

OpenGL-FinalProject

Generated by Doxygen 1.13.2

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Ragot::AreaLight Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 AreaLight()	8
4.2 Ragot::Camera Class Reference	8
4.2.1 Detailed Description	9
4.2.2 Constructor & Destructor Documentation	10
4.2.2.1 Camera() [1/3]	10
4.2.2.2 Camera() [2/3]	11
4.2.2.3 Camera() [3/3]	11
4.2.3 Member Function Documentation	11
4.2.3.1 get_far_z()	11
4.2.3.2 get_fov()	12
4.2.3.3 get_location()	12
4.2.3.4 get_near_z()	12
4.2.3.5 get_projection_matrix()	12
4.2.3.6 get_ratio()	12
4.2.3.7 get_target()	13
4.2.3.8 get_transform_matrix_inverse()	13
4.2.3.9 move()	13
4.2.3.10 reset()	13
4.2.3.11 rotate()	13
4.2.3.12 set_far_z()	14
4.2.3.13 set_fov()	14
4.2.3.14 set_location()	14
4.2.3.15 set_near_z()	14
4.2.3.16 set_ratio()	15
4.2.3.17 set_target()	15
4.3 Ragot::Color_Buffer< COLOR > Class Template Reference	15
4.3.1 Detailed Description	16
4.3.2 Constructor & Destructor Documentation	16
4.3.2.1 Color_Buffer()	16
4.3.3 Member Function Documentation	17

4.3.3.1 colors() [1/2]	17
4.3.3.2 colors() [2/2]	17
4.3.3.3 get() [1/2]	17
4.3.3.4 get() [2/2]	17
4.3.3.5 get_height()	18
4.3.3.6 get_width()	18
4.3.3.7 set()	18
4.4 Ragot::Component Class Reference	19
4.4.1 Detailed Description	19
4.4.2 Member Function Documentation	19
4.4.2.1 get_entity()	19
4.4.2.2 get_has_task()	20
4.4.2.3 set_entity()	20
4.5 Ragot::Critical_Task Class Reference	20
4.5.1 Detailed Description	21
4.5.2 Constructor & Destructor Documentation	21
4.5.2.1 Critical_Task()	21
4.5.3 Member Function Documentation	22
4.5.3.1 execute()	22
4.6 Ragot::DirectionalLight Class Reference	22
4.6.1 Detailed Description	23
4.6.2 Constructor & Destructor Documentation	23
4.6.2.1 DirectionalLight()	23
4.7 Ragot::Entity Class Reference	23
4.7.1 Detailed Description	24
4.7.2 Member Function Documentation	24
4.7.2.1 add_child()	24
4.7.2.2 add_component()	24
4.7.2.3 get_components()	25
4.7.2.4 get_scene() [1/2]	25
4.7.2.5 get_scene() [2/2]	25
4.7.2.6 remove_child()	25
4.7.2.7 remove_component()	25
4.7.2.8 set_scene()	26
4.7.2.9 set_transform_parent()	26
4.8 Ragot::Fragment_Shader Class Reference	26
4.8.1 Detailed Description	27
4.8.2 Constructor & Destructor Documentation	27
4.8.2.1 Fragment_Shader()	27
4.9 Ragot::Frame_Buffer Class Reference	28
4.9.1 Detailed Description	28
4.9.2 Constructor & Destructor Documentation	28

4.9.2.1 Frame_Buffer()	28
4.10 Ragot::Kernel Class Reference	29
4.10.1 Detailed Description	29
4.10.2 Member Function Documentation	29
4.10.2.1 add()	29
4.11 Ragot::Light Class Reference	29
4.11.1 Detailed Description	30
4.11.2 Constructor & Destructor Documentation	30
4.11.2.1 Light()	30
4.12 Ragot::Light_Task Class Reference	30
4.12.1 Detailed Description	31
4.12.2 Constructor & Destructor Documentation	31
4.12.2.1 Light_Task()	31
4.12.3 Member Function Documentation	32
4.12.3.1 execute()	32
4.13 Ragot::Material Class Reference	32
4.13.1 Detailed Description	33
4.13.2 Constructor & Destructor Documentation	33
4.13.2.1 Material()	33
4.13.3 Member Function Documentation	33
4.13.3.1 bind_texture()	33
4.13.3.2 get_color()	33
4.13.3.3 get_shader_program_id()	33
4.13.3.4 get_shader_program_uniform_location()	33
4.13.3.5 get shininess()	34
4.14 Ragot::Mesh Class Reference	34
4.14.1 Detailed Description	35
4.14.2 Member Enumeration Documentation	35
4.14.2.1 anonymous enum	35
4.14.3 Constructor & Destructor Documentation	35
4.14.3.1 Mesh()	35
4.14.4 Member Function Documentation	36
4.14.4.1 get_coordinates()	36
4.14.4.2 get_indices()	36
4.14.4.3 get_normals()	36
4.14.4.4 get_number_of_indices()	36
4.14.4.5 get_textures_uv()	37
4.14.4.6 get_vao_id()	37
4.15 Ragot::Model_Component Class Reference	37
4.15.1 Detailed Description	38
4.15.2 Constructor & Destructor Documentation	38
4.15.2.1 Model_Component()	38

4.15.3 Member Function Documentation	39
4.15.3.1 get_shader_program_id()	39
4.15.3.2 set_transparency()	39
4.16 Ragot::Once_Task Class Reference	39
4.16.1 Detailed Description	40
4.16.2 Constructor & Destructor Documentation	40
4.16.2.1 Once_Task()	40
4.16.3 Member Function Documentation	41
4.16.3.1 execute()	41
4.17 Ragot::Window::OpenGL_Context_Settings Struct Reference	41
4.17.1 Detailed Description	41
4.18 Ragot::PointLight Class Reference	42
4.18.1 Detailed Description	42
4.18.2 Constructor & Destructor Documentation	42
4.18.2.1 PointLight()	42
4.19 Ragot::Rgba8888 Union Reference	43
4.19.1 Detailed Description	43
4.20 Ragot::Scene Class Reference	43
4.20.1 Detailed Description	44
4.20.2 Member Function Documentation	44
4.20.2.1 add_entities()	44
4.20.2.2 get_camera()	45
4.20.2.3 get_entity()	45
4.20.2.4 on_click()	45
4.20.2.5 on_drag()	45
4.20.2.6 on_shift_pressed()	46
4.20.2.7 on_translation()	46
4.20.2.8 remove_entities()	46
4.20.2.9 resize()	46
4.21 Ragot::Shader Class Reference	47
4.21.1 Detailed Description	48
4.21.2 Constructor & Destructor Documentation	48
4.21.2.1 Shader()	48
4.21.3 Member Function Documentation	48
4.21.3.1 compile_shader()	48
4.21.3.2 get_error()	48
4.21.3.3 get_id()	48
4.21.3.4 is_ok()	49
4.22 Ragot::Shader_Program Class Reference	49
4.22.1 Detailed Description	49
4.22.2 Constructor & Destructor Documentation	49
4.22.2.1 Shader_Program()	49

4.22.3 Member Function Documentation	50
4.22.3.1 get_id()	50
4.22.3.2 get_uniform_location()	50
4.23 Ragot::Skybox Class Reference	50
4.23.1 Detailed Description	51
4.23.2 Constructor & Destructor Documentation	51
4.23.2.1 Skybox()	51
4.23.3 Member Function Documentation	51
4.23.3.1 set_camera()	51
4.24 Ragot::System Class Reference	52
4.24.1 Detailed Description	52
4.24.2 Constructor & Destructor Documentation	52
4.24.2.1 System()	52
4.24.3 Member Function Documentation	52
4.24.3.1 add_entities()	52
4.25 Ragot::Task Class Reference	53
4.25.1 Detailed Description	54
4.25.2 Constructor & Destructor Documentation	54
4.25.2.1 Task()	54
4.25.3 Member Function Documentation	54
4.25.3.1 execute()	54
4.25.3.2 shouldFinish()	54
4.25.3.3 shouldStop()	55
4.26 Ragot::Terrain Class Reference	55
4.26.1 Detailed Description	55
4.26.2 Constructor & Destructor Documentation	55
4.26.2.1 Terrain()	55
4.26.3 Member Function Documentation	56
4.26.3.1 set_camera()	56
4.27 Ragot::Texture2D< COLOR_FORMAT > Class Template Reference	56
4.27.1 Detailed Description	57
4.27.2 Constructor & Destructor Documentation	57
4.27.2.1 Texture2D()	57
4.27.3 Member Function Documentation	58
4.27.3.1 bind()	58
4.27.3.2 create_texture_2d()	58
4.27.3.3 is_ok()	58
4.27.3.4 load_image()	58
4.28 Ragot::Texture_Cube Class Reference	59
4.28.1 Detailed Description	60
4.28.2 Constructor & Destructor Documentation	60
4.28.2.1 Texture_Cube()	60

4.28.3 Member Function Documentation	60
4.28.3.1 bind()	60
4.29 Ragot::Transform_Component Class Reference	61
4.29.1 Detailed Description	62
4.29.2 Member Function Documentation	62
4.29.2.1 get_parent()	62
4.29.2.2 get_position()	62
4.29.2.3 get_rotation()	62
4.29.2.4 get_scale()	63
4.29.2.5 get_transform_matrix()	63
4.29.2.6 set_parent()	63
4.29.2.7 set_position()	63
4.29.2.8 set_rotation()	63
4.29.2.9 set_scale()	64
4.30 Ragot::Vertex_Shader Class Reference	64
4.30.1 Detailed Description	65
4.30.2 Constructor & Destructor Documentation	65
4.30.2.1 Vertex_Shader()	65
4.31 Ragot::Window Class Reference	65
4.31.1 Detailed Description	66
4.31.2 Member Enumeration Documentation	66
4.31.2.1 Position	66
4.31.3 Constructor & Destructor Documentation	67
4.31.3.1 Window() [1/3]	67
4.31.3.2 Window() [2/3]	67
4.31.3.3 Window() [3/3]	67
4.31.4 Member Function Documentation	68
4.31.4.1 get_height()	68
4.31.4.2 get_width()	68
4.31.4.3 operator=()	68
5 File Documentation	69
5.1 Ambient.hpp	69
5.2 Camera.hpp	71
5.3 Color.hpp	72
5.4 Color_Buffer.hpp	73
5.5 Component.hpp	74
5.6 Entity.hpp	76
5.7 Mesh.hpp	76
5.8 MyKernel.hpp	79
5.9 MySystem.hpp	80
5.10 Postprocess.hpp	81

5.11 Shader_Program.hpp	82
5.12 Task.hpp	84
5.13 Window.hpp	86

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Ragot::Camera	8
Ragot::Color_Buffer< COLOR >	15
Ragot::Component	19
Ragot::Model_Component	37
Ragot::Transform_Component	61
std::enable_shared_from_this	
Ragot::Entity	23
Ragot::Frame_Buffer	28
Ragot::Kernel	29
Ragot::Light	29
Ragot::AreaLight	7
Ragot::DirectionalLight	22
Ragot::PointLight	42
Ragot::Material	32
Ragot::Mesh	34
Ragot::Window::OpenGL_Context_Settings	41
Ragot::Rgba8888	43
Ragot::Scene	43
Ragot::Shader	47
Ragot::Fragment_Shader	26
Ragot::Vertex_Shader	64
Ragot::Shader_Program	49
Ragot::Skybox	50
Ragot::System	52
Ragot::Task	53
Ragot::Critical_Task	20
Ragot::Light_Task	30
Ragot::Once_Task	39
Ragot::Terrain	55
Ragot::Texture2D< COLOR_FORMAT >	56
Ragot::Texture2D< Rgba8888 >	56
Ragot::Texture_Cube	59
Ragot::Window	65

Chapter 2

Class Index

2.1 Class List

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

Ragot::AreaLight	
Class for area light	7
Ragot::Camera	
Class for managing a camera in OpenGL	8
Ragot::Color_Buffer< COLOR >	
Template class for managing a color buffer	15
Ragot::Component	
Base class for components	19
Ragot::Critical_Task	
Class to execute critical tasks such as rendering, which the Kernel can pause other tasks to execute. These tasks run in the main thread	20
Ragot::DirectionalLight	
Class for directional light	22
Ragot::Entity	
Class for managing entities in a scene	23
Ragot::Fragment_Shader	
Class for managing an OpenGL fragment shader	26
Ragot::Frame_Buffer	
Class for managing a frame buffer in OpenGL	28
Ragot::Kernel	
Class for managing the kernel that executes tasks	29
Ragot::Light	
Base class for different types of lights	29
Ragot::Light_Task	
Class to execute cyclic tasks such as Update or Input	30
Ragot::Material	
Class for managing a material	32
Ragot::Mesh	
Class for managing a 3D mesh	34
Ragot::Model_Component	
Component for managing models	37
Ragot::Once_Task	
Class for tasks that are executed only once	39
Ragot::Window::OpenGL_Context_Settings	
Struct for OpenGL context settings	41

Ragot::PointLight	
Class for point light	42
Ragot::Rgba8888	
Union for managing RGBA color values	43
Ragot::Scene	
Class for managing a scene in OpenGL	43
Ragot::Shader	
Class for managing an OpenGL shader	47
Ragot::Shader_Program	
Class for managing an OpenGL shader program	49
Ragot::Skybox	
Class for rendering a skybox in the scene	50
Ragot::System	
Class for managing the system in OpenGL	52
Ragot::Task	
Base class for managing tasks	53
Ragot::Terrain	
Class for rendering a terrain in the scene	55
Ragot::Texture2D< COLOR_FORMAT >	
Template class for managing a 2D texture	56
Ragot::Texture_Cube	
Class for managing a cube texture	59
Ragot::Transform_Component	
Component for managing transformations	61
Ragot::Vertex_Shader	
Class for managing an OpenGL vertex shader	64
Ragot::Window	
Class for managing an SDL window with OpenGL context	65

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Ambient.hpp	69
Camera.hpp	71
Color.hpp	72
Color_Buffer.hpp	73
Component.hpp	74
Entity.hpp	76
Mesh.hpp	76
MyKernel.hpp	79
MySystem.hpp	80
Postprocess.hpp	81
Shader_Program.hpp	82
Task.hpp	84
Window.hpp	86

Chapter 4

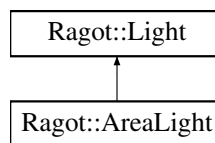
Class Documentation

4.1 Ragot::AreaLight Class Reference

Class for area light.

```
#include <Ambient.hpp>
```

Inheritance diagram for Ragot::AreaLight:



Public Member Functions

- [AreaLight](#) (const glm::vec3 &[color](#), const glm::vec3 &[position](#), const glm::vec3 &[size](#))
Constructor for the [AreaLight](#) class.

