TazGraph Project

v0.1.0

Generated by Doxygen 1.9.8

Chapter 1

TazGraph Project

create a build folder in root necessary programs: cmake, g++ WSL: `sudo apt-get install libsdl2-dev libsdl2-image-dev libsdl2-ttf-dev libsdl2-mixer-dev libglew-dev libglm-dev libopengl` if libopengl does not exist: `sudo apt install freeglut3-dev mesa-common-dev`

Start by cloning the repository with `git clone --recursive https://carvgit.ics.forth.gr/kotsonas/tazgraph`

In build/: `cmake -DCMAKE_BUILD_TYPE=Release .. make`

in TazGraph/TazGraph/: `../build/TazGraph/TazGraph`

For Windows: In root folder: `msbuild TazGraph.sln /p:Configuration=Debug /p:Platform=x64`

In TazGraph/: `../x64/Debug/TazGraph.exe`

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

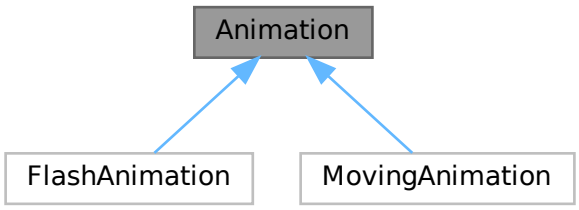
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Animation Struct Reference

Inheritance diagram for Animation:



Public Types

- enum **animType** { **ANIMTYPE_NONE** = 0 , **ANIMTYPE_PLAY_N_TIMES** = 1 , **ANIMTYPE_LOOPED** = 2 , **ANIMTYPE_BACK_FORTH** = 3 }

Public Member Functions

- **Animation** (int ix, int iy, size_t f, float s, const std::string _type, int _reps=0)
- **Animation** (int ix, int iy, size_t f, float s, const animType _type, int _reps=0)
- void **advanceFrame** (float deltaTime)
- void **resetFrameIndex** ()
- bool **hasFinished** ()

Public Attributes

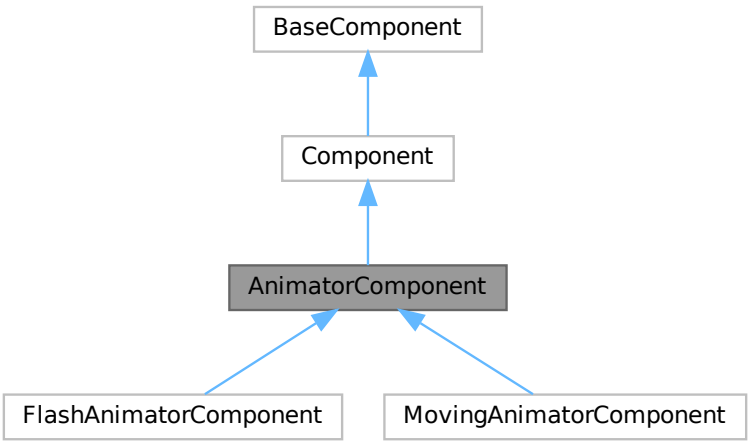
- int **indexX** = 0
- int **indexY** = 0
- size_t **total_frames** = 0
- float **speed** = 1.0f
- animType **type** = animType::ANIMTYPE_NONE
- int **reps** = 0
- int **frame_times_played** = 0
- int **cur_frame_index** = 0
- float **cur_frame_index_f** = 0
- int **times_played** = 0
- int **flow_direction** = 1
- bool **finished** = false

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

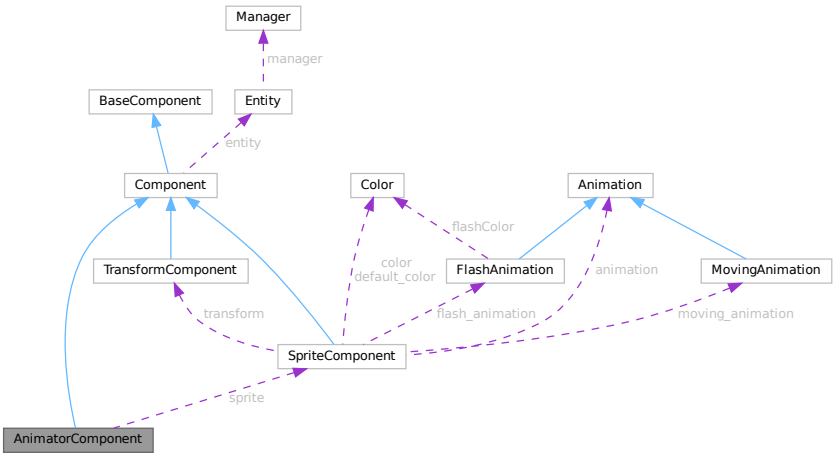
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/Animation.h

5.2 AnimatorComponent Class Reference

Inheritance diagram for AnimatorComponent:



Collaboration diagram for AnimatorComponent:



Public Member Functions

- **AnimatorComponent** (std::string id)
- void init () override
- void update (float deltaTime) override
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- void **Play** (std::string animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()
- void **DestroyTex** ()

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

Public Attributes

- SpriteComponent * **sprite** = nullptr
- std::string **textureid**
- std::string **animationName** = ""
- timestamp **resumeTime** = 0

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.2.1 Member Function Documentation

5.2.1.1 draw()

```
void AnimatorComponent::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.2.1.2 init()

```
void AnimatorComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.2.1.3 update()

```
void AnimatorComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/AnimatorComponent.↩
h

5.3 AnimatorManager Struct Reference

Public Member Functions

- void **InitializeAnimators** ()

Static Public Member Functions

- static AnimatorManager & **getInstance** ()

Public Attributes

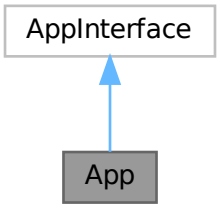
- std::map< std::string, Animation > **animations**
- std::map< std::string, MovingAnimation > **moving_animations**
- std::map< std::string, FlashAnimation > **flash_animations**

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

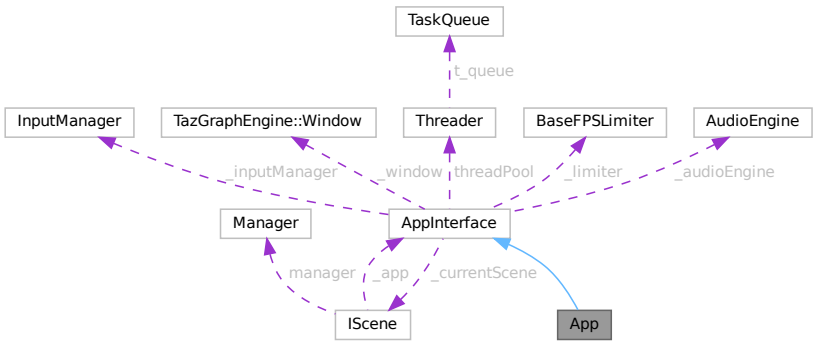
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/AnimatorManager.↩ h

5.4 App Class Reference

Inheritance diagram for App:



Collaboration diagram for App:



Public Member Functions

- **App** (int threadCount)
- virtual void onInit () override
- virtual void addScenes () override
- virtual void onExit () override

Public Member Functions inherited from AppInterface

- **AppInterface** (int threadCount)
- void **run** ()
- void **exitSimulator** ()
- void **onSDLEvent** (SDL_Event &evnt)
- BaseFPSLimiter & **getFPSLimiter** ()
- AudioEngine & **getAudioEngine** ()

Additional Inherited Members

Public Attributes inherited from AppInterface

- InputManager **_inputManager**
- TazGraphEngine::Window **_window**
- Threader **threadPool**

Protected Member Functions inherited from AppInterface

- virtual void **checkInput** ()
- virtual void **update** (float deltaTime)
- virtual void **draw** ()
- virtual void **updateUI** ()
- bool **init** ()
- bool **initSystems** ()

Protected Attributes inherited from AppInterface

- BaseFPSLimiter **_limiter**
- AudioEngine **_audioEngine**
- std::unique_ptr< SceneList > **_sceneList** = nullptr
- IScene * **_currentScene** = nullptr
- bool **_isRunning** = false
- const float **SCALE_SPEED** = 0.1f

5.4.1 Member Function Documentation

5.4.1.1 addScenes()

```
void App::addScenes ( )  [override], [virtual]
```

Implements AppInterface.

5.4.1.2 onExit()

```
void App::onExit ( )  [override], [virtual]
```

Implements AppInterface.

5.4.1.3 onInit()

```
void App::onInit ( )  [override], [virtual]
```

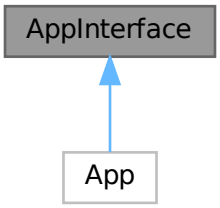
Implements AppInterface.

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

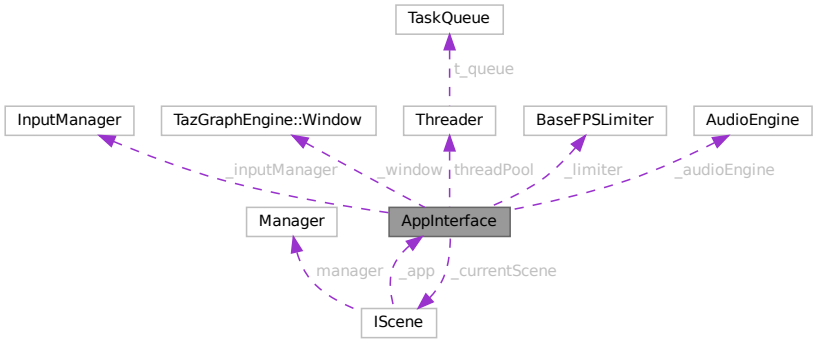
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/App/App.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/App/App.cpp

5.5 AppInterface Class Reference

Inheritance diagram for AppInterface:



Collaboration diagram for AppInterface:



Public Member Functions

- **AppInterface** (int threadCount)
- void **run** ()
- void **exitSimulator** ()
- virtual void **onInit** ()=0
- virtual void **addScenes** ()=0
- virtual void **onExit** ()=0
- void **onSDLEvent** (SDL_Event &evnt)
- BaseFPSLimiter & **getFPSLimiter** ()
- AudioEngine & **getAudioEngine** ()

Public Attributes

- InputManager **_inputManager**
- TazGraphEngine::Window **_window**
- Threader **threadPool**

Protected Member Functions

- virtual void **checkInput** ()
- virtual void **update** (float deltaTime)
- virtual void **draw** ()
- virtual void **updateUI** ()
- bool **init** ()
- bool **initSystems** ()

Protected Attributes

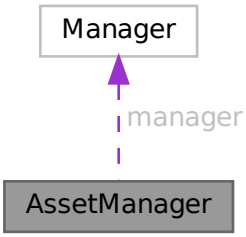
- BaseFPSLimiter **_limiter**
- AudioEngine **_audioEngine**
- std::unique_ptr< SceneList > **_sceneList** = nullptr
- IScene * **_currentScene** = nullptr
- bool **_isRunning** = false
- const float **SCALE_SPEED** = 0.1f

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GraphScreen/AppInterface.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GraphScreen/AppInterface.cpp

5.6 AssetManager Class Reference

Collaboration diagram for AssetManager:



Public Member Functions

- **AssetManager** (Manager ∗man, InputManager &inputManager, TazGraphEngine::Window &window)
- void **CreateWorldMap** (Entity &worldMap)
- void **CreateGroup** (Entity &groupNode, glm::vec3 centerGroup, float groupNodeSize, Grid::Level m_level)
- void **CreateGroupLink** (Entity &groupLink, Grid::Level m_level)
- void **createGroupLayout** (Grid::Level m_level)
- void **ungroupLayout** (Grid::Level m_level)

Public Attributes

- SDL_Color **black** = { 0, 0 ,0 ,255 }
- SDL_Color **white** = { 255, 255 ,255 ,255 }
- SDL_Color **red** = { 255, 0 ,0 ,255 }
- SDL_Color **green** = { 0, 255 ,0 ,255 }
- Manager ∗ **manager**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/AssetManager/AssetManager.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/AssetManager/AssetManager.cpp

5.7 AudioEngine Class Reference

Public Member Functions

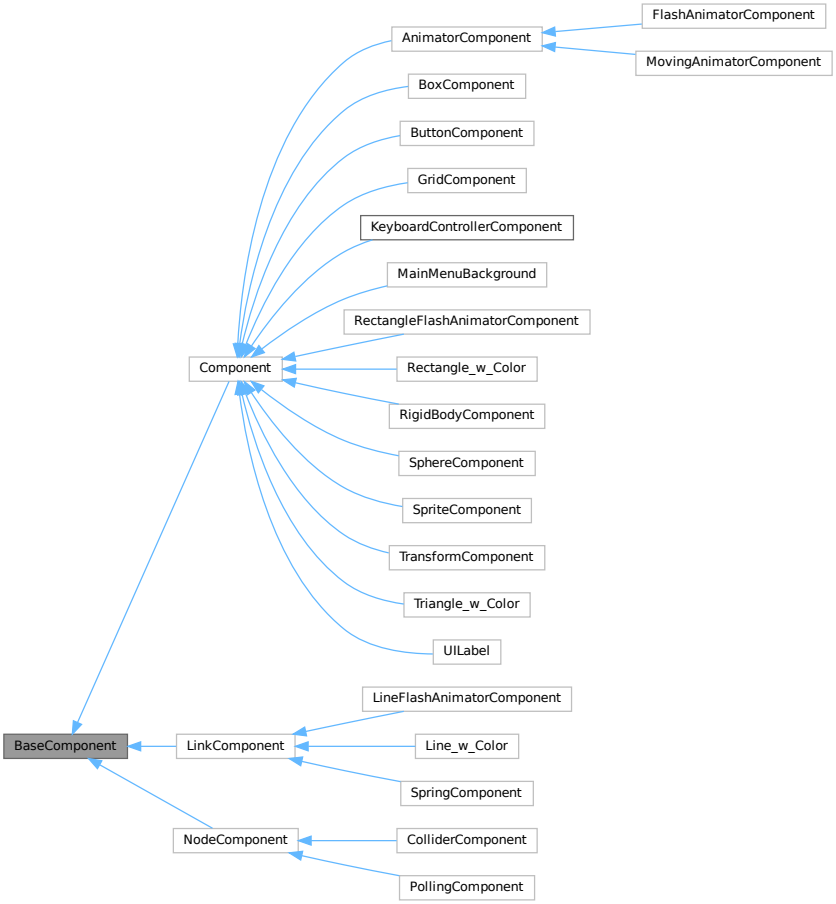
- void **init** ()
- void **destroy** ()
- SoundEffect **loadSoundEffect** (const std::string &filePath)
- Music **loadMusic** (const std::string &filePath)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/AudioEngine/AudioEngine.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/AudioEngine/AudioEngine.cpp

5.8 BaseComponent Class Reference

Inheritance diagram for BaseComponent:



Public Member Functions

- virtual void **init** ()
- virtual void **update** (float deltaTime)
- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

Public Attributes

- ComponentID **id** = 0u

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECS.h

5.9 BaseFPSLimiter Class Reference

Public Member Functions

- void **init** (float maxFPS)
- void **setMaxFPS** (float maxFPS)
- void **begin** ()
- float **end** ()
- void **setHistoryValue** (float currentFPS)

Public Attributes

- float **fpsHistory** [fps_history_count] = { 0 }
- int **fpsHistoryIndx** = 0
- float **fps**
- float **maxFPS**
- Uint32 **frameTime**
- Uint32 **startTicks**

Static Public Attributes

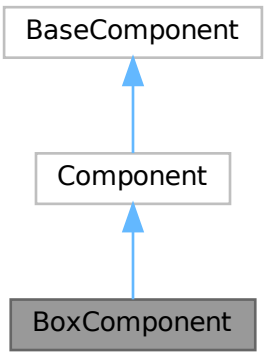
- static constexpr int **fps_history_count** = 100

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

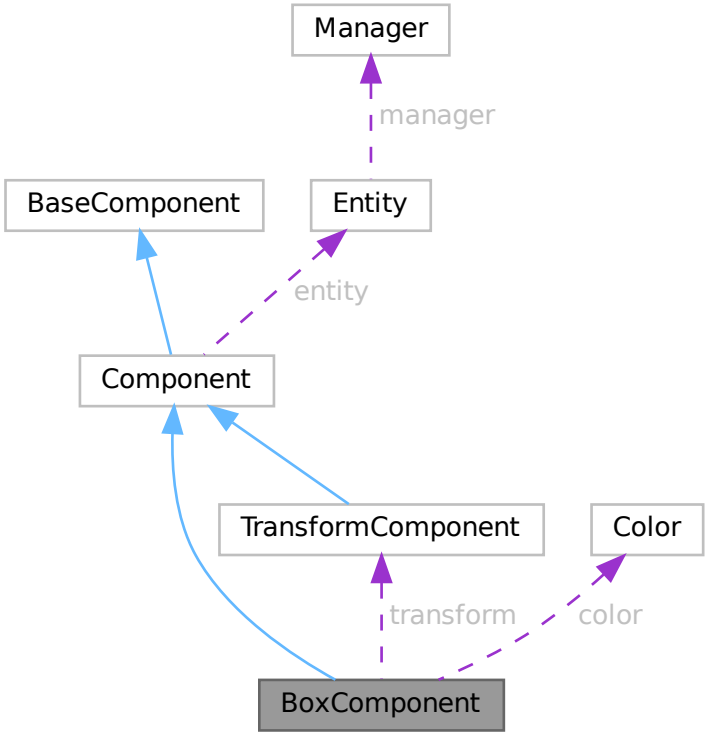
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/BaseFPSLimiter/BaseFPSLimiter.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/BaseFPSLimiter/BaseFPSLimiter.↩ cpp

5.10 BoxComponent Class Reference

Inheritance diagram for BoxComponent:



Collaboration diagram for BoxComponent:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void draw (size_t v_index, LightRenderer &batch, TazGraphEngine::Window &window)
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- Color **color** = { 255, 255, 255, 255 }
- SDL_Rect **destRect**
- TransformComponent * **transform** = nullptr
- float **temp_rotation** = 0.0f

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.10.1 Member Function Documentation

5.10.1.1 draw() [1/2]

