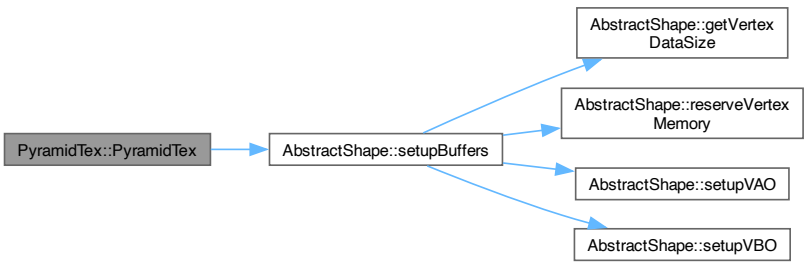
4.13.2 Constructor & Destructor Documentation

4.13.2.1 PyramidTex()

```
PyramidTex::PyramidTex ( )  [inline]
```

Definition at line 102 of file pyramid.h.

Here is the call graph for this function:



4.13.2.2 ~PyramidTex()

```
PyramidTex::~PyramidTex ( )  [inline], [override]
```

Definition at line 113 of file pyramid.h.

4.13.3 Member Function Documentation

4.13.3.1 getVertexDataSize()

```
size_t PyramidTex::getVertexDataSize ( ) const  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 120 of file pyramid.h.

4.13.3.2 render()

```
void PyramidTex::render ( ) const  [inline], [override], [virtual]
```

Implements AbstractShape.

Definition at line 107 of file pyramid.h.

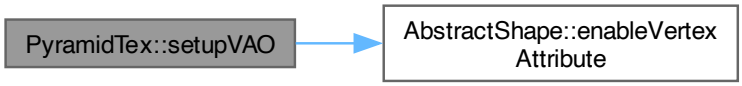
4.13.3.3 setupVAO()

```
void PyramidTex::setupVAO (
            unsigned int _VAO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 125 of file pyramid.h.

Here is the call graph for this function:



4.13.3.4 setupVBO()

```
void PyramidTex::setupVBO (
            unsigned int _VBO )  [inline], [override], [protected], [virtual]
```

Implements AbstractShape.

Definition at line 134 of file pyramid.h.

4.13.4 Member Data Documentation

4.13.4.1 VAO

```
unsigned int PyramidTex::VAO  [private]
```

Definition at line 144 of file pyramid.h.

4.13.4.2 VBO

```
unsigned int PyramidTex::VBO  [private]
```

Definition at line 143 of file pyramid.h.

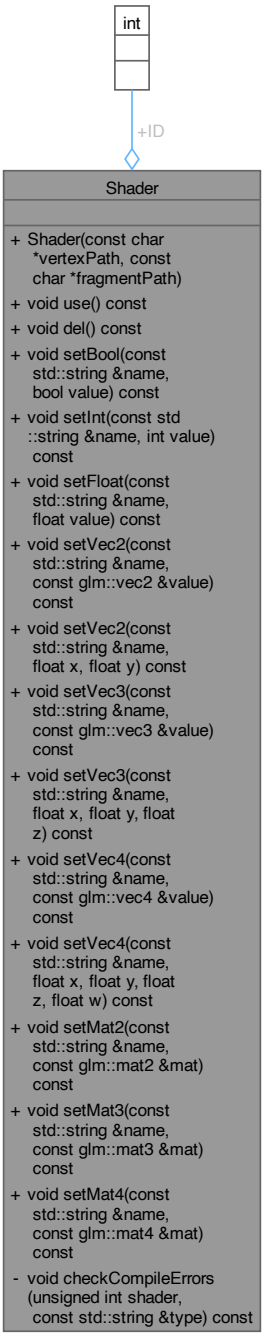
The documentation for this class was generated from the following file:

- vender/render/models/objects/pyramid/pyramid.h

4.14 Shader Class Reference

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

Collaboration diagram for Shader:



Public Member Functions

- Shader (const char ∗vertexPath, const char ∗fragmentPath)
- void use () const
- void del () const
- 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) const
- 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

Public Attributes

- unsigned int ID

Private Member Functions

- void checkCompileErrors (unsigned int shader, const std::string &type) const

4.14.1 Detailed Description

Definition at line 17 of file shader.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Shader()

```
Shader::Shader (
            const char * vertexPath,
            const char * fragmentPath )
```

Definition at line 7 of file shader.cpp.

Here is the call graph for this function:

4.14.3 Member Function Documentation

4.14.3.1 checkCompileErrors()

```
void Shader::checkCompileErrors (
            unsigned int shader,
            const std::string & type ) const  [private]
```

Definition at line 127 of file shader.cpp.

Here is the caller graph for this function:

4.14.3.2 del()

```
void Shader::del ( ) const
```

Definition at line 73 of file shader.cpp.

4.14.3.3 setBool()

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

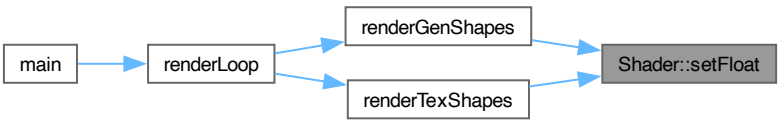
Definition at line 78 of file shader.cpp.

4.14.3.4 setFloat()

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

Definition at line 86 of file shader.cpp.

Here is the caller graph for this function:



4.14.3.5 setInt()

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

Definition at line 82 of file shader.cpp.

4.14.3.6 setMat2()

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

Definition at line 114 of file shader.cpp.

4.14.3.7 setMat3()

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

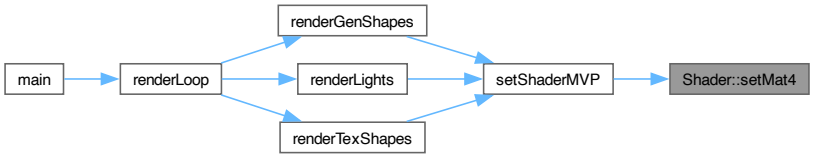
Definition at line 118 of file shader.cpp.

4.14.3.8 setMat4()

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

Definition at line 122 of file shader.cpp.

Here is the caller graph for this function:



4.14.3.9 setVec2() [1/2]

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

Definition at line 90 of file shader.cpp.

4.14.3.10 setVec2() [2/2]

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

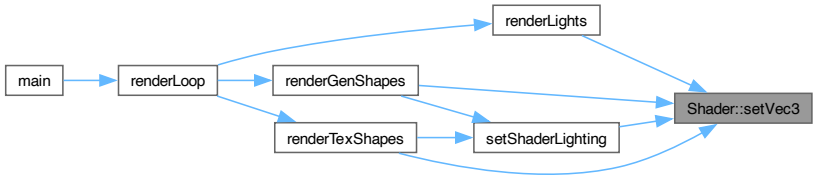
Definition at line 94 of file shader.cpp.

4.14.3.11 setVec3() [1/2]

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

Definition at line 98 of file shader.cpp.

Here is the caller graph for this function:



4.14.3.12 setVec3() [2/2]

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

Definition at line 102 of file shader.cpp.

4.14.3.13 setVec4() [1/2]

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

Definition at line 106 of file shader.cpp.

4.14.3.14 setVec4() [2/2]

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

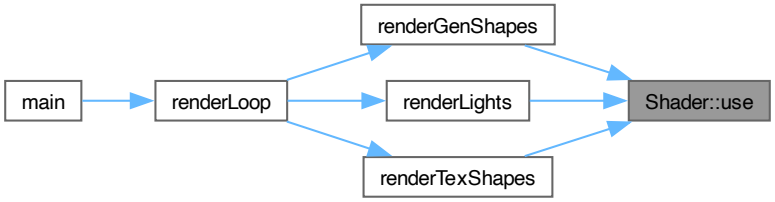
Definition at line 110 of file shader.cpp.

4.14.3.15 use()

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

Definition at line 69 of file shader.cpp.

Here is the caller graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 ID

```
unsigned int Shader::ID
```

Definition at line 21 of file shader.h.

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

- vender/shaders/shader.h
- vender/shaders/shader.cpp

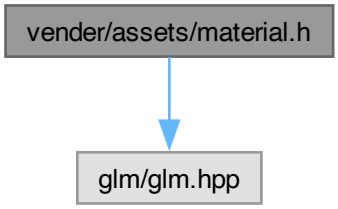
Chapter 5

File Documentation

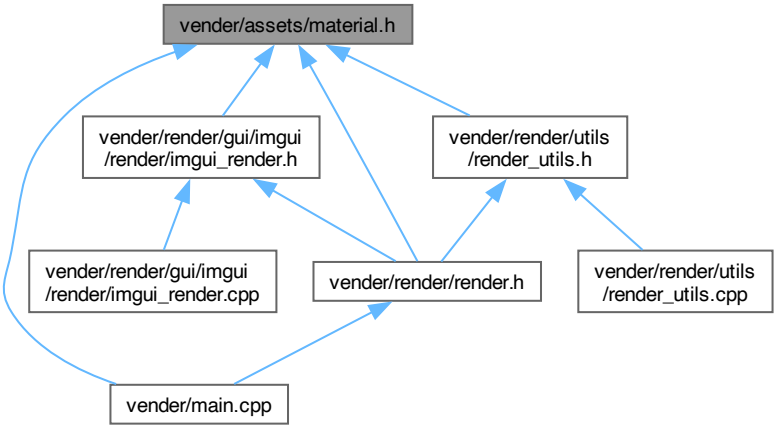
5.1 vender/assets/material.h File Reference

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

Include dependency graph for material.h:



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



Classes

- struct Material

Variables

- const Material mat_gold

5.1.1 Variable Documentation

5.1.1.1 mat_gold

```
const Material mat_gold
```

Initial value:

```
= {
    glm::vec3(1.0f, 0.84f, 0.0f),
    glm::vec3(1.0f, 0.84f, 0.0f),
    glm::vec3(0.5f, 0.5f, 0.5f),
    168.0f
}
```

Definition at line 14 of file material.h.

5.2 material.h

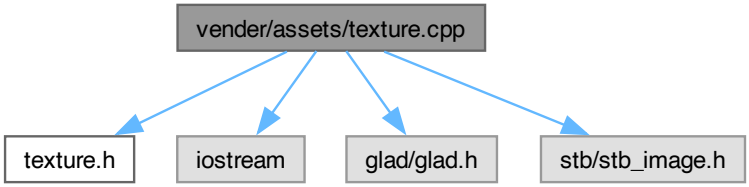
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 struct Material
00006 {
00007     glm::vec3 ambient = glm::vec3(1.0f, 0.5f, 0.5f);
00008     glm::vec3 diffuse = glm::vec3(1.0f, 0.5f, 0.5f);
00009     glm::vec3 specular = glm::vec3(0.5f, 0.5f, 0.5f);
00010     float shininess = 64.0f;
00011 };
00012
00013 // Metal approximation
00014 const Material mat_gold = {
00015     glm::vec3(1.0f, 0.84f, 0.0f), // ambient
00016     glm::vec3(1.0f, 0.84f, 0.0f), // diffuse
00017     glm::vec3(0.5f, 0.5f, 0.5f),  // specular
00018     168.0f                        // shininess
00019 };
```

5.3 vender/assets/texture.cpp File Reference

```
#include "texture.h"
#include <iostream>
#include <glad/glad.h>
```

`#include "stb/stb_image.h"`
Include dependency graph for texture.cpp:



Functions

- unsigned int loadTexture (char const *path)

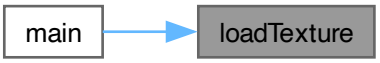
5.3.1 Function Documentation

5.3.1.1 loadTexture()

```
unsigned int loadTexture (
            char const * path )
```

Definition at line 7 of file texture.cpp.

Here is the caller graph for this function:



5.4 texture.cpp

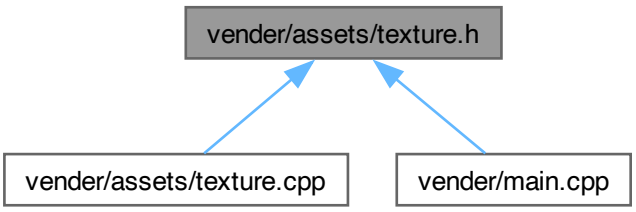
Go to the documentation of this file.

```
00001 #include "texture.h"
00002
00003 #include <iostream>
00004 #include <glad/glad.h>
00005 #include "stb/stb_image.h"
00006
00007 unsigned int loadTexture(char const *path)
00008 {
00009     unsigned int textureID;
00010     glGenTextures(1, &textureID);
00011
```

```
00012     int width;
00013     int height;
00014     int nrComponents;
00015     unsigned char *data = stbi_load(path, &width, &height, &nrComponents, 0);
00016     if (data)
00017     {
00018         GLenum format;
00019         if (nrComponents == 1)
00020             format = GL_RED;
00021         else if (nrComponents == 3)
00022             format = GL_RGB;
00023         else if (nrComponents == 4)
00024             format = GL_RGBA;
00025 
00026         glBindTexture(GL_TEXTURE_2D, textureID);
00027         glTexImage2D(GL_TEXTURE_2D, 0, format, width, height, 0, format, GL_UNSIGNED_BYTE, data);
00028         glGenerateMipmap(GL_TEXTURE_2D);
00029 
00030         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
00031         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
00032         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
00033         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
00034 
00035         stbi_image_free(data);
00036     }
00037     else
00038     {
00039         std::cout << "Texture failed to load at path: " << path << std::endl;
00040         stbi_image_free(data);
00041     }
00042 
00043     return textureID;
00044 }
```

5.5 vender/assets/texture.h File Reference

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



Functions

- unsigned int loadTexture (char const *path)

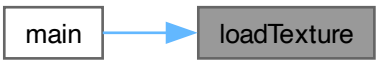
5.5.1 Function Documentation

5.5.1.1 loadTexture()

```
unsigned int loadTexture (
            char const * path )
```

Definition at line 7 of file texture.cpp.

Here is the caller graph for this function:



5.6 texture.h

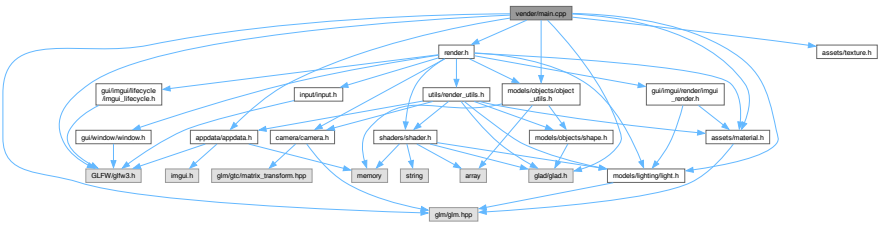
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 unsigned int loadTexture(char const *path);
```

5.7 vender/main.cpp File Reference

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include "appdata/appdata.h"
#include "render.h"
#include "assets/texture.h"
#include "assets/material.h"
#include "models/objects/object_utils.h"
#include "models/lighting/light.h"
```

Include dependency graph for main.cpp:



Macros

- #define GLFW_DLL

Functions

- int main ()

5.7.1 Macro Definition Documentation

5.7.1.1 GLFW_DLL

```
#define GLFW_DLL
```

Definition at line 1 of file main.cpp.

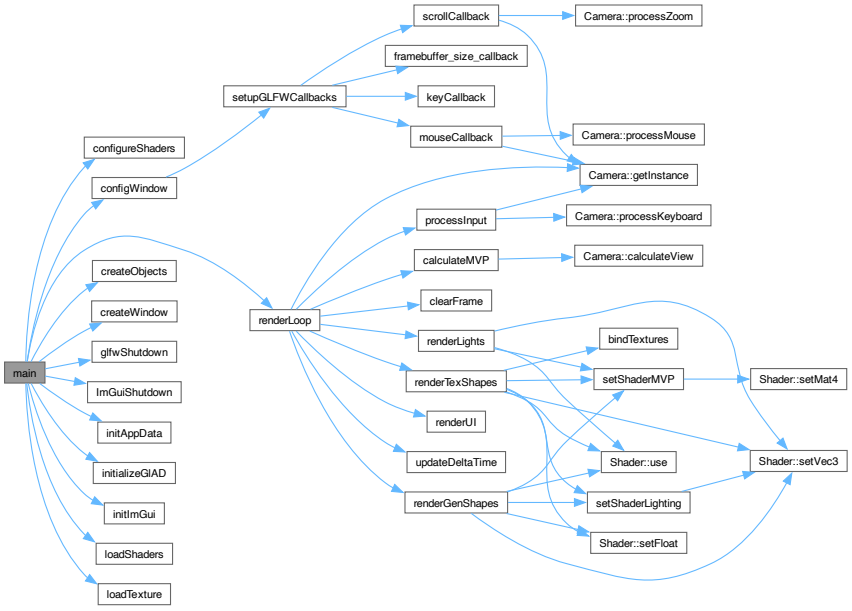
5.7.2 Function Documentation

5.7.2.1 main()

```
int main ( )
```

Definition at line 14 of file main.cpp.

Here is the call graph for this function:



5.8 main.cpp

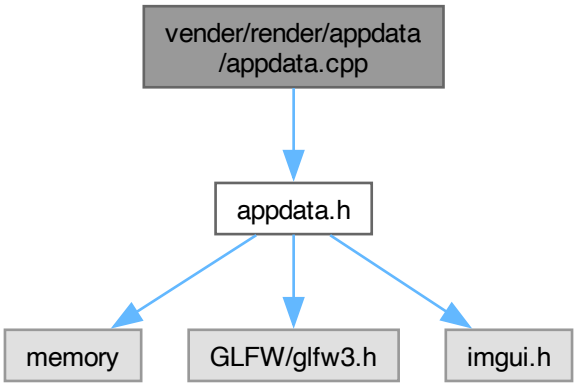
Go to the documentation of this file.

