Simple Renderer

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 IDrawable Class Reference

Public Member Functions

- virtual glm::vec2 getCenterPoint ()=0

    Used to retrieve the center of the drawable, for example for scaling. It is calculated by averaging all points forming the shape.
- virtual IDrawable ∗ clone ()=0

    Used to create a copy of the drawable.
- virtual std::string getTexPath ()=0

    Used to retrieve the texture path of the drawable.
- virtual std::vector< unsigned int > getElementData ()=0

    Used to retrieve the triangle formation data.
- virtual std::vector< glm::vec2 > getVertexData ()=0

    Used to retrieve the vertex position data.
- virtual std::vector< glm::vec2 > getTexCoordsData ()=0

    Used to retrieve the texture coordinate data.
- virtual glm::vec4 getColor ()=0

    Used to retrieve the current color of the drawable.
- virtual void setColor (glm::vec4 color)=0

    Sets the color of the drawable.
- virtual glm::vec2 getTranslation ()=0

    Retrieves the current translation of the drawable.
- virtual void setTranslation (glm::vec2 translation)=0

    Sets the translation of the drawable.
- virtual glm::vec2 getScale ()=0

    Used to retrieve the current scaling applied to the drawable.
- virtual void setScale (glm::vec2 scale)=0

    Sets the scaling of the drawable.

2.1.1 Member Function Documentation

2.1.1.1 clone()

```
virtual IDrawable* IDrawable::clone ( ) [pure virtual]
```

Used to create a copy of the drawable.

Returns

The copy.

2.1.1.2 getCenterPoint()

```
virtual glm::vec2 IDrawable::getCenterPoint ( ) [pure virtual]
```

Used to retrieve the center of the drawable, for example for scaling. It is calculated by averaging all points forming the shape.

Returns

The center of a shape.

2.1.1.3 getColor()

```
virtual glm::vec4 IDrawable::getColor ( ) [pure virtual]
```

Used to retrieve the current color of the drawable.

Returns

An rgba shape 4-dimensional vector.

2.1.1.4 getElementData()

```
virtual std::vector<unsigned int> IDrawable::getElementData ( ) [pure virtual]
```

Used to retrieve the triangle formation data.

Returns

a vector of vector index combinations that form the triangles of the drawable.

2.1.1.5 getScale()

```
virtual glm::vec2 IDrawable::getScale ( )  [pure virtual]
```

Used to retrieve the current scaling applied to the drawable.

Returns

The scale vector. vec2(1.0f, 1.0f) is the default.

2.1.1.6 getTexCoordsData()

```
virtual std::vector<glm::vec2> IDrawable::getTexCoordsData ( )  [pure virtual]
```

Used to retrieve the texture coordinate data.

Returns

A vector of texture coordinates for each vertex.

2.1.1.7 getTexPath()

```
virtual std::string IDrawable::getTexPath ( )  [pure virtual]
```

Used to retrieve the texture path of the drawable.

Returns

The texture path assigned to the drawable.

2.1.1.8 getTranslation()

```
virtual glm::vec2 IDrawable::getTranslation ( )  [pure virtual]
```

Retrieves the current translation of the drawable.

Returns

The translation vector. vec2(0.0f, 0.0f) is the default.

2.1.1.9 getVertexData()

```
virtual std::vector<glm::vec2> IDrawable::getVertexData ( )  [pure virtual]
```

Used to retrieve the vertex position data.

Returns

A vector of all vertex positions.

2.1.1.10 setColor()

```
virtual void IDrawable::setColor (
            glm::vec4 color )  [pure virtual]
```

Sets the color of the drawable.

Parameters

color	The color in the rgba format.

2.1.1.11 setScale()

```
virtual void IDrawable::setScale (
            glm::vec2 scale )  [pure virtual]
```

Sets the scaling of the drawable.

Parameters

scale	The scale vector. vec2(1.0f, 1.0f) is the default.

2.1.1.12 setTranslation()

```
virtual void IDrawable::setTranslation (
            glm::vec2 translation )  [pure virtual]
```

Sets the translation of the drawable.

Parameters

translation	The translation vector. vec2(0.0f, 0.0f) is the default.

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

- IDrawable.h

2.2 Renderer Class Reference

Public Member Functions

- virtual void TexInit (unsigned int maxTexWidth, unsigned int maxTexHeight, unsigned int numTextures)=0

  Initializes the texture functionality by creating immutable storage for textures.
- virtual void drawRectangle (glm::vec2 origin, glm::vec2 size, glm::vec4 color)=0

  Creates a rectangle on the stack that is immediately discarded after drawing.
- virtual void drawQuadrangle (glm::vec2 p1, glm::vec2 p2, glm::vec2 p3, glm::vec2 p4, glm::vec4 color)=0

  Creates a quadrangle out of any four points on the stack that is immediately discarded after drawing.
- virtual void drawRectangle (glm::vec2 origin, glm::vec2 size, std::string texPath)=0

  Creates a rectangle on the stack that is immediately discarded after drawing.