```
void BoxComponent::draw (
            size_t v_index,
            LightRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.10.1.2 draw() [2/2]

```
void BoxComponent::draw (
            size_t v_index,
            PlaneColorRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.10.1.3 GetComponentName()

```
std::string BoxComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.10.1.4 init()

```
void BoxComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.10.1.5 update()

```
void BoxComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

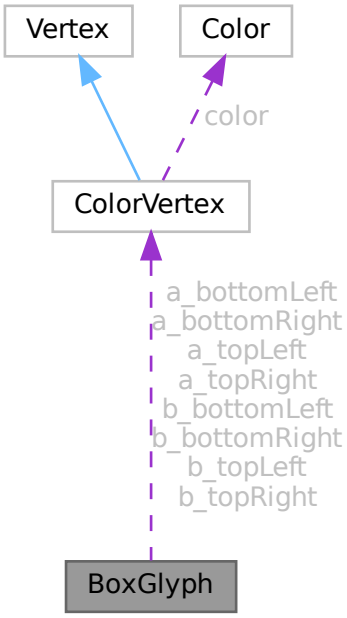
Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Basic/BoxComponent.h

5.11 BoxGlyph Class Reference

Collaboration diagram for BoxGlyph:



Public Member Functions

- **BoxGlyph** (const glm::vec3 &origin, const glm::vec3 &size, const Color &color, float angle)

Public Attributes

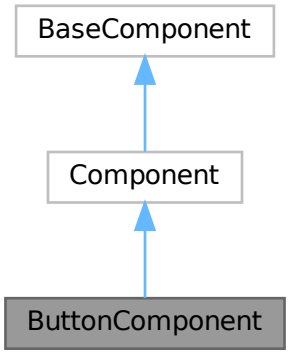
- ColorVertex **a_topLeft**
- ColorVertex **a_bottomLeft**
- ColorVertex **a_bottomRight**
- ColorVertex **a_topRight**
- ColorVertex **b_topLeft**
- ColorVertex **b_bottomLeft**
- ColorVertex **b_bottomRight**
- ColorVertex **b_topRight**

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

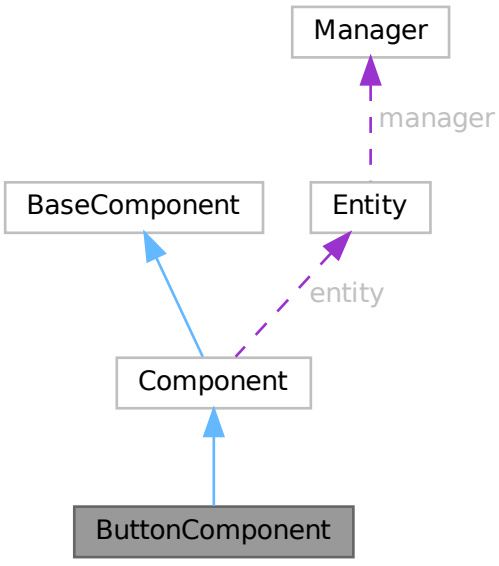
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LineRenderer/Line↩ Renderer.h

5.12 ButtonComponent Class Reference

Inheritance diagram for ButtonComponent:



Collaboration diagram for ButtonComponent:



Public Types

- enum class **ButtonState** { **NORMAL** , **HOVERED** , **PRESSED** }

Public Member Functions

- **ButtonComponent** (std::function< void()> onClick)
- **ButtonComponent** (std::function< void()> onClick, std::string button_label, glm::vec2 b_dimensions, Color b_background)
- void init () override
- void **setOnClick** (std::function< void()> newOnClick)
- void **setState** (ButtonState state)
- void update (float deltaTime) override
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Additional Inherited Members

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.12.1 Member Function Documentation

5.12.1.1 GetComponentName()

```
std::string ButtonComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.12.1.2 init()

```
void ButtonComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.12.1.3 update()

```
void ButtonComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Util/ButtonComponent.h

5.13 CameraManager Class Reference

Public Member Functions

- void **addCamera** (const std::string &name, std::shared_ptr< ICamera > camera)
- std::shared_ptr< ICamera > **getCamera** (const std::string &name)
- void **initializeCameras** ()

Static Public Member Functions

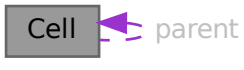
- static CameraManager & **getInstance** ()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Camera2.5D/CameraManager.h

5.14 Cell Struct Reference

Collaboration diagram for Cell:



Public Member Functions

- template<typename T >
  std::vector< T * > & **getEntityList** ()

Public Attributes

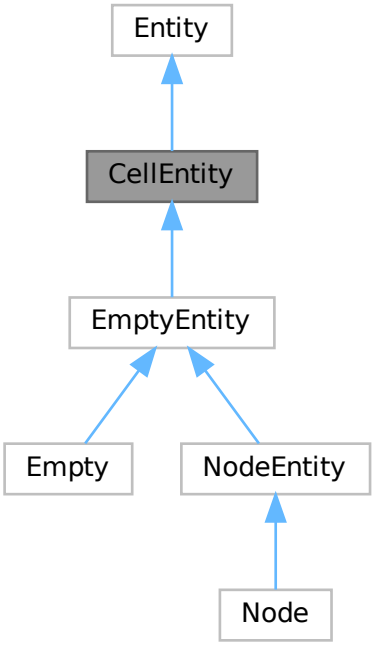
- std::vector< EmptyEntity * > **emptyEntities**
- std::vector< NodeEntity * > **nodes**
- std::vector< LinkEntity * > **links**
- glm::vec3 **boundingBox_origin** = glm::vec3(0)
- glm::vec3 **boundingBox_size** = glm::vec3(0)
- glm::vec3 **boundingBox_center** = glm::vec3(0)
- Cell * **parent** = nullptr
- std::vector< Cell * > **children**

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

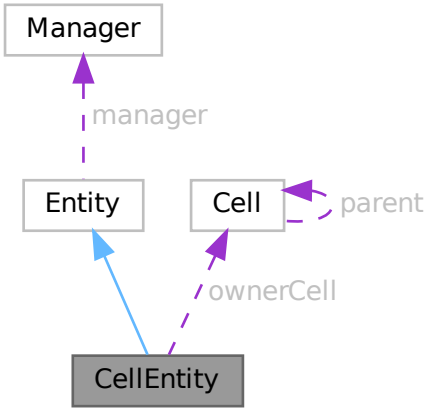
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/CellEntity.h

5.15 CellEntity Class Reference

Inheritance diagram for CellEntity:



Collaboration diagram for CellEntity:



Public Member Functions

- **CellEntity** (Manager &mManager)
- void **setOwnerCell** (Cell ∗cell)
- Cell ∗ getOwnerCell () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual void **update** (float deltaTime)
- virtual void **cellUpdate** ()
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- virtual void **destroy** ()
- bool **hasGroup** (Group mGroup)
- virtual void **addGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function

- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (Component *c)
- virtual void **setComponentEntity** (NodeComponent *c)
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual Entity * **getParentEntity** ()
- virtual void **setParentEntity** (Entity *pEntity)
- virtual void **imgui_print** ()
- virtual void **imgui_display** ()
- virtual void **removeEntity** ()

Public Attributes

- Cell * **ownerCell** = nullptr

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Additional Inherited Members

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

5.15.1 Member Function Documentation

5.15.1.1 getOwnerCell()

```
Cell * CellEntity::getOwnerCell ( ) const  [inline], [virtual]
```

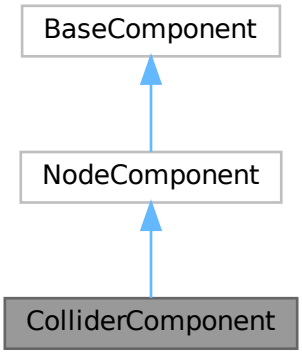
Reimplemented from Entity.

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

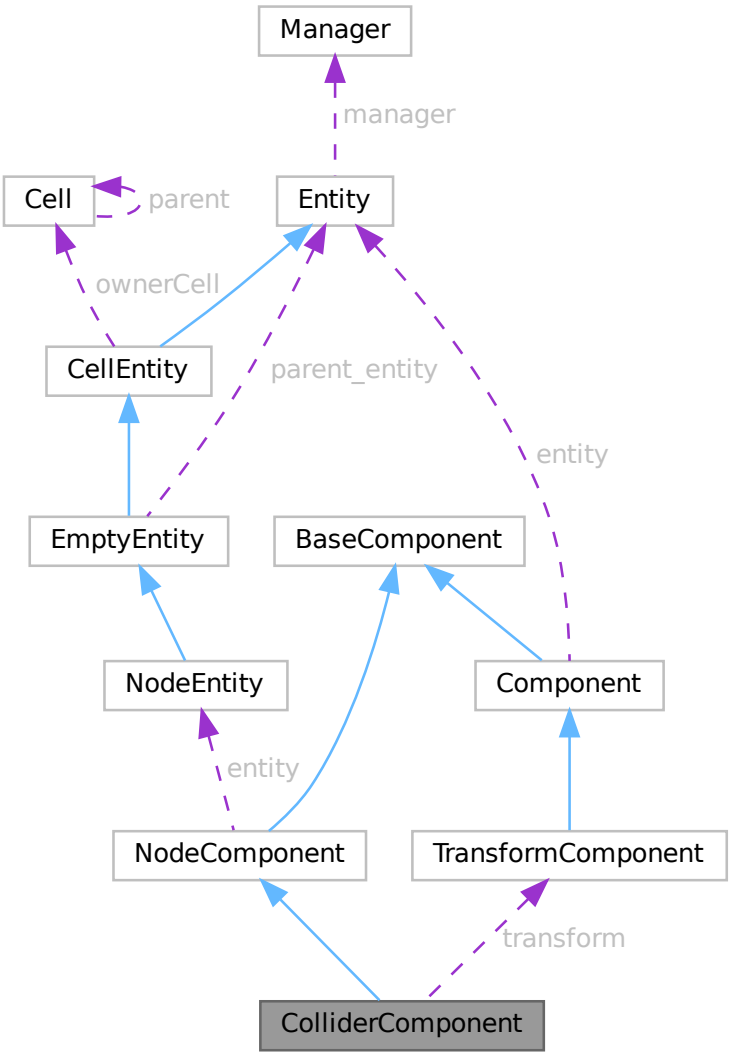
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/CellEntity.h

5.16 ColliderComponent Class Reference

Inheritance diagram for ColliderComponent:



Collaboration diagram for ColliderComponent:



Public Member Functions

- **ColliderComponent** (Manager *manager, glm::vec3 boxCollider_size)
- void init () override
- void update (float deltaTime) override
- void **collisionPhysics** ()
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- std::string GetComponentName () override
- void **addCollisionGroup** (Group g)
- void **removeCollisionGroup** (Group g)

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- glm::vec3 **box_collider** = glm::vec3(0.0f)
- TransformComponent * **transform** = nullptr

Public Attributes inherited from NodeComponent

- NodeEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.16.1 Member Function Documentation

5.16.1.1 draw()

```
void ColliderComponent::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.16.1.2 GetComponentName()

```
std::string ColliderComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.16.1.3 init()