Public Member Functions inherited from [Ragot::Light](#)

- [Light](#) (const glm::vec3 &[color](#))
Constructor for the [Light](#) class.
- virtual **~Light** ()=default
Virtual destructor for the [Light](#) class.

Public Attributes

- glm::vec3 **position**
Position of the light.
- glm::vec3 **size**
Size of the area light.

Public Attributes inherited from [Ragot::Light](#)

- `glm::vec3 color`
Color of the light.

4.1.1 Detailed Description

Class for area light.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 [AreaLight\(\)](#)

```
Ragot::AreaLight::AreaLight (
    const glm::vec3 & color,
    const glm::vec3 & position,
    const glm::vec3 & size) [inline]
```

Constructor for the [AreaLight](#) class.

Parameters

<i>color</i>	Color of the light.
<i>position</i>	Position of the light.
<i>size</i>	Size of the area light.

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

- `Ambient.hpp`

4.2 [Ragot::Camera](#) Class Reference

Class for managing a camera in OpenGL.

```
#include <Camera.hpp>
```

Public Member Functions

- [Camera](#) (float ratio=1.f)
Constructor with default ratio.
- [Camera](#) (float near_z, float far_z, float ratio=1.f)
Constructor with near and far clipping planes.
- [Camera](#) (float fov_degrees, float near_z, float far_z, float ratio)
Constructor with field of view, near and far clipping planes, and aspect ratio.
- float [get_fov](#) () const
Gets the field of view.
- float [get_near_z](#) () const
Gets the near clipping plane.
- float [get_far_z](#) () const
Gets the far clipping plane.
- float [get_ratio](#) () const
Gets the aspect ratio.
- const Point & [get_location](#) () const
Gets the location of the camera.
- const Point & [get_target](#) () const
Gets the target point the camera is looking at.
- void [set_fov](#) (float new_fov)
Sets the field of view and recalculates the projection matrix.
- void [set_near_z](#) (float new_near_z)
Sets the near clipping plane and recalculates the projection matrix.
- void [set_far_z](#) (float new_far_z)
Sets the far clipping plane and recalculates the projection matrix.
- void [set_ratio](#) (float new_ratio)
Sets the aspect ratio and recalculates the projection matrix.
- void [set_location](#) (float x, float y, float z)
Sets the location of the camera.
- void [set_target](#) (float x, float y, float z)
Sets the target point the camera is looking at.
- void [reset](#) (float new_fov, float new_near_z, float new_far_z, float new_ratio)
Resets the camera with new parameters and recalculates the projection matrix.
- void [move](#) (const glm::vec3 &translation)
Moves the camera by a given translation vector.
- void [rotate](#) (const glm::mat4 &rotation)
Rotates the camera by a given rotation matrix.
- const glm::mat4 & [get_projection_matrix](#) () const
Gets the projection matrix of the camera.
- glm::mat4 [get_transform_matrix_inverse](#) () const
Gets the inverse of the transformation matrix for the camera.

4.2.1 Detailed Description

Class for managing a camera in OpenGL.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Camera() [1/3]

```
Ragot::Camera::Camera (  
    float ratio = 1.f) [inline]
```

Constructor with default ratio.

Parameters

<i>ratio</i>	Aspect ratio (default is 1.0f).
--------------	---------------------------------

4.2.2.2 Camera() [2/3]

```
Ragot::Camera::Camera (
    float near_z,
    float far_z,
    float ratio = 1.f) [inline]
```

Constructor with near and far clipping planes.

Parameters

<i>near_z</i>	Near clipping plane.
<i>far_z</i>	Far clipping plane.
<i>ratio</i>	Aspect ratio (default is 1.0f).

4.2.2.3 Camera() [3/3]

```
Ragot::Camera::Camera (
    float fov_degrees,
    float near_z,
    float far_z,
    float ratio) [inline]
```

Constructor with field of view, near and far clipping planes, and aspect ratio.

Parameters

<i>fov_degrees</i>	Field of view in degrees.
<i>near_z</i>	Near clipping plane.
<i>far_z</i>	Far clipping plane.
<i>ratio</i>	Aspect ratio.

4.2.3 Member Function Documentation**4.2.3.1 get_far_z()**

```
float Ragot::Camera::get_far_z () const [inline]
```

Gets the far clipping plane.

Returns

Far clipping plane.

4.2.3.2 get_fov()

```
float Ragot::Camera::get_fov () const [inline]
```

Gets the field of view.

Returns

Field of view in degrees.

4.2.3.3 get_location()

```
const Point & Ragot::Camera::get_location () const [inline]
```

Gets the location of the camera.

Returns

Location of the camera.

4.2.3.4 get_near_z()

```
float Ragot::Camera::get_near_z () const [inline]
```

Gets the near clipping plane.

Returns

Near clipping plane.

4.2.3.5 get_projection_matrix()

```
const glm::mat4 & Ragot::Camera::get_projection_matrix () const [inline]
```

Gets the projection matrix of the camera.

Returns

Projection matrix.

4.2.3.6 get_ratio()

```
float Ragot::Camera::get_ratio () const [inline]
```

Gets the aspect ratio.

Returns

Aspect ratio.

4.2.3.7 get_target()

```
const Point & Ragot::Camera::get_target () const [inline]
```

Gets the target point the camera is looking at.

Returns

Target point.

4.2.3.8 get_transform_matrix_inverse()

```
glm::mat4 Ragot::Camera::get_transform_matrix_inverse () const [inline]
```

Gets the inverse of the transformation matrix for the camera.

Returns

Inverse of the transformation matrix.

4.2.3.9 move()

```
void Ragot::Camera::move (
    const glm::vec3 & translation) [inline]
```

Moves the camera by a given translation vector.

Parameters

<i>translation</i>	Translation vector.
--------------------	---------------------

4.2.3.10 reset()

```
void Ragot::Camera::reset (
    float new_fov,
    float new_near_z,
    float new_far_z,
    float new_ratio) [inline]
```

Resets the camera with new parameters and recalculates the projection matrix.

Parameters

<i>new_fov</i>	New field of view in degrees.
<i>new_near_z</i>	New near clipping plane.
<i>new_far_z</i>	New far clipping plane.
<i>new_ratio</i>	New aspect ratio.

4.2.3.11 rotate()

```
void Ragot::Camera::rotate (
    const glm::mat4 & rotation) [inline]
```

Rotates the camera by a given rotation matrix.

Parameters

<i>rotation</i>	Rotation matrix.
-----------------	------------------

4.2.3.12 set_far_z()

```
void Ragot::Camera::set_far_z (  
    float new_far_z) [inline]
```

Sets the far clipping plane and recalculates the projection matrix.

Parameters

<i>new_far_z</i>	New far clipping plane.
------------------	-------------------------

4.2.3.13 set_fov()

```
void Ragot::Camera::set_fov (  
    float new_fov) [inline]
```

Sets the field of view and recalculates the projection matrix.

Parameters

<i>new_fov</i>	New field of view in degrees.
----------------	-------------------------------

4.2.3.14 set_location()

```
void Ragot::Camera::set_location (  
    float x,  
    float y,  
    float z) [inline]
```

Sets the location of the camera.

Parameters

<i>x</i>	X-coordinate of the location.
<i>y</i>	Y-coordinate of the location.
<i>z</i>	Z-coordinate of the location.

4.2.3.15 set_near_z()

```
void Ragot::Camera::set_near_z (  
    float new_near_z) [inline]
```

Sets the near clipping plane and recalculates the projection matrix.

Parameters

<i>new_near</i> ↔ _z	New near clipping plane.
-------------------------	--------------------------

4.2.3.16 set_ratio()

```
void Ragot::Camera::set_ratio (
    float new_ratio) [inline]
```

Sets the aspect ratio and recalculates the projection matrix.

Parameters

<i>new_ratio</i>	New aspect ratio.
------------------	-------------------

4.2.3.17 set_target()

```
void Ragot::Camera::set_target (
    float x,
    float y,
    float z) [inline]
```

Sets the target point the camera is looking at.

Parameters

<i>x</i>	X-coordinate of the target.
<i>y</i>	Y-coordinate of the target.
<i>z</i>	Z-coordinate of the target.

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

- Camera.hpp

4.3 Ragot::Color_Buffer< COLOR > Class Template Reference

Template class for managing a color buffer.

```
#include <Color_Buffer.hpp>
```

Public Types

- using **Color** = COLOR
Type alias for the color format.

Public Member Functions

- [Color_Buffer](#) (unsigned width, unsigned height)
Constructor for the [Color_Buffer](#) class.
- unsigned [get_width](#) () const
Gets the width of the buffer.
- unsigned [get_height](#) () const
Gets the height of the buffer.
- [Color](#) * [colors](#) ()
Gets a pointer to the color data.
- const [Color](#) * [colors](#) () const
Gets a constant pointer to the color data.
- [Color](#) & [get](#) (unsigned offset)
Gets the color at a specific offset.
- const [Color](#) & [get](#) (unsigned offset) const
Gets the color at a specific offset (constant version).
- void [set](#) (unsigned offset, const [Color](#) &color)
Sets the color at a specific offset.

4.3.1 Detailed Description

```
template<typename COLOR>
class Ragot::Color_Buffer< COLOR >
```

Template class for managing a color buffer.

Template Parameters

<i>COLOR</i>	The color format for the buffer.
--------------	----------------------------------

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Color_Buffer()

```
template<typename COLOR>
Ragot::Color_Buffer< COLOR >::Color_Buffer (
    unsigned width,
    unsigned height) [inline]
```

Constructor for the [Color_Buffer](#) class.

Parameters

<i>width</i>	Width of the buffer.
<i>height</i>	Height of the buffer.

4.3.3 Member Function Documentation

4.3.3.1 colors() [1/2]

```
template<typename COLOR>
Color * Ragot::Color_Buffer< COLOR >::colors () [inline]
```

Gets a pointer to the color data.

Returns

Pointer to the color data.

4.3.3.2 colors() [2/2]

```
template<typename COLOR>
const Color * Ragot::Color_Buffer< COLOR >::colors () const [inline]
```

Gets a constant pointer to the color data.

Returns

Constant pointer to the color data.

4.3.3.3 get() [1/2]

```
template<typename COLOR>
Color & Ragot::Color_Buffer< COLOR >::get (
    unsigned offset) [inline]
```

Gets the color at a specific offset.

Parameters

<i>offset</i>	The offset in the buffer.
---------------	---------------------------

Returns

Reference to the color at the specified offset.

4.3.3.4 get() [2/2]

```
template<typename COLOR>
const Color & Ragot::Color_Buffer< COLOR >::get (
    unsigned offset) const [inline]
```

Gets the color at a specific offset (constant version).

Parameters

<i>offset</i>	The offset in the buffer.
---------------	---------------------------

Returns

Constant reference to the color at the specified offset.

4.3.3.5 get_height()

```
template<typename COLOR>
unsigned Ragot::Color_Buffer< COLOR >::get_height () const [inline]
```

Gets the height of the buffer.

Returns

Height of the buffer.

4.3.3.6 get_width()

```
template<typename COLOR>
unsigned Ragot::Color_Buffer< COLOR >::get_width () const [inline]
```

Gets the width of the buffer.

Returns

Width of the buffer.

4.3.3.7 set()

```
template<typename COLOR>
void Ragot::Color_Buffer< COLOR >::set (
    unsigned offset,
    const Color & color) [inline]
```

Sets the color at a specific offset.

Parameters

<i>offset</i>	The offset in the buffer.
<i>color</i>	The color to set.

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

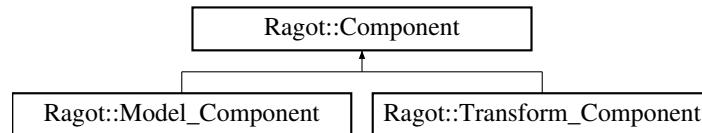
- Color_Buffer.hpp

4.4 Ragot::Component Class Reference

Base class for components.

```
#include <Component.hpp>
```

Inheritance diagram for Ragot::Component:



Public Member Functions

- virtual `~Component()` = default
Virtual destructor for the [Component](#) class.
- `std::shared_ptr< Entity > get_entity() const`
Gets the entity associated with this component.
- `void set_entity(std::shared_ptr< Entity > ent)`
Sets the entity associated with this component.
- `bool get_has_task() const`
Checks if the component has a task.

Protected Attributes

- `bool has_task = false`
Indicates whether the component has a task.

4.4.1 Detailed Description

Base class for components.

4.4.2 Member Function Documentation

4.4.2.1 get_entity()

```
std::shared_ptr< Entity > Ragot::Component::get_entity() const [inline]
```

Gets the entity associated with this component.

Returns

Shared pointer to the associated entity.

4.4.2.2 get_has_task()

```
bool Ragot::Component::get_has_task () const [inline]
```

Checks if the component has a task.

Returns

True if the component has a task, false otherwise.

4.4.2.3 set_entity()

```
void Ragot::Component::set_entity (
    std::shared_ptr< Entity > ent) [inline]
```

Sets the entity associated with this component.