- virtual void drawQuadrangle (glm::vec2 p1, glm::vec2 p2, glm::vec2 p3, glm::vec2 p4, std::string texPath)=0

  Creates a quadrangle out of any four points on the stack that is immediately discarded after drawing.
- virtual void Display ()=0

  Displays everything that was drawn before calling it.
- virtual void loadTexture (std::string texPath)=0

  Allows you to load a texture before using it.
- virtual void draw (IDrawable ∗drawable)=0

  Draws an IDrawable object on the screen.

2.2.1 Member Function Documentation

2.2.1.1 Display()

```
virtual void Renderer::Display ( )  [pure virtual]
```

Displays everything that was drawn before calling it.

Note

You need to re-draw shapes and display them for each frame you want to generate.

2.2.1.2 draw()

```
virtual void Renderer::draw (
             IDrawable ∗ drawable )  [pure virtual]
```

Draws an IDrawable object on the screen.

Parameters

drawable	A drawable object pointer.

2.2.1.3 drawQuadrangle() [1/2]

```
virtual void Renderer::drawQuadrangle (
             glm::vec2 p1,
             glm::vec2 p2,
             glm::vec2 p3,
             glm::vec2 p4,
             glm::vec4 color )  [pure virtual]
```

Creates a quadrangle out of any four points on the stack that is immediately discarded after drawing.

Note

The order needs to be counter clockwise.
Can be used as a shorter alternative to draw() without the need to pass an object.

Parameters

p1	Lower left corner. <-1.0, 1.0> range of coordinates.
p2	Lower right corner. <-1.0, 1.0> range of coordinates.
p3	Upper rigt corner. <-1.0, 1.0> range of coordinates.
p4	Upper left corner. <-1.0, 1.0> range of coordinates.
color	The color of the rectangle in rgba format.

2.2.1.4 drawQuadrangle() [2/2]

```
virtual void Renderer::drawQuadrangle (
             glm::vec2 p1,
             glm::vec2 p2,
             glm::vec2 p3,
             glm::vec2 p4,
             std::string texPath )  [pure virtual]
```

Creates a quadrangle out of any four points on the stack that is immediately discarded after drawing.

Note

The order needs to be counter clockwise. It should start from the lower left corner if you want the texture to be positioned upright.
Can be used as a shorter alternative to draw() without the need to pass an object.

Parameters

p1	Lower left corner. <-1.0, 1.0> range of coordinates.
p2	Lower right corner. <-1.0, 1.0> range of coordinates.
p3	Upper rigt corner. <-1.0, 1.0> range of coordinates.
p4	Upper left corner. <-1.0, 1.0> range of coordinates.
texPath	The path to the texture. A black shape will be drawn if it doesn't exist.

2.2.1.5 drawRectangle() [1/2]

```
virtual void Renderer::drawRectangle (
             glm::vec2 origin,
             glm::vec2 size,
             glm::vec4 color )  [pure virtual]
```

Creates a rectangle on the stack that is immediately discarded after drawing.

Note

Can be used as a shorter alternative to draw() without the need to pass an object.

Parameters

origin	The origin point (lower left corner). <-1.0, 1.0> range of coordinates.
size	The width and height of the rectangle.
color	The color of the rectangle in rgba format.

2.2.1.6 drawRectangle() [2/2]

```
virtual void Renderer::drawRectangle (
             glm::vec2 origin,
             glm::vec2 size,
             std::string texPath )  [pure virtual]
```

Creates a rectangle on the stack that is immediately discarded after drawing.

Note

Can be used as a shorter alternative to draw() without the need to pass an object.

Parameters

origin	The origin point (lower left corner). <-1.0, 1.0> range of coordinates.
size	The width and height of the rectangle.
texPath	The path to the texture. A black shape will be drawn if it doesn't exist.

2.2.1.7 loadTexture()

```
virtual void Renderer::loadTexture (
             std::string texPath )  [pure virtual]
```

Allows you to load a texture before using it.

Note

If this is not used, the texture will be loaded on its first use in the program, which can cause unwanted delays.

Parameters

texPath	The path to the texture.

2.2.1.8 TexInit()

```
virtual void Renderer::TexInit (
            unsigned int maxTexWidth,
            unsigned int maxTexHeight,
            unsigned int numTextures )  [pure virtual]
```

Initializes the texture functionality by creating immutable storage for textures.

This storage provides incredible speed at the cost of wasting vram if the textures aren't of similar sizes. This is because the width and height of the storage need to be able to fit the biggest texture. Smaller textures will use "slots" of the same size as the biggest texture.

Parameters

maxTexWidth	The height of the texture storage. Texture used in the program can't be higher than this.
maxTexHeight	The width of the texture storage. Texture used in the program can't be wider than this.
numTextures	The depth of the texture storage. There can't be more individual textures than this number in the program.

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

- Renderer.h