```
void ColliderComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.16.1.4 update()

```
void ColliderComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Util/ColliderComponent.h

5.17 Color Struct Reference

Public Member Functions

- **Color** (GLubyte R, GLubyte G, GLubyte B, GLubyte A)
- glm::vec4 **toVec4** () const
- Color **operator∗** (float scalar) const
- Color **operator+** (const Color &other) const
- bool **operator==** (const Color &other) const

Static Public Member Functions

- static Color **fromVec4** (const glm::vec4 &v)

Public Attributes

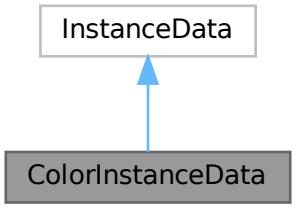
- GLubyte **r**
- GLubyte **g**
- GLubyte **b**
- GLubyte **a**

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

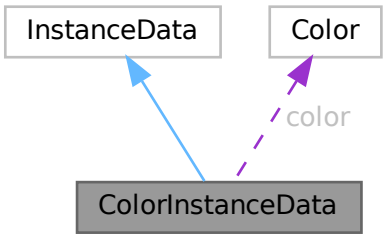
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Vertex.h

5.18 ColorInstanceData Struct Reference

Inheritance diagram for ColorInstanceData:



Collaboration diagram for ColorInstanceData:



Public Member Functions

- **ColorInstanceData** (glm::vec3 mSize, Position mBodyCenter, Rotation mRotation, Color mColor)
- **ColorInstanceData** (glm::vec2 mSize, Position mBodyCenter, Rotation mRotation, Color mColor)

Public Member Functions inherited from InstanceData

- **InstanceData** (glm::vec3 mSize, Position mBodyCenter, Rotation mRotation)
- **InstanceData** (glm::vec2 mSize, Position mBodyCenter, Rotation mRotation)

Public Attributes

- Color **color** = Color(255, 255, 255, 255)

Public Attributes inherited from InstanceData

- Size **size** = glm::vec3(0.0f)
- Position **bodyCenter** = glm::vec3(0.0f)
- Rotation **rotation** = glm::vec3(0.0f)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.19 ColorMeshRenderer Struct Reference

Public Attributes

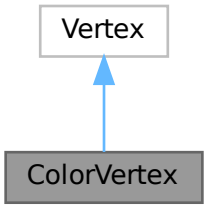
- size_t **meshIndices** = 0
- std::vector< ColorInstanceData > **instances**
- GLuint **vao**
- GLuint **vbo**
- GLuint **ibo**

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

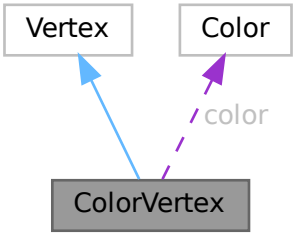
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.20 ColorVertex Struct Reference

Inheritance diagram for ColorVertex:



Collaboration diagram for ColorVertex:



Public Member Functions

- void **setColor** (GLubyte r, GLubyte g, GLubyte b, GLubyte a)

Public Member Functions inherited from Vertex

- void **setPosition** (Position m_position)

Public Attributes

- Color **color** = Color()

Public Attributes inherited from Vertex

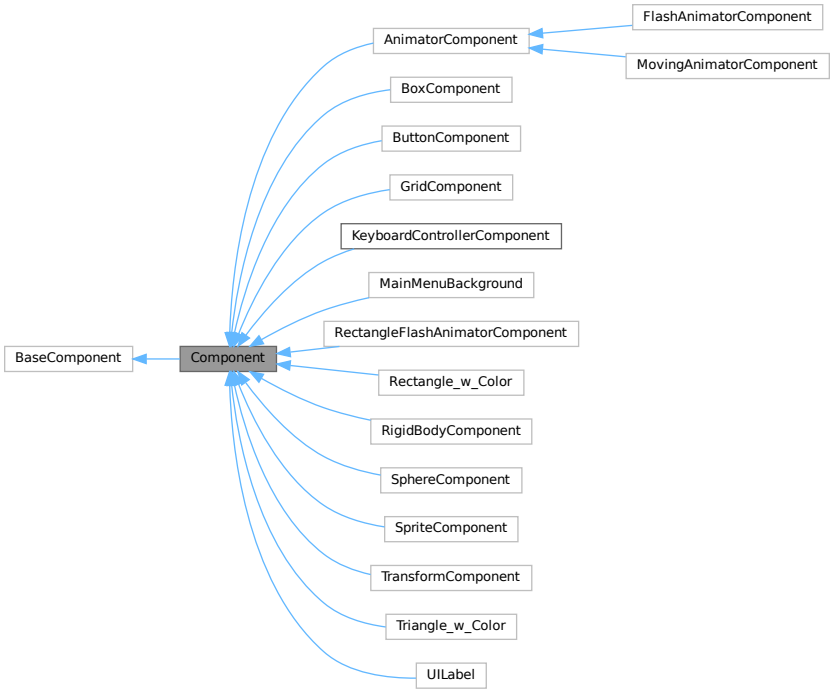
- Position **position** = Position(0)

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

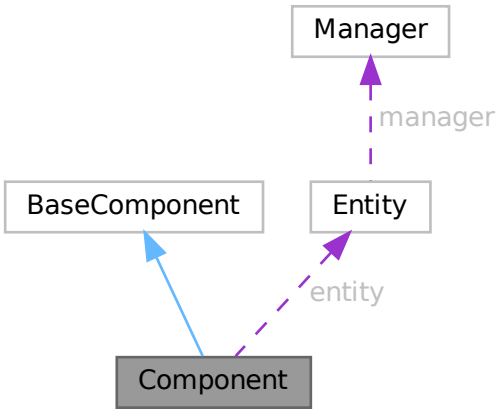
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Vertex.h

5.21 Component Class Reference

Inheritance diagram for Component:



Collaboration diagram for Component:



Public Attributes

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

Additional Inherited Members

Public Member Functions inherited from BaseComponent

- virtual void **init** ()
- virtual void **update** (float deltaTime)
- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECS.h

5.22 TazGraphEngine::ConsoleLogger Class Reference

Static Public Member Functions

- static void **log** (const std::string &message)
- static void **error** (const std::string &errorMessage)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/ConsoleLogger.h

5.23 CustomFunctions Class Reference

Public Member Functions

- void **renderUI** (Manager &manager, std::vector< std::pair< Entity *, glm::vec3 > > &m_selectedEntities)
- void **default_renderUI** ()
- void **CalculateDegree** (Manager &manager, std::vector< std::pair< Entity *, glm::vec3 > > &m_selected↩ Entities)
- void **CalculateSignals** ()
- void **CalculateHeatMap** ()
- void **DrawCandlestickChart** ()

Public Attributes

- bool **isScriptResultsOpen** = false
- int **activatedScriptShown** = 0

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/EditorIMGUI/CustomFunctions/CustomFunctions.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/EditorIMGUI/CustomFunctions/CustomFunctions.cpp

5.24 DataManager Class Reference

Static Public Member Functions

- static DataManager & **getInstance** ()

Public Attributes

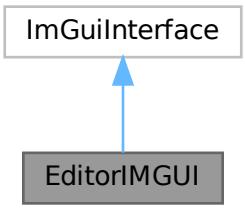
- std::string **mapToLoad**

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

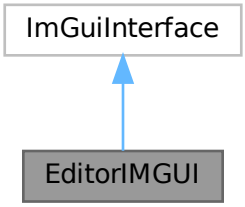
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/DataManager/DataManager.h

5.25 EditorIMGUI Class Reference

Inheritance diagram for EditorIMGUI:



Collaboration diagram for EditorIMGUI:



Public Member Functions

- bool **isSaving** ()
- void **setNewMap** (bool startingNew)
- bool **isStartingNew** ()
- bool **isLoading** ()
- void **setLoading** (bool loading)
- bool **isGoingBack** ()
- void **SetGoingBack** (bool goingBack)
- void **updateFileNamesInAssets** ()
- void **updatePollingFileNamesInAssets** ()
- bool * **getDockspaceRef** ()
- void **MenuBar** ()
- bool **isMouseOnWidget** (const std::string &widgetName)
- void **LeftColumnUIElement** (bool &renderDebug, bool &clusterLayout, glm::vec2 mouseCoords, glm::vec2 mouseCoords2, Manager &manager, Entity *selectedEntity, float(&backgroundColor)[4], int cell_size)
- void **RightColumnUIElement** (Manager &manager, float *nodeRadius)
- void **FPSCounter** (const BaseFPSLimiter &baseFPSLimiter)
- void **ReloadAccessibleFiles** ()
- void **SavingUI** (Map *map)
- void **NewMapUI** ()
- char * **LoadingUI** ()
- void **MainMenuUI** (std::function< void()> onStartSimulator, std::function< void()> onLoadSimulator, std::function< void()> onExitSimulator)
- void **ShowAllEntities** (Manager &manager, float &m_nodeRadius)
- void **availableFunctions** ()
- void **SceneViewport** (uint32_t textureId, ImVec2 &storedWindowPos, ImVec2 &storedWindowSize)
- void **scriptResultsVisualization** (Manager &manager, std::vector< std::pair< Entity *, glm::vec3 > > &m_selectedEntities)
- std::string **SceneTabs** (const std::vector< std::string > &graphNames, std::string &currentActive)
- void **ShowFunctionExecutionResults** ()
- void **updateIsMouseInSecondColumn** ()
- void **ShowEntityComponents** (glm::vec2 mousePos, Entity *displayedEntity, Manager &manager)
- void **ShowSceneControl** (glm::vec2 mousePos, Manager &manager)
- void **StartPollingComponent** (Entity *entity, const std::string &fileName)

Public Member Functions inherited from ImGuiInterface

- void **BeginRender** ()
- void **RenderUI** ()
- void **EndRender** ()

Public Attributes

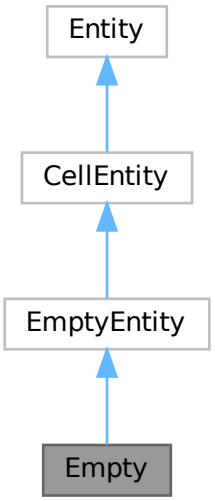
- float **cameraRotationZ** = 0
- int **newNodesCount** = 0
- int **newLinksCount** = 0
- float **interpolation** = 0.0f
- float **interpolation_speed** = 0.01f
- bool **interpolation_running** = false
- bool **isMouseInSecondColumn** = false
- int **last_activeLayout** = 0
- int **activeLayout** = 0

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

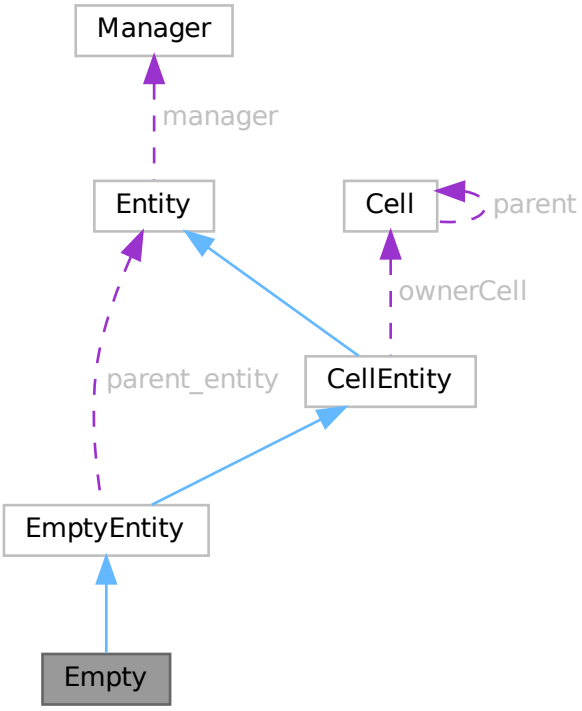
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/EditorIMGUI/EditorIMGUI.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/EditorIMGUI/EditorIMGUI.cpp

5.26 Empty Class Reference

Inheritance diagram for Empty:



Collaboration diagram for Empty:



Public Member Functions

- **Empty** (Manager &mManager)
- void addGroup (Group mGroup) override
- void update (float deltaTime)
- void cellUpdate () override
- void imgui_print () override
- void destroy ()

Public Member Functions inherited from EmptyEntity

- **EmptyEntity** (Manager &mManager)
- void setComponentEntity (Component *c) override
- Entity * getParentEntity () override
- void setParentEntity (Entity *pEntity) override
- void **removeFromCell** ()
- void removeEntity () override

Public Member Functions inherited from CellEntity

- **CellEntity** (Manager &mManager)
- void **setOwnerCell** (Cell *cell)
- Cell * getOwnerCell () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- bool **hasGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (NodeComponent *c)
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual void **imgui_display** ()

Additional Inherited Members

Public Attributes inherited from CellEntity

- Cell * **ownerCell** = nullptr

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Protected Attributes inherited from EmptyEntity

- Entity * **parent_entity** = nullptr

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

5.26.1 Member Function Documentation

5.26.1.1 addGroup()

```
void Empty::addGroup (
            Group mGroup )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.26.1.2 cellUpdate()

```
void Empty::cellUpdate ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.26.1.3 destroy()

```
void Empty::destroy ( )  [inline], [virtual]
```

Reimplemented from Entity.

5.26.1.4 imgui_print()

```
void Empty::imgui_print ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.26.1.5 update()

```
void Empty::update (
            float deltaTime )  [inline], [virtual]
```

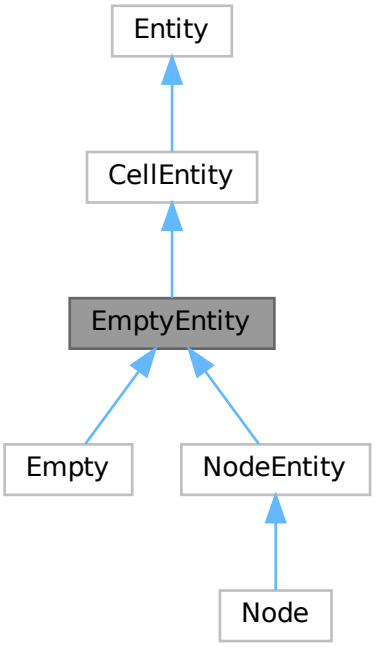
Reimplemented from Entity.

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

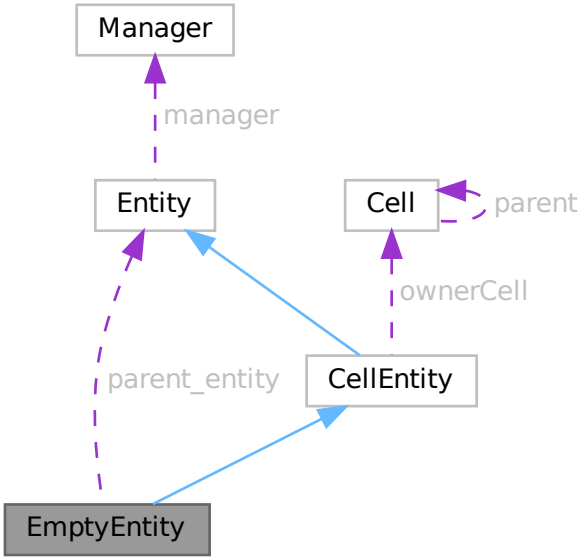
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSEntityTypes.h

5.27 EmptyEntity Class Reference

Inheritance diagram for EmptyEntity:



Collaboration diagram for EmptyEntity:



Public Member Functions

- **EmptyEntity** (Manager &mManager)
- void setComponentEntity (Component *c) override
- Entity * getParentEntity () override
- void setParentEntity (Entity *pEntity) override
- void **removeFromCell** ()
- void removeEntity () override

Public Member Functions inherited from CellEntity

- **CellEntity** (Manager &mManager)
- void **setOwnerCell** (Cell *cell)
- Cell * getOwnerCell () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual void **update** (float deltaTime)
- virtual void **cellUpdate** ()

- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- virtual void **destroy** ()
- bool **hasGroup** (Group mGroup)
- virtual void **addGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (NodeComponent *c)
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual void **imgui_print** ()
- virtual void **imgui_display** ()

Protected Attributes

- Entity * **parent_entity** = nullptr

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

Additional Inherited Members

Public Attributes inherited from CellEntity

- Cell * **ownerCell** = nullptr

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

5.27.1 Member Function Documentation

5.27.1.1 getParentEntity()

```
Entity * EmptyEntity::getParentEntity ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.27.1.2 removeEntity()

```
void EmptyEntity::removeEntity ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.27.1.3 setComponentEntity()

```
void EmptyEntity::setComponentEntity (
            Component * c )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.27.1.4 setParentEntity()