```
00001 #define GLFW_DLL
00002
00003 #include <glad/glad.h>
00004 #include <GLFW/glfw3.h>
00005 #include <glm/glm.hpp>
00006
00007 #include "appdata/appdata.h"
00008 #include "render.h"
00009 #include "assets/texture.h"
00010 #include "assets/material.h"
00011 #include "models/objects/object_utils.h"
00012 #include "models/lighting/light.h"
```

```
00013
00014 int main()
00015 {
00016   GLFWwindow *window = createWindow();
00017   configWindow(window);
00018   initializeGlAD();
00019   glEnable(GL_DEPTH_TEST);
00020   initImGui(window);
00021   const std::unique_ptr<AppData> appData = initAppData(window);
00022   glfwSetWindowUserPointer(window, appData.get());
00023
00024   unsigned int diffuseMap = loadTexture("../assets/textures/container.png");
00025   unsigned int specularMap = loadTexture("../assets/textures/container_specular.png");
00026   auto shaders = loadShaders();
00027   configureShaders(shaders);
00028   auto objects = createObjects();
00029
00030   Material material;
00031   Light light;
00032   int selectedMaterial = 0;
00033   int selectedShape = 0;
00034   auto clear_color = ImVec4(0.6f, 0.6f, 0.6f, 1.0f);
00035
00036   renderLoop(window,
00037              *appData,
00038              objects,
00039              shaders,
00040              selectedShape,
00041              material,
00042              selectedMaterial,
00043              light,
00044              clear_color,
00045              diffuseMap,
00046              specularMap);
00047
00048   ImGuiShutdown();
00049   glfwShutdown(window);
00050   return 0;
00051 }
```

5.9 vender/render/appdata/appdata.cpp File Reference

`#include "appdata.h"`
Include dependency graph for appdata.cpp:



Functions

- std::unique_ptr< AppData > initAppData (GLFWwindow *window)

5.9.1 Function Documentation

5.9.1.1 initAppData()

```
std::unique_ptr< AppData > initAppData (
            GLFWwindow * window )
```

Definition at line 3 of file appdata.cpp.

Here is the caller graph for this function:

5.10 appdata.cpp

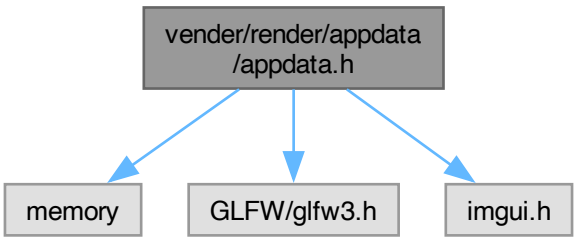
Go to the documentation of this file.

```
00001 #include "appdata.h"
00002
00003 std::unique_ptr<AppData> initAppData(GLFWwindow *window)
00004 {
00005     const ImGuiIO &io = ImGui::GetIO();
00006     int framebufferWidth;
00007     int framebufferHeight;
00008     glfwGetFramebufferSize(window, &framebufferWidth, &framebufferHeight);
00009     auto lastX = (float)framebufferWidth / 2;
00010     auto lastY = (float)framebufferHeight / 2;
00011
00012     auto appDataPtr = std::make_unique<AppData>(io, framebufferWidth, framebufferHeight, lastX,
      lastY);
00013     return appDataPtr;
00014 }
```

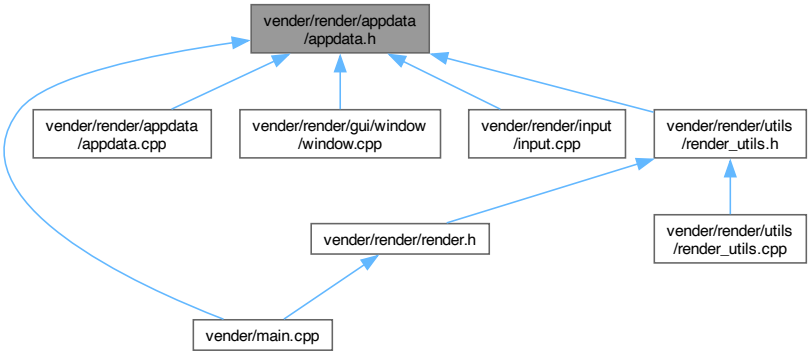
5.11 vender/render/appdata/appdata.h File Reference

```
#include <memory>
#include <GLFW/glfw3.h>
#include "imgui.h"
```

Include dependency graph for appdata.h:



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



Classes

- struct AppData

Functions

- std::unique_ptr< AppData > initAppData (GLFWwindow *window)

5.11.1 Function Documentation

5.11.1.1 initAppData()

```
std::unique_ptr< AppData > initAppData (
            GLFWwindow * window )
```

Definition at line 3 of file appdata.cpp.

Here is the caller graph for this function:

5.12 appdata.h

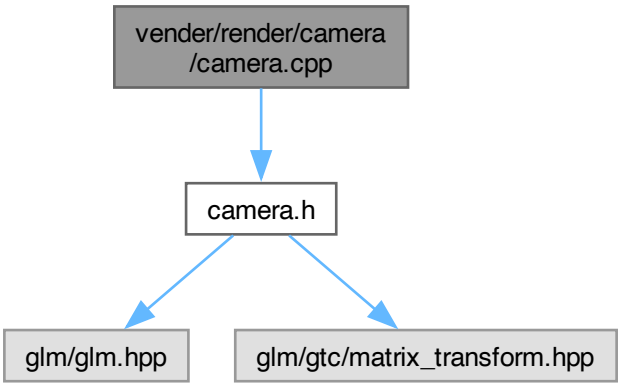
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <memory>
00004 #include <GLFW/glfw3.h>
00005 #include "imgui.h"
00006
00007 struct AppData
00008 {
00009     const ImGuiIO &io;
00010     int framebufferWidth;
00011     int framebufferHeight;
00012     float lastX;
00013     float lastY;
00014     float deltaTime = 0.0f;
00015     float lastFrame = 0.0f;
00016     bool firstMouse = true;
00017     bool debug_mode = false;
00018     AppData(
00019         const ImGuiIO &_io,
00020         int _fbw,
00021         int _fbh,
00022         float _lastX,
00023         float _lastY)
00024         : io(_io),
00025           framebufferWidth(_fbw),
00026           framebufferHeight(_fbh),
00027           lastX(_lastX),
00028           lastY(_lastY)
00029     {
00030     }
00031 };
00032
00033 std::unique_ptr<AppData> initAppData(GLFWwindow *window);
```

5.13 vender/render/camera/camera.cpp File Reference

`#include "camera.h"`

Include dependency graph for camera.cpp:



5.14 camera.cpp

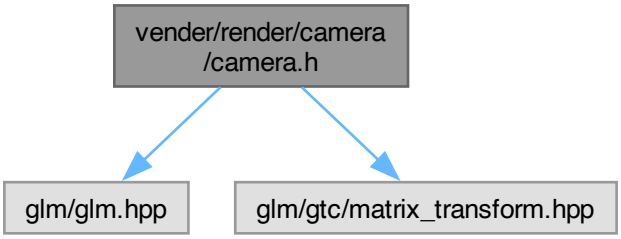
Go to the documentation of this file.

```
00001 #include "camera.h"
00002
00003 glm::mat4 Camera::calculateView() const
00004 {
00005     return glm::lookAt(cameraPos, cameraPos + cameraFront, cameraUp);
00006 }
00007
00008 void Camera::processKeyboard(Direction direction, float deltaTime)
00009 {
00010     using enum Direction;
00011     float cameraSpeed = speed * deltaTime;
00012     if (direction == UP)
00013         cameraPos += cameraSpeed * cameraFront;
00014     if (direction == DOWN)
00015         cameraPos -= cameraSpeed * cameraFront;
00016     if (direction == LEFT)
00017         cameraPos -= glm::normalize(glm::cross(cameraFront, cameraUp)) * cameraSpeed;
00018     if (direction == RIGHT)
00019         cameraPos += glm::normalize(glm::cross(cameraFront, cameraUp)) * cameraSpeed;
00020 }
00021
00022 void Camera::processMouse(float xoffset, float yoffset)
00023 {
00024
00025     xoffset *= sensitivity;
00026     yoffset *= sensitivity;
00027
00028     yaw += xoffset;
00029     pitch += yoffset;
00030
00031     if (pitch > 89.0f)
00032         pitch = 89.0f;
00033     if (pitch < -89.0f)
00034         pitch = -89.0f;
00035
00036     glm::vec3 direction;
00037     direction.x = cos(glm::radians(yaw)) * cos(glm::radians(pitch));
00038     direction.y = sin(glm::radians(pitch));
00039     direction.z = sin(glm::radians(yaw)) * cos(glm::radians(pitch));
00040     cameraFront = glm::normalize(direction);
00041 }
00042
00043 void Camera::processZoom(float yoffset)
00044 {
00045     fov -= yoffset;
00046     if (fov < 1.0f)
00047         fov = 1.0f;
00048     if (fov > 90.0f)
00049         fov = 90.0f;
00050 }
```

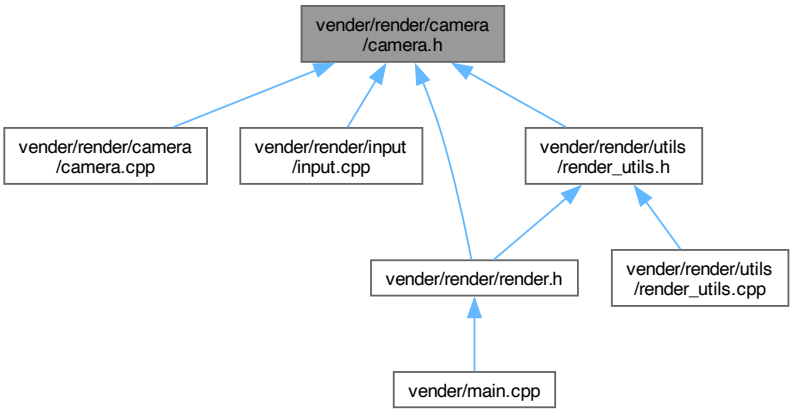
5.15 vender/render/camera/camera.h File Reference

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
```

Include dependency graph for camera.h:



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



Classes

- class Camera

Enumerations

- enum struct Direction { LEFT , RIGHT , UP , DOWN }

5.15.1 Enumeration Type Documentation

5.15.1.1 Direction

```
enum struct Direction  [strong]
```

Enumerator

LEFT	
RIGHT	
UP	
DOWN	

Definition at line 6 of file camera.h.

5.16 camera.h

Go to the documentation of this file.

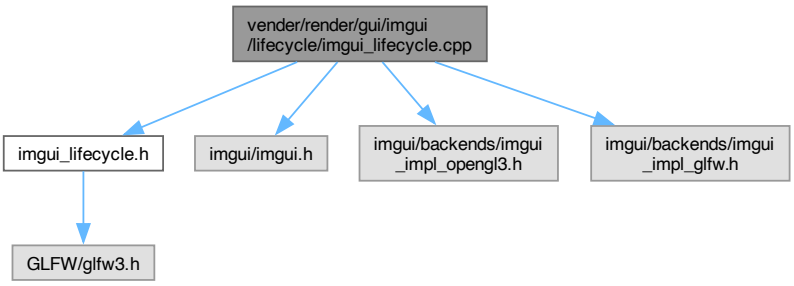
```
00001 #pragma once
00002
```

```
00003 #include <glm/glm.hpp>
00004 #include <glm/gtc/matrix_transform.hpp>
00005
00006 enum struct Direction
00007 {
00008     LEFT,
00009     RIGHT,
00010     UP,
00011     DOWN
00012 };
00013
00014 class Camera
00015 {
00016 public:
00017     float yaw = -90.0f;
00018     float pitch = 0.0f;
00019     float fov = 45.0f;
00020     float speed = 2.5f;
00021     float sensitivity = 0.1f;
00022
00023     glm::vec3 cameraPos = glm::vec3(0.0f, 0.0f, 3.0f);
00024     glm::vec3 cameraFront = glm::vec3(0.0f, 0.0f, -1.0f);
00025     glm::vec3 cameraUp = glm::vec3(0.0f, 1.0f, 0.0f);
00026
00027     static Camera &getInstance()
00028     {
00029         static Camera instance;
00030         return instance;
00031     }
00032
00033     Camera(Camera const &) = delete;
00034     Camera &operator=(Camera const &) = delete;
00035
00036     glm::mat4 calculateView() const;
00037     void processKeyboard(Direction direction, float deltaTime);
00038     void processMouse(float xoffset, float yoffset);
00039     void processZoom(float yoffset);
00040
00041 private:
00042     Camera() = default;
00043 };
```

5.17 vender/render/gui/imgui/lifecycle/imgui_lifecycle.cpp File Reference

```
#include "imgui_lifecycle.h"
#include "imgui/imgui.h"
#include "imgui/backends/imgui_impl_opengl3.h"
#include "imgui/backends/imgui_impl_glfw.h"
```

Include dependency graph for imgui_lifecycle.cpp:



Functions

- void initImGui (GLFWwindow *window)
- void ImGuiShutdown ()

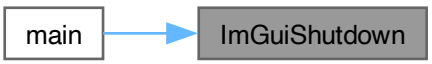
5.17.1 Function Documentation

5.17.1.1 ImGuiShutdown()

```
void ImGuiShutdown ( )
```

Definition at line 25 of file imgui_lifecycle.cpp.

Here is the caller graph for this function:

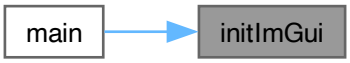


5.17.1.2 initImGui()

```
void initImGui (
            GLFWwindow * window )
```

Definition at line 7 of file imgui_lifecycle.cpp.

Here is the caller graph for this function:



5.18 imgui_lifecycle.cpp

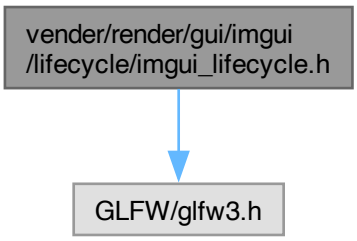
Go to the documentation of this file.

```
00001 #include "imgui_lifecycle.h"
00002
00003 #include "imgui/imgui.h"
00004 #include "imgui/backends/imgui_impl_opengl3.h"
00005 #include "imgui/backends/imgui_impl_glfw.h"
00006
00007 void initImGui(GLFWwindow *window)
00008 {
00009     const char *glsl_version = "#version 150";
00010     IMGUI_CHECKVERSION();
00011     ImGui::CreateContext();
00012     ImGuiIO &io = ImGui::GetIO();
00013     (void)io;
00014     io.ConfigFlags |= ImGuiConfigFlags_NavEnableKeyboard;
```

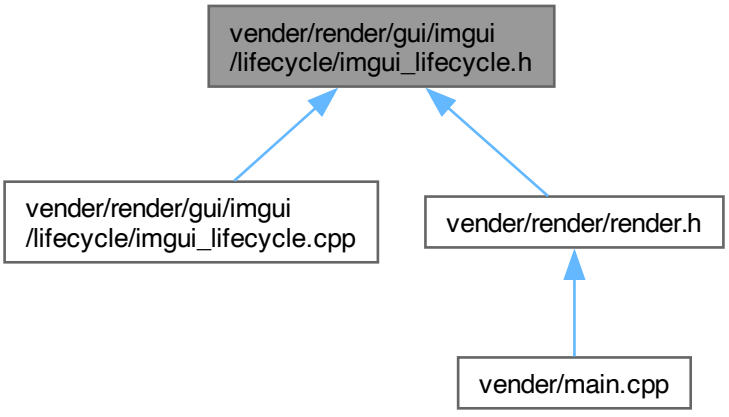
```
00015     io.ConfigFlags |= ImGuiConfigFlags_NavEnableGamepad;
00016
00017     ImGui::StyleColorsDark();
00018     ImGui::SetNextWindowCollapsed(true);
00019
00020     // Setup Platform/Renderer backends
00021     ImGui_ImplGlfw_InitForOpenGL(window, true); // Second param install_callback=true will install
      GLFW callbacks and chain to existing ones.
00022     ImGui_ImplOpenGL3_Init(glsl_version);
00023 }
00024
00025 void ImGuiShutdown()
00026 {
00027     ImGui_ImplOpenGL3_Shutdown();
00028     ImGui_ImplGlfw_Shutdown();
00029     ImGui::DestroyContext();
00030 }
```

5.19 vender/render/gui/imgui/lifecycle/imgui_lifecycle.h File Reference

`#include <GLFW/glfw3.h>`
Include dependency graph for imgui_lifecycle.h:



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



Functions

- void initImGui (GLFWwindow *window)
- void ImGuiShutdown ()

5.19.1 Function Documentation

5.19.1.1 ImGuiShutdown()

```
void ImGuiShutdown ( )
```

Definition at line 25 of file imgui_lifecycle.cpp.

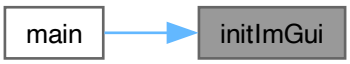
Here is the caller graph for this function:

5.19.1.2 initImGui()

```
void initImGui (
            GLFWwindow * window )
```

Definition at line 7 of file imgui_lifecycle.cpp.

Here is the caller graph for this function:



5.20 imgui_lifecycle.h

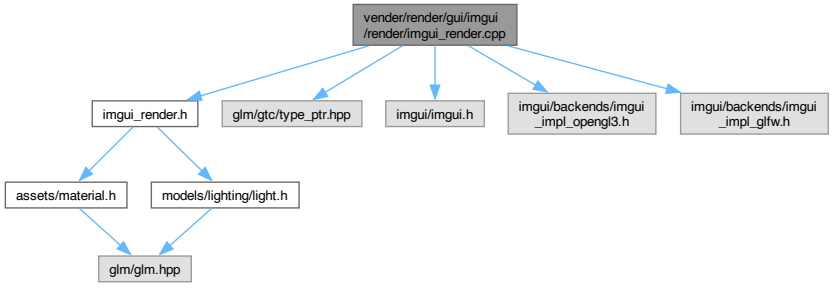
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <GLFW/glfw3.h>
00004
00005 void initImGui(GLFWwindow *window);
00006 void ImGuiShutdown();
```

5.21 vender/render/gui/imgui/render/imgui_render.cpp File Reference

```
#include "imgui_render.h"
#include <glm/gtc/type_ptr.hpp>
#include "imgui/imgui.h"
#include "imgui/backends/imgui_impl_opengl3.h"
#include "imgui/backends/imgui_impl_glfw.h"
```

Include dependency graph for imgui_render.cpp:



Functions

- void renderUI (const float framerate, Light &light, Material &material, int &selectedMaterial, int &selected↩ Shape)

5.21.1 Function Documentation

5.21.1.1 renderUI()

```
void renderUI (
            const float framerate,
            Light & light,
            Material & material,
            int & selectedMaterial,
            int & selectedShape )
```

Definition at line 8 of file imgui_render.cpp.