Parameters

<i>ent</i>	Shared pointer to the entity to associate.
------------	--

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

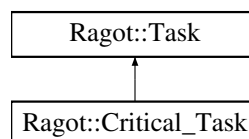
- Component.hpp

4.5 Ragot::Critical_Task Class Reference

Class to execute critical tasks such as rendering, which the [Kernel](#) can pause other tasks to execute. These tasks run in the main thread.

```
#include <Task.hpp>
```

Inheritance diagram for Ragot::Critical_Task:



Public Member Functions

- [Critical_Task](#) (function< void()> [task_func](#))
Constructor that calls the base class constructor.
- void [execute](#) () override
Specific execution function for critical tasks.

Public Member Functions inherited from Ragot::Task

- [Task](#) (function< void()> [task_func](#))
Constructor that accepts the function to run for this task.
- virtual `~Task` ()=default
Default destructor.
- void `stop_execution` ()
Stops execution of all tasks, even if executed by one thread.
- void `stop` ()
Stops execution temporarily for critical sections of code.
- void `resume` ()
Resumes execution after a stop.

Additional Inherited Members

Protected Member Functions inherited from Ragot::Task

- bool `shouldStop` ()
Checks if the task should stop execution.
- bool `shouldFinish` ()
Checks if the task should finish execution.
- void `wait_for_resume` ()
Waits for resume signal to continue execution.

Protected Attributes inherited from Ragot::Task

- function< void()> `task_func`
Function to run for this task.

4.5.1 Detailed Description

Class to execute critical tasks such as rendering, which the [Kernel](#) can pause other tasks to execute. These tasks run in the main thread.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Critical_Task()

```
Ragot::Critical_Task::Critical_Task (
    function< void()> task_func) [inline]
```

Constructor that calls the base class constructor.

Parameters

<code>task_func</code>	Function to run for this task.
------------------------	--------------------------------

4.5.3 Member Function Documentation

4.5.3.1 execute()

```
void Ragot::Critical_Task::execute () [override], [virtual]
```

Specific execution function for critical tasks.

Implements [Ragot::Task](#).

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

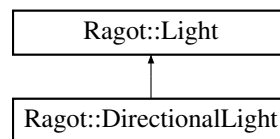
- Task.hpp
- Task.cpp

4.6 Ragot::DirectionalLight Class Reference

Class for directional light.

```
#include <Ambient.hpp>
```

Inheritance diagram for Ragot::DirectionalLight:



Public Member Functions

- [DirectionalLight](#) (const glm::vec3 &[color](#), const glm::vec3 [direction](#))
Constructor for the [DirectionalLight](#) class.

Public Member Functions inherited from [Ragot::Light](#)

- [Light](#) (const glm::vec3 &[color](#))
Constructor for the [Light](#) class.
- virtual **~Light** ()=default
Virtual destructor for the [Light](#) class.

Public Attributes

- glm::vec3 **direction**
Direction of the light.

Public Attributes inherited from Ragot::Light

- glm::vec3 **color**
Color of the light.

4.6.1 Detailed Description

Class for directional light.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 DirectionalLight()

```
Ragot::DirectionalLight::DirectionalLight (
    const glm::vec3 & color,
    const glm::vec3 direction) [inline]
```

Constructor for the [DirectionalLight](#) class.

Parameters

<i>color</i>	Color of the light.
<i>direction</i>	Direction of the light.

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

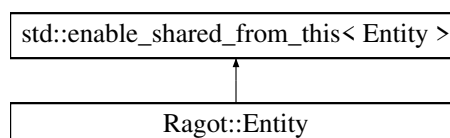
- Ambient.hpp

4.7 Ragot::Entity Class Reference

Class for managing entities in a scene.

```
#include <Entity.hpp>
```

Inheritance diagram for Ragot::Entity:



Public Member Functions

- void `set_scene` (`Scene` *scene)
Sets the scene for this entity.
- const `Scene` * `get_scene` ()
Gets the scene this entity belongs to.
- const `Scene` * `get_scene` () const
Gets the scene this entity belongs to (const version).
- void `add_component` (shared_ptr< `Component` > component, const string &name)
Adds a component to the entity.
- void `remove_component` (const string &name)
Removes a component from the entity.
- void `add_child` (shared_ptr< `Entity` > child)
Adds a child entity.
- void `remove_child` (shared_ptr< `Entity` > child)
Removes a child entity.
- void `set_transform_parent` (`Transform_Component` *parent)
Sets the parent transform component.
- const map< string, shared_ptr< `Component` > > & `get_components` () const
Gets the components associated with this entity.

Public Attributes

- `Transform_Component` `transform`
Transform component of the entity.

4.7.1 Detailed Description

Class for managing entities in a scene.

4.7.2 Member Function Documentation

4.7.2.1 `add_child()`

```
void Ragot::Entity::add_child (
    shared_ptr< Entity > child) [inline]
```

Adds a child entity.

Parameters

<i>child</i>	Shared pointer to the child entity.
--------------	-------------------------------------

4.7.2.2 `add_component()`

```
void Ragot::Entity::add_component (
    shared_ptr< Component > component,
    const string & name)
```

Adds a component to the entity.

Parameters

<i>component</i>	Shared pointer to the component.
<i>name</i>	Name of the component.

4.7.2.3 get_components()

```
const map< string, shared_ptr< Component > > & Ragot::Entity::get_components () const [inline]
```

Gets the components associated with this entity.

Returns

Map of components.

4.7.2.4 get_scene() [1/2]

```
const Scene * Ragot::Entity::get_scene () [inline]
```

Gets the scene this entity belongs to.

Returns

Pointer to the scene.

4.7.2.5 get_scene() [2/2]

```
const Scene * Ragot::Entity::get_scene () const [inline]
```

Gets the scene this entity belongs to (const version).

Returns

Pointer to the scene.

4.7.2.6 remove_child()

```
void Ragot::Entity::remove_child (  
    shared_ptr< Entity > child) [inline]
```

Removes a child entity.

Parameters

<i>child</i>	Shared pointer to the child entity.
--------------	-------------------------------------

4.7.2.7 remove_component()

```
void Ragot::Entity::remove_component (  
    const string & name)
```

Removes a component from the entity.

Parameters

<i>name</i>	Name of the component.
-------------	------------------------

4.7.2.8 set_scene()

```
void Ragot::Entity::set_scene (
    Scene * scene) [inline]
```

Sets the scene for this entity.

Parameters

<i>scene</i>	Pointer to the scene.
--------------	-----------------------

4.7.2.9 set_transform_parent()

```
void Ragot::Entity::set_transform_parent (
    Transform_Component * parent) [inline]
```

Sets the parent transform component.

Parameters

<i>parent</i>	Pointer to the parent transform component.
---------------	--

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

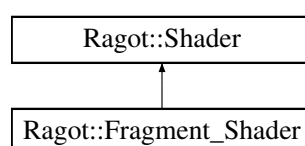
- Entity.hpp
- Entity.cpp

4.8 Ragot::Fragment_Shader Class Reference

Class for managing an OpenGL fragment shader.

```
#include <Shader_Program.hpp>
```

Inheritance diagram for Ragot::Fragment_Shader:



Public Member Functions

- [Fragment_Shader](#) (const vector< string > &source_code)
Constructor for the [Fragment_Shader](#) class.

Public Member Functions inherited from [Ragot::Shader](#)

- **Shader** ()=delete
Deleted default constructor.
- **~Shader** ()
Destructor for the [Shader](#) class.
- GLuint [get_id](#) () const
Gets the shader ID.
- string * [get_error](#) ()
Gets the compilation error message.
- bool [is_ok](#) () const
Checks if the shader is compiled successfully.

Additional Inherited Members

Protected Member Functions inherited from [Ragot::Shader](#)

- [Shader](#) (const vector< string > &source_code, GLenum type)
Constructor for the [Shader](#) class.
- GLuint [compile_shader](#) ()
Compiles the shader.
- void **show_compilation_error** ()
Displays compilation errors.

4.8.1 Detailed Description

Class for managing an OpenGL fragment shader.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 [Fragment_Shader](#)()

```
Ragot::Fragment_Shader::Fragment_Shader (
    const vector< string > & source_code) [inline]
```

Constructor for the [Fragment_Shader](#) class.

Parameters

<i>source_code</i>	Vector of fragment shader source code.
--------------------	--

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

- [Shader_Program.hpp](#)

4.9 Ragot::Frame_Buffer Class Reference

Class for managing a frame buffer in OpenGL.

```
#include <Postprocess.hpp>
```

Public Member Functions

- [Frame_Buffer](#) (unsigned width, unsigned height)
Constructor for the [Frame_Buffer](#) class.
- **Frame_Buffer** ()=delete
Default constructor is deleted.
- **~Frame_Buffer** ()
Destructor for the [Frame_Buffer](#) class.
- void **bind_frame_buffer** () const
Binds the frame buffer.
- void **unbind_frame_buffer** () const
Unbinds the frame buffer.
- void **bind_texture** () const
Binds the texture.
- void **unbind_texture** () const
Unbinds the texture.
- void **render** ()
Renders the frame buffer.

4.9.1 Detailed Description

Class for managing a frame buffer in OpenGL.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Frame_Buffer()

```
Ragot::Frame_Buffer::Frame_Buffer (
    unsigned width,
    unsigned height)
```

Constructor for the [Frame_Buffer](#) class.

Parameters

<i>width</i>	Width of the frame buffer.
<i>height</i>	Height of the frame buffer.

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

- Postprocess.hpp
- Postprocess.cpp

4.10 Ragot::Kernel Class Reference

Class for managing the kernel that executes tasks.

```
#include <MyKernel.hpp>
```

Public Member Functions

- void **add** (std::shared_ptr< **Task** > new_task)
Adds a new task to the kernel.
- void **run** ()
Runs the kernel, executing all tasks.
- void **stop** ()
Stops the kernel and all tasks.
- void **execute_critical** ()
Executes all critical functions at once.

4.10.1 Detailed Description

Class for managing the kernel that executes tasks.

4.10.2 Member Function Documentation

4.10.2.1 add()

```
void Ragot::Kernel::add (
    std::shared_ptr< Task > new_task)
```

Adds a new task to the kernel.

Parameters

<i>new_task</i>	Shared pointer to the new task to add.
-----------------	--

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

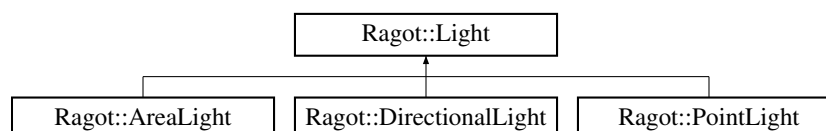
- MyKernel.hpp
- MyKernel.cpp

4.11 Ragot::Light Class Reference

Base class for different types of lights.

```
#include <Ambient.hpp>
```

Inheritance diagram for Ragot::Light:



Public Member Functions

- [Light](#) (const glm::vec3 &[color](#))
Constructor for the [Light](#) class.
- virtual ~**Light** ()=default
Virtual destructor for the [Light](#) class.

Public Attributes

- glm::vec3 **color**
Color of the light.

4.11.1 Detailed Description

Base class for different types of lights.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Light()

```
Ragot::Light::Light (
    const glm::vec3 & color) [inline]
```

Constructor for the [Light](#) class.

Parameters

<i>color</i>	Color of the light.
--------------	---------------------

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

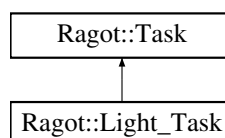
- Ambient.hpp

4.12 Ragot::Light_Task Class Reference

Class to execute cyclic tasks such as Update or Input.

```
#include <Task.hpp>
```

Inheritance diagram for Ragot::Light_Task:



Public Member Functions

- [Light_Task](#) (function< void()> [task_func](#))
Constructor that calls the base class constructor.
- void [execute](#) () override
Specific execution function for light tasks.

Public Member Functions inherited from [Ragot::Task](#)

- [Task](#) (function< void()> [task_func](#))
Constructor that accepts the function to run for this task.
- virtual [~Task](#) ()=default
Default destructor.
- void [stop_execution](#) ()
Stops execution of all tasks, even if executed by one thread.
- void [stop](#) ()
Stops execution temporarily for critical sections of code.
- void [resume](#) ()
Resumes execution after a stop.

Additional Inherited Members

Protected Member Functions inherited from [Ragot::Task](#)

- bool [shouldStop](#) ()
Checks if the task should stop execution.
- bool [shouldFinish](#) ()
Checks if the task should finish execution.
- void [wait_for_resume](#) ()
Waits for resume signal to continue execution.

Protected Attributes inherited from [Ragot::Task](#)

- function< void()> [task_func](#)
Function to run for this task.

4.12.1 Detailed Description

Class to execute cyclic tasks such as Update or Input.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 [Light_Task](#)()

```
Ragot::Light_Task::Light_Task (
    function< void()> task\_func) [inline]
```

Constructor that calls the base class constructor.

Parameters

<code>task_func</code>	Function to run for this task.
------------------------	--------------------------------

4.12.3 Member Function Documentation

4.12.3.1 execute()

```
void Ragot::Light_Task::execute () [override], [virtual]
```

Specific execution function for light tasks.

Implements [Ragot::Task](#).

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

- Task.hpp
- Task.cpp

4.13 Ragot::Material Class Reference

Class for managing a material.

```
#include <Mesh.hpp>
```

Public Member Functions

- **Material** ()=delete
Deleted default constructor.
- [Material](#) (const vector< string > &source_code_vertex, const vector< string > &source_code_fragment, const string &texture_base_path)
Constructor for the [Material](#) class.
- **~Material** ()=default
Default destructor for the [Material](#) class.
- void **use_shader_program** ()
Uses the shader program.
- GLint [get_shader_program_uniform_location](#) (const string &uniform)
Gets the uniform location in the shader program.
- GLuint [get_shader_program_id](#) () const
Gets the shader program ID.
- const bool [bind_texture](#) () const
Binds the texture.
- const glm::vec3 [get_color](#) ()
Gets the color of the material.
- const float [get_shininess](#) ()
Gets the shininess of the material.