```
void EmptyEntity::setParentEntity (
            Entity * pEntity )  [inline], [override], [virtual]
```

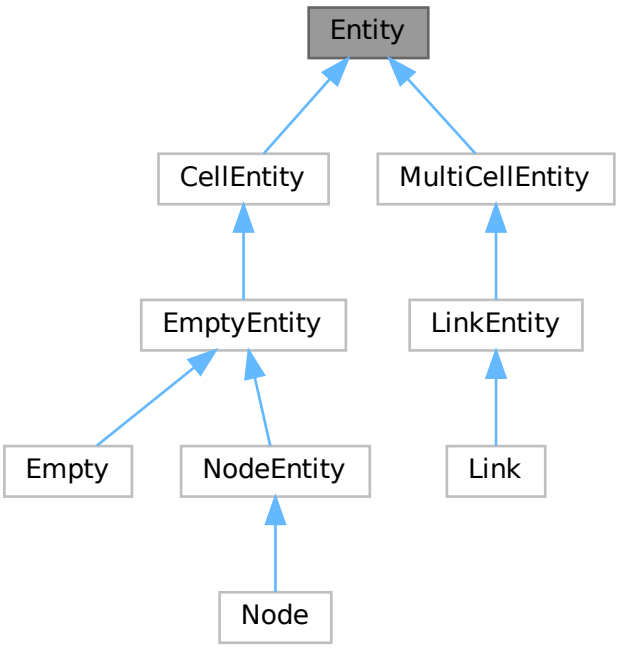
Reimplemented from Entity.

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

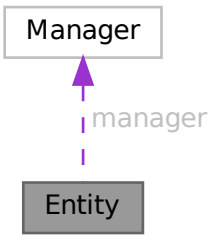
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSEntity.h

5.28 Entity Class Reference

Inheritance diagram for Entity:



Collaboration diagram for Entity:



Public Member Functions

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()

- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual void **update** (float deltaTime)
- virtual void **cellUpdate** ()
- virtual Cell * **getOwnerCell** () const
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- virtual void **destroy** ()
- bool **hasGroup** (Group mGroup)
- virtual void **addGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (Component *c)
- virtual void **setComponentEntity** (NodeComponent *c)
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual Entity * **getParentEntity** ()
- virtual void **setParentEntity** (Entity *pEntity)
- virtual void **imgui_print** ()
- virtual void **imgui_display** ()
- virtual void **removeEntity** ()

Public Attributes

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Protected Attributes

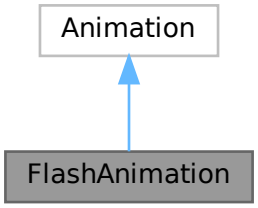
- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

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

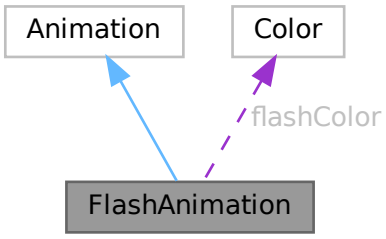
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECS.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECS.cpp

5.29 FlashAnimation Struct Reference

Inheritance diagram for FlashAnimation:



Collaboration diagram for FlashAnimation:



Public Types

- enum class **FlashState** { **FLASH_OUT** , **EASE_IN** , **FLASH_IN** , **EASE_OUT** }

Public Types inherited from Animation

- enum **animType** { **ANIMTYPE_NONE** = 0 , **ANIMTYPE_PLAY_N_TIMES** = 1 , **ANIMTYPE_LOOPED** = 2 , **ANIMTYPE_BACK_FORTH** = 3 }

Public Member Functions

- **FlashAnimation** (int ix, int iy, size_t f, float s, const std::string _type, const std::vector< float > &flashTimes, Color flashC, int _reps=0)
- **FlashAnimation** (int ix, int iy, size_t f, float s, const animType _type, const std::vector< float > &flashTimes, Color flashC, int _reps=0)
- void **advanceFrame** (float deltaTime)
- std::vector< float > **getSpeedsAsVector** () const

Public Member Functions inherited from Animation

- **Animation** (int ix, int iy, size_t f, float s, const std::string _type, int _reps=0)
- **Animation** (int ix, int iy, size_t f, float s, const animType _type, int _reps=0)
- void **advanceFrame** (float deltaTime)
- void **resetFrameIndex** ()
- bool **hasFinished** ()

Public Attributes

- float **interpolation_a** = 0.0f
- std::map< FlashState, float > **speeds**
- FlashState **currentSpeedIndex** = FlashState::FLASH_OUT
- Color **flashColor** = Color(255,255,255,255)

Public Attributes inherited from Animation

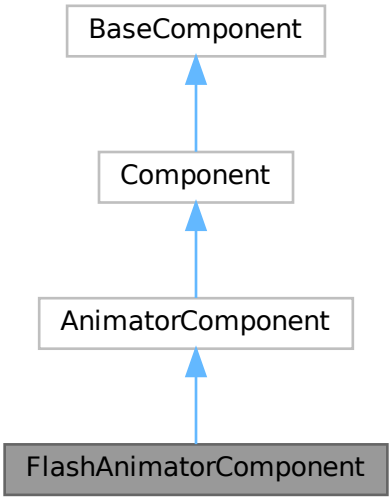
- int **indexX** = 0
- int **indexY** = 0
- size_t **total_frames** = 0
- float **speed** = 1.0f
- animType **type** = animType::ANIMTYPE_NONE
- int **reps** = 0
- int **frame_times_played** = 0
- int **cur_frame_index** = 0
- float **cur_frame_index_f** = 0
- int **times_played** = 0
- int **flow_direction** = 1
- bool **finished** = false

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

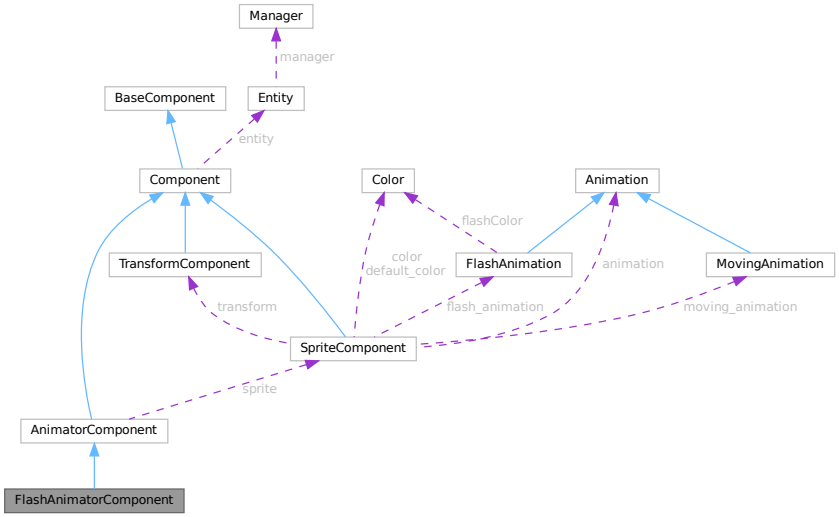
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/FlashAnimation.h

5.30 FlashAnimatorComponent Class Reference

Inheritance diagram for FlashAnimatorComponent:



Collaboration diagram for FlashAnimatorComponent:



Public Member Functions

- **FlashAnimatorComponent** ()

also we use MovingAnimator instead of simple Animator so that entities use less memory and we use it to entities that have triggers that change their animation

- **FlashAnimatorComponent** (std::string id)
- void init () override
- void update (float deltaTime) override
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- void **Play** (std::string animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()
- void **DestroyTex** ()

Public Member Functions inherited from AnimatorComponent

- **AnimatorComponent** (std::string id)
- void **Play** (std::string animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()
- void **DestroyTex** ()

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

Additional Inherited Members

Public Attributes inherited from AnimatorComponent

- SpriteComponent * **sprite** = nullptr
- std::string **textureid**
- std::string **animationName** = ""
- timestamp **resumeTime** = 0

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.30.1 Member Function Documentation

5.30.1.1 draw()

```
void FlashAnimatorComponent::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from AnimatorComponent.

5.30.1.2 init()

```
void FlashAnimatorComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from AnimatorComponent.

5.30.1.3 update()

```
void FlashAnimatorComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from AnimatorComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/FlashAnimator↩ Component.h

5.31 Framebuffer Class Reference

Public Member Functions

- void init (int windowWidth, int windowHeight)
- void **Bind** ()
- void **Unbind** ()

Public Attributes

- uint32_t **_framebufferTexture**

5.31.1 Member Function Documentation

5.31.1.1 init()

```
void Framebuffer::init (
            int windowWidth,
            int windowHeight )
```

invalidate()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/FrameBuffer/Framebuffer.↩ h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/FrameBuffer/Framebuffer.↩ cpp

5.32 GLSLProgram Class Reference

Public Member Functions

- void **compileShaders** (const std::string &vertexShaderFilePath, const std::string &fragmentShaderFilePath)
- void **compileShadersFromSource** (const char ∗vertexSource, const char ∗fragmentSource)
- void **linkShaders** ()
- void **addAttribute** (const std::string &attributeName)
- GLint **getUniformLocation** (const std::string &uniformName)
- void **use** ()
- void **unuse** ()
- void **dispose** ()
- GLuint **getProgramID** ()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.33 GLTexture Struct Reference

Public Attributes

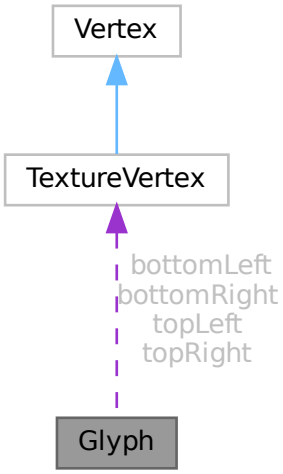
- GLuint **id**
- int **width**
- int **height**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLTexture.h

5.34 Glyph Class Reference

Collaboration diagram for Glyph:



Public Member Functions

- **Glyph** (const glm::vec2 &rectSize, const glm::vec3 &mRotation, const glm::vec4 &uvRect, GLuint texture, float Depth)

Public Attributes

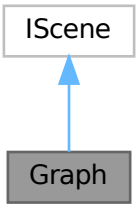
- GLuint **texture** = 0
- TextureVertex **topLeft**
- TextureVertex **bottomLeft**
- TextureVertex **topRight**
- TextureVertex **bottomRight**

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

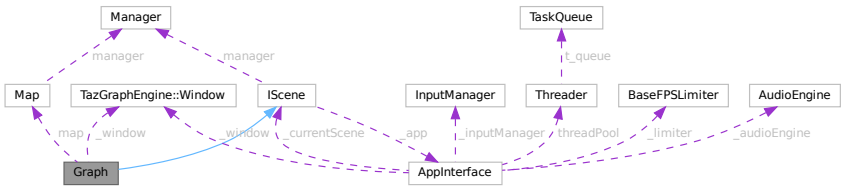
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/PlaneModelRenderer/Plane↩ ModelRenderer.h

5.35 Graph Class Reference

Inheritance diagram for Graph:



Collaboration diagram for Graph:



Public Member Functions

- **Graph** (TazGraphEngine::Window ∗window)
- virtual int getNextSceneIndex () const override
- virtual int getPreviousSceneIndex () const override
- virtual void build () override
- virtual void destroy () override
- virtual void onEntry () override
- virtual void onExit () override
- virtual void update (float deltaTime) override
- virtual void draw () override
- virtual void BeginRender () override
- virtual void updateUI () override
- virtual void EndRender () override
- void renderBatch (const std::vector< LinkEntity ∗ > &entities, LineRenderer &batch)
- void **renderBatch** (const std::vector< EmptyEntity ∗ > &entities, PlaneColorRenderer &batch)
- void **renderBatch** (const std::vector< NodeEntity ∗ > &entities, PlaneColorRenderer &batch)
- void **renderBatch** (const std::vector< EmptyEntity ∗ > &entities, PlaneModelRenderer &batch)
- void **renderBatch** (const std::vector< NodeEntity ∗ > &entities, PlaneModelRenderer &batch)
- void **renderBatch** (const std::vector< EmptyEntity ∗ > &entities, LightRenderer &batch)
- void **drawHUD** (const std::vector< NodeEntity ∗ > &entities)

Public Member Functions inherited from IScene

- int **getSceneIndex** () const
- void **setRunning** ()
- SceneState **getState** () const
- void **setParentApp** (AppInterface *app)
- AppInterface * **getApp** () const
- void **setManager** (std::string m_managerName)

Public Attributes

- Map * **map** = nullptr

Static Public Attributes

- static TazGraphEngine::Window * **_window** = nullptr

Additional Inherited Members

Protected Attributes inherited from IScene

- SceneState **_currentState** = SceneState::NONE
- AppInterface * **_app** = nullptr
- int **_sceneIndex** = -1
- std::unordered_map< std::string, Manager * > managers
- Manager * **manager** = nullptr
- std::string **managerName** = ""
- bool **_renderDebug** = false
- bool **_clusterLayout** = false

5.35.1 Member Function Documentation

5.35.1.1 BeginRender()

```
void Graph::BeginRender ( )  [override], [virtual]
```

Implements IScene.

5.35.1.2 build()

```
void Graph::build ( )  [override], [virtual]
```

Implements IScene.

5.35.1.3 destroy()

```
void Graph::destroy ( )  [override], [virtual]
```

Implements IScene.

5.35.1.4 draw()

```
void Graph::draw ( )  [override], [virtual]
```

Line Renderer Init

Color Renderer Init

Model Renderer Init

Light Renderer Init

this reduces a bit fps

Implements IScene.

5.35.1.5 EndRender()

```
void Graph::EndRender ( )  [override], [virtual]
```

Implements IScene.

5.35.1.6 getNextSceneIndex()

```
int Graph::getNextSceneIndex ( ) const  [override], [virtual]
```

Implements IScene.

5.35.1.7 getPreviousSceneIndex()

```
int Graph::getPreviousSceneIndex ( ) const  [override], [virtual]
```

Implements IScene.

5.35.1.8 onEntry()

```
void Graph::onEntry ( )  [override], [virtual]
```

Implements IScene.

5.35.1.9 onExit()

```
void Graph::onExit ( )  [override], [virtual]
```

Implements IScene.

5.35.1.10 renderBatch()

```
void Graph::renderBatch (
            const std::vector< LinkEntity * > & entities,
            LineRenderer & batch )
```

activate threads near the end, where we have completed everything else

5.35.1.11 update()

```
void Graph::update (
            float deltaTime )  [override], [virtual]
```

Implements IScene.