Here is the caller graph for this function:

5.22 imgui_render.cpp

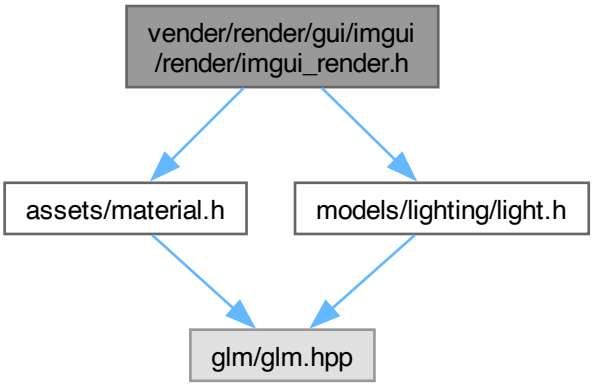
Go to the documentation of this file.

```
00001 #include "imgui_render.h"
00002 #include <glm/gtc/type_ptr.hpp>
00003
00004 #include "imgui/imgui.h"
00005 #include "imgui/backends/imgui_impl_opengl3.h"
00006 #include "imgui/backends/imgui_impl_glfw.h"
00007
00008 void renderUI(const float framerate, Light &light, Material &material, int &selectedMaterial, int
      &selectedShape)
00009 {
00010     ImGui_ImplOpenGL3_NewFrame();
00011     ImGui_ImplGlfw_NewFrame();
00012     ImGui::NewFrame();
00013     ImGui::Begin("Controls");
00014     ImGui::Text("FPS = %f", framerate);
00015     if (ImGui::CollapsingHeader("Light"))
00016     {
00017         ImGui::SliderFloat3("Pos", glm::value_ptr(light.pos), -2.0f, 2.0f);
00018         ImGui::ColorPicker3("Color", glm::value_ptr(light.color));
00019     }
00020
00021     if (ImGui::CollapsingHeader("Object"))
00022     {
00023         ImGui::Columns(2, nullptr, false);
00024         ImGui::RadioButton("Cube", &selectedShape, 0);
00025         ImGui::RadioButton("Pyramid", &selectedShape, 1);
00026
00027         ImGui::NextColumn();
00028         if (ImGui::RadioButton("Generic", &selectedMaterial, 0))
00029         {
00030             material = Material();
00031         }
00032         else if (ImGui::RadioButton("Gold", &selectedMaterial, 1))
00033         {
00034             material = mat_gold;
00035         }
00036         ImGui::RadioButton("Container", &selectedMaterial, 2);
00037         ImGui::Columns(1);
00038         ImGui::Dummy(ImVec2(0.0f, 15.0f));
00039
00040         ImGui::ColorPicker3("Ambient", glm::value_ptr(material.ambient));
00041         ImGui::ColorPicker3("Diffuse", glm::value_ptr(material.diffuse));
00042         ImGui::ColorPicker3("SpecularColor", glm::value_ptr(material.specular));
00043         ImGui::SliderFloat("Shininess", &material.shininess, 1.0f, 200.0f);
00044     }
00045     ImGui::End();
00046     ImGui::Render();
00047     ImGui_ImplOpenGL3_RenderDrawData(ImGui::GetDrawData());
00048 }
```

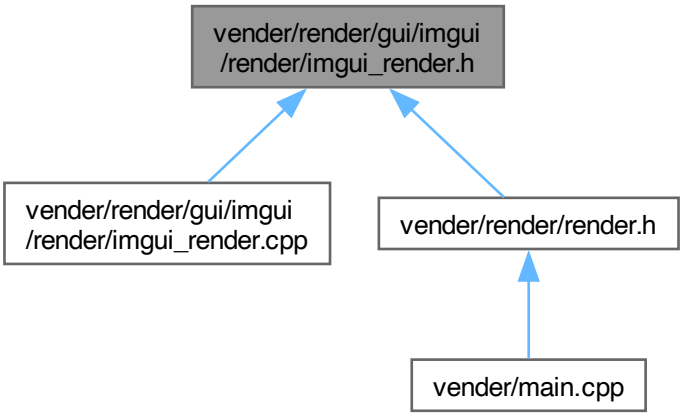
5.23 vender/render/gui/imgui/render/imgui_render.h File Reference

```
#include "assets/material.h"
#include "models/lighting/light.h"
```

Include dependency graph for imgui_render.h:



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



Functions

- void renderUI (const float framerate, Light &light, Material &material, int &selectedMaterial, int &selected↩ Shape)

5.23.1 Function Documentation

5.23.1.1 renderUI()

```
void renderUI (
            const float framerate,
```

```
            Light & light,
            Material & material,
            int & selectedMaterial,
            int & selectedShape )
```

Definition at line 8 of file imgui_render.cpp.

Here is the caller graph for this function:

5.24 imgui_render.h

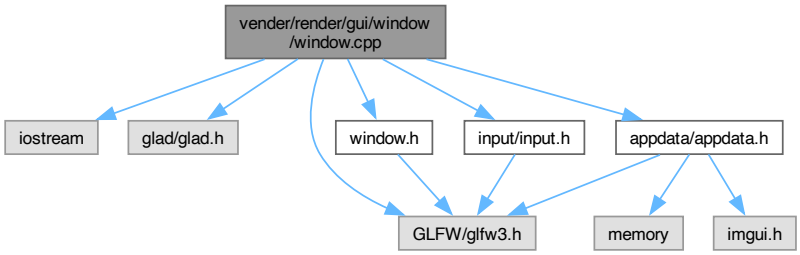
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "assets/material.h"
00004 #include "models/lighting/light.h"
00005
00006 void renderUI(const float framerate, Light &light, Material &material, int &selectedMaterial, int
      &selectedShape);
```

5.25 vender/render/gui/window/window.cpp File Reference

```
#include <iostream>
#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include "window.h"
#include "appdata/appdata.h"
#include "input/input.h"
```

Include dependency graph for window.cpp:



Functions

- GLFWwindow * createWindow (GLint scr_width, GLint scr_height)
- void configWindow (GLFWwindow *window)
- bool initializeGlAD ()
- void glfwShutdown (GLFWwindow *window)
- void setupGLFWCallbacks (GLFWwindow *window)
- void framebuffer_size_callback (GLFWwindow *window, int width, int height)

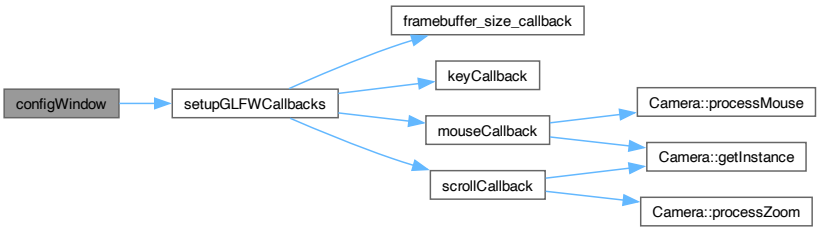
5.25.1 Function Documentation

5.25.1.1 configWindow()

```
void configWindow (
            GLFWwindow * window )
```

Definition at line 26 of file window.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.25.1.2 createWindow()

```
GLFWwindow * createWindow (
            GLint scr_width,
            GLint scr_height )
```

Definition at line 9 of file window.cpp.

Here is the caller graph for this function:

5.25.1.3 framebuffer_size_callback()

```
void framebuffer_size_callback (
            GLFWwindow * window,
            int width,
            int height )
```

Definition at line 59 of file window.cpp.

Here is the caller graph for this function:

5.25.1.4 glfwShutdown()

```
void glfwShutdown (
            GLFWwindow * window )
```

Definition at line 44 of file window.cpp.

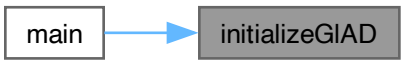
Here is the caller graph for this function:

5.25.1.5 initializeGlAD()

```
bool initializeGlAD ( )
```

Definition at line 34 of file window.cpp.

Here is the caller graph for this function:

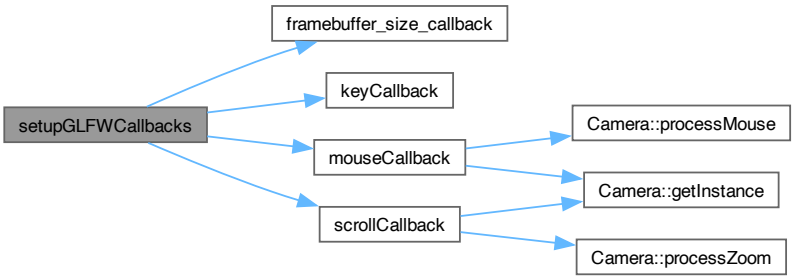


5.25.1.6 setupGLFWCallbacks()

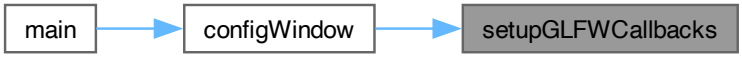
```
void setupGLFWCallbacks (
            GLFWwindow * window )
```

Definition at line 50 of file window.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.26 window.cpp

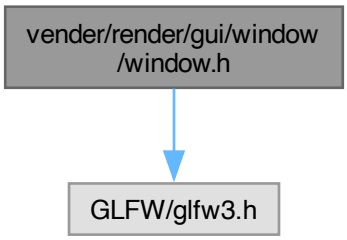
Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include <glad/glad.h>
00003 #include <GLFW/glfw3.h>
00004
00005 #include "window.h"
00006 #include "appdata/appdata.h"
00007 #include "input/input.h"
00008
00009 GLFWwindow *createWindow(GLint scr_width, GLint scr_height)
00010 {
00011     glfwInit();
00012     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
00013     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
00014     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
00015     glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE); // safe on mac
00016     GLFWwindow *window = glfwCreateWindow(scr_width, scr_height, "vender", nullptr, nullptr);
00017
00018     if (window == nullptr)
00019     {
00020         std::cout « "Failed to create GLFW window" « std::endl;
00021         glfwTerminate();
00022     }
00023     return window;
00024 }
00025
00026 void configWindow(GLFWwindow *window)
00027 {
00028     glfwMakeContextCurrent(window);
00029     setupGLFWCallbacks(window);
00030     glfwSwapInterval(1);
00031     glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED);
00032 }
00033
00034 bool initializeGlAD()
00035 {
00036     if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
00037     {
00038         std::cout « "Failed to initialize GLAD" « std::endl;
00039         return false;
00040     }
00041     return true;
00042 }
00043
00044 void glfwShutdown(GLFWwindow *window)
00045 {
00046     glfwDestroyWindow(window);
00047     glfwTerminate();
00048 }
00049
00050 void setupGLFWCallbacks(GLFWwindow *window)
00051 {
00052     glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
00053     glfwSetCursorPosCallback(window, mouseCallback);
00054     glfwSetScrollCallback(window, scrollCallback);
00055     glfwSetKeyCallback(window, keyCallback);
00056 }
00057
00058 // glfw: window size changed, callback executes
00059 void framebuffer_size_callback(GLFWwindow *window, int width, int height)
00060 {
00061     auto appData = (AppData *)glfwGetWindowUserPointer(window);
00062     appData->framebufferWidth = width;
00063     appData->framebufferHeight = height;
00064     glViewport(0, 0, width, height);
00065 }
```

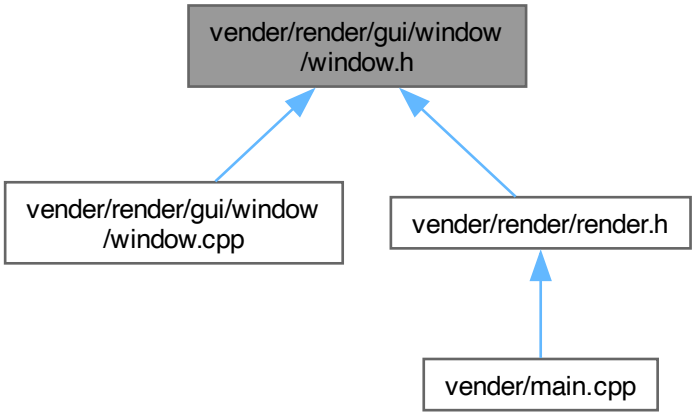
5.27 vender/render/gui/window/window.h File Reference

```
#include <GLFW/glfw3.h>
```

Include dependency graph for window.h:



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



Functions

- GLFWwindow * createWindow (GLint scr_width=800, GLint scr_height=600)
- void configWindow (GLFWwindow *window)
- bool initializeGlAD ()
- void glfwShutdown (GLFWwindow *window)
- void setupGLFWCallbacks (GLFWwindow *window)
- void framebuffer_size_callback (GLFWwindow *window, int width, int height)

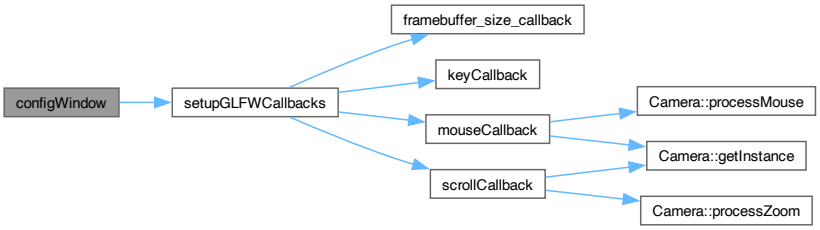
5.27.1 Function Documentation

5.27.1.1 configWindow()

```
void configWindow (
            GLFWwindow * window )
```

Definition at line 26 of file window.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.27.1.2 createWindow()

```
GLFWwindow * createWindow (
            GLint scr_width = 800,
            GLint scr_height = 600 )
```

Definition at line 9 of file window.cpp.

Here is the caller graph for this function:

5.27.1.3 framebuffer_size_callback()

```
void framebuffer_size_callback (
            GLFWwindow * window,
            int width,
            int height )
```

Definition at line 59 of file window.cpp.

Here is the caller graph for this function:

5.27.1.4 glfwShutdown()

```
void glfwShutdown (
            GLFWwindow * window )
```

Definition at line 44 of file window.cpp.

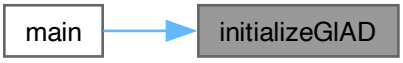
Here is the caller graph for this function:

5.27.1.5 initializeGlAD()

```
bool initializeGlAD ( )
```

Definition at line 34 of file window.cpp.

Here is the caller graph for this function:

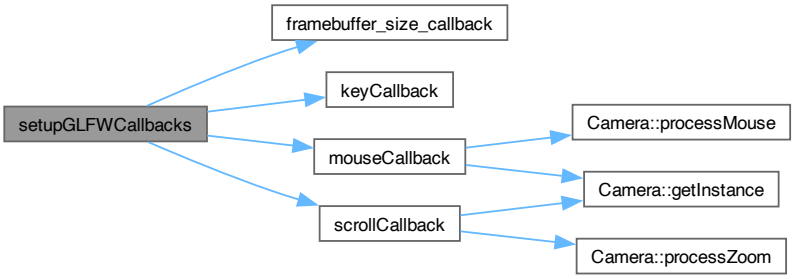


5.27.1.6 setupGLFWCallbacks()

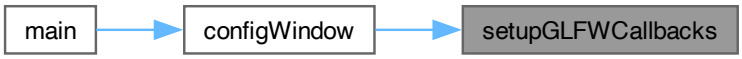
```
void setupGLFWCallbacks (
            GLFWwindow * window )
```

Definition at line 50 of file window.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.28 window.h

Go to the documentation of this file.

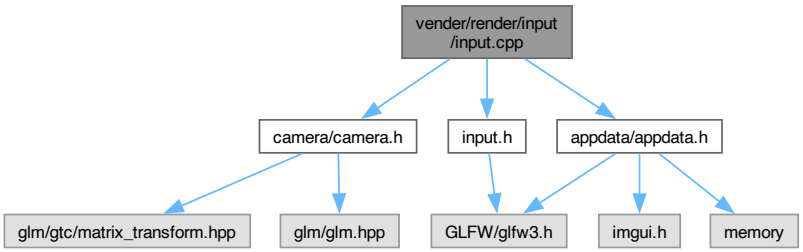
```
00001 #pragma once
00002
00003 #include <GLFW/glfw3.h>
00004
00005 GLFWwindow *createWindow(GLint scr_width = 800, GLint scr_height = 600);
00006 void configWindow(GLFWwindow *window);
00007 bool initializeGlAD();
00008 void glfwShutdown(GLFWwindow *window);
00009 void setupGLFWCallbacks(GLFWwindow *window);
00010 void framebuffer_size_callback(GLFWwindow *window, int width, int height);
```

5.29 vender/render/input/input.cpp File Reference

```
#include "input.h"
#include "camera/camera.h"
```

```
#include "appdata/appdata.h"
```

Include dependency graph for input.cpp:



Functions

- void processInput (GLFWwindow ∗window)
- void keyCallback (GLFWwindow ∗window, int key, int, int action, int)
- void mouseCallback (GLFWwindow ∗window, double xpos, double ypos)
- void scrollCallback (GLFWwindow ∗window, double, double yoffset)

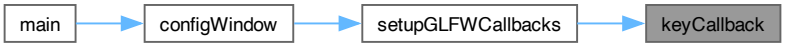
5.29.1 Function Documentation

5.29.1.1 keyCallback()

```
void keyCallback (
            GLFWwindow * window,
            int key,
            int ,
            int action,
            int  )
```

Definition at line 30 of file input.cpp.

Here is the caller graph for this function:

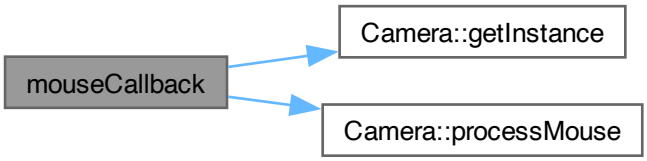


5.29.1.2 mouseCallback()

```
void mouseCallback (
            GLFWwindow * window,
            double xpos,
            double ypos )
```

Definition at line 48 of file input.cpp.