4.13.1 Detailed Description

Class for managing a material.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Material()

```
Ragot::Material::Material (
    const vector< string > & source_code_vertex,
    const vector< string > & source_code_fragment,
    const string & texture_base_path)
```

Constructor for the [Material](#) class.

Parameters

<i>source_code_vertex</i>	Vector of vertex shader source code.
<i>source_code_fragment</i>	Vector of fragment shader source code.
<i>texture_base_path</i>	Path to the base texture file.

4.13.3 Member Function Documentation

4.13.3.1 bind_texture()

```
const bool Ragot::Material::bind_texture () const [inline]
```

Binds the texture.

Returns

True if the texture is successfully bound, false otherwise.

4.13.3.2 get_color()

```
const glm::vec3 Ragot::Material::get_color () [inline]
```

Gets the color of the material.

Returns

Color of the material.

4.13.3.3 get_shader_program_id()

```
GLuint Ragot::Material::get_shader_program_id () const [inline]
```

Gets the shader program ID.

Returns

[Shader](#) program ID.

4.13.3.4 get_shader_program_uniform_location()

```
GLint Ragot::Material::get_shader_program_uniform_location (
    const string & uniform) [inline]
```

Gets the uniform location in the shader program.

Parameters

<i>uniform</i>	Name of the uniform.
----------------	----------------------

Returns

Uniform location.

4.13.3.5 `get_shininess()`

```
const float Ragot::Material::get_shininess () [inline]
```

Gets the shininess of the material.

Returns

Shininess of the material.

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

- Mesh.hpp
- Mesh.cpp

4.14 Ragot::Mesh Class Reference

Class for managing a 3D mesh.

```
#include <Mesh.hpp>
```

Public Member Functions

- **Mesh** ()=default
Default constructor for the [Mesh](#) class.
- **Mesh** (const std::string &mesh_file_path)
Constructor for the [Mesh](#) class.
- **~Mesh** ()
Destructor for the [Mesh](#) class.
- const vector< glm::vec3 > & [get_coordinates](#) () const
Gets the vertex coordinates.
- const vector< glm::vec3 > & [get_normals](#) () const
Gets the vertex normals.
- const vector< glm::vec2 > & [get_textures_uv](#) () const
Gets the texture coordinates.
- const vector< GLuint > & [get_indices](#) () const
Gets the indices.
- const GLuint [get_vao_id](#) () const
Gets the Vertex Array Object ID.
- const GLsizei [get_number_of_indices](#) () const
Gets the number of indices.

Protected Types

- enum {
[COORDINATES_VBO](#) , [NORMALS_VBO](#) , [TEXTURE_UVS_VBO](#) , [INDICES_EBO](#) ,
[VBO_COUNT](#) }
Enum for VBO indices.

Protected Attributes

- vector< glm::vec3 > **coordinates**
Vector of vertex coordinates.
- vector< glm::vec3 > **normals**
Vector of vertex normals.
- vector< glm::vec2 > **texture_coords**
Vector of texture coordinates.
- vector< GLuint > **indices**
Vector of indices.

4.14.1 Detailed Description

Class for managing a 3D mesh.

4.14.2 Member Enumeration Documentation

4.14.2.1 anonymous enum

anonymous enum [protected]

Enum for VBO indices.

Enumerator

COORDINATES_VBO	VBO index for coordinates.
NORMALS_VBO	VBO index for normals.
TEXTURE_UVS_VBO	VBO index for texture UVs.
INDICES_EBO	VBO index for indices.
VBO_COUNT	Total number of VBOs.

4.14.3 Constructor & Destructor Documentation

4.14.3.1 Mesh()

```
Ragot::Mesh::Mesh (
    const std::string & mesh_file_path)
```

Constructor for the [Mesh](#) class.

Parameters

<i>mesh_file_path</i>	Path to the mesh file.
-----------------------	------------------------

4.14.4 Member Function Documentation

4.14.4.1 `get_coordinates()`

```
const vector< glm::vec3 > & Ragot::Mesh::get_coordinates () const [inline]
```

Gets the vertex coordinates.

Returns

Vector of vertex coordinates.

4.14.4.2 `get_indices()`

```
const vector< GLuint > & Ragot::Mesh::get_indices () const [inline]
```

Gets the indices.

Returns

Vector of indices.

4.14.4.3 `get_normals()`

```
const vector< glm::vec3 > & Ragot::Mesh::get_normals () const [inline]
```

Gets the vertex normals.

Returns

Vector of vertex normals.

4.14.4.4 `get_number_of_indices()`

```
const GLsizei Ragot::Mesh::get_number_of_indices () const [inline]
```

Gets the number of indices.

Returns

Number of indices.

4.14.4.5 get_textures_uv()

```
const vector< glm::vec2 > & Ragot::Mesh::get_textures_uv () const [inline]
```

Gets the texture coordinates.

Returns

Vector of texture coordinates.

4.14.4.6 get_vao_id()

```
const GLuint Ragot::Mesh::get_vao_id () const [inline]
```

Gets the Vertex Array Object ID.

Returns

VAO ID.

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

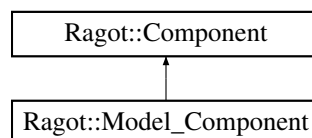
- Mesh.hpp
- Mesh.cpp

4.15 Ragot::Model_Component Class Reference

[Component](#) for managing models.

```
#include <Component.hpp>
```

Inheritance diagram for Ragot::Model_Component:



Public Member Functions

- **Model_Component** ()=delete
Deleted default constructor.
- [Model_Component](#) (const string &model_file_path, const string &texture_file_path)
Constructor for the [Model_Component](#) class.
- const GLuint [get_shader_program_id](#) () const
Gets the shader program ID.
- void [set_transparency](#) (bool trans)
Sets the transparency of the model.

Public Member Functions inherited from Ragot::Component

- virtual **~Component** ()=default
Virtual destructor for the [Component](#) class.
- std::shared_ptr< [Entity](#) > **get_entity** () const
Gets the entity associated with this component.
- void **set_entity** (std::shared_ptr< [Entity](#) > ent)
Sets the entity associated with this component.
- bool **get_has_task** () const
Checks if the component has a task.

Public Attributes

- [Critical_Task](#) **render_task**
Task for rendering the model.
- [Light_Task](#) **update_task**

Additional Inherited Members

Protected Attributes inherited from Ragot::Component

- bool **has_task** = false
Indicates whether the component has a task.

4.15.1 Detailed Description

[Component](#) for managing models.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Model_Component()

```
Ragot::Model_Component::Model_Component (
    const string & model_file_path,
    const string & texture_file_path)
```

Constructor for the [Model_Component](#) class.

Parameters

<i>model_file_path</i>	Path to the model file.
<i>texture_file_path</i>	Path to the texture file.

4.15.3 Member Function Documentation

4.15.3.1 get_shader_program_id()

```
const GLuint Ragot::Model_Component::get_shader_program_id () const [inline]
```

Gets the shader program ID.

Returns

Shader program ID.

4.15.3.2 set_transparency()

```
void Ragot::Model_Component::set_transparency (
    bool trans) [inline]
```

Sets the transparency of the model.

Parameters

<i>trans</i>	True to set the model as transparent, false otherwise.
--------------	--

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

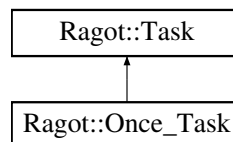
- Component.hpp
- Component.cpp

4.16 Ragot::Once_Task Class Reference

Class for tasks that are executed only once.

```
#include <Task.hpp>
```

Inheritance diagram for Ragot::Once_Task:



Public Member Functions

- [Once_Task](#) (function< void()> [task_func](#))
Constructor that calls the base class constructor.
- void [execute](#) () override
Specific execution function for once-only tasks.

Public Member Functions inherited from [Ragot::Task](#)

- [Task](#) (function< void()> [task_func](#))
Constructor that accepts the function to run for this task.
- virtual `~Task` ()=default
Default destructor.
- void **stop_execution** ()
Stops execution of all tasks, even if executed by one thread.
- void **stop** ()
Stops execution temporarily for critical sections of code.
- void **resume** ()
Resumes execution after a stop.

Additional Inherited Members

Protected Member Functions inherited from [Ragot::Task](#)

- bool [shouldStop](#) ()
Checks if the task should stop execution.
- bool [shouldFinish](#) ()
Checks if the task should finish execution.
- void **wait_for_resume** ()
Waits for resume signal to continue execution.

Protected Attributes inherited from [Ragot::Task](#)

- function< void()> **task_func**
Function to run for this task.

4.16.1 Detailed Description

Class for tasks that are executed only once.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 `Once_Task()`

```
Ragot::Once_Task::Once_Task (
    function< void()> task_func) [inline]
```

Constructor that calls the base class constructor.

Parameters

<i>task_func</i>	Function to run for this task.
------------------	--------------------------------

4.16.3 Member Function Documentation

4.16.3.1 execute()

```
void Ragot::Once_Task::execute () [override], [virtual]
```

Specific execution function for once-only tasks.

Implements [Ragot::Task](#).

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

- Task.hpp
- Task.cpp

4.17 Ragot::Window::OpenGL_Context_Settings Struct Reference

Struct for OpenGL context settings.

```
#include <Window.hpp>
```

Public Attributes

- unsigned **version_major** = 3
Major version of OpenGL.
- unsigned **version_minor** = 3
Minor version of OpenGL.
- bool **core_profile** = true
Core profile flag.
- unsigned **depth_buffer_size** = 24
Depth buffer size.
- unsigned **stencil_buffer_size** = 0
Stencil buffer size.
- bool **enable_vsync** = true
V-Sync enable flag.

4.17.1 Detailed Description

Struct for OpenGL context settings.

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

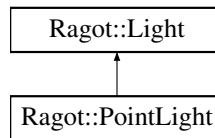
- Window.hpp

4.18 Ragot::PointLight Class Reference

Class for point light.

```
#include <Ambient.hpp>
```

Inheritance diagram for Ragot::PointLight:



Public Member Functions

- [PointLight](#) (const glm::vec3 &[color](#), const glm::vec3 &[position](#))
Constructor for the [PointLight](#) class.

Public Member Functions inherited from [Ragot::Light](#)

- [Light](#) (const glm::vec3 &[color](#))
Constructor for the [Light](#) class.
- virtual `~Light ()=default`
Virtual destructor for the [Light](#) class.

Public Attributes

- glm::vec3 [position](#)
Position of the light.

Public Attributes inherited from [Ragot::Light](#)

- glm::vec3 [color](#)
Color of the light.

4.18.1 Detailed Description

Class for point light.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 PointLight()

```
Ragot::PointLight::PointLight (
    const glm::vec3 & color,
    const glm::vec3 & position) [inline]
```

Constructor for the [PointLight](#) class.

Parameters

<i>color</i>	Color of the light.
<i>position</i>	Position of the light.

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

- Ambient.hpp

4.19 Ragot::Rgba8888 Union Reference

Union for managing RGBA color values.

```
#include <Color.hpp>
```

Public Types

- enum { **RED** , **GREEN** , **BLUE** , **ALPHA** }
Enum for the RGBA component indices.

Public Attributes

- uint32_t **value**
32-bit RGBA color value.
- uint8_t **components** [4]
Array of RGBA components.

4.19.1 Detailed Description

Union for managing RGBA color values.

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

- Color.hpp

4.20 Ragot::Scene Class Reference

Class for managing a scene in OpenGL.

```
#include <MySystem.hpp>
```

Public Member Functions

- void **resize** (int width, int height)
Resizes the scene.
- void **on_drag** (int pointer_x, int pointer_y)
Handles pointer drag events.
- void **on_click** (int pointer_x, int pointer_y, bool down)
Handles pointer click events.
- void **on_translation** (glm::vec3 translation)
Handles translation events.
- void **on_shift_pressed** (bool down)
Handles shift key press events.
- void **update** ()
Updates the scene.
- void **render** ()
Renders the scene.
- void **postprocess** ()
Post-processes the scene.
- **Scene** ()
Constructor for the [Scene](#) class.
- void **add_entities** (shared_ptr< [Entity](#) > entity, const string &name)
Adds an entity to the scene.
- void **remove_entities** (const string &name)
Removes an entity from the scene.
- shared_ptr< [Entity](#) > **get_entity** (const string &name) const
Gets an entity from the scene.
- shared_ptr< [Camera](#) > **get_camera** () const
Gets the camera of the scene.

4.20.1 Detailed Description

Class for managing a scene in OpenGL.

4.20.2 Member Function Documentation

4.20.2.1 add_entities()

```
void Ragot::Scene::add_entities (
    shared_ptr< Entity > entity,
    const string & name)
```

Adds an entity to the scene.

Parameters

<i>entity</i>	Shared pointer to the entity.
<i>name</i>	Name of the entity.

4.20.2.2 get_camera()

```
shared_ptr< Camera > Ragot::Scene::get_camera () const [inline]
```

Gets the camera of the scene.

Returns

Shared pointer to the camera.

4.20.2.3 get_entity()

```
shared_ptr< Entity > Ragot::Scene::get_entity (
    const string & name) const
```

Gets an entity from the scene.

Parameters

<i>name</i>	Name of the entity.
-------------	---------------------

Returns

Shared pointer to the entity.

4.20.2.4 on_click()

```
void Ragot::Scene::on_click (
    int pointer_x,
    int pointer_y,
    bool down)
```

Handles pointer click events.

Parameters

<i>pointer_x</i>	X-coordinate of the pointer.
<i>pointer_y</i>	Y-coordinate of the pointer.
<i>down</i>	Indicates if the pointer is pressed down.

4.20.2.5 on_drag()

```
void Ragot::Scene::on_drag (
    int pointer_x,
    int pointer_y)
```

Handles pointer drag events.

Parameters

<i>pointer</i> ↔ _x	X-coordinate of the pointer.
<i>pointer</i> ↔ _y	Y-coordinate of the pointer.

4.20.2.6 on_shift_pressed()

```
void Ragot::Scene::on_shift_pressed (  
    bool down)
```

Handles shift key press events.