5.35.1.12 updateUI()

```
void Graph::updateUI ( )  [override], [virtual]
```

Implements IScene.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/Graph.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/Graph.cpp

5.36 Grid Class Reference

Public Types

- enum **Level** { **Basic** , **Outer1** , **Outer2** }

Public Member Functions

- **Grid** (int width, int height, int depth, int cellSize)
- void **setSize** (int cellSize)
- void **init** (int width, int height, int depth, int cellSize)
- void **createCells** (Grid::Level size)
- void **addLink** (LinkEntity *link, Grid::Level m_level)
- std::vector< Cell * > **getLinkCells** (const LinkEntity &link, Grid::Level m_level)
- void **addLink** (LinkEntity *link, std::vector< Cell * > cell)
- void **addEmpty** (EmptyEntity *entity, Grid::Level m_level)
- void **addNode** (NodeEntity *entity, Grid::Level m_level)
- void **addEmpty** (EmptyEntity *entity, Cell *cell)
- void **addNode** (NodeEntity *entity, Cell *cell)
- Cell * **getCell** (int x, int y, int z, Grid::Level m_level)
- Cell * **getCell** (const Entity &position, Grid::Level m_level)
- std::vector< Cell * > **getAdjacentCells** (int x, int y, int z, Grid::Level m_level)
- std::vector< Cell * > **getAdjacentCells** (const Entity &entity, Grid::Level m_level)

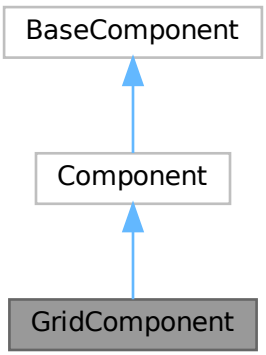
- std::vector< Cell > & **getCells** (Grid::Level m_level)
- int **getCellSize** ()
- int **getNumXCells** ()
- int **getNumYCells** ()
- int **getNumZCells** ()
- bool **setIntersectedCameraCells** (ICamera &camera)
- std::vector< Cell * > **getIntersectedCameraCells** (ICamera &camera)
- template<typename T >
  std::vector< T * > **getRevealedEntitiesInCameraCells** ()
- template<typename T >
  std::vector< T * > **getEntitiesInCameraCells** ()
- std::vector< LinkEntity * > **getLinksInCameraCells** ()
- bool **gridLevelChanged** ()
- Level **getGridLevel** ()
- void **setGridLevel** (Level newLevel)
- int **getLevelCellScale** ()
- int **getLevelCellScale** (Level level)

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

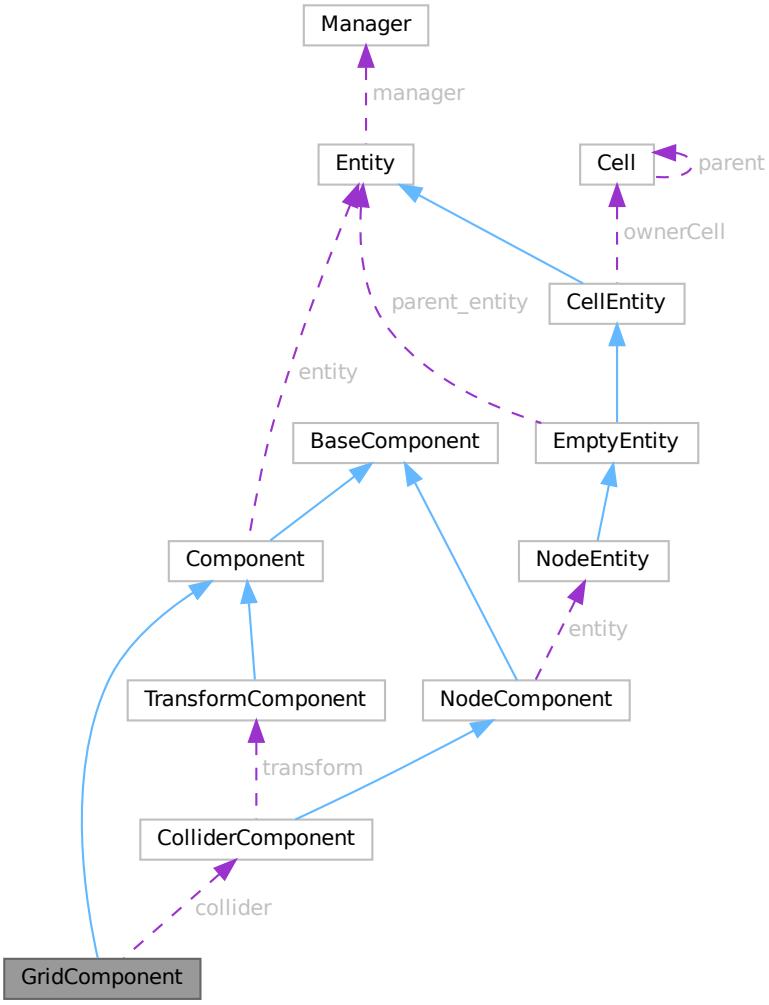
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Grid/Grid.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Grid/Grid.cpp

5.37 GridComponent Class Reference

Inheritance diagram for GridComponent:



Collaboration diagram for GridComponent:



Public Member Functions

- **GridComponent** (int xpos, int ypos, int tscaled, std::bitset< GRID_CELL_NUM > collider_bitSet=std::bitset< GRID_CELL_NUM >())
- void init () override
- void update (float deltaTime) override
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- glm::ivec2 **position**
- int **scaledTile**
- ColliderComponent * **collider** = nullptr

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.37.1 Member Function Documentation

5.37.1.1 GetComponentName()

```
std::string GridComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.37.1.2 init()

```
void GridComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.37.1.3 update()

```
void GridComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Util/GridComponent.h

5.38 GridLevelData Struct Reference

Public Attributes

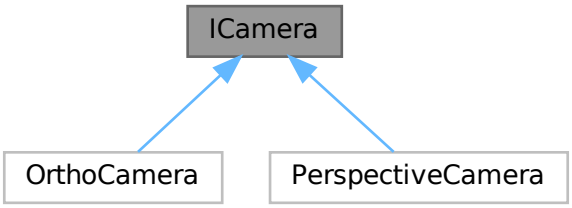
- float **numXCells**
- float **numYCells**
- float **numZCells**
- float **startX**
- float **endX**
- float **startY**
- float **endY**
- float **startZ**
- float **endZ**
- float **cameraMargin** = 0.0f

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Grid/Grid.h

5.39 ICamera Class Reference

Inheritance diagram for ICamera:



Public Member Functions

- virtual void **init** ()=0
- virtual void **update** ()=0
- virtual glm::vec2 **convertScreenToWorld** (glm::vec2 screenCoords) const =0
- virtual glm::ivec2 **getCameraDimensions** () const =0
- virtual SDL_FRect **getCameraRect** () const =0
- virtual glm::vec3 **getPosition** () const =0
- virtual void **setPosition** (const glm::vec3 newPosition)=0
- virtual void **setPosition_X** (const float newPosition)=0
- virtual void **setPosition_Y** (const float newPosition)=0
- virtual void **setPosition_Z** (const float newPosition)=0
- virtual float **getScale** () const =0

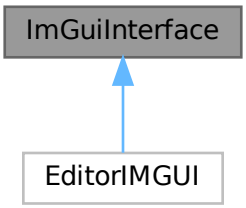
- virtual glm::mat4 **getCameraMatrix** () const =0
- virtual void **setScale** (float scale)=0
- virtual bool **isPointInCameraView** (const glm::vec4 point, float margin)=0
- virtual void **makeCameraDirty** ()=0
- virtual bool **hasChanged** ()=0
- virtual void **refreshCamera** ()=0

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Camera2.5D/ICamera.h

5.40 ImGuiInterface Class Reference

Inheritance diagram for ImGuiInterface:



Public Member Functions

- void **BeginRender** ()
- void **RenderUI** ()
- void **EndRender** ()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/ImGuiInterface/ImGuiInterface.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/ImGuiInterface/ImGuiInterface.cpp

5.41 InputManager Class Reference

Public Member Functions

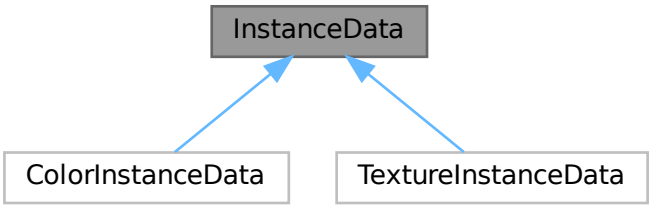
- void **update** ()
- void **pressKey** (unsigned int keyID)
- void **releaseKey** (unsigned int keyID)
- bool **isKeyDown** (unsigned int keyID)
- bool **isKeyPressed** (unsigned int keyID)
- bool **checkMouseCollision** (glm::vec2 position, glm::ivec2 tr_size)
- void **setMouseCoords** (float x, float y)
- glm::vec2 **getMouseCoords** () const
- void **setPanningPoint** (glm::vec2 position)
- glm::vec2 **calculatePanningDelta** (glm::vec2 position)
- void **setObjectRelativePos** (glm::vec2 relativeObjectPos)
- glm::vec2 **getObjectRelativePos** ()
- glm::vec2 **convertWindowToCameraCoords** (glm::vec2 mousePos, glm::vec2 viewportSize, glm::vec2 windowDimensions, const glm::vec2 &windowPos, const glm::vec2 &windowSize, const ICamera &camera)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/InputManager/InputManager.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/InputManager/InputManager.cpp

5.42 InstanceData Struct Reference

Inheritance diagram for InstanceData:



Public Member Functions

- **InstanceData** (glm::vec3 mSize, Position mBodyCenter, Rotation mRotation)
- **InstanceData** (glm::vec2 mSize, Position mBodyCenter, Rotation mRotation)

Public Attributes

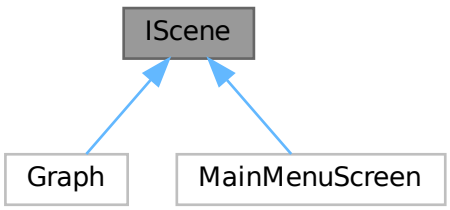
- Size **size** = glm::vec3(0.0f)
- Position **bodyCenter** = glm::vec3(0.0f)
- Rotation **rotation** = glm::vec3(0.0f)

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

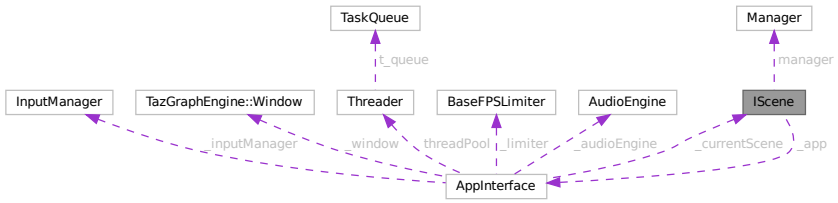
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.43 IScene Class Reference

Inheritance diagram for IScene:



Collaboration diagram for IScene:



Public Member Functions

- virtual int **getNextSceneIndex** () const =0
- virtual int **getPreviousSceneIndex** () const =0
- virtual void **build** ()=0
- virtual void **destroy** ()=0
- virtual void **onEntry** ()=0
- virtual void **onExit** ()=0
- virtual void **checkInput** ()=0

- virtual void **update** (float deltaTime)=0
- virtual void draw ()=0
- virtual void **BeginRender** ()=0
- virtual void **updateUI** ()=0
- virtual void **EndRender** ()=0
- int **getSceneIndex** () const
- void **setRunning** ()
- SceneState **getState** () const
- void **setParentApp** (AppInterface *app)
- AppInterface * **getApp** () const
- void **setManager** (std::string m_managerName)

Protected Attributes

- SceneState **_currentState** = SceneState::NONE
- AppInterface * **_app** = nullptr
- int **_sceneIndex** = -1
- std::unordered_map< std::string, Manager * > managers
- Manager * **manager** = nullptr
- std::string **managerName** = ""
- bool **_renderDebug** = false
- bool **_clusterLayout** = false

Friends

- class **SceneList**

5.43.1 Member Function Documentation

5.43.1.1 draw()

```
virtual void IScene::draw ( )  [pure virtual]
```

Implemented in Graph.

5.43.2 Member Data Documentation

5.43.2.1 managers

```
std::unordered_map<std::string, Manager*> IScene::managers  [protected]
```

Initial value:

```
= {
    }
```

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GraphScreen/IScene.h

5.44 JsonParser Class Reference

Public Member Functions

- **JsonParser** (const std::string &input)
- **JsonParser** (std::ifstream &file)
- JsonValue **parse** ()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/JsonParser/JsonParser.h

5.45 JsonValue Struct Reference

Public Attributes

- JsonType **type** = JsonType::Object
- std::map< std::string, JsonValue, NumericStringCompare > **obj**
- std::vector< JsonValue > **arr**
- std::string **str** = ""
- double **num** = -1
- bool **boolean** = false

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

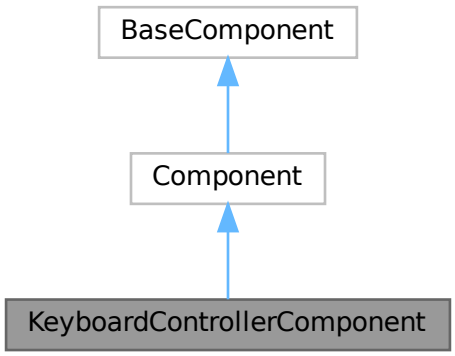
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/JsonParser/JsonParser.h

5.46 KeyboardControllerComponent Class Reference

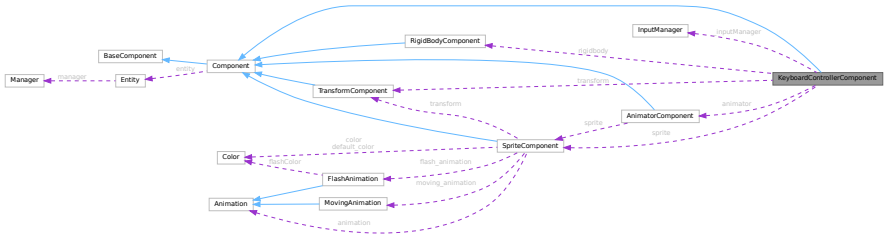
moving animation

`#include <KeyboardControllerComponent.h>`

Inheritance diagram for KeyboardControllerComponent:



Collaboration diagram for KeyboardControllerComponent:



Public Member Functions

- **KeyboardControllerComponent** (InputManager *inputManager, SDL_KeyCode walkUpKey, SDL_KeyCode walkLeftKey, SDL_KeyCode walkRightKey, SDL_KeyCode walkDownKey)
- void init () override
- void update (float deltaTime) override
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- InputManager * **_inputManager** = nullptr
- TransformComponent * **transform** = nullptr
- AnimatorComponent * **animator** = nullptr
- RigidBodyComponent * **rigidbody** = nullptr
- SpriteComponent * **sprite** = nullptr
- SDL_KeyCode **walkUpKey** = SDLK_UNKNOWN
- SDL_KeyCode **walkLeftKey** = SDLK_UNKNOWN
- SDL_KeyCode **walkRightKey** = SDLK_UNKNOWN
- SDL_KeyCode **walkDownKey** = SDLK_UNKNOWN

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.46.1 Detailed Description

moving animation

5.46.2 Member Function Documentation

5.46.2.1 GetComponentName()

```
std::string KeyboardControllerComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.46.2.2 init()

```
void KeyboardControllerComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.46.2.3 update()

```
void KeyboardControllerComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Util/KeyboardControllerComponent.h

5.47 LightRenderer Class Reference

Public Member Functions

- void **init** ()
- void **begin** ()
- void **end** ()
- void **initLightTriangleBatch** (size_t mSize)
- void **initLightQuadBatch** (size_t mSize)
- void **initLightBoxBatch** (size_t mSize)
- void **initLightSphereBatch** (size_t mSize)
- void **initBatchSize** ()
- void **drawTriangle** (size_t v_index, const glm::vec3 &depth, const glm::vec3 &cpuRotation, const Color &color)
- void **draw** (size_t v_index, const glm::vec2 &rectSize, const glm::vec3 &bodyCenter, const glm::vec3 &m↩ Rotation, const Color &color)

  draws are needed to convert the pos and size to vertices
- void **drawBox** (size_t v_index, const glm::vec3 &boxSize, const glm::vec3 &bodyCenter, const glm::vec3 &mRotation, const Color &color)
- void **drawSphere** (size_t v_index, const glm::vec3 &sphereSize, const glm::vec3 &bodyCenter, const glm↩ ::vec3 &mRotation, const Color &color)
- void **renderBatch** (GLSLProgram *glsl_program)
- void **dispose** ()