Here is the call graph for this function:



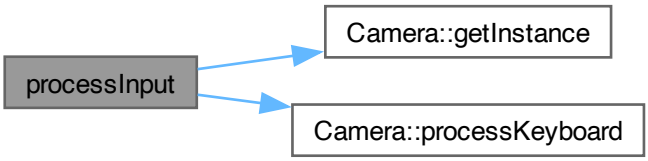
Here is the caller graph for this function:

5.29.1.3 processInput()

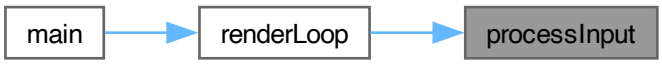
```
void processInput (
            GLFWwindow * window )
```

Definition at line 5 of file input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

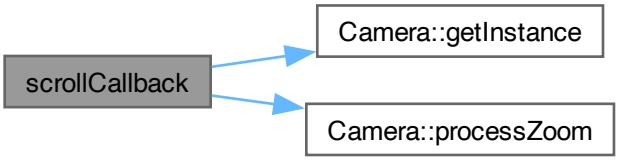


5.29.1.4 scrollCallback()

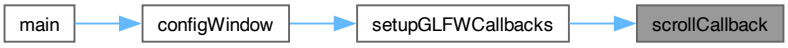
```
void scrollCallback (
            GLFWwindow * window,
            double xoffset,
            double yoffset )
```

Definition at line 74 of file input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.30 input.cpp

Go to the documentation of this file.

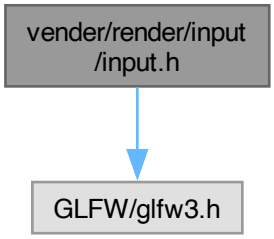
```
00001 #include "input.h"
00002 #include "camera/camera.h"
00003 #include "appdata/appdata.h"
00004
```

```
00005 void processInput(GLFWwindow *window)
00006 {
00007     using enum Direction;
00008     Camera &camera = Camera::getInstance();
00009     auto appData = (AppData *)glfwGetWindowUserPointer(window);
00010 
00011     if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
00012         glfwSetWindowShouldClose(window, true);
00013     if (glfwGetKey(window, GLFW_KEY_1) == GLFW_PRESS)
00014         glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
00015     if (glfwGetKey(window, GLFW_KEY_2) == GLFW_PRESS)
00016         glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
00017     if (glfwGetKey(window, GLFW_KEY_3) == GLFW_PRESS)
00018         glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
00019 
00020     if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
00021         camera.processKeyboard(UP, appData->deltaTime);
00022     if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
00023         camera.processKeyboard(DOWN, appData->deltaTime);
00024     if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
00025         camera.processKeyboard(LEFT, appData->deltaTime);
00026     if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
00027         camera.processKeyboard(RIGHT, appData->deltaTime);
00028 }
00029 
00030 void keyCallback(GLFWwindow *window, int key, int, int action, int)
00031 {
00032     auto appData = (AppData *)glfwGetWindowUserPointer(window);
00033     if (key == GLFW_KEY_M && action == GLFW_PRESS)
00034     {
00035         if (appData->debug_mode)
00036         {
00037             appData->debug_mode = false;
00038             glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED);
00039         }
00040         else
00041         {
00042             appData->debug_mode = true;
00043             glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_NORMAL);
00044         }
00045     }
00046 }
00047 
00048 void mouseCallback(GLFWwindow *window, double xpos, double ypos)
00049 {
00050     Camera &camera = Camera::getInstance();
00051     auto appData = (AppData *)glfwGetWindowUserPointer(window);
00052 
00053     if (appData->io.WantCaptureMouse || appData->debug_mode)
00054     {
00055         appData->firstMouse = true;
00056         return;
00057     }
00058 
00059     if (appData->firstMouse)
00060     {
00061         appData->lastX = (float)xpos;
00062         appData->lastY = (float)ypos;
00063         appData->firstMouse = false;
00064     }
00065 
00066     float xoffset = (float)xpos - appData->lastX;
00067     float yoffset = appData->lastY - (float)ypos;
00068     appData->lastX = (float)xpos;
00069     appData->lastY = (float)ypos;
00070 
00071     camera.processMouse(xoffset, yoffset);
00072 }
00073 
00074 void scrollCallback(GLFWwindow *window, double, double yoffset)
00075 {
00076     Camera &camera = Camera::getInstance();
00077 
00078     if (auto appData = (AppData *)glfwGetWindowUserPointer(window);
00079         appData->io.WantCaptureMouse || appData->debug_mode)
00080     {
00081         return;
00082     }
00083     camera.processZoom((float)yoffset);
00084 }
```

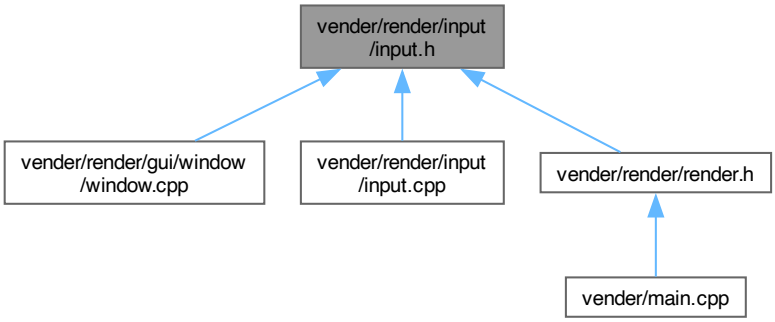
5.31 vender/render/input/input.h File Reference

```
#include <GLFW/glfw3.h>
```

Include dependency graph for input.h:



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



Functions

- void processInput (GLFWwindow ∗window)
- void keyCallback (GLFWwindow ∗window, int key, int, int action, int)
- void mouseCallback (GLFWwindow ∗, double xpos, double ypos)
- void scrollCallback (GLFWwindow ∗window, double xoffset, double yoffset)

5.31.1 Function Documentation

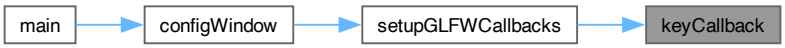
5.31.1.1 keyCallback()

```
void keyCallback (
            GLFWwindow * window,
```

```
            int key,
            int ,
            int action,
            int  )
```

Definition at line 30 of file input.cpp.

Here is the caller graph for this function:

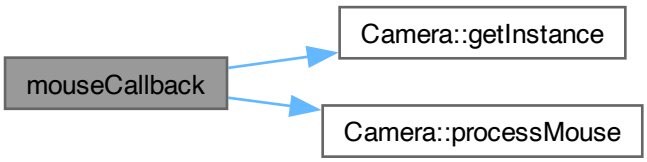


5.31.1.2 mouseCallback()

```
void mouseCallback (
            GLFWwindow * window,
            double xpos,
            double ypos )
```

Definition at line 48 of file input.cpp.

Here is the call graph for this function:



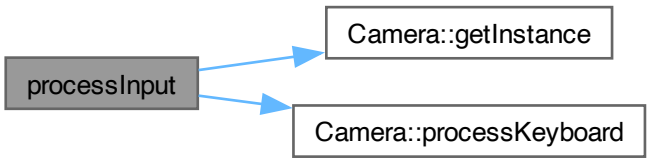
Here is the caller graph for this function:

5.31.1.3 processInput()

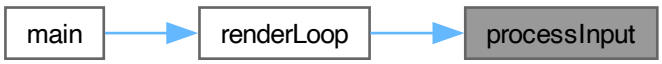
```
void processInput (
            GLFWwindow * window )
```

Definition at line 5 of file input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

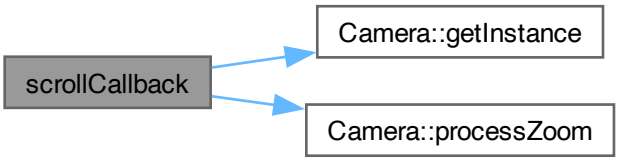


5.31.1.4 scrollCallback()

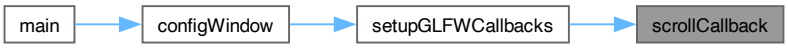
```
void scrollCallback (
            GLFWwindow * window,
            double xoffset,
            double yoffset )
```

Definition at line 74 of file input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



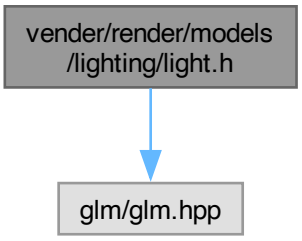
5.32 input.h

Go to the documentation of this file.

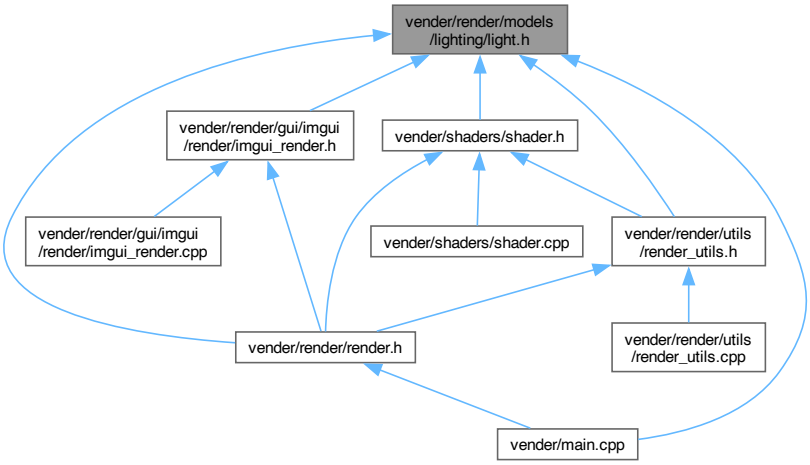
```
00001 #pragma once
00002
00003 #include <GLFW/glfw3.h>
00004
00005 void processInput(GLFWwindow *window);
00006 void keyCallback(GLFWwindow *window, int key, int, int action, int);
00007 void mouseCallback(GLFWwindow *, double xpos, double ypos);
00008 void scrollCallback(GLFWwindow *window, double xoffset, double yoffset);
```

5.33 vender/render/models/lighting/light.h File Reference

`#include <glm/glm.hpp>`
Include dependency graph for light.h:



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



Classes

- struct Light

5.34 light.h

Go to the documentation of this file.

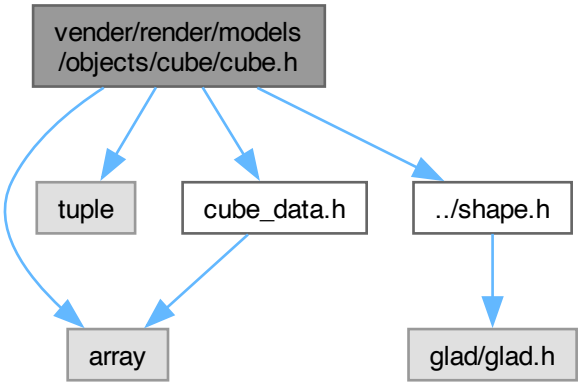
```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 struct Light
00006 {
00007     glm::vec3 pos = glm::vec3(1.0f, 0.17f, 1.6f);
00008     glm::vec3 color = glm::vec3(1.0f, 1.0f, 1.0f);
00009     float ambient = 0.2f;
00010     float diffuse = 0.5f;
00011     float specular = 1.0f;
00012 };
```

5.35 vender/render/models/objects/cube/cube.h File Reference

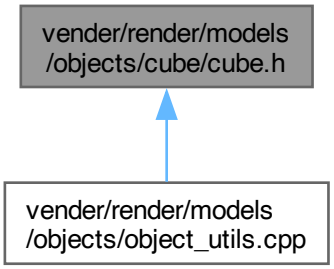
```
#include <array>
#include <tuple>
#include "../shape.h"
```

```
#include "cube_data.h"
```

Include dependency graph for cube.h:



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



Classes

- class CubeDefault
- class CubeNorm
- class CubeTex

5.36 cube.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <array>
```

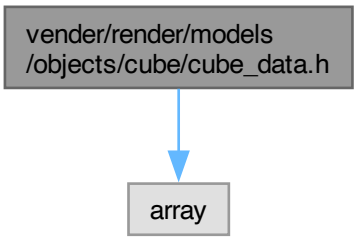
```
00004 #include <tuple>
00005
00006 #include "../shape.h"
00007 #include "cube_data.h"
00008
00009 class CubeDefault : public AbstractShape
00010 {
00011 public:
00012     CubeDefault()
00013     {
00014         setupBuffers();
00015     }
00016
00017     void render() const override
00018     {
00019         glBindVertexArray(VAO);
00020         glDrawArrays(GL_TRIANGLES, 0, 36);
00021     }
00022     ~CubeDefault() override
00023     {
00024         glDeleteBuffers(1, &VBO);
00025         glDeleteVertexArrays(1, &VAO);
00026     };
00027
00028 protected:
00029     size_t getVertexDataSize() const override
00030     {
00031         return CubeData::vertPosSize;
00032     }
00033
00034     void setupVAO(unsigned int _VAO) override
00035     {
00036         VAO = _VAO;
00037         glBindVertexArray(VAO);
00038         enableVertexAttribute(0, 3, 3 * sizeof(float), 0);
00039     }
00040
00041     void setupVBO(unsigned int _VBO) override
00042     {
00043         VBO = _VBO;
00044         glBufferSubData(GL_ARRAY_BUFFER, 0, CubeData::vertPosSize, CubeData::vertPos.data());
00045     }
00046
00047 private:
00048     unsigned int VBO;
00049     unsigned int VAO;
00050 };
00051
00052 class CubeNorm : public AbstractShape
00053 {
00054 public:
00055     CubeNorm()
00056     {
00057         setupBuffers();
00058     }
00059
00060     void render() const override
00061     {
00062         glBindVertexArray(VAO);
00063         glDrawArrays(GL_TRIANGLES, 0, 36);
00064     }
00065
00066     ~CubeNorm() override
00067     {
00068         glDeleteBuffers(1, &VBO);
00069         glDeleteVertexArrays(1, &VAO);
00070     };
00071
00072 protected:
00073     size_t getVertexDataSize() const override
00074     {
00075         return CubeData::vertPosSize + CubeData::vertNormSize + CubeData::texCoordSize;
00076     }
00077
00078     void setupVAO(unsigned int _VAO) override
00079     {
00080         VAO = _VAO;
00081         glBindVertexArray(VAO);
00082         enableVertexAttribute(0, 3, 3 * sizeof(float), 0);
00083         enableVertexAttribute(1, 3, 3 * sizeof(float), CubeData::vertPosSize);
00084     }
00085
00086     void setupVBO(unsigned int _VBO) override
00087     {
00088         VBO = _VBO;
00089         glBufferSubData(GL_ARRAY_BUFFER, 0, CubeData::vertPosSize, CubeData::vertPos.data());
00090         glBufferSubData(GL_ARRAY_BUFFER, CubeData::vertPosSize, CubeData::vertNormSize,
```

```
      CubeData::vertNorm.data());
00091     }
00092
00093 private:
00094     unsigned int VBO;
00095     unsigned int VAO;
00096 };
00097
00098 class CubeTex : public AbstractShape
00099 {
00100 public:
00101     CubeTex()
00102     {
00103         setupBuffers();
00104     }
00105
00106     void render() const override
00107     {
00108         glBindVertexArray(VAO);
00109         glDrawArrays(GL_TRIANGLES, 0, 36);
00110     }
00111
00112     ~CubeTex() override
00113     {
00114         glDeleteBuffers(1, &VBO);
00115         glDeleteVertexArrays(1, &VAO);
00116     };
00117
00118 protected:
00119     size_t getVertexDataSize() const override
00120     {
00121         return CubeData::vertPosSize + CubeData::vertNormSize + CubeData::texCoordSize;
00122     }
00123
00124     void setupVAO(unsigned int _VAO) override
00125     {
00126         VAO = _VAO;
00127         glBindVertexArray(VAO);
00128         enableVertexAttribute(0, 3, 3 * sizeof(float), 0);
00129         enableVertexAttribute(1, 3, 3 * sizeof(float), CubeData::vertPosSize);
00130         enableVertexAttribute(2, 2, 2 * sizeof(float), CubeData::vertPosSize +
      CubeData::vertNormSize);
00131     }
00132
00133     void setupVBO(unsigned int _VBO) override
00134     {
00135         VBO = _VBO;
00136         glBufferSubData(GL_ARRAY_BUFFER, 0, CubeData::vertPosSize, CubeData::vertPos.data());
00137         glBufferSubData(GL_ARRAY_BUFFER, CubeData::vertPosSize, CubeData::vertNormSize,
      CubeData::vertNorm.data());
00138         glBufferSubData(GL_ARRAY_BUFFER, CubeData::vertPosSize + CubeData::vertNormSize,
      CubeData::texCoordSize, CubeData::texCoords.data());
00139     }
00140
00141 private:
00142     unsigned int VBO;
00143     unsigned int VAO;
00144 };
```

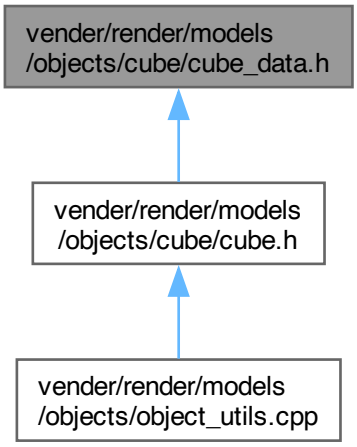
5.37 vender/render/models/objects/cube/cube_data.h File Reference

```
#include <array>
```

Include dependency graph for cube_data.h:



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



Classes

- struct CubeData

5.38 cube_data.h

Go to the documentation of this file.