Parameters

<i>down</i>	Indicates if the shift key is pressed down.
-------------	---

4.20.2.7 on_translation()

```
void Ragot::Scene::on_translation (  
    glm::vec3 translation)
```

Handles translation events.

Parameters

<i>translation</i>	Translation vector.
--------------------	---------------------

4.20.2.8 remove_entities()

```
void Ragot::Scene::remove_entities (  
    const string & name)
```

Removes an entity from the scene.

Parameters

<i>name</i>	Name of the entity.
-------------	---------------------

4.20.2.9 resize()

```
void Ragot::Scene::resize (  
    int width,  
    int height)
```

Resizes the scene.

Parameters

<i>width</i>	New width of the scene.
<i>height</i>	New height of the scene.

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

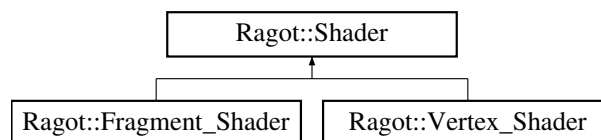
- MySystem.hpp
- MySystem.cpp

4.21 Ragot::Shader Class Reference

Class for managing an OpenGL shader.

```
#include <Shader_Program.hpp>
```

Inheritance diagram for Ragot::Shader:



Public Member Functions

- **Shader** ()=delete
Deleted default constructor.
- **~Shader** ()
Destructor for the [Shader](#) class.
- GLuint **get_id** () const
Gets the shader ID.
- string * **get_error** ()
Gets the compilation error message.
- bool **is_ok** () const
Checks if the shader is compiled successfully.

Protected Member Functions

- **Shader** (const vector< string > &source_code, GLenum type)
Constructor for the [Shader](#) class.
- GLuint **compile_shader** ()
Compiles the shader.
- void **show_compilation_error** ()
Displays compilation errors.

4.21.1 Detailed Description

Class for managing an OpenGL shader.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Shader()

```
Ragot::Shader::Shader (
    const vector< string > & source_code,
    GLenum type) [protected]
```

Constructor for the [Shader](#) class.

Parameters

<i>source_code</i>	Vector of shader source code.
<i>type</i>	Shader type (e.g., GL_VERTEX_SHADER, GL_FRAGMENT_SHADER).

4.21.3 Member Function Documentation

4.21.3.1 compile_shader()

```
GLuint Ragot::Shader::compile_shader () [protected]
```

Compiles the shader.

Returns

[Shader](#) ID.

4.21.3.2 get_error()

```
string * Ragot::Shader::get_error () [inline]
```

Gets the compilation error message.

Returns

Pointer to the error message string.

4.21.3.3 get_id()

```
GLuint Ragot::Shader::get_id () const [inline]
```

Gets the shader ID.

Returns

[Shader](#) ID.

4.21.3.4 is_ok()

```
bool Ragot::Shader::is_ok () const [inline]
```

Checks if the shader is compiled successfully.

Returns

True if compilation succeeded, false otherwise.

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

- Shader_Program.hpp
- Shader_Program.cpp

4.22 Ragot::Shader_Program Class Reference

Class for managing an OpenGL shader program.

```
#include <Shader_Program.hpp>
```

Public Member Functions

- [Shader_Program](#) (const vector< string > &source_code_vertex, const vector< string > &source_code_fragment)
Constructor for the [Shader_Program](#) class.
- **Shader_Program** ()=delete
Deleted default constructor.
- **~Shader_Program** ()
Destructor for the [Shader_Program](#) class.
- void **use** () const
Uses the shader program.
- GLuint **get_id** () const
Gets the shader program ID.
- GLuint **get_uniform_location** (string uniform_name) const
Gets the uniform location in the shader program.

4.22.1 Detailed Description

Class for managing an OpenGL shader program.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 Shader_Program()

```
Ragot::Shader_Program::Shader_Program (
    const vector< string > & source_code_vertex,
    const vector< string > & source_code_fragment)
```

Constructor for the [Shader_Program](#) class.

Parameters

<i>source_code_vertex</i>	Vector of vertex shader source code.
<i>source_code_fragment</i>	Vector of fragment shader source code.

4.22.3 Member Function Documentation

4.22.3.1 `get_id()`

```
GLuint Ragot::Shader_Program::get_id () const [inline]
```

Gets the shader program ID.

Returns

[Shader](#) program ID.

4.22.3.2 `get_uniform_location()`

```
GLuint Ragot::Shader_Program::get_uniform_location (
    string uniform_name) const [inline]
```

Gets the uniform location in the shader program.

Parameters

<i>uniform_name</i>	Name of the uniform.
---------------------	----------------------

Returns

Uniform location.

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

- Shader_Program.hpp
- Shader_Program.cpp

4.23 Ragot::Skybox Class Reference

Class for rendering a skybox in the scene.

```
#include <Ambient.hpp>
```

Public Member Functions

- [Skybox](#) (const string &texture_path)
Constructor for the [Skybox](#) class.
- [~Skybox](#) ()
Destructor for the [Skybox](#) class.
- void [set_camera](#) (shared_ptr< [Camera](#) > cam)
Sets the camera for the skybox.
- void [render](#) ()
Renders the skybox.

4.23.1 Detailed Description

Class for rendering a skybox in the scene.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Skybox()

```
Ragot::Skybox::Skybox (
    const string & texture_path)
```

Constructor for the [Skybox](#) class.

Parameters

<i>texture_path</i>	Path to the texture for the skybox.
---------------------	-------------------------------------

4.23.3 Member Function Documentation

4.23.3.1 set_camera()

```
void Ragot::Skybox::set_camera (
    shared_ptr< Camera > cam) [inline]
```

Sets the camera for the skybox.

Parameters

<i>cam</i>	Shared pointer to the camera.
------------	-------------------------------

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

- Ambient.hpp
- Ambient.cpp

4.24 Ragot::System Class Reference

Class for managing the system in OpenGL.

```
#include <MySystem.hpp>
```

Public Member Functions

- [System](#) (const string &Window_Name, const int width, const int height)
Constructor for the [System](#) class.
- **System** ()
Default constructor for the [System](#) class.
- **~System** ()
Destructor for the [System](#) class.
- void [add_entities](#) (shared_ptr< [Entity](#) > entity, const string &name)
Adds an entity to the system.
- void **run** ()
Runs the system.
- void **stop** ()
Stops the system.

4.24.1 Detailed Description

Class for managing the system in OpenGL.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 System()

```
Ragot::System::System (
    const string & Window_Name,
    const int width,
    const int height)
```

Constructor for the [System](#) class.

Parameters

<i>Window_Name</i>	Name of the window.
<i>width</i>	Width of the window.
<i>height</i>	Height of the window.

4.24.3 Member Function Documentation

4.24.3.1 add_entities()

```
void Ragot::System::add_entities (
    shared_ptr< Entity > entity,
    const string & name)
```

Adds an entity to the system.

Parameters

<i>entity</i>	Shared pointer to the entity.
<i>name</i>	Name of the entity.

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

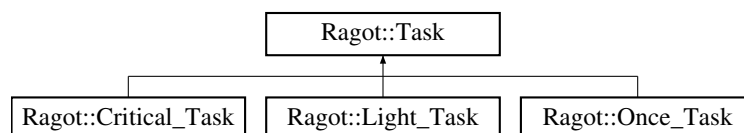
- MySystem.hpp
- MySystem.cpp

4.25 Ragot::Task Class Reference

Base class for managing tasks.

```
#include <Task.hpp>
```

Inheritance diagram for Ragot::Task:



Public Member Functions

- **Task** (function< void()> **task_func**)
Constructor that accepts the function to run for this task.
- virtual **~Task** ()=default
Default destructor.
- virtual void **execute** ()=0
Virtual function to execute the task, to be overridden by derived classes.
- void **stop_execution** ()
Stops execution of all tasks, even if executed by one thread.
- void **stop** ()
Stops execution temporarily for critical sections of code.
- void **resume** ()
Resumes execution after a stop.

Protected Member Functions

- bool **shouldStop** ()
Checks if the task should stop execution.
- bool **shouldFinish** ()
Checks if the task should finish execution.
- void **wait_for_resume** ()
Waits for resume signal to continue execution.

Protected Attributes

- `function< void()> task_func`
Function to run for this task.

4.25.1 Detailed Description

Base class for managing tasks.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Task()

```
Ragot::Task::Task (  
    function< void()> task_func) [inline]
```

Constructor that accepts the function to run for this task.

Parameters

<code>task_func</code>	Function to run for this task.
------------------------	--------------------------------

4.25.3 Member Function Documentation

4.25.3.1 execute()

```
virtual void Ragot::Task::execute () [pure virtual]
```

Virtual function to execute the task, to be overridden by derived classes.

Implemented in [Ragot::Critical_Task](#), [Ragot::Light_Task](#), and [Ragot::Once_Task](#).

4.25.3.2 shouldFinish()

```
bool Ragot::Task::shouldFinish () [inline], [protected]
```

Checks if the task should finish execution.

Returns

True if the task should finish, false otherwise.

4.25.3.3 shouldStop()

```
bool Ragot::Task::shouldStop () [inline], [protected]
```

Checks if the task should stop execution.

Returns

True if the task should stop, false otherwise.

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

- Task.hpp
- Task.cpp

4.26 Ragot::Terrain Class Reference

Class for rendering a terrain in the scene.

```
#include <Ambient.hpp>
```

Public Member Functions

- [Terrain](#) (float width, float depth, unsigned x_slices, unsigned z_slices)
Constructor for the [Terrain](#) class.
- [~Terrain](#) ()
Destructor for the [Terrain](#) class.
- void [set_camera](#) (shared_ptr< [Camera](#) > cam)
Sets the camera for the terrain.
- void [render](#) ()
Renders the terrain.

4.26.1 Detailed Description

Class for rendering a terrain in the scene.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 Terrain()

```
Ragot::Terrain::Terrain (  
    float width,  
    float depth,  
    unsigned x_slices,  
    unsigned z_slices)
```

Constructor for the [Terrain](#) class.

Parameters

<i>width</i>	Width of the terrain.
<i>depth</i>	Depth of the terrain.
<i>x_slices</i>	Number of slices along the x-axis.
<i>z_slices</i>	Number of slices along the z-axis.

4.26.3 Member Function Documentation**4.26.3.1 set_camera()**

```
void Ragot::Terrain::set_camera (
    shared_ptr< Camera > cam) [inline]
```

Sets the camera for the terrain.

Parameters

<i>cam</i>	Shared pointer to the camera.
------------	-------------------------------

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

- Ambient.hpp
- Ambient.cpp

4.27 Ragot::Texture2D< COLOR_FORMAT > Class Template Reference

Template class for managing a 2D texture.

```
#include <Mesh.hpp>
```

Public Member Functions

- [Texture2D](#) (const string &texture_base_path)
Constructor for the [Texture2D](#) class.
- [~Texture2D](#) ()
Destructor for the [Texture2D](#) class.
- bool [is_ok](#) () const
Checks if the texture is loaded.
- virtual bool [bind](#) () const
Binds the texture.

Protected Types

- typedef [Color_Buffer](#)< COLOR_FORMAT > [Color_Buffer](#)
Type alias for color buffer.

Protected Member Functions

- **Texture2D ()**
Default constructor for the [Texture2D](#) class.
- GLint [create_texture_2d](#) (const string &texture_path)
Creates a 2D texture from a file.
- unique_ptr< [Color_Buffer](#) > [load_image](#) (const string &texture_path)
Loads an image from a file.

Protected Attributes

- GLuint **texture_id**
Texture ID.
- bool **texture_is_loaded**
Indicates if the texture is loaded.

4.27.1 Detailed Description

```
template<typename COLOR_FORMAT>
class Ragot::Texture2D< COLOR_FORMAT >
```

Template class for managing a 2D texture.

Template Parameters

<i>COLOR_FORMAT</i>	Color format of the texture.
---------------------	------------------------------

4.27.2 Constructor & Destructor Documentation

4.27.2.1 Texture2D()

```
template<typename COLOR_FORMAT>
Ragot::Texture2D< COLOR_FORMAT >::Texture2D (
    const string & texture_base_path)
```

Constructor for the [Texture2D](#) class.

Parameters

<i>texture_base_path</i>	Path to the base texture file.
--------------------------	--------------------------------

4.27.3 Member Function Documentation

4.27.3.1 bind()

```
template<typename COLOR_FORMAT>
virtual bool Ragot::Texture2D< COLOR_FORMAT >::bind () const [inline], [virtual]
```

Binds the texture.

Returns

True if the texture is successfully bound, false otherwise.

Reimplemented in [Ragot::Texture_Cube](#).

4.27.3.2 create_texture_2d()

```
template<typename COLOR_FORMAT>
GLint Ragot::Texture2D< COLOR_FORMAT >::create_texture_2d (
    const string & texture_path) [protected]
```

Creates a 2D texture from a file.

Parameters

<i>texture_path</i>	Path to the texture file.
---------------------	---------------------------

Returns

The texture ID.

4.27.3.3 is_ok()

```
template<typename COLOR_FORMAT>
bool Ragot::Texture2D< COLOR_FORMAT >::is_ok () const [inline]
```

Checks if the texture is loaded.

Returns

True if the texture is loaded, false otherwise.

4.27.3.4 load_image()

```
template<typename COLOR_FORMAT>
unique_ptr< Color_Buffer< COLOR_FORMAT > > Ragot::Texture2D< COLOR_FORMAT >::load_image (
    const string & texture_path) [protected]
```

Loads an image from a file.

Parameters

<code>texture_path</code>	Path to the texture file.
---------------------------	---------------------------

Returns

Unique pointer to the color buffer.

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

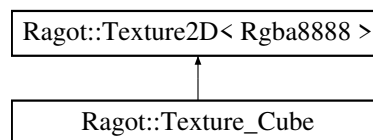
- Mesh.hpp
- Mesh.cpp

4.28 Ragot::Texture_Cube Class Reference

Class for managing a cube texture.

```
#include <Mesh.hpp>
```

Inheritance diagram for Ragot::Texture_Cube:



Public Member Functions

- [Texture_Cube](#) (const string &texture_base_path)
Constructor for the [Texture_Cube](#) class.
- bool [bind](#) () const override
Binds the cube texture.