Public Attributes

- std::vector< LightVertex > sphereVertices
- std::vector< GLuint > sphereIndices

5.47.1 Member Data Documentation

5.47.1.1 sphereIndices

```
std::vector<GLuint> LightRenderer::sphereIndices
```

Initial value:

```
= {

    }
```

5.47.1.2 sphereVertices

```
std::vector<LightVertex> LightRenderer::sphereVertices
```

Initial value:

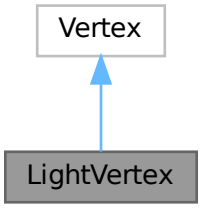
```
= {

    }
```

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

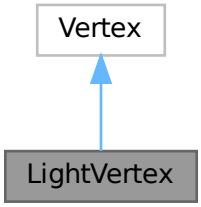
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LightRenderer/Light↩ Renderer.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LightRenderer/Light↩ Renderer.cpp

5.48 LightVertex Struct Reference

Inheritance diagram for LightVertex:



Collaboration diagram for LightVertex:



Public Attributes

- Normal **normal** = Normal()

Public Attributes inherited from Vertex

- Position **position** = Position(0)

Additional Inherited Members

Public Member Functions inherited from Vertex

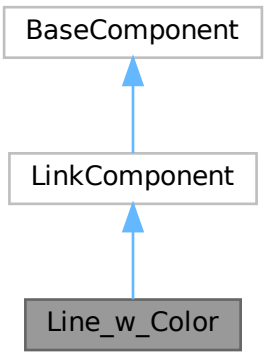
- void **setPosition** (Position m_position)

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

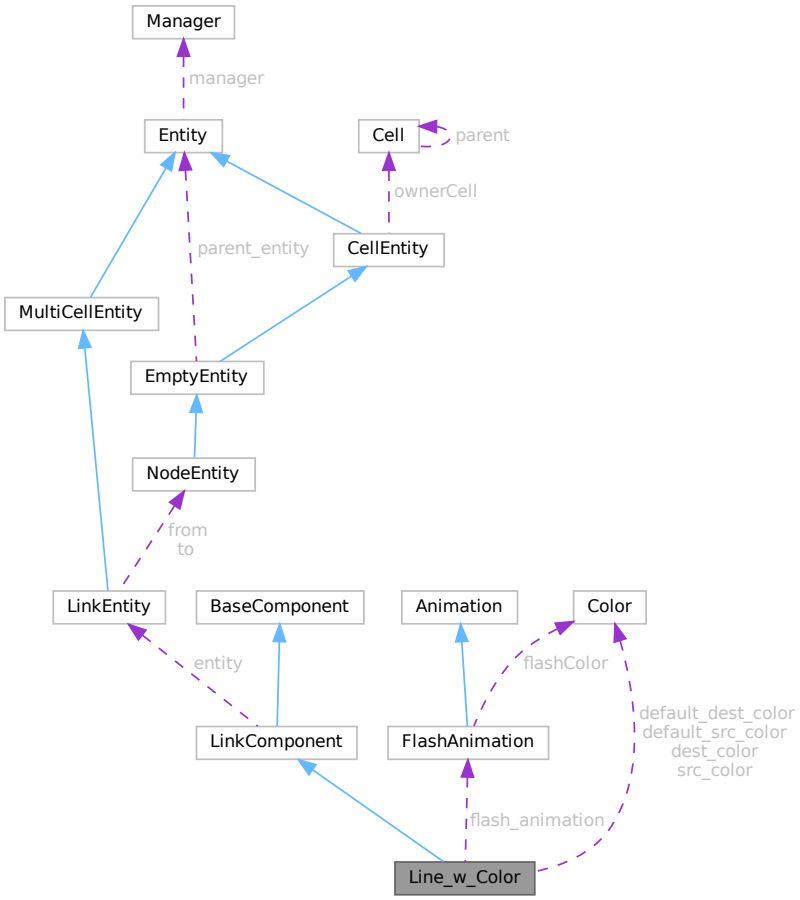
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Vertex.h

5.49 Line_w_Color Class Reference

Inheritance diagram for Line_w_Color:



Collaboration diagram for Line_w_Color:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **drawWithPorts** (size_t v_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **setSrcColor** (Color clr)
- void **setDestColor** (Color clr)
- void **SetFlashAnimation** (int idX, int idY, size_t fr, float sp, const Animation::animType type, const std::vector< float > &flashTimes, Color flashC, int reps=0)
- void **setFlashFrame** ()
- std::string GetComponentName () override
- void showGUI () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)

Public Attributes

- std::string **temp_lineParsed** = ""
- Color **default_src_color** = { 255, 255, 255, 255 }
- Color **src_color** = { 255, 255, 255, 255 }
- Color **default_dest_color** = { 255, 255, 255, 255 }
- Color **dest_color** = { 255, 255, 255, 255 }
- FlashAnimation **flash_animation**

Public Attributes inherited from LinkComponent

- LinkEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.49.1 Member Function Documentation

5.49.1.1 draw()

```
void Line_w_Color::draw (
            size_t v_index,
            LineRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.49.1.2 GetComponentName()

```
std::string Line_w_Color::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.49.1.3 init()

```
void Line_w_Color::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.49.1.4 showGUI()

```
void Line_w_Color::showGUI ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.49.1.5 update()

```
void Line_w_Color::update (
            float deltaTime )  [inline], [override], [virtual]
```

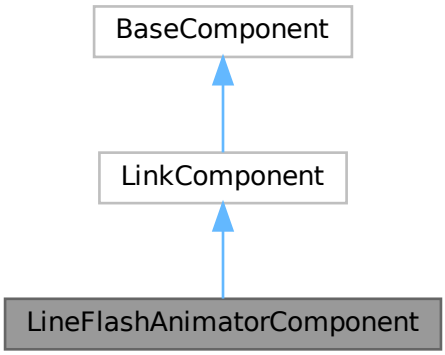
Reimplemented from BaseComponent.

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

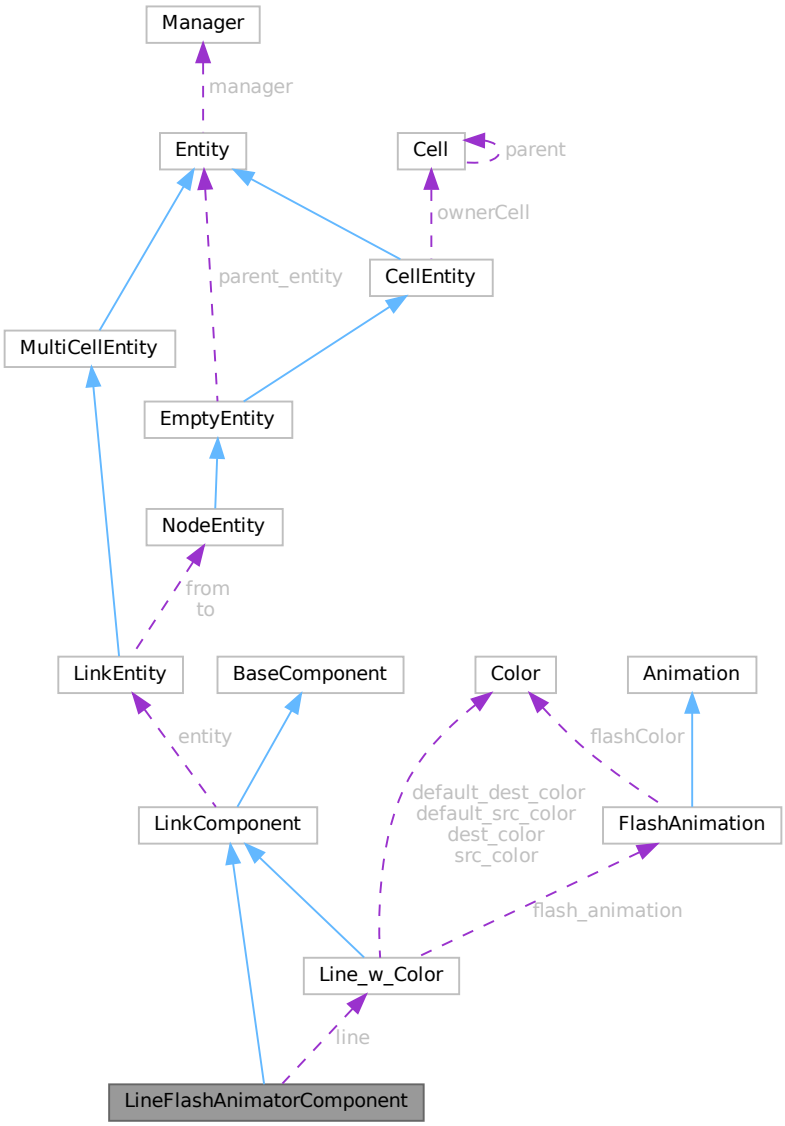
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Link/Basic/Line↩ _w_Color.h

5.50 LineFlashAnimatorComponent Class Reference

Inheritance diagram for LineFlashAnimatorComponent:



Collaboration diagram for LineFlashAnimatorComponent:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- void **Play** (const char *animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)

- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

Public Attributes

- Line_w_Color * **line** = nullptr

  also we use MovingAnimator instead of simple Animator so that entities use less memory and we use it to entities that have triggers that change their animation
- std::string **animationName** = ""
- timestamp **resumeTime** = 0

Public Attributes inherited from LinkComponent

- LinkEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.50.1 Member Function Documentation

5.50.1.1 draw()

```
void LineFlashAnimatorComponent::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.50.1.2 init()

```
void LineFlashAnimatorComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.50.1.3 update()

```
void LineFlashAnimatorComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

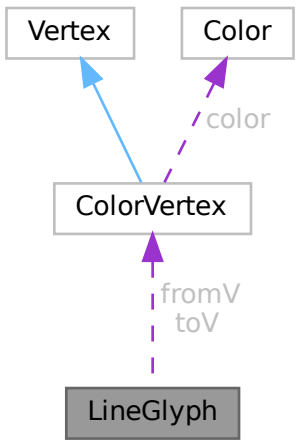
Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/LineAnimators/Line↩ FlashAnimatorComponent.h

5.51 LineGlyph Class Reference

Collaboration diagram for LineGlyph:



Public Member Functions

- **LineGlyph** (const glm::vec3 &fromPosition, const glm::vec3 &toPosition, const Color &srcColor, const Color &destColor)

Public Attributes

- ColorVertex **fromV**
- ColorVertex **toV**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LineRenderer/Line↩ Renderer.h

5.52 LineRenderer Class Reference

Public Member Functions

- void **init** ()
- void **begin** ()
- void **end** ()
- void **initBatchLines** (size_t msize)
- void **initBatchSquares** (size_t msize)
- void **initBatchBoxes** (size_t msize)

- void **drawLine** (size_t v_index, const glm::vec3 srcPosition, const glm::vec3 destPosition, const Color &src↩ Color, const Color &destColor)
- void **drawRectangle** (size_t v_index, const glm::vec4 &destRect, const Color &color, float angle, float z↩ Index=0.0f)
- void **drawBox** (size_t v_index, const glm::vec3 &origin, const glm::vec3 &size, const Color &color, float angle)
- void **drawCircle** (const glm::vec2 &center, const Color &color, float radius)
- void **initBatchSize** ()
- void **renderBatch** (float lineWidth)
- void **dispose** ()

Public Attributes

- const char * VERT_SRC
- const char * FRAG_SRC
- int box_edgePairs [12][2]

5.52.1 Member Data Documentation

5.52.1.1 box_edgePairs

```
int LineRenderer::box_edgePairs[12][2]
```

Initial value:

```
= {
            {0, 1}, {1, 2}, {2, 3}, {3, 0},
            {4, 5}, {5, 6}, {6, 7}, {7, 4},
            {0, 4}, {1, 5}, {2, 6}, {3, 7}
    }
```

5.52.1.2 FRAG_SRC

```
const char* LineRenderer::FRAG_SRC
```

Initial value:

```
= R"(#version 400

in vec4 fragmentColor;

out vec4 color; //rgb value

void main() {
    color = vec4(fragmentColor.rgb, fragmentColor.a);
})"
```

5.52.1.3 VERT_SRC

```
const char* LineRenderer::VERT_SRC
```

Initial value:

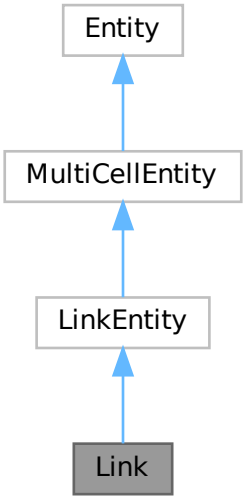
```
= R"(#version 400

in vec3 vertexPosition; //vec3 is array of 3 floats
in vec4 vertexColor;

out vec4 fragmentColor;

uniform mat4 projection;

void main() {
    gl_Position = projection * vec4(vertexPosition.xyz, 1.0);