```
00001 #pragma once
```

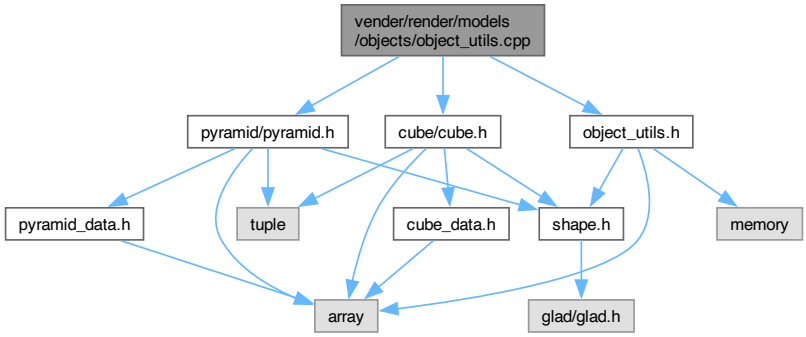
```
00002
00003 #include <array>
00004
00005 struct CubeData
00006 {
00007     static constexpr std::array<float, 108> vertPos = {
00008         -0.5f, -0.5f, -0.5f,
00009         0.5f, -0.5f, -0.5f,
00010         0.5f, 0.5f, -0.5f,
00011         0.5f, 0.5f, -0.5f,
00012         -0.5f, 0.5f, -0.5f,
00013         -0.5f, -0.5f, -0.5f,
00014
00015         -0.5f, -0.5f, 0.5f,
00016         0.5f, -0.5f, 0.5f,
00017         0.5f, 0.5f, 0.5f,
00018         0.5f, 0.5f, 0.5f,
00019         -0.5f, 0.5f, 0.5f,
00020         -0.5f, -0.5f, 0.5f,
00021
00022         -0.5f, 0.5f, 0.5f,
00023         -0.5f, 0.5f, -0.5f,
00024         -0.5f, -0.5f, -0.5f,
00025         -0.5f, -0.5f, -0.5f,
00026         -0.5f, -0.5f, 0.5f,
00027         -0.5f, 0.5f, 0.5f,
00028
00029         0.5f, 0.5f, 0.5f,
00030         0.5f, 0.5f, -0.5f,
00031         0.5f, -0.5f, -0.5f,
00032         0.5f, -0.5f, -0.5f,
00033         0.5f, -0.5f, 0.5f,
00034         0.5f, 0.5f, 0.5f,
00035
00036         -0.5f, -0.5f, -0.5f,
00037         0.5f, -0.5f, -0.5f,
00038         0.5f, -0.5f, 0.5f,
00039         0.5f, -0.5f, 0.5f,
00040         -0.5f, -0.5f, 0.5f,
00041         -0.5f, -0.5f, -0.5f,
00042
00043         -0.5f, 0.5f, -0.5f,
00044         0.5f, 0.5f, -0.5f,
00045         0.5f, 0.5f, 0.5f,
00046         0.5f, 0.5f, 0.5f,
00047         -0.5f, 0.5f, 0.5f,
00048         -0.5f, 0.5f, -0.5f};
00049
00050     static constexpr std::array<float, 108> vertNorm = {
00051         0.0f, 0.0f, -1.0f,
00052         0.0f, 0.0f, -1.0f,
00053         0.0f, 0.0f, -1.0f,
00054         0.0f, 0.0f, -1.0f,
00055         0.0f, 0.0f, -1.0f,
00056         0.0f, 0.0f, -1.0f,
00057
00058         0.0f, 0.0f, 1.0f,
00059         0.0f, 0.0f, 1.0f,
00060         0.0f, 0.0f, 1.0f,
00061         0.0f, 0.0f, 1.0f,
00062         0.0f, 0.0f, 1.0f,
00063         0.0f, 0.0f, 1.0f,
00064
00065         -1.0f, 0.0f, 0.0f,
00066         -1.0f, 0.0f, 0.0f,
00067         -1.0f, 0.0f, 0.0f,
00068         -1.0f, 0.0f, 0.0f,
00069         -1.0f, 0.0f, 0.0f,
00070         -1.0f, 0.0f, 0.0f,
00071
00072         1.0f, 0.0f, 0.0f,
00073         1.0f, 0.0f, 0.0f,
00074         1.0f, 0.0f, 0.0f,
00075         1.0f, 0.0f, 0.0f,
00076         1.0f, 0.0f, 0.0f,
00077         1.0f, 0.0f, 0.0f,
00078
00079         0.0f, -1.0f, 0.0f,
00080         0.0f, -1.0f, 0.0f,
00081         0.0f, -1.0f, 0.0f,
00082         0.0f, -1.0f, 0.0f,
00083         0.0f, -1.0f, 0.0f,
00084         0.0f, -1.0f, 0.0f,
00085
00086         0.0f, 1.0f, 0.0f,
00087         0.0f, 1.0f, 0.0f,
00088         0.0f, 1.0f, 0.0f,
```

```
00089         0.0f, 1.0f, 0.0f,
00090         0.0f, 1.0f, 0.0f,
00091         0.0f, 1.0f, 0.0f};
00092
00093     static constexpr std::array<float, 72> texCoords = {
00094         0.0f, 0.0f,
00095         1.0f, 0.0f,
00096         1.0f, 1.0f,
00097         1.0f, 1.0f,
00098         0.0f, 1.0f,
00099         0.0f, 0.0f,
00100
00101         0.0f, 0.0f,
00102         1.0f, 0.0f,
00103         1.0f, 1.0f,
00104         1.0f, 1.0f,
00105         0.0f, 1.0f,
00106         0.0f, 0.0f,
00107
00108         1.0f, 0.0f,
00109         1.0f, 1.0f,
00110         0.0f, 1.0f,
00111         0.0f, 1.0f,
00112         0.0f, 0.0f,
00113         1.0f, 0.0f,
00114
00115         1.0f, 0.0f,
00116         1.0f, 1.0f,
00117         0.0f, 1.0f,
00118         0.0f, 1.0f,
00119         0.0f, 0.0f,
00120         1.0f, 0.0f,
00121
00122         0.0f, 1.0f,
00123         1.0f, 1.0f,
00124         1.0f, 0.0f,
00125         1.0f, 0.0f,
00126         0.0f, 0.0f,
00127         0.0f, 1.0f,
00128
00129         0.0f, 1.0f,
00130         1.0f, 1.0f,
00131         1.0f, 0.0f,
00132         1.0f, 0.0f,
00133         0.0f, 0.0f,
00134         0.0f, 1.0f};
00135
00136     static constexpr size_t vertPosSize = sizeof(vertPos);
00137     static constexpr size_t vertNormSize = sizeof(vertNorm);
00138     static constexpr size_t texCoordSize = sizeof(texCoords);
00139 };
```

5.39 vender/render/models/objects/object_utils.cpp File Reference

```
#include "object_utils.h"
#include "cube/cube.h"
#include "pyramid/pyramid.h"
```

Include dependency graph for object_utils.cpp:



Functions

- std::array< std::unique_ptr< AbstractShape >, 5 > createObjects ()

5.39.1 Function Documentation

5.39.1.1 createObjects()

```
std::array< std::unique_ptr< AbstractShape >, 5 > createObjects ( )
```

Definition at line 6 of file object_utils.cpp.

Here is the caller graph for this function:

5.40 object_utils.cpp

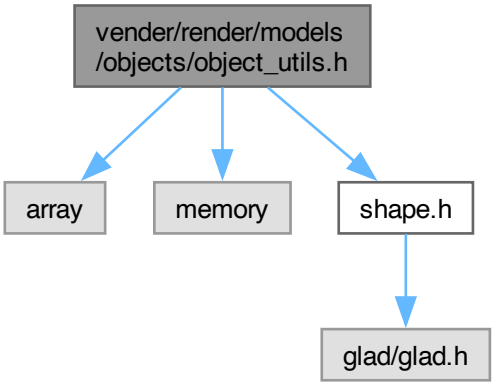
Go to the documentation of this file.

```
00001 #include "object_utils.h"
00002
00003 #include "cube/cube.h"
00004 #include "pyramid/pyramid.h"
00005
00006 std::array<std::unique_ptr<AbstractShape>, 5> createObjects()
00007 {
00008     std::array<std::unique_ptr<AbstractShape>, 5> objects = {
00009         std::make_unique<CubeNorm>(),
00010         std::make_unique<CubeTex>(),
00011         std::make_unique<PyramidNorm>(),
00012         std::make_unique<PyramidTex>(),
00013         std::make_unique<CubeDefault>()};
00014     return objects;
00015 };
```

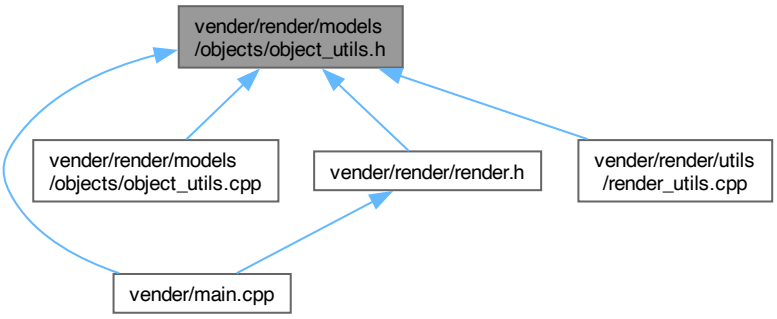
5.41 vender/render/models/objects/object_utils.h File Reference

```
#include <array>
#include <memory>
#include "shape.h"
```

Include dependency graph for object_utils.h:



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



Enumerations

- enum struct ObjectIdx {
  cubeNorm = 0 , cubeTex = 1 , pyramidNorm = 2 , pyramidTex = 3 ,
  lightCube = 4 }

Functions

- std::array< std::unique_ptr< AbstractShape >, 5 > createObjects ()

5.41.1 Enumeration Type Documentation

5.41.1.1 ObjectIdx

```
enum struct ObjectIdx  [strong]
```

Enumerator

cubeNorm	
cubeTex	
pyramidNorm	
pyramidTex	
lightCube	

Definition at line 8 of file object_utils.h.

5.41.2 Function Documentation

5.41.2.1 createObjects()

```
std::array< std::unique_ptr< AbstractShape >, 5 > createObjects ( )
```

Definition at line 6 of file object_utils.cpp.

Here is the caller graph for this function:

5.42 object_utils.h

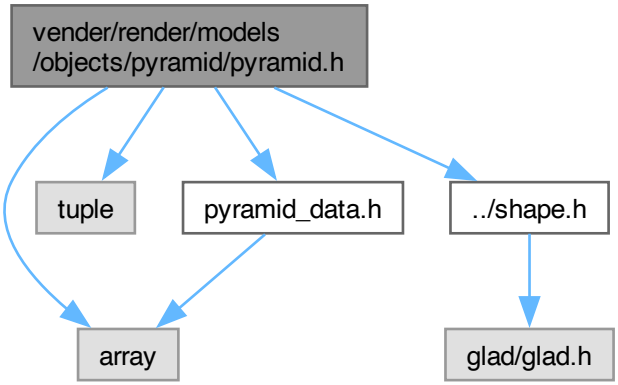
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <array>
00004 #include <memory>
00005
00006 #include "shape.h"
00007
00008 enum struct ObjectIdx
00009 {
00010     cubeNorm = 0,
00011     cubeTex = 1,
00012     pyramidNorm = 2,
00013     pyramidTex = 3,
00014     lightCube = 4,
00015 };
00016
00017 std::array<std::unique_ptr<AbstractShape>, 5> createObjects();
```

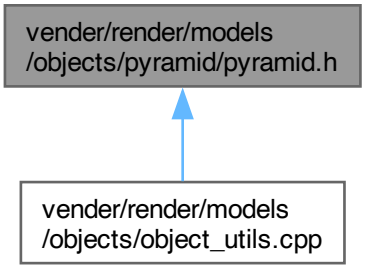
5.43 vender/render/models/objects/pyramid/pyramid.h File Reference

```
#include <array>
#include <tuple>
#include "../shape.h"
#include "pyramid_data.h"
```

Include dependency graph for pyramid.h:



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



Classes

- class PyramidDefault
- class PyramidNorm
- class PyramidTex

5.44 pyramid.h

Go to the documentation of this file.

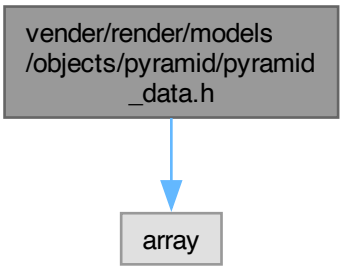
```
00001 #pragma once
00002
00003 #include <array>
00004 #include <tuple>
00005
00006 #include "../shape.h"
00007 #include "pyramid_data.h"
00008
00009 class PyramidDefault : public AbstractShape
00010 {
00011 public:
00012     PyramidDefault()
00013     {
00014         setupBuffers();
00015     }
00016
00017     void render() const override
00018     {
00019         glBindVertexArray(VAO);
00020         glDrawArrays(GL_TRIANGLES, 0, 18);
00021     }
00022
00023     ~PyramidDefault() override
00024     {
00025         glDeleteBuffers(1, &VBO);
00026         glDeleteVertexArrays(1, &VAO);
00027     }
00028
00029 protected:
00030     size_t getVertexDataSize() const override
00031     {
00032         return PyramidData::vertPosSize;
00033     }
00034
00035     void setupVAO(unsigned int _VAO) override
00036     {
00037         VAO = _VAO;
00038         glBindVertexArray(VAO);
00039         enableVertexAttribute(0, 3, 3 * sizeof(float), 0);
00040     }
00041
00042     void setupVBO(unsigned int _VBO) override
00043     {
00044         VBO = _VBO;
00045         glBufferSubData(GL_ARRAY_BUFFER, 0, PyramidData::vertPosSize, PyramidData::vertPos.data());
00046     }
00047
00048 private:
00049     unsigned int VBO;
00050     unsigned int VAO;
00051 };
00052
00053 class PyramidNorm : public AbstractShape
00054 {
00055 public:
00056     PyramidNorm()
00057     {
00058         setupBuffers();
00059     }
00060
00061     void render() const override
00062     {
00063         glBindVertexArray(VAO);
00064         glDrawArrays(GL_TRIANGLES, 0, 18);
00065     }
00066
00067     ~PyramidNorm() override
00068     {
00069         glDeleteBuffers(1, &VBO);
00070         glDeleteVertexArrays(1, &VAO);
00071     }
00072
00073 protected:
00074     size_t getVertexDataSize() const override
00075     {
00076         return PyramidData::vertPosSize + PyramidData::vertNormSize + PyramidData::texCoordSize;
00077     }
00078
00079     void setupVAO(unsigned int _VAO) override
00080     {
00081         VAO = _VAO;
00082         glBindVertexArray(VAO);
```

```
00083         enableVertexAttribute(0, 3, 3 * sizeof(float), 0);
00084         enableVertexAttribute(1, 3, 3 * sizeof(float), PyramidData::vertPosSize);
00085     }
00086
00087     void setupVBO(unsigned int _VBO) override
00088     {
00089         VBO = _VBO;
00090         glBufferSubData(GL_ARRAY_BUFFER, 0, PyramidData::vertPosSize, PyramidData::vertPos.data());
00091         glBufferSubData(GL_ARRAY_BUFFER, PyramidData::vertPosSize, PyramidData::vertNormSize,
      PyramidData::vertNorm.data());
00092     }
00093
00094 private:
00095     unsigned int VBO;
00096     unsigned int VAO;
00097 };
00098
00099 class PyramidTex : public AbstractShape
00100 {
00101 public:
00102     PyramidTex()
00103     {
00104         setupBuffers();
00105     }
00106
00107     void render() const override
00108     {
00109         glBindVertexArray(VAO);
00110         glDrawArrays(GL_TRIANGLES, 0, 18);
00111     }
00112
00113     ~PyramidTex() override
00114     {
00115         glDeleteBuffers(1, &VBO);
00116         glDeleteVertexArrays(1, &VAO);
00117     }
00118
00119 protected:
00120     size_t getVertexDataSize() const override
00121     {
00122         return PyramidData::vertPosSize + PyramidData::vertNormSize + PyramidData::texCoordSize;
00123     }
00124
00125     void setupVAO(unsigned int _VAO) override
00126     {
00127         VAO = _VAO;
00128         glBindVertexArray(VAO);
00129         enableVertexAttribute(0, 3, 3 * sizeof(float), 0);
00130         enableVertexAttribute(1, 3, 3 * sizeof(float), PyramidData::vertPosSize);
00131         enableVertexAttribute(2, 2, 2 * sizeof(float), PyramidData::vertPosSize +
      PyramidData::vertNormSize);
00132     }
00133
00134     void setupVBO(unsigned int _VBO) override
00135     {
00136         VBO = _VBO;
00137         glBufferSubData(GL_ARRAY_BUFFER, 0, PyramidData::vertPosSize, PyramidData::vertPos.data());
00138         glBufferSubData(GL_ARRAY_BUFFER, PyramidData::vertPosSize, PyramidData::vertNormSize,
      PyramidData::vertNorm.data());
00139         glBufferSubData(GL_ARRAY_BUFFER, PyramidData::vertPosSize + PyramidData::vertNormSize,
      PyramidData::texCoordSize, PyramidData::texCoords.data());
00140     }
00141
00142 private:
00143     unsigned int VBO;
00144     unsigned int VAO;
00145 };
```

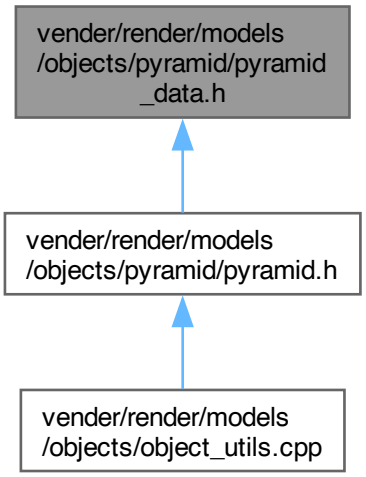
5.45 vender/render/models/objects/pyramid/pyramid_data.h File Reference

```
#include <array>
```

Include dependency graph for pyramid_data.h:



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



Classes

- struct PyramidData

5.46 pyramid_data.h

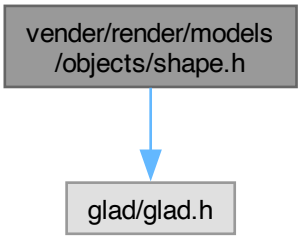
Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <array>
00004
00005 struct PyramidData
00006 {
00007     static constexpr std::array<float, 54> vertPos = {
00008         0.0f, 0.5f, 0.0f,
00009         -0.5f, -0.5f, 0.5f,
00010         0.5f, -0.5f, 0.5f,
00011
00012         0.0f, 0.5f, 0.0f,
00013         0.5f, -0.5f, 0.5f,
00014         0.5f, -0.5f, -0.5f,
00015
00016         0.0f, 0.5f, 0.0f,
00017         0.5f, -0.5f, -0.5f,
00018         -0.5f, -0.5f, -0.5f,
00019
00020         0.0f, 0.5f, 0.0f,
00021         -0.5f, -0.5f, -0.5f,
00022         -0.5f, -0.5f, 0.5f,
00023
00024         -0.5f, -0.5f, -0.5f,
00025         0.5f, -0.5f, -0.5f,
00026         0.5f, -0.5f, 0.5f,
00027
00028         0.5f, -0.5f, 0.5f,
00029         -0.5f, -0.5f, 0.5f,
00030         -0.5f, -0.5f, -0.5f};
00031
00032     static constexpr std::array<float, 54> vertNorm = {
00033         0.0f, 0.71f, 0.71f,
00034         0.0f, 0.71f, 0.71f,
00035         0.0f, 0.71f, 0.71f,
00036
00037         0.71f, 0.71f, 0.0f,
00038         0.71f, 0.71f, 0.0f,
00039         0.71f, 0.71f, 0.0f,
00040
00041         0.0f, 0.71f, -0.71f,
00042         0.0f, 0.71f, -0.71f,
00043         0.0f, 0.71f, -0.71f,
00044
00045         -0.71f, 0.71f, 0.0f,
00046         -0.71f, 0.71f, 0.0f,
00047         -0.71f, 0.71f, 0.0f,
00048
00049         0.0f, -1.0f, 0.0f,
00050         0.0f, -1.0f, 0.0f,
00051         0.0f, -1.0f, 0.0f,
00052
00053         0.0f, -1.0f, 0.0f,
00054         0.0f, -1.0f, 0.0f,
00055         0.0f, -1.0f, 0.0f};
00056
00057     static constexpr std::array<float, 36> texCoords = {
00058         0.5f, 1.0f,
00059         0.0f, 0.0f,
00060         1.0f, 0.0f,
00061
00062         0.5f, 1.0f,
00063         0.0f, 0.0f,
00064         1.0f, 0.0f,
00065
00066         0.5f, 1.0f,
00067         0.0f, 0.0f,
00068         1.0f, 0.0f,
00069
00070         0.5f, 1.0f,
00071         0.0f, 0.0f,
00072         1.0f, 0.0f,
00073
00074         0.0f, 1.0f,
00075         1.0f, 1.0f,
00076         1.0f, 0.0f,
00077         1.0f, 0.0f,
00078         0.0f, 0.0f,
00079         0.0f, 1.0f};
00080
00081     static constexpr size_t vertPosSize = sizeof(vertPos);
00082     static constexpr size_t vertNormSize = sizeof(vertNorm);
```

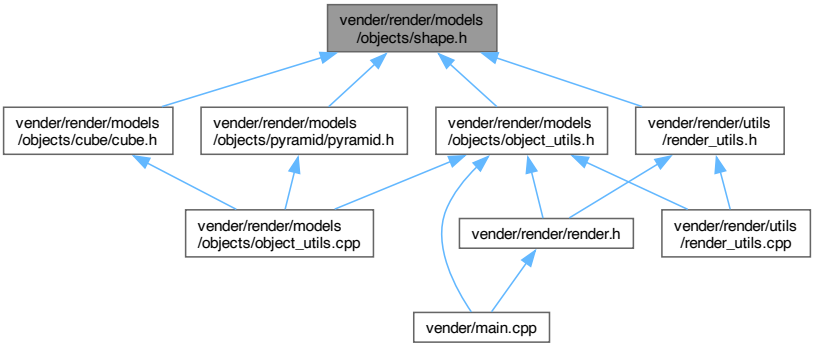
```
00083     static constexpr size_t texCoordSize = sizeof(texCoords);
00084 };
```

5.47 vender/render/models/objects/shape.h File Reference

`#include <glad/glad.h>`
Include dependency graph for shape.h:



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



Classes

- class AbstractShape

5.48 shape.h

Go to the documentation of this file.

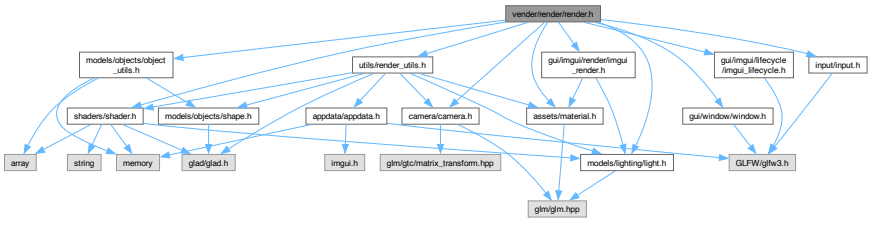
```
00001 #pragma once
00002
00003 #include <glad/glad.h>
```

```
00004 class AbstractShape
00005 {
00006 public:
00007     virtual void render() const = 0;
00008     virtual ~AbstractShape() = default;
00009
00010 protected:
00011     void setupBuffers()
00012     {
00013         const auto BUFFER_SIZE = getVertexDataSize();
00014         auto [VAO, VBO] = reserveVertexMemory(BUFFER_SIZE);
00015         setupVAO(VAO);
00016         setupVBO(VBO);
00017     };
00018
00019     virtual size_t getVertexDataSize() const = 0;
00020
00021     std::tuple<unsigned int, unsigned int> reserveVertexMemory(size_t BUFFER_SIZE)
00022     {
00023         unsigned int VAO;
00024         unsigned int VBO;
00025         glGenVertexArrays(1, &VAO);
00026         glGenBuffers(1, &VBO);
00027         glBindBuffer(GL_ARRAY_BUFFER, VBO);
00028         glBufferData(GL_ARRAY_BUFFER, BUFFER_SIZE, nullptr, GL_STATIC_DRAW);
00029         return {VAO, VBO};
00030     }
00031
00032     virtual void setupVAO(unsigned int VAO) = 0;
00033     virtual void setupVBO(unsigned int VBO) = 0;
00034
00035     void enableVertexAttribute(unsigned int index, unsigned int numComponents, size_t stride, size_t
      offset) const
00036     {
00037         glVertexAttribPointer(index, numComponents, GL_FLOAT, GL_FALSE, stride, (GLvoid *)offset);
00038         glEnableVertexAttribArray(index);
00039     }
00040 };
```

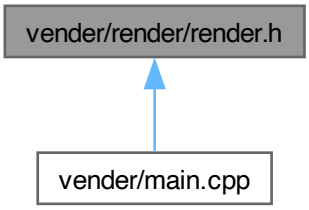
5.49 vender/render/render.h File Reference

```
#include "utils/render_utils.h"
#include "gui/window/window.h"
#include "gui/imgui/render/imgui_render.h"
#include "gui/imgui/lifecycle/imgui_lifecycle.h"
#include "input/input.h"
#include "camera/camera.h"
#include "shaders/shader.h"
#include "assets/material.h"
#include "models/objects/object_utils.h"
#include "models/lighting/light.h"
```

Include dependency graph for render.h:



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



Functions

- void renderLoop (GLFWwindow *window, AppData &appData, const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const std::array< std::unique_ptr< Shader >, 3 > &shaders, int &selectedShape, Material &material, int &selectedMaterial, Light &light, const ImVec4 &clear_color, unsigned int diffuseMap, unsigned int specularMap)

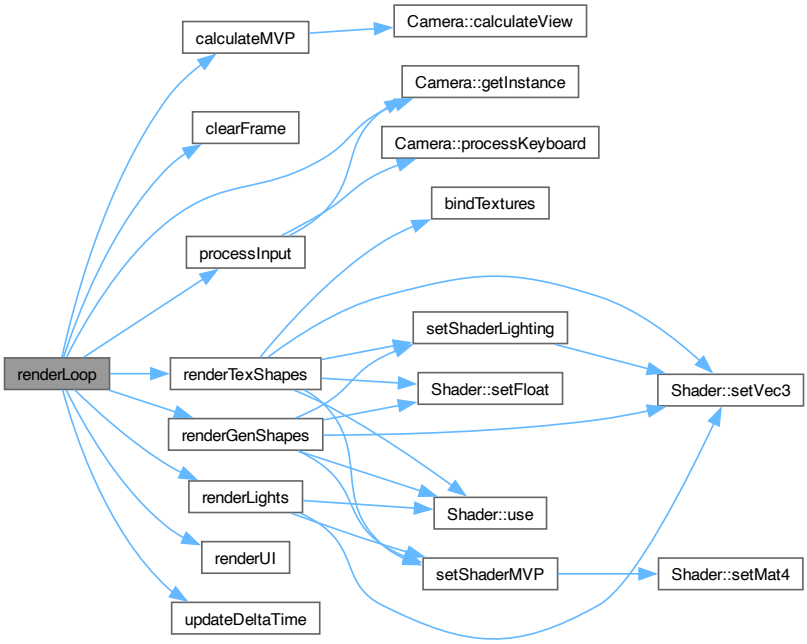
5.49.1 Function Documentation

5.49.1.1 renderLoop()

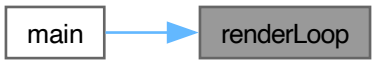
```
void renderLoop (
            GLFWwindow * window,
            AppData & appData,
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const std::array< std::unique_ptr< Shader >, 3 > & shaders,
            int & selectedShape,
            Material & material,
            int & selectedMaterial,
            Light & light,
            const ImVec4 & clear_color,
            unsigned int diffuseMap,
            unsigned int specularMap )
```

Definition at line 14 of file render.h.

Here is the call graph for this function:



Here is the caller graph for this function:



5.50 render.h

Go to the documentation of this file.

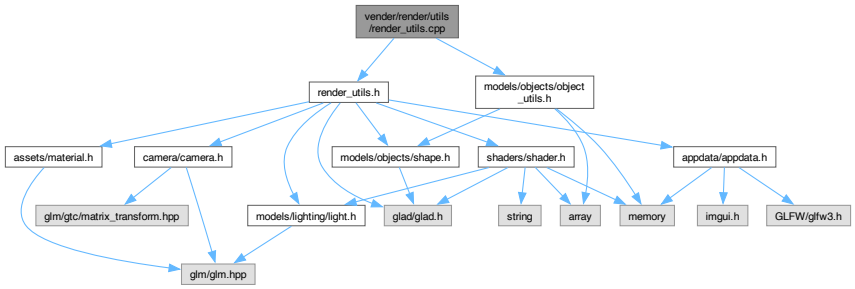
```
00001 #pragma once
00002
00003 #include "utils/render_utils.h"
00004 #include "gui/window/window.h"
00005 #include "gui/imgui/render/imgui_render.h"
00006 #include "gui/imgui/lifecycle/imgui_lifecycle.h"
00007 #include "input/input.h"
00008 #include "camera/camera.h"
00009 #include "shaders/shader.h"
00010 #include "assets/material.h"
00011 #include "models/objects/object_utils.h"
00012 #include "models/lighting/light.h"
00013
00014 void renderLoop(GLFWwindow *window, AppData &appData,
00015                 const std::array<std::unique_ptr<AbstractShape>, 5> &objects,
00016                 const std::array<std::unique_ptr<Shader>, 3> &shaders,
00017                 int &selectedShape,
```

```
00018                 Material &material,
00019                 int &selectedMaterial,
00020                 Light &light,
00021                 const ImVec4 &clear_color,
00022                 unsigned int diffuseMap,
00023                 unsigned int specularMap)
00024 {
00025     while (!glfwWindowShouldClose(window))
00026     {
00027         const Camera &camera = Camera::getInstance();
00028         updateDeltaTime(appData);
00029         clearFrame(clear_color);
00030         processInput(window);
00031 
00032         auto MVP = calculateMVP(camera, (float)appData.framebufferWidth /
      (float)appData.framebufferHeight);
00033         renderLights(objects, *shaders[(size_t)ShaderIdx::light], MVP, light);
00034         if (selectedMaterial < 2)
00035         {
00036             renderGenShapes(objects, selectedShape, *shaders[(size_t)ShaderIdx::generic], camera,
      material, MVP, light);
00037         }
00038         else
00039         {
00040             renderTexShapes(objects, selectedShape, *shaders[(size_t)ShaderIdx::tex], camera,
      material, MVP, light, diffuseMap, specularMap);
00041         }
00042 
00043         renderUI(appData.io.Framerate, light, material, selectedMaterial, selectedShape);
00044         glfwSwapBuffers(window);
00045         glfwPollEvents();
00046     }
00047 }
```

5.51 vender/render/utils/render_utils.cpp File Reference

```
#include "render_utils.h"
#include "models/objects/object_utils.h"
```

Include dependency graph for render_utils.cpp:



Functions

- void renderGenShapes (const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const int &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const std::array< glm::mat4, 3 > &MVP, const Light &light)
- void renderTexShapes (const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const int &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const std::array< glm::mat4, 3 > &MVP, const Light &light, const unsigned int &diffuseMap, const unsigned int &specularMap)
- void renderLights (const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const Shader &shader, const std::array< glm::mat4, 3 > &MVP, const Light &light)
- void clearFrame (const ImVec4 &clear_color)
- std::array< glm::mat4, 3 > calculateMVP (const Camera &camera, float ratio, const glm::vec3 &pos, float scale)
- void updateDeltaTime (AppData &appData)

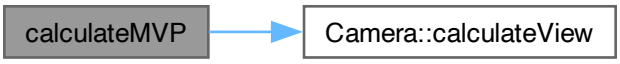
5.51.1 Function Documentation

5.51.1.1 calculateMVP()

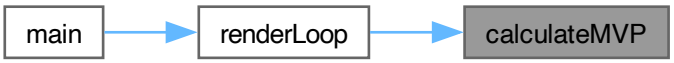
```
std::array< glm::mat4, 3 > calculateMVP (
            const Camera & camera,
            float ratio,
            const glm::vec3 & pos,
            float scale )
```

Definition at line 66 of file render_utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

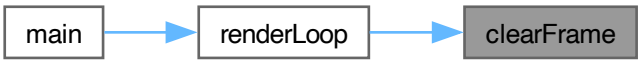


5.51.1.2 clearFrame()

```
void clearFrame (
            const ImVec4 & clear_color )
```

Definition at line 60 of file render_utils.cpp.

Here is the caller graph for this function:

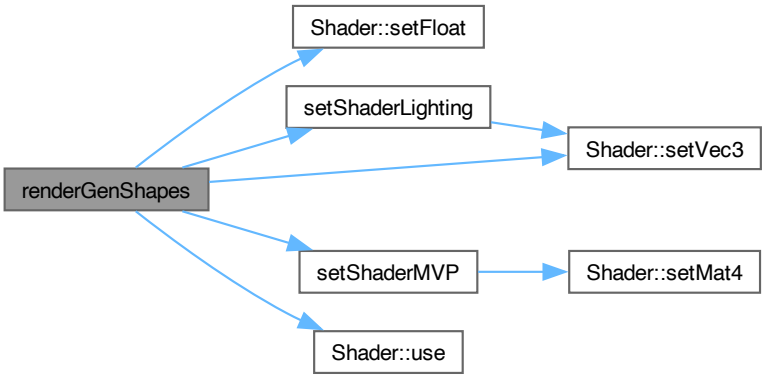


5.51.1.3 renderGenShapes()

```
void renderGenShapes (
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const int & selectedShape,
            const Shader & shader,
            const Camera & camera,
            const Material & material,
            const std::array< glm::mat4, 3 > & MVP,
            const Light & light )
```

Definition at line 4 of file render_utils.cpp.

Here is the call graph for this function:



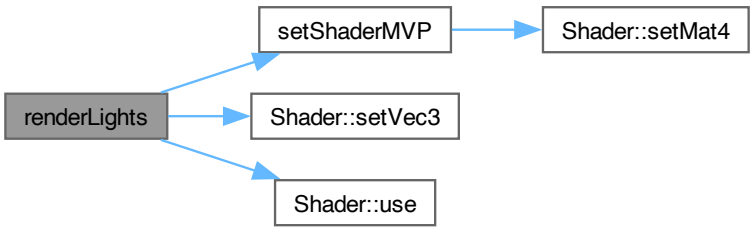
Here is the caller graph for this function:

5.51.1.4 renderLights()

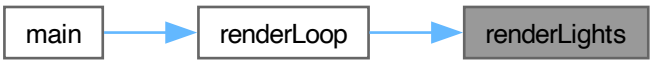
```
void renderLights (
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const Shader & shader,
            const std::array< glm::mat4, 3 > & MVP,
            const Light & light )
```

Definition at line 48 of file render_utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

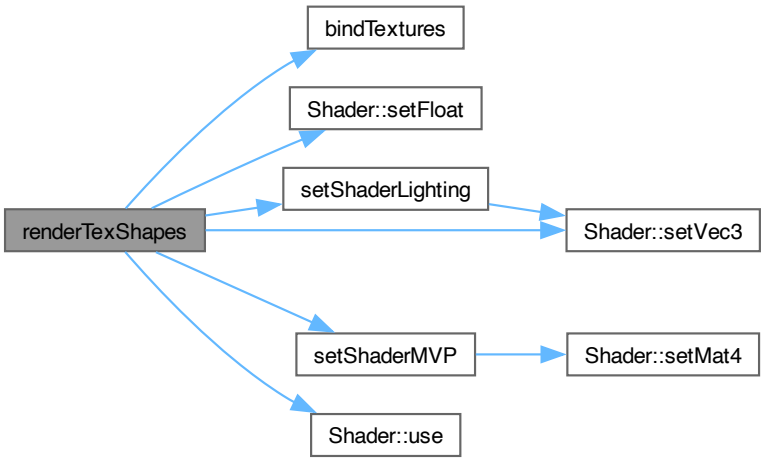


5.51.1.5 renderTexShapes()

```
void renderTexShapes (
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const int & selectedShape,
            const Shader & shader,
            const Camera & camera,
            const Material & material,
            const std::array< glm::mat4, 3 > & MVP,
            const Light & light,
            const unsigned int & diffuseMap,
            const unsigned int & specularMap )
```

Definition at line 26 of file render_utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.51.1.6 updateDeltaTime()