Public Member Functions inherited from [Ragot::Texture2D< Rgba8888 >](#)

- [Texture2D](#) (const string &texture_base_path)
Constructor for the [Texture2D](#) class.
- [~Texture2D](#) ()
Destructor for the [Texture2D](#) class.
- bool [is_ok](#) () const
Checks if the texture is loaded.

Additional Inherited Members

Protected Types inherited from [Ragot::Texture2D< Rgba8888 >](#)

- typedef [Color_Buffer< Rgba8888 >](#) [Color_Buffer](#)
Type alias for color buffer.

Protected Member Functions inherited from [Ragot::Texture2D< Rgba8888 >](#)

- **Texture2D ()**
Default constructor for the [Texture2D](#) class.
- GLint [create_texture_2d](#) (const string &texture_path)
Creates a 2D texture from a file.
- unique_ptr< [Color_Buffer](#) > [load_image](#) (const string &texture_path)
Loads an image from a file.

Protected Attributes inherited from [Ragot::Texture2D< Rgba8888 >](#)

- GLuint **texture_id**
Texture ID.
- bool **texture_is_loaded**
Indicates if the texture is loaded.

4.28.1 Detailed Description

Class for managing a cube texture.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 Texture_Cube()

```
Ragot::Texture_Cube::Texture_Cube (
    const string & texture_base_path)
```

Constructor for the [Texture_Cube](#) class.

Parameters

<i>texture_base_path</i>	Path to the base texture file.
--------------------------	--------------------------------

4.28.3 Member Function Documentation

4.28.3.1 bind()

```
bool Ragot::Texture_Cube::bind () const [inline], [override], [virtual]
```

Binds the cube texture.

Returns

True if the texture is successfully bound, false otherwise.

Reimplemented from [Ragot::Texture2D< Rgba8888 >](#).

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

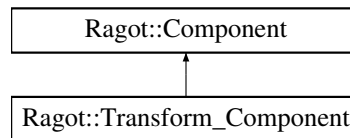
- Mesh.hpp
- Mesh.cpp

4.29 Ragot::Transform_Component Class Reference

[Component](#) for managing transformations.

```
#include <Component.hpp>
```

Inheritance diagram for Ragot::Transform_Component:



Public Member Functions

- **Transform_Component ()**
Constructor for the [Transform_Component](#) class.
- mat4 [get_transform_matrix](#) ()
Gets the transformation matrix.
- void [set_position](#) (const vec3 &pos)
Sets the position.
- vec3 [get_position](#) () const
Gets the position.
- void [set_rotation](#) (const vec3 &rot)
Sets the rotation.
- vec3 [get_rotation](#) () const
Gets the rotation.
- void [set_scale](#) (const vec3 &scal)
Sets the scale.
- vec3 [get_scale](#) () const
Gets the scale.
- void [set_parent](#) (Transform_Component *par)
Sets the parent transformation component.
- Transform_Component * [get_parent](#) () const
Gets the parent transformation component.

Public Member Functions inherited from [Ragot::Component](#)

- virtual ~**Component** ()=default
Virtual destructor for the [Component](#) class.
- std::shared_ptr< [Entity](#) > [get_entity](#) () const
Gets the entity associated with this component.
- void [set_entity](#) (std::shared_ptr< [Entity](#) > ent)
Sets the entity associated with this component.
- bool [get_has_task](#) () const
Checks if the component has a task.

Additional Inherited Members

Protected Attributes inherited from [Ragot::Component](#)

- bool **has_task** = false
Indicates whether the component has a task.

4.29.1 Detailed Description

[Component](#) for managing transformations.

4.29.2 Member Function Documentation

4.29.2.1 `get_parent()`

```
Transform\_Component * Ragot::Transform_Component::get_parent () const [inline]
```

Gets the parent transformation component.

Returns

Pointer to the parent transformation component.

4.29.2.2 `get_position()`

```
vec3 Ragot::Transform_Component::get_position () const [inline]
```

Gets the position.

Returns

Current position.

4.29.2.3 `get_rotation()`

```
vec3 Ragot::Transform_Component::get_rotation () const [inline]
```

Gets the rotation.

Returns

Current rotation.

4.29.2.4 get_scale()

```
vec3 Ragot::Transform_Component::get_scale () const [inline]
```

Gets the scale.

Returns

Current scale.

4.29.2.5 get_transform_matrix()

```
mat4 Ragot::Transform_Component::get_transform_matrix () [inline]
```

Gets the transformation matrix.

Returns

Transformation matrix.

4.29.2.6 set_parent()

```
void Ragot::Transform_Component::set_parent (  
    Transform_Component * par) [inline]
```

Sets the parent transformation component.

Parameters

<i>par</i>	Pointer to the parent transformation component.
------------	---

4.29.2.7 set_position()

```
void Ragot::Transform_Component::set_position (  
    const vec3 & pos) [inline]
```

Sets the position.

Parameters

<i>pos</i>	New position.
------------	---------------

4.29.2.8 set_rotation()

```
void Ragot::Transform_Component::set_rotation (  
    const vec3 & rot) [inline]
```

Sets the rotation.

Parameters

<i>rot</i>	New rotation.
------------	---------------

4.29.2.9 set_scale()

```
void Ragot::Transform_Component::set_scale (
    const vec3 & scal) [inline]
```

Sets the scale.

Parameters

<i>scal</i>	New scale.
-------------	------------

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

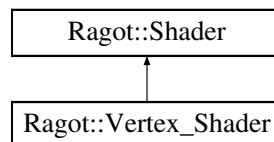
- Component.hpp

4.30 Ragot::Vertex_Shader Class Reference

Class for managing an OpenGL vertex shader.

```
#include <Shader_Program.hpp>
```

Inheritance diagram for Ragot::Vertex_Shader:

**Public Member Functions**

- [Vertex_Shader](#) (const vector< string > &source_code)

Constructor for the [Vertex_Shader](#) class.

Public Member Functions inherited from [Ragot::Shader](#)

- **Shader** ()=delete
Deleted default constructor.
- **~Shader** ()
Destructor for the [Shader](#) class.
- GLuint [get_id](#) () const
Gets the shader ID.
- string * [get_error](#) ()
Gets the compilation error message.
- bool [is_ok](#) () const
Checks if the shader is compiled successfully.

Additional Inherited Members

Protected Member Functions inherited from [Ragot::Shader](#)

- [Shader](#) (const vector< string > &source_code, GLenum type)
Constructor for the [Shader](#) class.
- GLuint [compile_shader](#) ()
Compiles the shader.
- void [show_compilation_error](#) ()
Displays compilation errors.

4.30.1 Detailed Description

Class for managing an OpenGL vertex shader.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 Vertex_Shader()

```
Ragot::Vertex_Shader::Vertex_Shader (
    const vector< string > & source_code) [inline]
```

Constructor for the [Vertex_Shader](#) class.

Parameters

<i>source_code</i>	Vector of vertex shader source code.
--------------------	--------------------------------------

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

- Shader_Program.hpp

4.31 Ragot::Window Class Reference

Class for managing an SDL window with OpenGL context.

```
#include <Window.hpp>
```

Classes

- struct [OpenGL_Context_Settings](#)
Struct for OpenGL context settings.

Public Types

- enum [Position](#) { [UNDEFINED](#) = SDL_WINDOWPOS_UNDEFINED , [CENTERED](#) = SDL_WINDOWPOS_↵
CENTERED }

Enum for window position.

Public Member Functions

- [Window](#) (const std::string &title, int left_x, int top_y, unsigned width, unsigned height, const [OpenGL_Context_Settings](#) &context_details)
Constructor for the [Window](#) class.
- [Window](#) (const char *title, int left_x, int top_y, unsigned width, unsigned height, const [OpenGL_Context_Settings](#) &context_details)
Constructor for the [Window](#) class.
- [~Window](#) ()
Destructor for the [Window](#) class.
- [Window](#) (const [Window](#) &)=delete
Deleted copy constructor.
- [Window](#) & [operator=](#) (const [Window](#) &)=delete
Deleted copy assignment operator.
- [Window](#) ([Window](#) &&other) noexcept
Move constructor for the [Window](#) class.
- [Window](#) & [operator=](#) ([Window](#) &&other) noexcept
Move assignment operator for the [Window](#) class.
- void [swap_buffers](#) ()
Swaps the OpenGL buffers.
- unsigned [get_width](#) ()
Gets the width of the window.
- unsigned [get_height](#) ()
Gets the height of the window.

4.31.1 Detailed Description

Class for managing an SDL window with OpenGL context.

4.31.2 Member Enumeration Documentation

4.31.2.1 Position

```
enum Ragot::Window::Position
```

Enum for window position.

Enumerator

UNDEFINED	Undefined position.
CENTERED	Centered position.

4.31.3 Constructor & Destructor Documentation

4.31.3.1 Window() [1/3]

```
Ragot::Window::Window (
    const std::string & title,
    int left_x,
    int top_y,
    unsigned width,
    unsigned height,
    const OpenGL_Context_Settings & context_details) [inline]
```

Constructor for the [Window](#) class.

Parameters

<i>title</i>	Title of the window.
<i>left_x</i>	X coordinate of the window position.
<i>top_y</i>	Y coordinate of the window position.
<i>width</i>	Width of the window.
<i>height</i>	Height of the window.
<i>context_details</i>	OpenGL context settings.

4.31.3.2 Window() [2/3]

```
Ragot::Window::Window (
    const char * title,
    int left_x,
    int top_y,
    unsigned width,
    unsigned height,
    const OpenGL_Context_Settings & context_details)
```

Constructor for the [Window](#) class.

Parameters

<i>title</i>	Title of the window.
<i>left_x</i>	X coordinate of the window position.
<i>top_y</i>	Y coordinate of the window position.
<i>width</i>	Width of the window.
<i>height</i>	Height of the window.
<i>context_details</i>	OpenGL context settings.

4.31.3.3 Window() [3/3]

```
Ragot::Window::Window (
    Window && other) [inline], [noexcept]
```

Move constructor for the [Window](#) class.

Parameters

<i>other</i>	Other window to move from.
--------------	----------------------------

4.31.4 Member Function Documentation

4.31.4.1 `get_height()`

```
unsigned Ragot::Window::get_height () [inline]
```

Gets the height of the window.

Returns

Height of the window.

4.31.4.2 `get_width()`

```
unsigned Ragot::Window::get_width () [inline]
```

Gets the width of the window.

Returns

Width of the window.

4.31.4.3 `operator=()`

```
Window & Ragot::Window::operator= (  
    Window && other) [inline], [noexcept]
```

Move assignment operator for the [Window](#) class.

Parameters

<i>other</i>	Other window to move from.
--------------	----------------------------

Returns

Reference to the moved window.

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

- `Window.hpp`
- `Window.cpp`

Chapter 5

File Documentation

5.1 Ambient.hpp

```
00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include "Camera.hpp"
00032 #include "Shader_Program.hpp"
00033 #include "Mesh.hpp"
00034 #include "Color.hpp"
00035
00036 #include <glad/glad.h>
00037 #include <string>
00038
00039 namespace Ragot
00040 {
00041     using namespace std;
00042
00043     class Skybox
00044     {
00045     private:
00046         Shader_Program shader_program;
00047         static const GLfloat coordinates[];
00048         static const string vertex_shader_code;
00049         static const string fragment_shader_code;
00050         GLuint vbo_id;
00051         GLuint vao_id;
00052         GLint model_view_matrix_id;
00053         GLint projection_matrix_id;
00054         shared_ptr<Camera> camera = nullptr;
00055         Texture_Cube texture_cube;
00056
00057     public:
```

```

00066         Skybox(const string & texture_path);
00067
00071         ~Skybox();
00072
00077         void set_camera(shared_ptr<Camera> cam) { camera = cam; }
00078
00082         void render();
00083     };
00084
00089     class Terrain
00090     {
00091     private:
00092         enum
00093         {
00094             COORDINATES_VBO,
00095             TEXTURE_UVS_VBO,
00096             INDICES_EBO,
00097             VBO_COUNT
00098         };
00099
00100         Shader_Program shader_program;
00101         static const string vertex_shader_code;
00102         static const string fragment_shader_code;
00103         GLsizei number_of_vertices;
00104         GLsizei number_of_indices;
00105         GLuint vbo_ids[VBO_COUNT];
00106         GLuint vao_id;
00107         GLint model_view_matrix_id;
00108         GLint projection_matrix_id;
00109         GLint view_position_id;
00110         GLint light_position_id;
00111         shared_ptr< Camera > camera = nullptr;
00112         Texture2D< Monochrome8 > texture;
00113
00114     public:
00122         Terrain(float width, float depth, unsigned x_slices, unsigned z_slices);
00123
00127         ~Terrain();
00128
00133         void set_camera(shared_ptr<Camera> cam) { camera = cam; }
00134
00138         void render();
00139     };
00140
00145     class Light
00146     {
00147     public:
00148         glm::vec3 color;
00149
00154         Light(const glm::vec3 & color) : color(color) {}
00155
00159         virtual ~Light() = default;
00160     };
00161
00166     class DirectionalLight : public Light
00167     {
00168     public:
00169         glm::vec3 direction;
00170
00176         DirectionalLight(const glm::vec3 & color, const glm::vec3 direction)
00177             : Light(color), direction(direction) {}
00178     };
00179
00184     class PointLight : public Light
00185     {
00186     public:
00187         glm::vec3 position;
00188
00194         PointLight(const glm::vec3 & color, const glm::vec3 & position)
00195             : Light(color), position(position) {}
00196     };
00197
00202     class AreaLight : public Light
00203     {
00204     public:
00205         glm::vec3 position;
00206         glm::vec3 size;
00207
00214         AreaLight(const glm::vec3 & color, const glm::vec3 & position, const glm::vec3 & size)
00215             : Light(color), position(position), size(size) {}
00216     };
00217 }