    fragmentColor = vertexColor;
})"
```

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

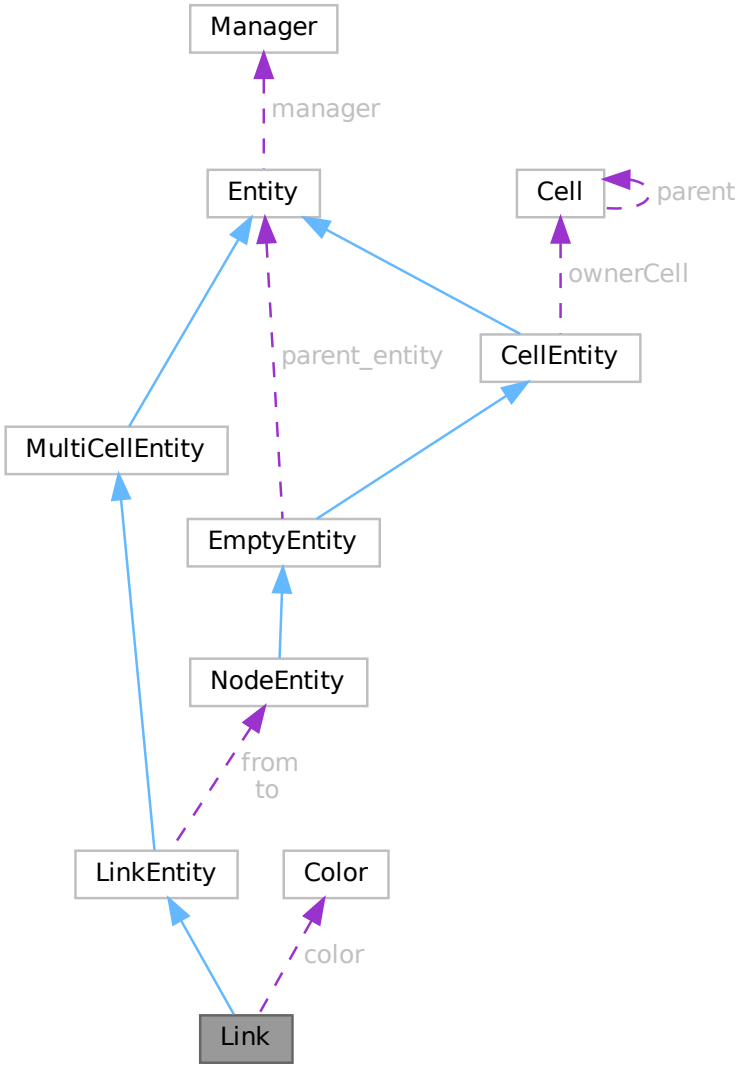
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LineRenderer/LineRenderer.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LineRenderer/LineRenderer.cpp

5.53 Link Class Reference

Inheritance diagram for Link:



Collaboration diagram for Link:



Public Member Functions

- **Link** (Manager &mManager)
- **Link** (Manager &mManager, unsigned int mfromId, unsigned int mtoId)
- **Link** (Manager &mManager, Entity *mfrom, Entity *mto)
- **Link** (Manager &mManager, NodeEntity *mfrom, NodeEntity *mto)
- void addGroup (Group mGroup) override
- void update (float deltaTime) override
- void cellUpdate () override
- void updateArrowHeads () override
- void updateLinkToPorts () override
- void addArrowHead () override
- void removeArrowHead () override

- void updateLinkToNodes () override
- std::string **getBestPortForConnection** (const glm::vec3 &fromPos, const glm::vec3 &toPos)
- void imgui_print () override
- void imgui_display () override
- void destroy ()

Public Member Functions inherited from LinkEntity

- **LinkEntity** (Manager &mManager)
- **LinkEntity** (Manager &mManager, unsigned int mfromId, unsigned int mtoId)
- **LinkEntity** (Manager &mManager, NodeEntity *mfrom, NodeEntity *mto)
- void setComponentEntity (LinkComponent *c) override
- void **removeFromCells** ()
- void removeEntity () override
- NodeEntity * **getFromNode** () const
- NodeEntity * **getToNode** () const
- EmptyEntity * **getFromPort** ()
- EmptyEntity * **getToPort** ()

Public Member Functions inherited from MultiCellEntity

- **MultiCellEntity** (Manager &mManager)
- void **setOwnerCells** (std::vector< Cell * > cells)
- Cell * **getOwnerCells** () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual Cell * **getOwnerCell** () const
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- bool **hasGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (Component *c)
- virtual void **setComponentEntity** (NodeComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual Entity * **getParentEntity** ()
- virtual void **setParentEntity** (Entity *pEntity)

Public Attributes

- Color **color** = {}

Public Attributes inherited from LinkEntity

- std::string **fromPort**
- std::string **toPort**

Public Attributes inherited from MultiCellEntity

- std::vector< Cell * > **ownerCells** = {}

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Additional Inherited Members

Protected Attributes inherited from LinkEntity

- unsigned int **fromId** = 0
- unsigned int **toId** = 0
- NodeEntity * **from** = nullptr
- NodeEntity * **to** = nullptr

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

5.53.1 Member Function Documentation

5.53.1.1 addArrowHead()

```
void Link::addArrowHead ( )  [inline], [override], [virtual]
```

Reimplemented from LinkEntity.

5.53.1.2 addGroup()

```
void Link::addGroup (
            Group mGroup )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.53.1.3 cellUpdate()

```
void Link::cellUpdate ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.53.1.4 destroy()

```
void Link::destroy ( )  [inline], [virtual]
```

Reimplemented from Entity.

5.53.1.5 imgui_display()

```
void Link::imgui_display ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.53.1.6 imgui_print()

```
void Link::imgui_print ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.53.1.7 removeArrowHead()

```
void Link::removeArrowHead ( )  [inline], [override], [virtual]
```

Reimplemented from LinkEntity.

5.53.1.8 update()

```
void Link::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.53.1.9 updateArrowHeads()

```
void Link::updateArrowHeads ( )  [inline], [override], [virtual]
```

Reimplemented from LinkEntity.

5.53.1.10 updateLinkToNodes()

```
void Link::updateLinkToNodes ( )  [inline], [override], [virtual]
```

Reimplemented from LinkEntity.

5.53.1.11 updateLinkToPorts()

```
void Link::updateLinkToPorts ( )  [inline], [override], [virtual]
```

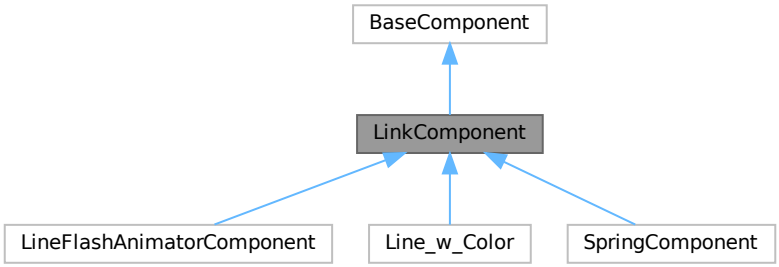
Reimplemented from LinkEntity.

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

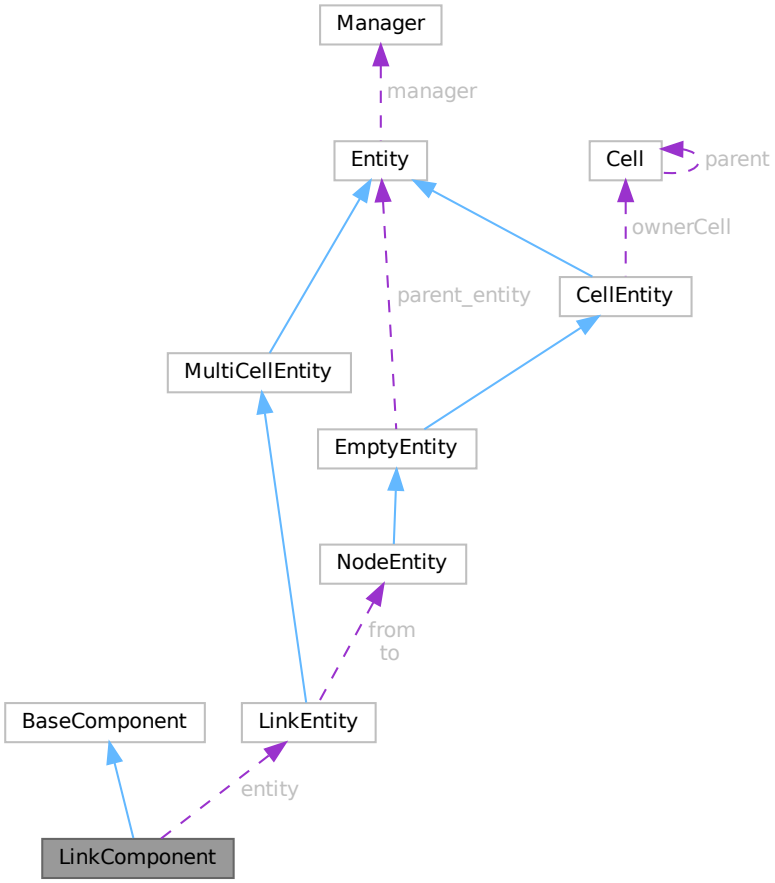
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSEntityTypes.h

5.54 LinkComponent Class Reference

Inheritance diagram for LinkComponent:



Collaboration diagram for LinkComponent:



Public Attributes

- LinkEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

Additional Inherited Members

Public Member Functions inherited from BaseComponent

- virtual void **init** ()
- virtual void **update** (float deltaTime)
- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)

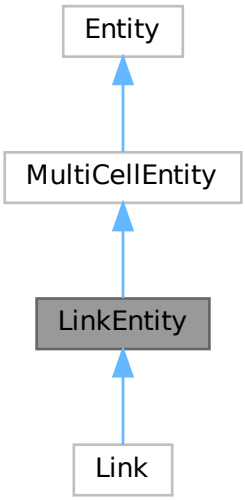
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

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

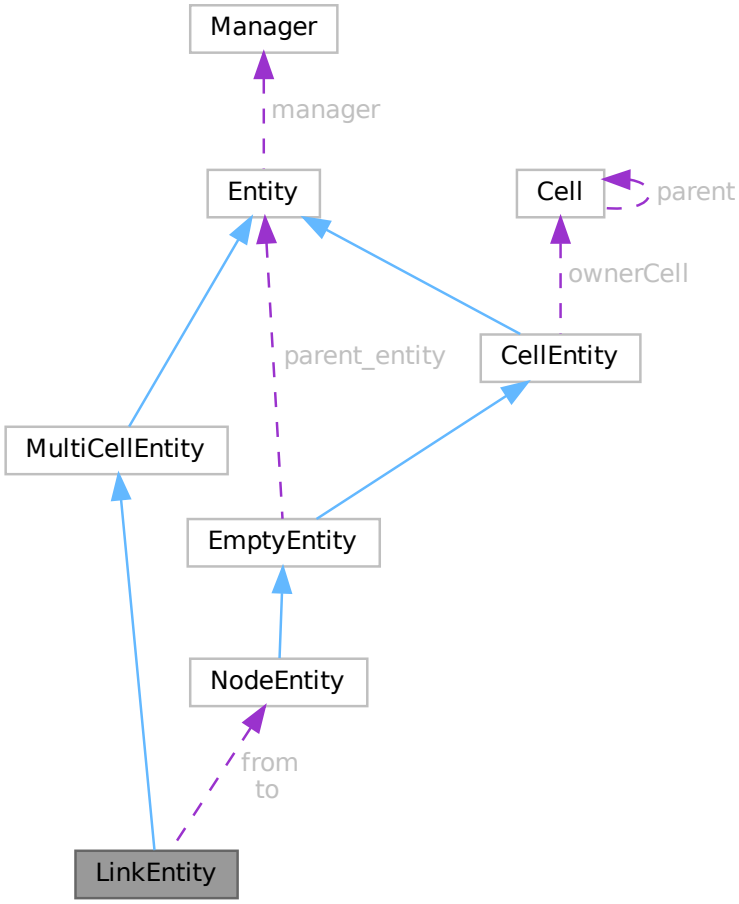
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECS.h

5.55 LinkEntity Class Reference

Inheritance diagram for LinkEntity:



Collaboration diagram for LinkEntity:



Public Member Functions

- **LinkEntity** (Manager &mManager)
- **LinkEntity** (Manager &mManager, unsigned int mfromId, unsigned int mtoId)
- **LinkEntity** (Manager &mManager, NodeEntity *mfrom, NodeEntity *mto)
- void setComponentEntity (LinkComponent *c) override
- void **removeFromCells** ()
- void removeEntity () override
- NodeEntity * **getFromNode** () const
- NodeEntity * **getToNode** () const
- EmptyEntity * **getFromPort** ()
- EmptyEntity * **getToPort** ()
- virtual void **updateLinkToPorts** ()
- virtual void **updateLinkToNodes** ()
- virtual void **updateArrowHeads** ()
- virtual void **addArrowHead** ()
- virtual void **removeArrowHead** ()

Public Member Functions inherited from MultiCellEntity

- **MultiCellEntity** (Manager &mManager)
- void **setOwnerCells** (std::vector< Cell * > cells)
- Cell * **getOwnerCells** () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual void **update** (float deltaTime)
- virtual void **cellUpdate** ()
- virtual Cell * **getOwnerCell** () const
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- virtual void **destroy** ()
- bool **hasGroup** (Group mGroup)
- virtual void **addGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (Component *c)
- virtual void **setComponentEntity** (NodeComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual Entity * **getParentEntity** ()
- virtual void **setParentEntity** (Entity *pEntity)
- virtual void **imgui_print** ()
- virtual void **imgui_display** ()

Public Attributes

- std::string **fromPort**
- std::string **toPort**

Public Attributes inherited from MultiCellEntity

- std::vector< Cell * > **ownerCells** = {}

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Protected Attributes

- unsigned int **fromId** = 0
- unsigned int **toId** = 0
- NodeEntity * **from** = nullptr
- NodeEntity * **to** = nullptr

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

5.55.1 Member Function Documentation

5.55.1.1 removeEntity()

```
void LinkEntity::removeEntity ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.55.1.2 setComponentEntity()

```
void LinkEntity::setComponentEntity (
            LinkComponent * c )  [inline], [override], [virtual]
```

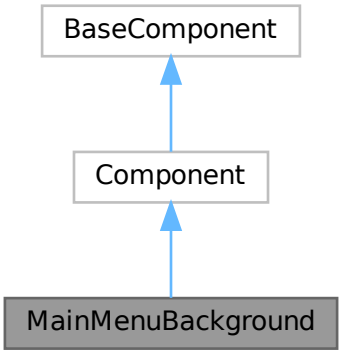
Reimplemented from Entity.

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

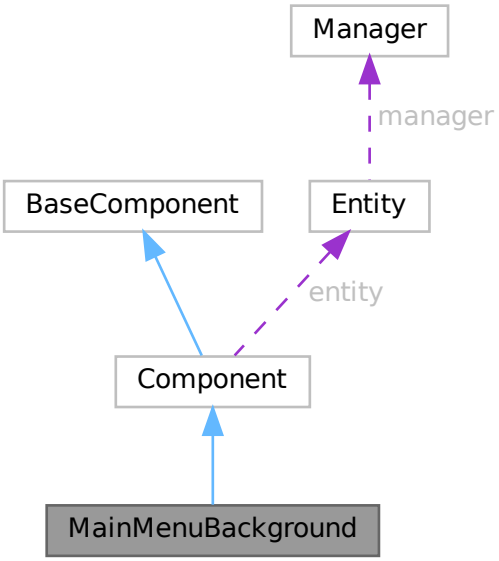
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSEntity.h

5.56 MainMenuBackground Class Reference

Inheritance diagram for MainMenuBackground:



Collaboration diagram for MainMenuBackground:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Additional Inherited Members

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.56.1 Member Function Documentation

5.56.1.1 GetComponentName()

```
std::string MainMenuBackground::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.56.1.2 init()

```
void MainMenuBackground::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.56.1.3 update()

```
void MainMenuBackground::update (
            float deltaTime )  [inline], [override], [virtual]
```

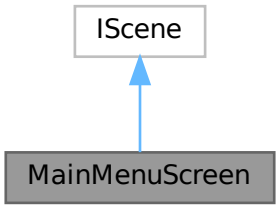
Reimplemented from BaseComponent.

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

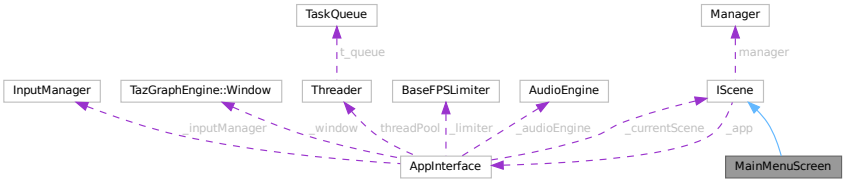
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/GECS/Scripts/MainMenuBackground.↩ h

5.57 MainMenuScreen Class Reference

Inheritance diagram for MainMenuScreen:



Collaboration diagram for MainMenuScreen:



Public Member Functions

- **MainMenuScreen** (TazGraphEngine::Window ∗window)
- virtual int getNextSceneIndex () const override
- virtual int getPreviousSceneIndex () const override
- virtual void build () override
- virtual void destroy () override
- virtual void onEntry () override
- virtual void onExit () override
- virtual void update (float deltaTime) override
- virtual void draw () override
- virtual void BeginRender () override
- virtual void updateUI () override
- virtual void EndRender () override
- void **renderBatch** (const std::vector< EmptyEntity ∗ > &entities)

Public Member Functions inherited from IScene

- int **getSceneIndex** () const
- void **setRunning** ()
- SceneState **getState** () const
- void **setParentApp** (AppInterface ∗app)
- AppInterface ∗ **getApp** () const
- void **setManager** (std::string m_managerName)

Additional Inherited Members

Protected Attributes inherited from IScene

- SceneState **_currentState** = SceneState::NONE
- AppInterface * **_app** = nullptr
- int **_sceneIndex** = -1
- std::unordered_map< std::string, Manager * > managers
- Manager * **manager** = nullptr
- std::string **managerName** = ""
- bool **_renderDebug** = false
- bool **_clusterLayout** = false

5.57.1 Member Function Documentation

5.57.1.1 BeginRender()