```
void updateDeltaTime (
            AppData & appData )
```

Definition at line 79 of file render_utils.cpp.

Here is the caller graph for this function:

5.52 render_utils.cpp

Go to the documentation of this file.

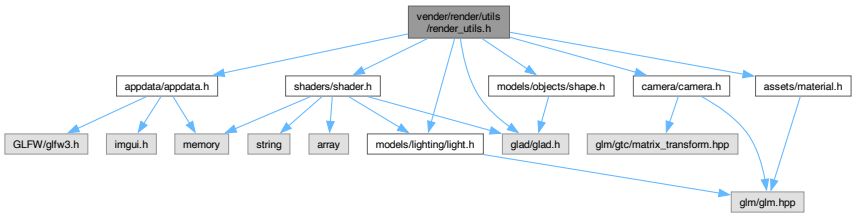
```
00001 #include "render_utils.h"
00002 #include "models/objects/object_utils.h"
00003
00004 void renderGenShapes(const std::array<std::unique_ptr<AbstractShape>, 5> &objects, const int
      &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const
      std::array<glm::mat4, 3> &MVP, const Light &light)
00005 {
00006     shader.use();
00007     setShaderLighting(shader, light);
00008     shader.setVec3("viewPos", camera.cameraPos);
00009     setShaderMVP(shader, MVP[0], MVP[1], MVP[2]);
00010
00011     shader.setVec3("material.ambient", material.ambient);
00012     shader.setVec3("material.diffuse", material.diffuse);
00013     shader.setVec3("material.specular", material.specular);
00014     shader.setFloat("material.shininess", material.shininess);
00015
00016     if (selectedShape < 1)
00017     {
00018         objects[(size_t)ObjectIdx::cubeNorm]->render();
00019     }
00020     else
00021     {
00022         objects[(size_t)ObjectIdx::pyramidNorm]->render();
00023     }
00024 }
00025
00026 void renderTexShapes(const std::array<std::unique_ptr<AbstractShape>, 5> &objects, const int
      &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const
      std::array<glm::mat4, 3> &MVP, const Light &light, const unsigned int &diffuseMap, const unsigned int
      &specularMap)
00027 {
00028     shader.use();
00029     setShaderLighting(shader, light);
00030     shader.setVec3("viewPos", camera.cameraPos);
00031     setShaderMVP(shader, MVP[0], MVP[1], MVP[2]);
00032
00033     shader.setVec3("material.specular", material.specular);
00034     shader.setFloat("material.shininess", material.shininess);
00035
00036     bindTextures(diffuseMap, specularMap);
00037
00038     if (selectedShape < 1)
00039     {
00040         objects[(size_t)ObjectIdx::cubeTex]->render();
00041     }
00042     else
00043     {
00044         objects[(size_t)ObjectIdx::pyramidTex]->render();
00045     }
00046 }
00047
00048 void renderLights(const std::array<std::unique_ptr<AbstractShape>, 5> &objects, const Shader &shader,
      const std::array<glm::mat4, 3> &MVP, const Light &light)
00049 {
00050     shader.use();
00051     auto model = glm::mat4(1.0f);
00052     model = glm::translate(model, light.pos);
00053     model = glm::scale(model, glm::vec3(0.2f));
00054     setShaderMVP(shader, model, MVP[1], MVP[2]);
00055     shader.setVec3("lightColor", light.color);
00056
00057     objects[(size_t)ObjectIdx::lightCube]->render();
00058 }
00059
00060 void clearFrame(const ImVec4 &clear_color)
00061 {
00062     glClearColor(clear_color.x * clear_color.w, clear_color.y * clear_color.w, clear_color.z *
      clear_color.w, clear_color.w);
00063     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
00064 }
00065
00066 std::array<glm::mat4, 3> calculateMVP(const Camera &camera, float ratio, const glm::vec3 &pos, float
      scale)
00067 {
00068     glm::mat4 projection = glm::perspective(glm::radians(camera.fov), ratio, 0.1f, 100.0f);
00069     glm::mat4 view = camera.calculateView();
00070     auto model = glm::mat4(1.0f);
00071     model = glm::translate(model, pos);
00072     model = glm::scale(model, glm::vec3(scale));
00073     return std::array<glm::mat4, 3>{
00074         model,
```

```
00075         view,
00076         projection};
00077 }
00078
00079 void updateDeltaTime(AppData &appData)
00080 {
00081     auto currentFrame = static_cast<float>(glfwGetTime());
00082     appData.deltaTime = currentFrame - appData.lastFrame;
00083     appData.lastFrame = currentFrame;
00084 }
```

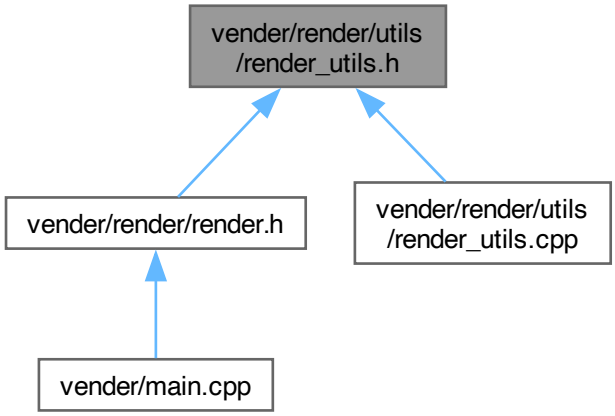
5.53 vender/render/utils/render_utils.h File Reference

```
#include <glad/glad.h>
#include "appdata/appdata.h"
#include "camera/camera.h"
#include "shaders/shader.h"
#include "assets/material.h"
#include "models/objects/shape.h"
#include "models/lighting/light.h"
```

Include dependency graph for render_utils.h:



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



Functions

- void renderGenShapes (const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const int &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const std::array< glm::mat4, 3 > &MVP, const Light &light)
- void renderTexShapes (const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const int &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const std::array< glm::mat4, 3 > &MVP, const Light &light, const unsigned int &diffuseMap, const unsigned int &specularMap)
- void renderLights (const std::array< std::unique_ptr< AbstractShape >, 5 > &objects, const Shader &shader, const std::array< glm::mat4, 3 > &MVP, const Light &light)
- void clearFrame (const ImVec4 &clear_color)
- std::array< glm::mat4, 3 > calculateMVP (const Camera &camera, float ratio, const glm::vec3 &pos=glm::vec3(0.0f, 0.0f, 0.0f), float scale=1.0f)
- void updateDeltaTime (AppData &appData)

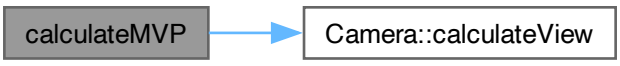
5.53.1 Function Documentation

5.53.1.1 calculateMVP()

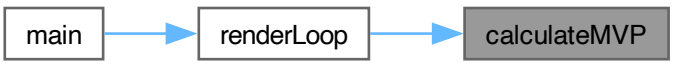
```
std::array< glm::mat4, 3 > calculateMVP (
            const Camera & camera,
            float ratio,
            const glm::vec3 & pos = glm::vec3(0.0f, 0.0f, 0.0f),
            float scale = 1.0f )
```

Definition at line 66 of file render_utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.53.1.2 clearFrame()

```
void clearFrame (
            const ImVec4 & clear_color )
```

Definition at line 60 of file render_utils.cpp.

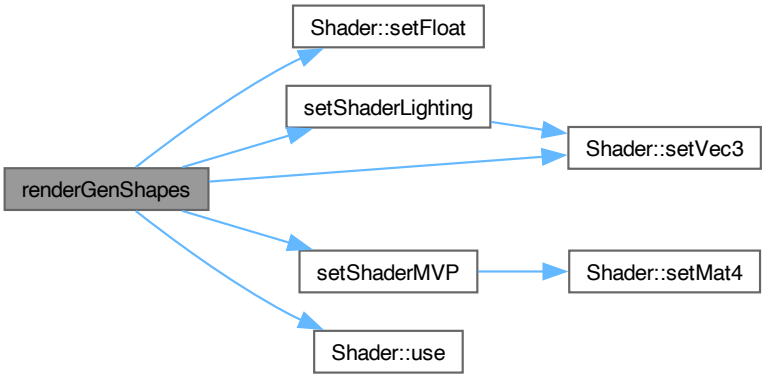
Here is the caller graph for this function:

5.53.1.3 renderGenShapes()

```
void renderGenShapes (
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const int & selectedShape,
            const Shader & shader,
            const Camera & camera,
            const Material & material,
            const std::array< glm::mat4, 3 > & MVP,
            const Light & light )
```

Definition at line 4 of file render_utils.cpp.

Here is the call graph for this function:



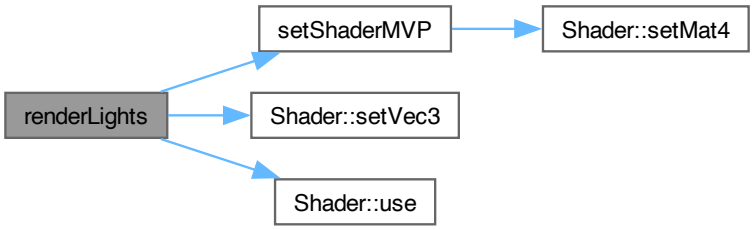
Here is the caller graph for this function:

5.53.1.4 renderLights()

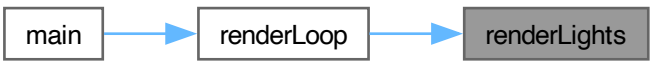
```
void renderLights (
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const Shader & shader,
            const std::array< glm::mat4, 3 > & MVP,
            const Light & light )
```

Definition at line 48 of file render_utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

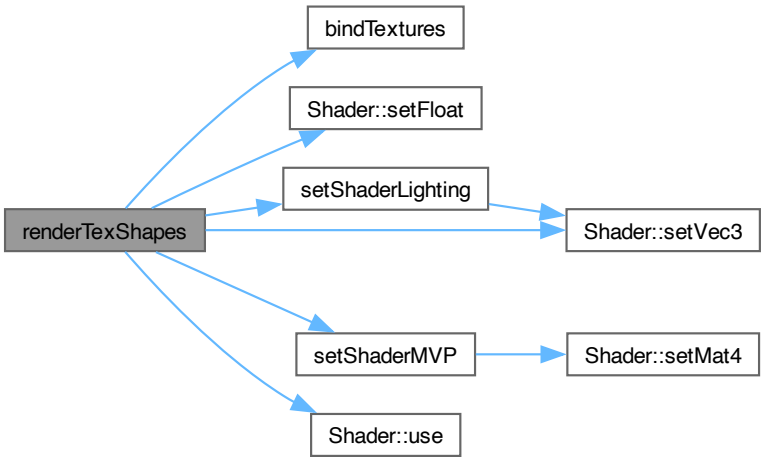


5.53.1.5 renderTexShapes()

```
void renderTexShapes (
            const std::array< std::unique_ptr< AbstractShape >, 5 > & objects,
            const int & selectedShape,
            const Shader & shader,
            const Camera & camera,
            const Material & material,
            const std::array< glm::mat4, 3 > & MVP,
            const Light & light,
            const unsigned int & diffuseMap,
            const unsigned int & specularMap )
```

Definition at line 26 of file render_utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.53.1.6 updateDeltaTime()

```
void updateDeltaTime (
            AppData & appData )
```

Definition at line 79 of file render_utils.cpp.

Here is the caller graph for this function:

5.54 render_utils.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <glad/glad.h>
00004 #include "appdata/appdata.h"
00005 #include "camera/camera.h"
00006 #include "shaders/shader.h"
00007 #include "assets/material.h"
00008 #include "models/objects/shape.h"
00009 #include "models/lighting/light.h"
00010
00011 void renderGenShapes(const std::array<std::unique_ptr<AbstractShape>, 5> &objects, const int
      &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const
      std::array<glm::mat4, 3> &MVP, const Light &light);
00012 void renderTexShapes(const std::array<std::unique_ptr<AbstractShape>, 5> &objects, const int
      &selectedShape, const Shader &shader, const Camera &camera, const Material &material, const
      std::array<glm::mat4, 3> &MVP, const Light &light, const unsigned int &diffuseMap, const unsigned int
      &specularMap);
00013 void renderLights(const std::array<std::unique_ptr<AbstractShape>, 5> &objects, const Shader &shader,
      const std::array<glm::mat4, 3> &MVP, const Light &light);
00014 void clearFrame(const ImVec4 &clear_color);
00015 std::array<glm::mat4, 3> calculateMVP(const Camera &camera, float ratio, const glm::vec3 &pos =
      glm::vec3(0.0f, 0.0f, 0.0f), float scale = 1.0f);
00016 void updateDeltaTime(AppData &appData);
```

5.55 vender/shaders/fragment/obj_generic.fs File Reference

5.56 obj_generic.fs

Go to the documentation of this file.

```
00001 #version 330 core
00002 out vec4 FragColor;
00003
00004 struct Material {
00005     vec3 ambient;
00006     vec3 diffuse;
00007     vec3 specular;
00008     float shininess;
00009 };
00010
00011 struct Light {
00012     vec3 pos;
00013     vec3 color;
00014     vec3 ambient;
00015     vec3 diffuse;
00016     vec3 specular;
00017 };
00018
00019 in vec3 FragPos;
00020 in vec3 Normal;
```

```
00021
00022 uniform vec3 viewPos;
00023 uniform Material material;
00024 uniform Light light;
00025
00026 void main()
00027 {
00028     // Ambient
00029     vec3 ambient = light.ambient * material.ambient;
00030
00031     // Diffuse
00032     vec3 norm = normalize(Normal);
00033     vec3 lightDir = normalize(light.pos - FragPos);
00034     float diff = max(dot(norm, lightDir), 0.0);
00035     vec3 diffuse = light.diffuse * (diff * material.diffuse);
00036
00037     // Specular
00038     vec3 viewDir = normalize(viewPos - FragPos);
00039     vec3 reflectDir = reflect(-lightDir, norm);
00040     float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
00041     vec3 specular = light.specular * material.specular * spec;
00042
00043     vec3 result = ambient + diffuse + specular;
00044     FragColor = vec4(result, 1.0);
00045 }
```

5.57 vender/shaders/fragment/obj_textured.fs File Reference

5.58 obj_textured.fs

Go to the documentation of this file.

```
00001 #version 330 core
00002 out vec4 FragColor;
00003
00004 struct Material {
00005     sampler2D diffuse;
00006     sampler2D specular;
00007     float shininess;
00008 };
00009
00010 struct Light {
00011     vec3 pos;
00012     vec3 color;
00013     vec3 ambient;
00014     vec3 diffuse;
00015     vec3 specular;
00016 };
00017
00018 in vec3 FragPos;
00019 in vec3 Normal;
00020 in vec2 TexCoords;
00021
00022 uniform vec3 viewPos;
00023 uniform Material material;
00024 uniform Light light;
00025
00026 void main()
00027 {
00028     // Ambient
00029     vec3 ambient = light.ambient * texture(material.diffuse, TexCoords).rgb;
00030
00031     // Diffuse
00032     vec3 norm = normalize(Normal);
00033     vec3 lightDir = normalize(light.pos - FragPos);
00034     float diff = max(dot(norm, lightDir), 0.0);
00035     vec3 diffuse = light.diffuse * diff * texture(material.diffuse, TexCoords).rgb;
00036
00037     // Specular
00038     vec3 viewDir = normalize(viewPos - FragPos);
00039     vec3 reflectDir = reflect(-lightDir, norm);
00040     float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
00041     vec3 specular = light.specular * spec * texture(material.specular, TexCoords).rgb;
00042
00043     vec3 result = ambient + diffuse + specular;
00044     FragColor = vec4(result, 1.0);
00045 }
```

5.59 vender/shaders/fragment/point_light.fs File Reference

5.60 point_light.fs

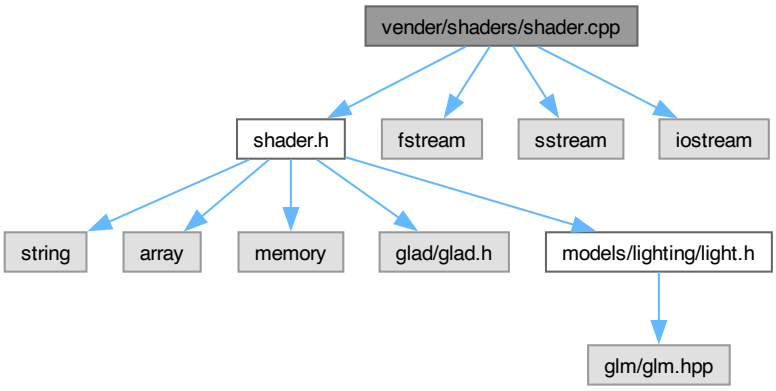
Go to the documentation of this file.

```
00001 #version 330 core
00002
00003 uniform vec3 lightColor;
00004
00005 out vec4 FragColor;
00006
00007 void main()
00008 {
00009     FragColor = vec4(lightColor, 1.0);
00010 }
```

5.61 vender/shaders/shader.cpp File Reference

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

Include dependency graph for shader.cpp:



Functions

- std::array< std::unique_ptr< Shader >, 3 > loadShaders ()
- void configureShaders (std::array< std::unique_ptr< Shader >, 3 > &shaders)
- void setShaderLighting (const Shader &shader, const Light &light)
- void setShaderMVP (const Shader &shader, const glm::mat4 &model, const glm::mat4 &view, const glm::mat4 &projection)
- void bindTextures (unsigned int diffuseMap, unsigned int specularMap)

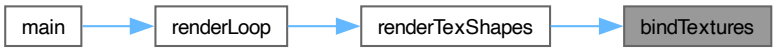
5.61.1 Function Documentation

5.61.1.1 bindTextures()

```
void bindTextures (
            unsigned int diffuseMap,
            unsigned int specularMap )
```

Definition at line 181 of file shader.cpp.

Here is the caller graph for this function:



5.61.1.2 configureShaders()

```
void configureShaders (
            std::array< std::unique_ptr< Shader >, 3 > & shaders )
```

Definition at line 161 of file shader.cpp.

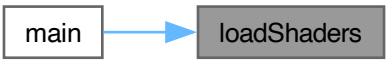
Here is the caller graph for this function:

5.61.1.3 loadShaders()

```
std::array< std::unique_ptr< Shader >, 3 > loadShaders ( )
```

Definition at line 153 of file shader.cpp.