```


5.2 Camera.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include <glm.hpp> // vec3, vec4, ivec4, mat4
00032 #include <gtc/matrix_transform.hpp> // translate, rotate, scale, perspective
00033 #include <gtc/type_ptr.hpp> // value_ptr
00034
00035 namespace Ragot
00036 {
00037     class Camera
00038     {
00039     public:
00040         using Point = glm::vec4;
00041         using Vector = glm::vec4;
00042         using Matrix44 = glm::mat4;
00043
00044     private:
00045         float fov;
00046         float near_z;
00047         float far_z;
00048         float ratio;
00049
00050         Point location;
00051         Point target;
00052
00053         Matrix44 projection_matrix;
00054
00055     public:
00056         Camera(float ratio = 1.f)
00057         {
00058             reset(60.f, 0.1f, 1000.f, ratio);
00059         }
00060
00061         Camera(float near_z, float far_z, float ratio = 1.f)
00062         {
00063             reset(60.f, near_z, far_z, ratio);
00064         }
00065
00066         Camera(float fov_degrees, float near_z, float far_z, float ratio)
00067         {
00068             reset(fov_degrees, near_z, far_z, ratio);
00069         }
00070
00071         float get_fov() const { return fov; }
00072
00073         float get_near_z() const { return near_z; }
00074
00075         float get_far_z() const { return far_z; }
00076
00077         float get_ratio() const { return ratio; }
00078
00079         const Point & get_location() const { return location; }
00080
00081         const Point & get_target() const { return target; }
00082
00083         void set_fov(float new_fov) { fov = new_fov; calculate_projection_matrix(); }
00084
00085     };
00086 }

```

```

00137     void set_near_z(float new_near_z) { near_z = new_near_z; calculate_projection_matrix(); }
00138
00143     void set_far_z(float new_far_z) { far_z = new_far_z; calculate_projection_matrix(); }
00144
00149     void set_ratio(float new_ratio) { ratio = new_ratio; calculate_projection_matrix(); }
00150
00157     void set_location(float x, float y, float z) { location[0] = x; location[1] = y; location[2] =
z; }
00158
00165     void set_target(float x, float y, float z) { target[0] = x; target[1] = y; target[2] = z; }
00166
00174     void reset(float new_fov, float new_near_z, float new_far_z, float new_ratio)
00175     {
00176         set_fov(new_fov);
00177         set_near_z(new_near_z);
00178         set_far_z(new_far_z);
00179         set_ratio(new_ratio);
00180         set_location(0.f, 0.f, 0.f);
00181         set_target(0.f, 0.f, -1.f);
00182         calculate_projection_matrix();
00183     }
00184
00189     void move(const glm::vec3 & translation)
00190     {
00191         location += glm::vec4(translation, 0.f);
00192         target += glm::vec4(translation, 1.f);
00193     }
00194
00199     void rotate(const glm::mat4 & rotation)
00200     {
00201         target = location + rotation * (target - location);
00202     }
00203
00208     const glm::mat4 & get_projection_matrix() const
00209     {
00210         return projection_matrix;
00211     }
00212
00217     glm::mat4 get_transform_matrix_inverse() const
00218     {
00219         return glm::lookAt(
00220             glm::vec3(location[0], location[1], location[2]),
00221             glm::vec3(target[0], target[1], target[2]),
00222             glm::vec3(0.0f, 1.0f, 0.0f)
00223         );
00224     }
00225
00226 private:
00230     void calculate_projection_matrix()
00231     {
00232         projection_matrix = glm::perspective(glm::radians(fov), ratio, near_z, far_z);
00233     }
00234 };
00235 }

```

5.3 Color.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrs Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrs Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.

```

```

00027  */
00028
00029 #pragma once
00030
00031 #include <cstdint>
00032
00033 namespace Ragot
00034 {
00038     using Monochrome8 = uint8_t;
00039
00043     union Rgba8888
00044     {
00048         enum { RED, GREEN, BLUE, ALPHA };
00049
00050         uint32_t value;
00051         uint8_t components[4];
00052     };
00053 }

```

5.4 Color_Buffer.hpp

```

00001  /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrs Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrs Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include <vector>
00032
00033 namespace Ragot
00034 {
00039     template<typename COLOR>
00040     class Color_Buffer
00041     {
00042     public:
00043         using Color = COLOR;
00044
00045     private:
00046         unsigned width;
00047         unsigned height;
00048
00049         std::vector<Color> buffer;
00050
00051     public:
00057         Color_Buffer(unsigned width, unsigned height)
00058             :
00059             width(width),
00060             height(height),
00061             buffer(width * height)
00062         {
00063         }
00064
00069         unsigned get_width() const
00070         {
00071             return width;
00072         }
00073
00078         unsigned get_height() const
00079         {

```

```

00080         return height;
00081     }
00082
00083     Color* colors()
00084     {
00085         return buffer.data();
00086     }
00087
00088     const Color* colors() const
00089     {
00090         return buffer.data();
00091     }
00092
00093     Color& get(unsigned offset)
00094     {
00095         return buffer[offset];
00096     }
00097
00098     const Color& get(unsigned offset) const
00099     {
00100         return buffer[offset];
00101     }
00102
00103     void set(unsigned offset, const Color& color)
00104     {
00105         buffer[offset] = color;
00106     }
00107 };
00108
00109 }
```

5.5 Component.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027 */
00028
00029 #pragma once
00030
00031 #include "MySystem.hpp"
00032 #include "Mesh.hpp"
00033 #include <glm.hpp>
00034 #include <gtc/matrix_transform.hpp> // translate, rotate, scale, perspective
00035 #include <gtc/type_ptr.hpp> // value_ptr, quat
00036 #include "Camera.hpp"
00037 #include "Shader_Program.hpp"
00038
00039 using glm::vec3;
00040 using glm::mat4;
00041
00042 namespace Ragot
00043 {
00044     class Component
00045     {
00046     public:
00047         virtual ~Component() = default;
00048
00049         std::shared_ptr<Entity> get_entity() const { return entity.lock(); }
00050
00051         void set_entity(std::shared_ptr<Entity> ent) { entity = ent; }
00052     };
00053 }
```

```

00066
00071     bool get_has_task() const { return has_task; }
00072
00073 private:
00074     std::weak_ptr<Entity> entity;
00075
00076 protected:
00077     bool has_task = false;
00078 };
00079
00083 class Transform_Component : public Component
00084 {
00085 public:
00089     Transform_Component() : position(0.f), rotation(0.f), scale(1.0f), parent(nullptr) {}
00090
00095     mat4 get_transform_matrix()
00096     {
00097         mat4 transform_matrix(1);
00098         transform_matrix = glm::translate(transform_matrix, position);
00099         transform_matrix = glm::scale(transform_matrix, scale);
00100
00101         glm::quat quaternion_rotation = glm::quat(glm::radians(rotation));
00102         transform_matrix *= glm::mat4_cast(quaternion_rotation);
00103
00104         if (parent)
00105             transform_matrix = parent->get_transform_matrix() * transform_matrix;
00106
00107         return transform_matrix;
00108     }
00109
00114     void set_position(const vec3 &pos) { position = pos; }
00115
00120     vec3 get_position() const { return position; }
00121
00126     void set_rotation(const vec3 &rot) { rotation = rot; }
00127
00132     vec3 get_rotation() const { return rotation; }
00133
00138     void set_scale(const vec3 &scal) { scale = scal; }
00139
00144     vec3 get_scale() const { return scale; }
00145
00150     void set_parent(Transform_Component *par) { parent = par; }
00151
00156     Transform_Component* get_parent() const { return parent; }
00157
00158 private:
00159     vec3 position;
00160     vec3 rotation;
00161     vec3 scale;
00162     Transform_Component* parent;
00163 };
00164
00168 class Model_Component : public Component
00169 {
00170 public:
00174     Model_Component() = delete;
00175
00181     Model_Component(const string &model_file_path, const string &texture_file_path);
00182
00183     Critical_Task render_task;
00184     Light_Task update_task;
00185
00190     const GLuint get_shader_program_id() const { return material.get_shader_program_id(); }
00191
00196     void set_transparency(bool trans) { is_transparent = trans; }
00197
00198 private:
00199     Mesh mesh;
00200     Material material;
00201     bool is_transparent = false;
00202     float vertical_position;
00203     float vertical_speed = 0.0000005f;
00204
00205     float orbit_angle;
00206     float orbit_speed = 0.00005f;
00207
00208     static const string vertex_shader_code;
00209     static const string fragment_shader_code;
00210
00211     GLint model_view_matrix_id;
00212     GLint projection_matrix_id;
00213     GLint normal_matrix_id;
00214     GLint view_pos_id;
00215
00216 private:
00220     void configure_material();

```

```

00221
00225         void render();
00226
00230         void update();
00231     };
00232 }

```

5.6 Entity.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include "MySystem.hpp"
00032 #include "Component.hpp"
00033
00034 namespace Ragot
00035 {
00039     class Entity : public std::enable_shared_from_this<Entity>
00040     {
00041     public:
00042         Scene* scene;
00043
00044     public:
00045         Transform_Component transform;
00046
00047     private:
00048         map<string, shared_ptr<Component>> components;
00049         vector<shared_ptr<Entity>> children;
00050
00051     public:
00052         void set_scene(Scene* scene) { this->scene = scene; }
00053
00054         const Scene* get_scene() { return scene; }
00055
00056         const Scene* get_scene() const { return scene; }
00057
00058         void add_component(shared_ptr<Component> component, const string& name);
00059
00060         void remove_component(const string& name);
00061
00062         void add_child(shared_ptr<Entity> child)
00063         {
00064             child->set_transform_parent(&transform);
00065             children.push_back(child);
00066         }
00067
00068         void remove_child(shared_ptr<Entity> child)
00069         {
00070             child->set_transform_parent(nullptr);
00071             children.erase(remove(children.begin(), children.end(), child), children.end());
00072         }
00073
00074         void set_transform_parent(Transform_Component* parent)
00075         {
00076             transform.set_parent(parent);
00077         }
00078     }
00079 }
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110

```

```

00115         const map<string, shared_ptr<Component>&& get_components() const { return components; }
00116     };
00117 }

```

5.7 Mesh.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include <glad/glad.h>
00032 #include <glm.hpp>
00033 #include <vector>
00034 #include <string>
00035 #include "Camera.hpp"
00036 #include "Shader_Program.hpp"
00037 #include "Color.hpp"
00038 #include "Color_Buffer.hpp"
00039
00040 namespace Ragot
00041 {
00042     using std::vector;
00043
00044     class Mesh
00045     {
00046     protected:
00047         enum
00048         {
00049             COORDINATES_VBO,
00050             NORMALS_VBO,
00051             TEXTURE_UVS_VBO,
00052             INDICES_EBO,
00053             VBO_COUNT
00054         };
00055
00056         vector<glm::vec3> coordinates;
00057         vector<glm::vec3> normals;
00058         vector<glm::vec2> texture_coords;
00059         vector<GLuint> indices;
00060
00061     private:
00062         GLuint vbo_ids[VBO_COUNT];
00063         GLuint vao_id;
00064
00065         GLsizei number_of_indices;
00066
00067         float angle;
00068
00069         void load_mesh(const std::string& mesh_file_path);
00070
00071     private:
00072         Mesh(const Mesh&) = delete;
00073         Mesh& operator=(const Mesh&) = delete;
00074
00075     public:
00076         Mesh() = default;
00077
00078     };
00079
00080 }

```

```

00095         Mesh(const std::string& mesh_file_path);
00096
00100     ~Mesh()
00101     {
00102         glDeleteVertexArrays(1, &vao_id);
00103         glDeleteBuffers(VBO_COUNT, vbo_ids);
00104     }
00105
00106 public:
00111     const vector<glm::vec3>& get_coordinates() const { return coordinates; }
00112
00117     const vector<glm::vec3>& get_normals() const { return normals; }
00118
00123     const vector<glm::vec2>& get_textures_uv() const { return texture_coords; }
00124
00129     const vector<GLuint>& get_indices() const { return indices; }
00130
00135     const GLuint get_vao_id() const { return vao_id; }
00136
00141     const GLsizei get_number_of_indices() const { return number_of_indices; }
00142 };
00143
00148 template <typename COLOR_FORMAT>
00149 class Texture2D
00150 {
00151 protected:
00152     typedef Color_Buffer<COLOR_FORMAT> Color_Buffer;
00153
00154     GLuint texture_id;
00155     bool texture_is_loaded;
00156
00157 private:
00158     bool is_uint8 = false;
00159
00160 public:
00165     Texture2D(const string& texture_base_path);
00166
00170     ~Texture2D();
00171
00172 private:
00173     Texture2D(const Texture2D&) = delete;
00174     Texture2D& operator=(const Texture2D&) = delete;
00175
00176 protected:
00180     Texture2D() : texture_id(0), texture_is_loaded(false) {}
00181
00182 public:
00187     bool is_ok() const
00188     {
00189         return texture_is_loaded;
00190     }
00191
00196     virtual bool bind() const
00197     {
00198         return texture_is_loaded ? glBindTexture(GL_TEXTURE_2D, texture_id), true : false;
00199     }
00200
00201 protected:
00207     GLint create_texture_2d(const string& texture_path);
00208
00214     unique_ptr<Color_Buffer> load_image(const string& texture_path);
00215 };
00216
00220 class Texture_Cube : public Texture2D<Rgba8888>
00221 {
00222 public:
00227     Texture_Cube(const string& texture_base_path);
00228
00233     bool bind() const override
00234     {
00235         return texture_is_loaded ? glBindTexture(GL_TEXTURE_CUBE_MAP, texture_id), true : false;
00236     }
00237 };
00238
00242 class Material
00243 {
00244 private:
00245     Shader_Program shader_program;
00246     Texture2D<Rgba8888> texture;
00247     glm::vec3 color;
00248     float shininess;
00249
00250 public:
00251     Material() = delete;
00252
00259     Material(const vector<string>& source_code_vertex, const vector<string>& source_code_fragment,
const string& texture_base_path);

```



```

00260
00264     ~Material() = default;
00265
00269     void use_shader_program() { shader_program.use(); }
00270
00276     GLint get_shader_program_uniform_location(const string& uniform) { return
shader_program.get_uniform_location(uniform); }
00277
00282     GLuint get_shader_program_id() const { return shader_program.get_id(); }
00283
00288     const bool bind_texture() const { return texture.bind(); }
00289
00294     const glm::vec3 get_color() { return color; }
00295
00300     const float get_shininess() { return shininess; }
00301 };
00302 }

```

5.8 MyKernel.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include "Task.hpp"
00032 #include <thread>
00033 #include <vector>
00034 #include <memory>
00035 #include <mutex>
00036
00037 namespace Ragot
00038 {
00039     using std::vector;
00040
00041     class Kernel
00042     {
00043     {
00047         vector<std::shared_ptr<Task>> tasks;
00048         vector<std::shared_ptr<Critical_Task>> render_tasks;
00049
00050         std::mutex tasks_mutex;
00051
00052         std::atomic<bool> exit;
00053         std::atomic<bool> is_running;
00054
00055         std::condition_variable cv;
00056
00057     public:
00062         void add(std::shared_ptr<Task> new_task);
00063
00067         void run();
00072         void stop()
00073         {
00074             tasks.front()->stop_execution();
00075             exit = true;
00076             cv.notify_all();

```

```

00077     }
00078
00082     void execute_critical();
00083 };
00084 }

```

5.9 MySystem.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include "Window.hpp"
00032 #include "Task.hpp"
00033 #include "MyKernel.hpp"
00034 #include "Camera.hpp"
00035 #include "Ambient.hpp"
00036 #include "Postprocess.hpp"
00037 #include <string>
00038 #include <memory>
00039 #include <map>
00040 #include <vector>
00041
00042 #include <mutex>
00043 #include <condition_variable>
00044 #include <queue>
00045
00046 namespace Ragot
00047 {
00048     using namespace std;
00049
00050     // Declaración adelantada de Entity
00051     class Entity;
00052
00053     class Scene
00054     {
00055     private:
00056         shared_ptr<Camera> camera = make_shared<Camera>();
00057         map<string, shared_ptr<Entity>> entities;
00058         Frame_Buffer framebuffer;
00059         Skybox skybox;
00060         Terrain terrain;
00061         vector<shared_ptr<Light>> lights;
00062
00063         mutex scene_mutex;
00064
00065         int width;
00066         int height;
00067
00068         float angle_around_x;
00069         float angle_around_y;
00070         float angle_delta_x;
00071         float angle_delta_y;
00072
00073         float camera_speed = 0.025f;
00074
00075         bool pointer_pressed;

```

```

00079         int last_pointer_x;
00080         int last_pointer_y;
00081         bool turbo;
00082
00083         float camera_turbo_speed = 2.f;
00084
00085         glm::vec3 camera_translation;
00086
00087     public:
00093         void resize(int width, int height);
00094
00100         void on_drag(int pointer_x, int pointer_y);
00101
00108         void on_click(int pointer_x, int pointer_y, bool down);
00109
00114         void on_translation(glm::vec3 translation);
00115
00120         void on_shift_pressed(bool down);
00121
00122     public:
00126         void update();
00127
00131         void render();
00132
00136         void postprocess();
00137
00138     public:
00142         Scene();
00143
00144     public:
00150         void add_entities(shared_ptr<Entity> entity, const string& name);
00151
00156         void remove_entities(const string& name);
00157
00163         shared_ptr<Entity> get_entity(const string& name) const;
00164
00169         shared_ptr<Camera> get_camera() const { return camera; }
00170
00171     private:
00176         void set_lights(GLuint shader_program_id);
00177     };
00178
00182     class System
00183     {
00184     public:
00185         Critical_Task buffer_swap;
00186         Critical_Task handle_events;
00187         Critical_Task scene_render;
00188         Light_Task scene_update;
00189         Light_Task process_events;
00190
00191         queue<SDL_Event> eventQueue;
00192         mutex queueMutex;
00193         condition_variable queueCondition;
00194
00195         Window window;
00196         Scene scene;
00197
00198         Kernel kernel;
00199     private:
00203         void input();
00204
00208         void initialize();
00209
00213         void sdl_events();
00214
00215     public:
00222         System(const string& Window_Name, const int width, const int height);
00223
00227         System();
00228
00232         ~System() { SDL_Quit(); }
00233
00234     public:
00240         void add_entities(shared_ptr<Entity> entity, const string& name);
00241
00245         void run()
00246         {
00247             kernel.run();
00248         }
00249
00253         void stop()
00254         {
00255             kernel.stop();
00256         }
00257     };
00258 }

```

5.10 Postprocess.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030 #include "Shader_Program.hpp"
00031
00032 namespace Ragot
00033 {
00034     class Frame_Buffer
00035     {
00036     private:
00037         enum
00038         {
00039             COORDINATES_VBO,
00040             UV_COORDINATES_VBO,
00041             VBO_COUNT
00042         };
00043
00044         static float vertices[];
00045         static float uv_coordenates[];
00046
00047         static const string vertex_code_shader;
00048         static const string fragment_code_shader;
00049
00050         GLuint frame_buffer_id;
00051         GLuint texture_id;
00052         GLuint depthbuffer_id;
00053         GLint current_time_id;
00054
00055         Shader_Program shader_program;
00056
00057         GLuint vbo_id[VBO_COUNT];
00058         GLuint vao_id;
00059
00060     public:
00061         Frame_Buffer(unsigned width, unsigned height);
00062
00063         Frame_Buffer() = delete;
00064
00065         ~Frame_Buffer();
00066
00067         void bind_frame_buffer() const { glBindFramebuffer(GL_FRAMEBUFFER, frame_buffer_id); }
00068
00069         void unbind_frame_buffer() const { glBindFramebuffer(GL_FRAMEBUFFER, 0); }
00070
00071         void bind_texture() const { glBindTexture(GL_TEXTURE_2D, texture_id); }
00072
00073         void unbind_texture() const { glBindTexture(GL_TEXTURE_2D, 0); }
00074
00075         void render();
00076     };
00077 }

```

5.11 Shader_Program.hpp

```

00001 /*

```

```

00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.a
00027  */
00028
00029 #pragma once
00030
00031 #include <glad/glad.h>
00032
00033 #include <string>
00034 #include <vector>
00035
00036 namespace Ragot
00037 {
00038     using namespace std;
00039
00040     class Shader
00041     {
00042     private:
00043         GLuint id;
00044         string error;
00045         bool compilation_succeeded;
00046
00047     protected:
00048         Shader(const vector<string>& source_code, GLenum type);
00049
00050         GLuint compile_shader();
00051
00052         void show_compilation_error();
00053
00054     public:
00055         Shader() = delete;
00056
00057         ~Shader()
00058         {
00059             glDeleteShader(id);
00060         }
00061
00062         GLuint get_id() const
00063         {
00064             return id;
00065         }
00066
00067         string* get_error()
00068         {
00069             return error.empty() ? nullptr : &error;
00070         }
00071
00072         bool is_ok() const
00073         {
00074             return compilation_succeeded;
00075         }
00076     };
00077
00078     class Vertex_Shader : public Shader
00079     {
00080     public:
00081         Vertex_Shader(const vector<string>& source_code) : Shader(source_code, GL_VERTEX_SHADER)
00082         {
00083         }
00084     };
00085
00086     class Fragment_Shader : public Shader
00087     {
00088     public:

```

```

00133         Fragment_Shader(const vector<string>& source_code) : Shader(source_code, GL_FRAGMENT_SHADER)
00134     {
00135     }
00136 };
00137
00141 class Shader_Program
00142 {
00143 private:
00144     GLuint program_id;
00145
00146 public:
00152     Shader_Program(const vector<string>& source_code_vertex, const vector<string>&
source_code_fragment);
00153
00154     Shader_Program() = delete;
00155
00159     ~Shader_Program()
00160     {
00161         glDeleteProgram(program_id);
00162     }
00163
00167     void use() const
00168     {
00169         glUseProgram(program_id);
00170     }
00171
00176     GLuint get_id() const
00177     {
00178         return program_id;
00179     }
00180
00186     GLuint get_uniform_location(string uniform_name) const
00187     {
00188         return glGetUniformLocation(program_id, uniform_name.c_str());
00189     }
00190
00191 private:
00192     Shader_Program(const Shader_Program&) = delete;
00193     Shader_Program& operator=(const Shader_Program&) = delete;
00194
00200     void initialize(GLuint vertex_shader_id, GLuint fragment_shader_id);
00201
00205     void show_linkage_error();
00206 };
00207 }

```

5.12 Task.hpp

```

00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031
00032 #include <mutex>
00033 #include <condition_variable>
00034 #include <atomic>
00035 #include <functional>

```

```

00036
00037 namespace Ragot
00038 {
00039     using std::condition_variable;
00040     using std::mutex;
00041     using std::atomic;
00042     using std::function;
00043
00044     class Task
00045     {
00046     public:
00047         static condition_variable cv;
00048         static mutex mtx;
00049         static atomic<bool> is_stop;
00050         static atomic<bool> finish_execution;
00051
00052     public:
00053         Task(function<void()> task_func) : task_func(task_func) {}
00054
00055         virtual ~Task() = default;
00056
00057         virtual void execute() = 0;
00058
00059         void stop_execution()
00060         {
00061             std::lock_guard<mutex> lock(mtx);
00062             finish_execution = true;
00063             cv.notify_all();
00064         }
00065
00066         void stop()
00067         {
00068             std::lock_guard<mutex> lock(mtx);
00069             is_stop = true;
00070             cv.notify_all();
00071         }
00072
00073         void resume()
00074         {
00075             std::lock_guard<mutex> lock(mtx);
00076             is_stop = false;
00077             cv.notify_all();
00078         }
00079
00080     protected:
00081         bool shouldStop()
00082         {
00083             return is_stop.load();
00084         }
00085
00086         bool shouldFinish()
00087         {
00088             return finish_execution.load();
00089         }
00090
00091         void wait_for_resume()
00092         {
00093             std::unique_lock<mutex> lock(mtx);
00094             cv.wait(lock, [this] {return !shouldStop() || shouldFinish(); });
00095         }
00096
00097     protected:
00098         function<void()> task_func;
00099     };
00100
00101     class Light_Task : public Task
00102     {
00103     public:
00104         Light_Task(function<void()> task_func) : Task(task_func) {}
00105
00106         void execute() override;
00107     };
00108
00109     class Critical_Task : public Task
00110     {
00111     public:
00112         Critical_Task(function<void()> task_func) : Task(task_func) {}
00113
00114         void execute() override;
00115     };
00116
00117     class Once_Task : public Task
00118     {
00119     public:
00120         Once_Task(function<void()> task_func) : Task(task_func) {}
00121
00122         void execute() override;
00123     };
00124
00125 }

```

```
00186 }
00187
```

5.13 Window.hpp

```
00001 /*
00002  * This file is part of OpenGL-FinalProject
00003  *
00004  * Developed by Andrés Ragot - github.com/andresragot
00005  *
00006  * MIT License
00007  *
00008  * Copyright (c) 2024 Andrés Ragot
00009  *
00010  * Permission is hereby granted, free of charge, to any person obtaining a copy
00011  * of this software and associated documentation files (the "Software"), to deal
00012  * in the Software without restriction, including without limitation the rights
00013  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00014  * copies of the Software, and to permit persons to whom the Software is
00015  * furnished to do so, subject to the following conditions:
00016  *
00017  * The above copyright notice and this permission notice shall be included in all
00018  * copies or substantial portions of the Software.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00021  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00022  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00023  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00024  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00025  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00026  * SOFTWARE.
00027  */
00028
00029 #pragma once
00030
00031 #include <SDL.h>
00032 #include <string>
00033 #include <utility>
00034
00035 #include <iostream>
00036
00037 namespace Ragot
00038 {
00039     class Window
00040     {
00041     public:
00042         enum Position
00043         {
00044             UNDEFINED = SDL_WINDOWPOS_UNDEFINED,
00045             CENTERED = SDL_WINDOWPOS_CENTERED,
00046         };
00047
00048         struct OpenGL_Context_Settings
00049         {
00050             unsigned version_major = 3;
00051             unsigned version_minor = 3;
00052             bool core_profile = true;
00053             unsigned depth_buffer_size = 24;
00054             unsigned stencil_buffer_size = 0;
00055             bool enable_vsync = true;
00056         };
00057
00058     private:
00059         SDL_Window* window_handle;
00060         SDL_GLContext opengl_context;
00061
00062         unsigned width;
00063         unsigned height;
00064
00065     public:
00066         Window(const std::string& title, int left_x, int top_y, unsigned width, unsigned height, const
OpenGL_Context_Settings& context_details)
00067             : Window(title.c_str(), left_x, top_y, width, height, context_details)
00068         {
00069         }
00070
00071         Window(const char* title, int left_x, int top_y, unsigned width, unsigned height, const
OpenGL_Context_Settings& context_details);
00072
00073         ~Window();
00074
00075     public:
00076         Window(const Window&) = delete;
00077     };
00078 }
```



```
00107         Window& operator=(const Window&) = delete;
00108
00113         Window(Window&& other) noexcept
00114         {
00115             this->window_handle = std::exchange(other.window_handle, nullptr);
00116             this->opengl_context = std::exchange(other.opengl_context, nullptr);
00117         }
00118
00124         Window& operator=(Window&& other) noexcept
00125         {
00126             this->window_handle = std::exchange(other.window_handle, nullptr);
00127             this->opengl_context = std::exchange(other.opengl_context, nullptr);
00128
00129             return *this;
00130         }
00131
00135         void swap_buffers()
00136         {
00137             SDL_GL_SwapWindow(window_handle);
00138         }
00139
00144         unsigned get_width() { return width; }
00145
00150         unsigned get_height() { return height; }
00151     };
00152 }
```