Here is the caller graph for this function:



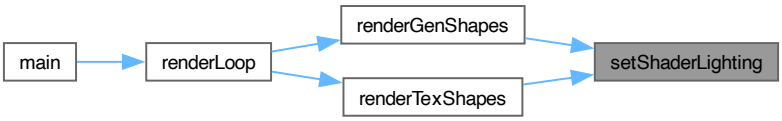
5.61.1.4 setShaderLighting()

```
void setShaderLighting (
            const Shader & shader,
            const Light & light )
```

Definition at line 168 of file shader.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

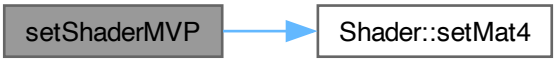


5.61.1.5 setShaderMVP()

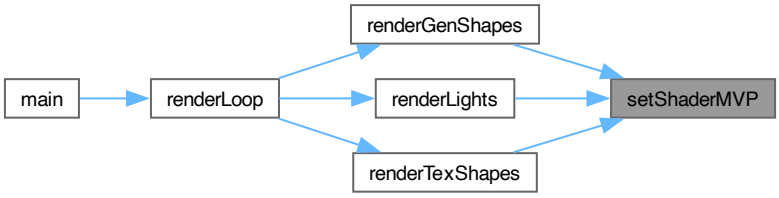
```
void setShaderMVP (
            const Shader & shader,
            const glm::mat4 & model,
            const glm::mat4 & view,
            const glm::mat4 & projection )
```

Definition at line 175 of file shader.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.62 shader.cpp

Go to the documentation of this file.

```
00001 #include "shader.h"
00002
00003 #include <fstream>
00004 #include <sstream>
00005 #include <iostream>
00006
00007 Shader::Shader(const char *vertexPath, const char *fragmentPath)
00008 {
00009     // 1. retrieve the vertex/fragment source code from filePath
00010     std::string vertexCode;
00011     std::string fragmentCode;
00012     std::ifstream vShaderFile;
00013     std::ifstream fShaderFile;
00014     // ensure ifstream objects can throw exceptions:
00015     vShaderFile.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00016     fShaderFile.exceptions(std::ifstream::failbit | std::ifstream::badbit);
00017     try
00018     {
00019         // open files
00020         vShaderFile.open(vertexPath);
00021         fShaderFile.open(fragmentPath);
00022         std::stringstream vShaderStream;
00023         std::stringstream fShaderStream;
00024         // read file's buffer contents into streams
00025         vShaderStream << vShaderFile.rdbuf();
00026         fShaderStream << fShaderFile.rdbuf();
00027         // close file handlers
00028         vShaderFile.close();
00029         fShaderFile.close();
00030         // convert stream into string
00031         vertexCode = vShaderStream.str();
00032         fragmentCode = fShaderStream.str();
00033     }
00034     catch (const std::ifstream::failure &e)
00035     {
00036         std::cout << "ERROR::SHADER::FILE_NOT_SUCCESFULLY_READ" << e.what() << std::endl;
00037     }
00038     const char *vShaderCode = vertexCode.c_str();
00039     const char *fShaderCode = fragmentCode.c_str();
00040
00041     // 2. compile shaders
00042     unsigned int vertex;
00043     unsigned int fragment;
00044
00045     // Vertex Shader
00046     vertex = glCreateShader(GL_VERTEX_SHADER);
00047     glShaderSource(vertex, 1, &vShaderCode, nullptr);
00048     glCompileShader(vertex);
00049     checkCompileErrors(vertex, "VERTEX");
00050
00051     // Fragment Shader
00052     fragment = glCreateShader(GL_FRAGMENT_SHADER);
00053     glShaderSource(fragment, 1, &fShaderCode, nullptr);
00054     glCompileShader(fragment);
00055     checkCompileErrors(fragment, "FRAGMENT");
00056
00057     // Shader Program
```

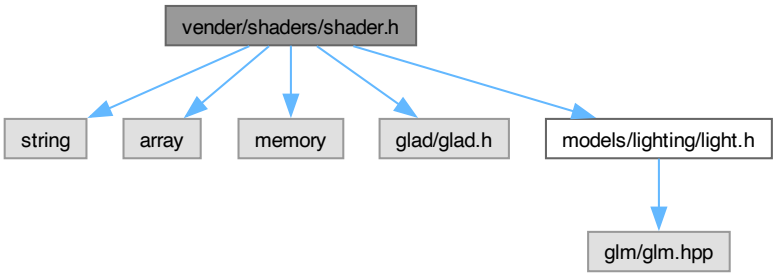
```
00058     ID = glCreateProgram();
00059     glAttachShader(ID, vertex);
00060     glAttachShader(ID, fragment);
00061     glLinkProgram(ID);
00062     checkCompileErrors(ID, "PROGRAM");
00063 
00064     glDeleteShader(vertex);
00065     glDeleteShader(fragment);
00066 }
00067 
00068 // use/activate the shader
00069 void Shader::use() const
00070 {
00071     glUseProgram(ID);
00072 }
00073 void Shader::del() const
00074 {
00075     glDeleteProgram(ID);
00076 }
00077 // utility uniform functions
00078 void Shader::setBool(const std::string &name, bool value) const
00079 {
00080     glUniform1i(glGetUniformLocation(ID, name.c_str()), (int)value);
00081 }
00082 void Shader::setInt(const std::string &name, int value) const
00083 {
00084     glUniform1i(glGetUniformLocation(ID, name.c_str()), value);
00085 }
00086 void Shader::setFloat(const std::string &name, float value) const
00087 {
00088     glUniform1f(glGetUniformLocation(ID, name.c_str()), value);
00089 }
00090 void Shader::setVec2(const std::string &name, const glm::vec2 &value) const
00091 {
00092     glUniform2fv(glGetUniformLocation(ID, name.c_str()), 1, &value[0]);
00093 }
00094 void Shader::setVec2(const std::string &name, float x, float y) const
00095 {
00096     glUniform2f(glGetUniformLocation(ID, name.c_str()), x, y);
00097 }
00098 void Shader::setVec3(const std::string &name, const glm::vec3 &value) const
00099 {
00100     glUniform3fv(glGetUniformLocation(ID, name.c_str()), 1, &value[0]);
00101 }
00102 void Shader::setVec3(const std::string &name, float x, float y, float z) const
00103 {
00104     glUniform3f(glGetUniformLocation(ID, name.c_str()), x, y, z);
00105 }
00106 void Shader::setVec4(const std::string &name, const glm::vec4 &value) const
00107 {
00108     glUniform4fv(glGetUniformLocation(ID, name.c_str()), 1, &value[0]);
00109 }
00110 void Shader::setVec4(const std::string &name, float x, float y, float z, float w) const
00111 {
00112     glUniform4f(glGetUniformLocation(ID, name.c_str()), x, y, z, w);
00113 }
00114 void Shader::setMat2(const std::string &name, const glm::mat2 &mat) const
00115 {
00116     glUniformMatrix2fv(glGetUniformLocation(ID, name.c_str()), 1, GL_FALSE, &mat[0][0]);
00117 }
00118 void Shader::setMat3(const std::string &name, const glm::mat3 &mat) const
00119 {
00120     glUniformMatrix3fv(glGetUniformLocation(ID, name.c_str()), 1, GL_FALSE, &mat[0][0]);
00121 }
00122 void Shader::setMat4(const std::string &name, const glm::mat4 &mat) const
00123 {
00124     glUniformMatrix4fv(glGetUniformLocation(ID, name.c_str()), 1, GL_FALSE, &mat[0][0]);
00125 }
00126 
00127 void Shader::checkCompileErrors(unsigned int shader, const std::string &type) const
00128 {
00129     int success;
00130     std::string infoLog;
00131     if (type != "PROGRAM")
00132     {
00133         glGetShaderiv(shader, GL_COMPILE_STATUS, &success);
00134         if (!success)
00135         {
00136             glGetShaderInfoLog(shader, 1024, nullptr, infoLog.data());
00137             std::cout « "ERROR::SHADER_COMPILATION_ERROR of type: " « type « "\n"
00138                      « infoLog « "\n -- --------------------------------------------------- -- " «
    std::endl;
00139         }
00140     }
00141     else
00142     {
00143         glGetProgramiv(shader, GL_LINK_STATUS, &success);
```

```
00144         if (!success)
00145         {
00146             glGetProgramInfoLog(shader, 1024, nullptr, infoLog.data());
00147             std::cout « "ERROR::PROGRAM_LINKING_ERROR of type: " « type « "\n"
00148                      « infoLog « "\n -- --------------------------------------------------- -- " «
      std::endl;
00149         }
00150     }
00151 }
00152
00153 std::array<std::unique_ptr<Shader>, 3> loadShaders()
00154 {
00155     std::array<std::unique_ptr<Shader>, 3> shaders = {
00156         std::make_unique<Shader>("../vender/shaders/vertex/obj_generic.vs",
      "../vender/shaders/fragment/obj_generic.fs"),
00157         std::make_unique<Shader>("../vender/shaders/vertex/obj_textured.vs",
      "../vender/shaders/fragment/obj_textured.fs"),
00158         std::make_unique<Shader>("../vender/shaders/vertex/obj_generic.vs",
      "../vender/shaders/fragment/point_light.fs")};
00159     return shaders;
00160 }
00161 void configureShaders(std::array<std::unique_ptr<Shader>, 3> &shaders)
00162 {
00163     using enum ShaderIdx;
00164     shaders[(size_t)tex]->use();
00165     shaders[(size_t)tex]->setInt("material.diffuse", 0);
00166     shaders[(size_t)tex]->setInt("material.specular", 1);
00167 }
00168 void setShaderLighting(const Shader &shader, const Light &light)
00169 {
00170     shader.setVec3("light.pos", light.pos);
00171     shader.setVec3("light.ambient", light.ambient * light.color);
00172     shader.setVec3("light.diffuse", light.diffuse * light.color);
00173     shader.setVec3("light.specular", light.specular * light.color);
00174 }
00175 void setShaderMVP(const Shader &shader, const glm::mat4 &model, const glm::mat4 &view, const glm::mat4
      &projection)
00176 {
00177     shader.setMat4("model", model);
00178     shader.setMat4("view", view);
00179     shader.setMat4("projection", projection);
00180 }
00181 void bindTextures(unsigned int diffuseMap, unsigned int specularMap)
00182 {
00183     glActiveTexture(GL_TEXTURE0);
00184     glBindTexture(GL_TEXTURE_2D, diffuseMap);
00185     glActiveTexture(GL_TEXTURE1);
00186     glBindTexture(GL_TEXTURE_2D, specularMap);
00187 }
```

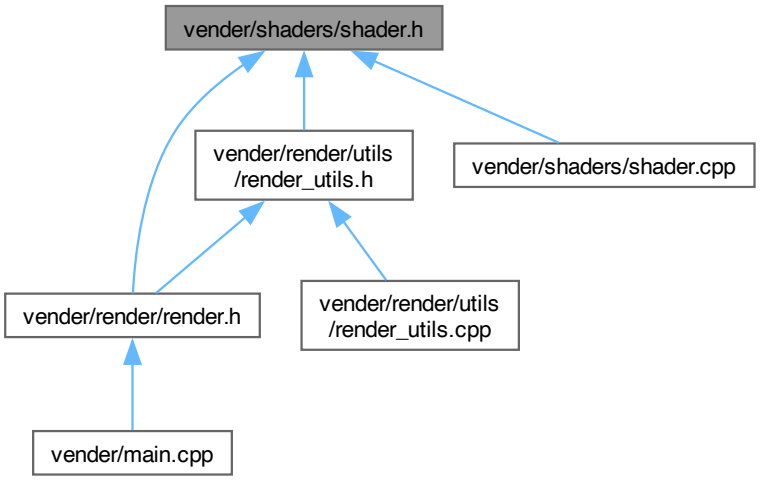
5.63 vender/shaders/shader.h File Reference

```
#include <string>
#include <array>
#include <memory>
#include <glad/glad.h>
#include "models/lighting/light.h"
```

Include dependency graph for shader.h:



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



Classes

- class Shader

Enumerations

- enum struct ShaderIdx { generic = 0 , tex = 1 , light = 2 }

Functions

- std::array< std::unique_ptr< Shader >, 3 > loadShaders ()
- void configureShaders (std::array< std::unique_ptr< Shader >, 3 > &shaders)
- void setShaderLighting (const Shader &shader, const Light &light)
- void setShaderMVP (const Shader &shader, const glm::mat4 &model, const glm::mat4 &view, const glm::mat4 &projection)
- void bindTextures (unsigned int diffuseMap, unsigned int specularMap)

5.63.1 Enumeration Type Documentation

5.63.1.1 ShaderIdx

```
enum struct ShaderIdx  [strong]
```

Enumerator

generic	
tex	
light	

Definition at line 10 of file shader.h.

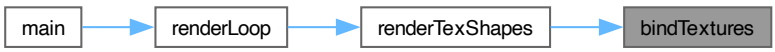
5.63.2 Function Documentation

5.63.2.1 bindTextures()

```
void bindTextures (
            unsigned int diffuseMap,
            unsigned int specularMap )
```

Definition at line 181 of file shader.cpp.

Here is the caller graph for this function:



5.63.2.2 configureShaders()

```
void configureShaders (
            std::array< std::unique_ptr< Shader >, 3 > & shaders )
```

Definition at line 161 of file shader.cpp.

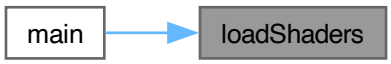
Here is the caller graph for this function:

5.63.2.3 loadShaders()

```
std::array< std::unique_ptr< Shader >, 3 > loadShaders ( )
```

Definition at line 153 of file shader.cpp.

Here is the caller graph for this function:

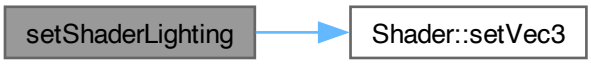


5.63.2.4 setShaderLighting()

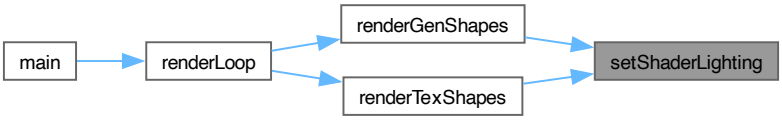
```
void setShaderLighting (
            const Shader & shader,
            const Light & light )
```

Definition at line 168 of file shader.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

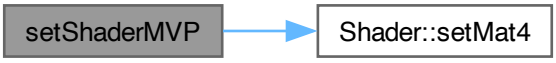


5.63.2.5 setShaderMVP()

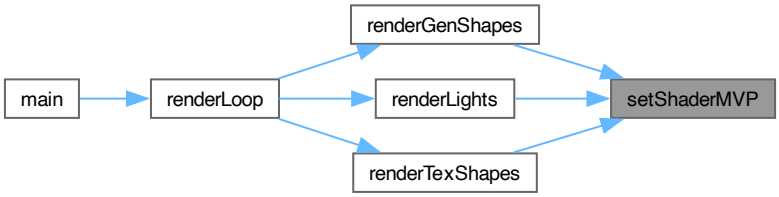
```
void setShaderMVP (
            const Shader & shader,
            const glm::mat4 & model,
            const glm::mat4 & view,
            const glm::mat4 & projection )
```

Definition at line 175 of file shader.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.64 shader.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <string>
00004 #include <array>
00005 #include <memory>
00006
00007 #include <glad/glad.h>
00008 #include "models/lighting/light.h"
00009
00010 enum struct ShaderIdx
00011 {
00012     generic = 0,
00013     tex = 1,
00014     light = 2,
00015 };
00016
00017 class Shader
00018 {
00019 public:
00020     // The program ID
00021     unsigned int ID;
```

```
00022
00023     // constructor reads and builds the shader
00024     Shader(const char *vertexPath, const char *fragmentPath);
00025
00026     // Use/activate the shader
00027     void use() const;
00028     void del() const;
00029
00030     // Utility uniform functions
00031     void setBool(const std::string &name, bool value) const;
00032     void setInt(const std::string &name, int value) const;
00033     void setFloat(const std::string &name, float value) const;
00034     void setVec2(const std::string &name, const glm::vec2 &value) const;
00035     void setVec2(const std::string &name, float x, float y) const;
00036     void setVec3(const std::string &name, const glm::vec3 &value) const;
00037     void setVec3(const std::string &name, float x, float y, float z) const;
00038     void setVec4(const std::string &name, const glm::vec4 &value) const;
00039     void setVec4(const std::string &name, float x, float y, float z, float w) const;
00040     void setMat2(const std::string &name, const glm::mat2 &mat) const;
00041     void setMat3(const std::string &name, const glm::mat3 &mat) const;
00042     void setMat4(const std::string &name, const glm::mat4 &mat) const;
00043
00044 private:
00045     void checkCompileErrors(unsigned int shader, const std::string &type) const;
00046 };
00047
00048 std::array<std::unique_ptr<Shader>, 3> loadShaders();
00049 void configureShaders(std::array<std::unique_ptr<Shader>, 3> &shaders);
00050 void setShaderLighting(const Shader &shader, const Light &light);
00051 void setShaderMVP(const Shader &shader, const glm::mat4 &model, const glm::mat4 &view, const glm::mat4
      &projection);
00052 void bindTextures(unsigned int diffuseMap, unsigned int specularMap);
```

5.65 vender/shaders/vertex/obj_generic.vs File Reference

5.66 obj_generic.vs

Go to the documentation of this file.

```
00001 #version 330 core
00002 layout (location = 0) in vec3 aPos;
00003 layout (location = 1) in vec3 aNormal;
00004
00005 out vec3 FragPos;
00006 out vec3 Normal;
00007
00008 uniform mat4 model;
00009 uniform mat4 view;
00010 uniform mat4 projection;
00011
00012 void main()
00013 {
00014     gl_Position = projection * view * model * vec4(aPos, 1.0);
00015     FragPos = vec3(model * vec4(aPos, 1.0));
00016     // TODO: Calculate the normal matrix in CPU
00017     Normal = mat3(transpose(inverse(model))) * aNormal;
00018 }
```

5.67 vender/shaders/vertex/obj_textured.vs File Reference

5.68 obj_textured.vs

Go to the documentation of this file.

```
00001 #version 330 core
00002 layout (location = 0) in vec3 aPos;
00003 layout (location = 1) in vec3 aNormal;
00004 layout (location = 2) in vec2 aTexCoords;
00005
00006 out vec3 FragPos;
00007 out vec3 Normal;
```

```
00008 out vec2 TexCoords;
00009
00010 uniform mat4 model;
00011 uniform mat4 view;
00012 uniform mat4 projection;
00013
00014 void main()
00015 {
00016     FragPos = vec3(model * vec4(aPos, 1.0));
00017     Normal = mat3(transpose(inverse(model))) * aNormal;
00018     TexCoords = aTexCoords;
00019
00020     gl_Position = projection * view * model * vec4(aPos, 1.0);
00021 }
```

Index