```
void MainMenuScreen::BeginRender ( )  [override], [virtual]
```

Implements IScene.

5.57.1.2 build()

```
void MainMenuScreen::build ( )  [override], [virtual]
```

Implements IScene.

5.57.1.3 destroy()

```
void MainMenuScreen::destroy ( )  [override], [virtual]
```

Implements IScene.

5.57.1.4 draw()

```
void MainMenuScreen::draw ( )  [override], [virtual]
```

Implements IScene.

5.57.1.5 EndRender()

```
void MainMenuScreen::EndRender ( )  [override], [virtual]
```

Implements IScene.

5.57.1.6 getNextSceneIndex()

```
int MainMenuScreen::getNextSceneIndex ( ) const  [override], [virtual]
```

Implements IScene.

5.57.1.7 getPreviousSceneIndex()

```
int MainMenuScreen::getPreviousSceneIndex ( ) const  [override], [virtual]
```

Implements IScene.

5.57.1.8 onEntry()

```
void MainMenuScreen::onEntry ( )  [override], [virtual]
```

Implements IScene.

5.57.1.9 onExit()

```
void MainMenuScreen::onExit ( )  [override], [virtual]
```

Implements IScene.

5.57.1.10 update()

```
void MainMenuScreen::update (
            float deltaTime )  [override], [virtual]
```

Implements IScene.

5.57.1.11 updateUI()

```
void MainMenuScreen::updateUI ( )  [override], [virtual]
```

Implements IScene.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/MainMenuScreen/MainMenuScreen.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/MainMenuScreen/MainMenuScreen.↩ cpp

5.58 Manager Class Reference

Public Types

- enum **groupLabels** : std::size_t {
  groupBackgroundLayer , **panelBackground** , **groupLinks_0** , **groupGroupLinks_0** ,
  groupGroupLinks_1 , **groupArrowHeads_0** , **groupNodes_0** , **groupGroupNodes_0** ,
  groupGroupNodes_1 , **groupColliders** , **groupEmpties** , **groupSphereEmpties** ,
  groupRenderSprites , **buttonLabels** }

Public Member Functions

- void **setThreader** (Threader &mthreader)
- void update (float deltaTime=1.0f)
- void **updateFully** (float deltaTime=1.0f)
- void **refresh** (ICamera *camera=nullptr)
- void **aboutTo_updateActiveEntities** ()
- void **updateActiveEntities** ()
- void **updateVisibleEntities** ()
- void **AddToGroup** (EmptyEntity *mEntity, Group mGroup)
- void **AddToGroup** (NodeEntity *mEntity, Group mGroup)
- void **AddLinkToGroup** (LinkEntity *mEntity, Group mGroup)
- const std::vector< std::unique_ptr< Entity > > & **getEntities** () const
- template<typename T >
  std::vector< T * > **getVisible** ()
- template<typename T >
  std::vector< T * > & **getVisibleGroup** (Group mGroup)
- template<typename T >
  std::vector< T * > & **getGroup** (Group mGroup)
- template<typename T , typename... TArgs>
  T & **addEntityNoId** (TArgs &&... mArgs)
- template<typename T , typename... TArgs>
  T & **addEntity** (TArgs &&... mArgs)
- void **resetEntityId** ()
- Entity * **getEntityFromId** (unsigned int mId)
- void **clearAllEntities** ()
- void **removeAllEntites** ()
- void **removeAllEntitiesFromGroup** (Group mGroup)
- void **removeAllEntitiesFromLinkGroup** (Group mGroup)
- std::vector< Entity * > **adjacentEntities** (Entity *mainEntity, Group group)
- std::string **getGroupName** (Group mGroup) const
- void **scanComponentNames** (const std::string &folderPath)
- void **setComponentNames** ()

Public Attributes

- std::vector< NodeEntity * > **movedNodes**
- std::mutex **movedNodesMutex**
- bool **arrowheadsEnabled** = false
- bool **last_arrowheadsEnabled** = false
- std::unordered_map< std::string, std::vector< std::string > > **componentNames**
- std::unique_ptr< Grid > **grid**
- const std::unordered_map< Group, std::string > groupNames

5.58.1 Member Function Documentation

5.58.1.1 update()

```
void Manager::update (
            float deltaTime = 1.0f )  [inline]
```

THREADER CHECK

CELL UPDATE

UPDATE LINK CELLS

UPDATE

FOR MAIN MENU

CELL UPDATE

5.58.2 Member Data Documentation

5.58.2.1 groupNames

```
const std::unordered_map<Group, std::string> Manager::groupNames
```

Initial value:

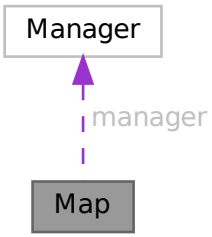
```
= {
        {groupBackgroundLayer, "groupBackgroundLayer" },
        {panelBackground, "panelBackground"},


        { groupLinks_0,"groupLinks_0" },
        {groupGroupLinks_0, "groupGroupLinks_0"},
        {groupGroupLinks_1, "groupGroupLinks_1"},

        {groupArrowHeads_0, "groupArrowHeads_0"},

        { groupNodes_0,"groupNodes_0" },
        { groupGroupNodes_0, "groupGroupNodes_0"},
        { groupGroupNodes_1, "groupGroupNodes_1"},

        { groupEmpties,"groupEmpties" },
        { groupSphereEmpties,"groupSphereEmpties" },

        { groupColliders,"groupColliders" },
        { groupRenderSprites,"groupRenderSprites" },



        { buttonLabels,"buttonLabels" },
    }
```

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSManager.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSManager.cpp

5.59 Map Class Reference

Collaboration diagram for Map:



Public Member Functions

- **Map** (Manager &m_manager, int ms, int ns)
- void **saveMapAsText** (const char *fileName)
- void **ProcessFile** (std::ifstream &mapFile, void(Map::*addNodeFunction)(Entity &, glm::vec3 mPosition), void(Map::*addLinkFunction)(Entity &))
- void **ProcessPythonFile** (std::ifstream &mapFile, void(Map::*addNodeFunction)(Entity &, glm::vec3 mPosition), void(Map::*addLinkFunction)(Entity &))
- void **loadTextMap** (const char *fileName)
- void **loadPythonMap** (const char *fileName)
- void **AddDefaultNode** (Entity &node, glm::vec3 mPosition)
- void **AddTreeNode** (Entity &node, glm::vec3 mPosition)
- void **AddDefaultLink** (Entity &node)
- void **AddTreeLink** (Entity &link)

Public Attributes

- Manager * **manager**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/Map/Map.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/Map/Map.cpp

5.60 MeshRenderer Struct Reference

Public Attributes

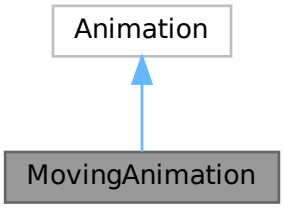
- size_t **meshIndices** = 0
- std::vector< InstanceData > **instances**
- GLuint **vao**
- GLuint **vbo**
- GLuint **ibo**

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

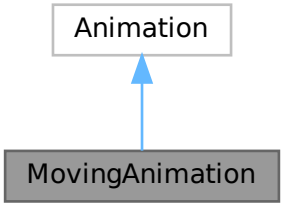
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.61 MovingAnimation Class Reference

Inheritance diagram for MovingAnimation:



Collaboration diagram for MovingAnimation:



Public Member Functions

- **MovingAnimation** (int ix, int iy, int f, float s, const std::string _type, int dx, int dy, int _reps=0)
- **MovingAnimation** (int ix, int iy, int f, float s, const animType _type, int dx, int dy, int _reps=0)
- **MovingAnimation** (int ix, int iy, int f, float s, const std::string _type, const std::vector< glm::vec2 > &_↩ positions, const std::vector< int > &_zIndices, const std::vector< int > &_rotations, int _reps=0)
- **MovingAnimation** (int ix, int iy, size_t f, float s, const animType _type, const std::vector< glm::vec2 > &_↩ positions, const std::vector< int > &_zIndices, const std::vector< int > &_rotations, int _reps=0)

Public Member Functions inherited from Animation

- **Animation** (int ix, int iy, size_t f, float s, const std::string _type, int _reps=0)
- **Animation** (int ix, int iy, size_t f, float s, const animType _type, int _reps=0)
- void **advanceFrame** (float deltaTime)
- void **resetFrameIndex** ()
- bool **hasFinished** ()

Public Attributes

- std::vector< glm::vec2 > **positions**
- std::vector< int > **zIndices**
- std::vector< int > **rotations**

Public Attributes inherited from Animation

- int **indexX** = 0
- int **indexY** = 0
- size_t **total_frames** = 0
- float **speed** = 1.0f
- animType **type** = animType::ANIMTYPE_NONE
- int **reps** = 0
- int **frame_times_played** = 0
- int **cur_frame_index** = 0
- float **cur_frame_index_f** = 0
- int **times_played** = 0
- int **flow_direction** = 1
- bool **finished** = false

Additional Inherited Members

Public Types inherited from Animation

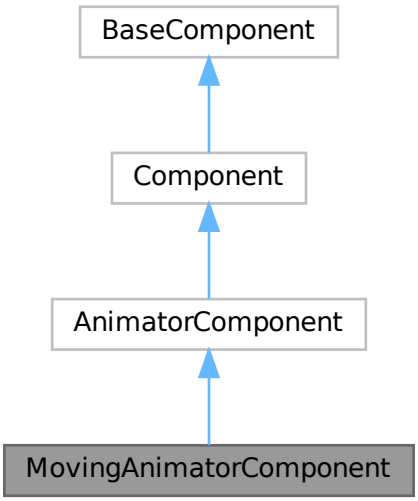
- enum **animType** { **ANIMTYPE_NONE** = 0 , **ANIMTYPE_PLAY_N_TIMES** = 1 , **ANIMTYPE_LOOPED** = 2 , **ANIMTYPE_BACK_FORTH** = 3 }

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

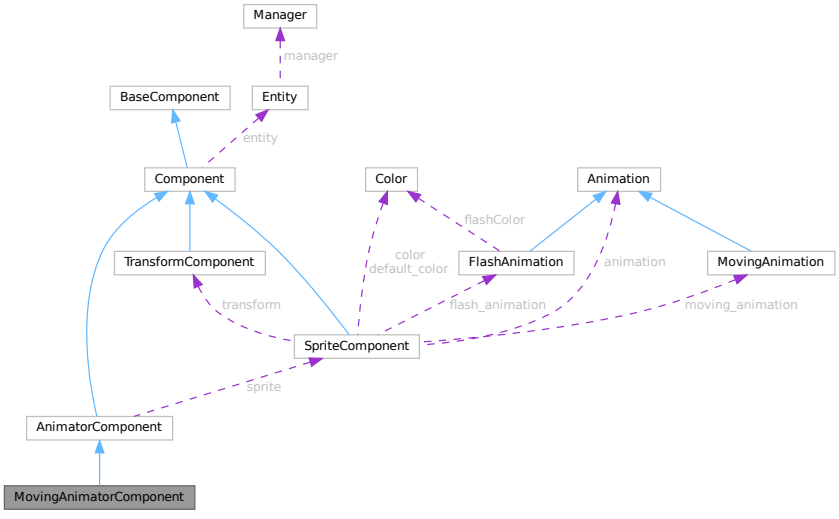
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/MovingAnimation.↩ h

5.62 MovingAnimatorComponent Class Reference

Inheritance diagram for MovingAnimatorComponent:



Collaboration diagram for MovingAnimatorComponent:



Public Member Functions

- **MovingAnimatorComponent** ()

  also we use MovingAnimator instead of simple Animator so that entities use less memory and we use it to entities that have triggers that change their animation

- **MovingAnimatorComponent** (std::string id)
- void init () override
- void update (float deltaTime) override
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- void **Play** (const char *animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()
- void **DestroyTex** ()

Public Member Functions inherited from AnimatorComponent

- **AnimatorComponent** (std::string id)
- void **Play** (std::string animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()
- void **DestroyTex** ()

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

Additional Inherited Members

Public Attributes inherited from AnimatorComponent

- SpriteComponent * **sprite** = nullptr
- std::string **textureid**
- std::string **animationName** = ""
- timestamp **resumeTime** = 0

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.62.1 Member Function Documentation

5.62.1.1 draw()

```
void MovingAnimatorComponent::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from AnimatorComponent.

5.62.1.2 init()

```
void MovingAnimatorComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from AnimatorComponent.

5.62.1.3 update()

```
void MovingAnimatorComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

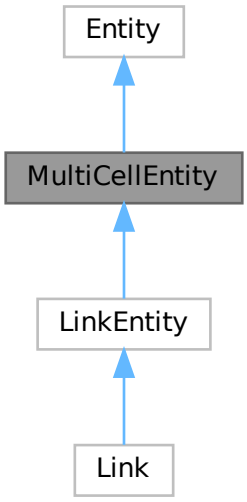
Reimplemented from AnimatorComponent.

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

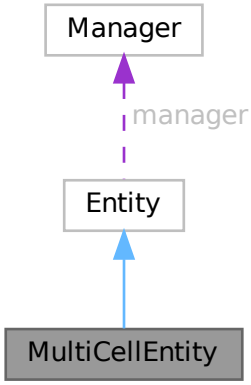
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/MovingAnimator↩ Component.h

5.63 MultiCellEntity Class Reference

Inheritance diagram for MultiCellEntity:



Collaboration diagram for MultiCellEntity:



Public Member Functions

- **MultiCellEntity** (Manager &mManager)
- void **setOwnerCells** (std::vector< Cell * > cells)
- Cell * **getOwnerCells** () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual void **update** (float deltaTime)
- virtual void **cellUpdate** ()
- virtual Cell * **getOwnerCell** () const
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- virtual void **destroy** ()
- bool **hasGroup** (Group mGroup)
- virtual void **addGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function

- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (Component *c)
- virtual void **setComponentEntity** (NodeComponent *c)
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual Entity * **getParentEntity** ()
- virtual void **setParentEntity** (Entity *pEntity)
- virtual void **imgui_print** ()
- virtual void **imgui_display** ()
- virtual void **removeEntity** ()

Public Attributes

- std::vector< Cell * > **ownerCells** = {}

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Additional Inherited Members

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/CellEntity.h

5.64 Music Class Reference

Public Member Functions

- void **play** (int loops=1)

Static Public Member Functions

- static void **pause** ()
- static void **stop** ()
- static void **resume** ()

Friends

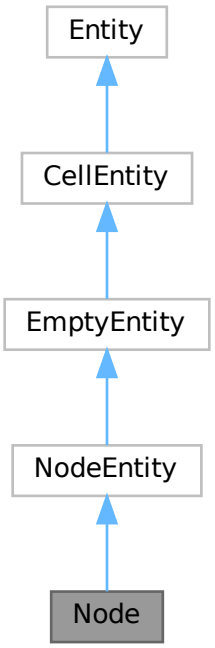
- class **AudioEngine**

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

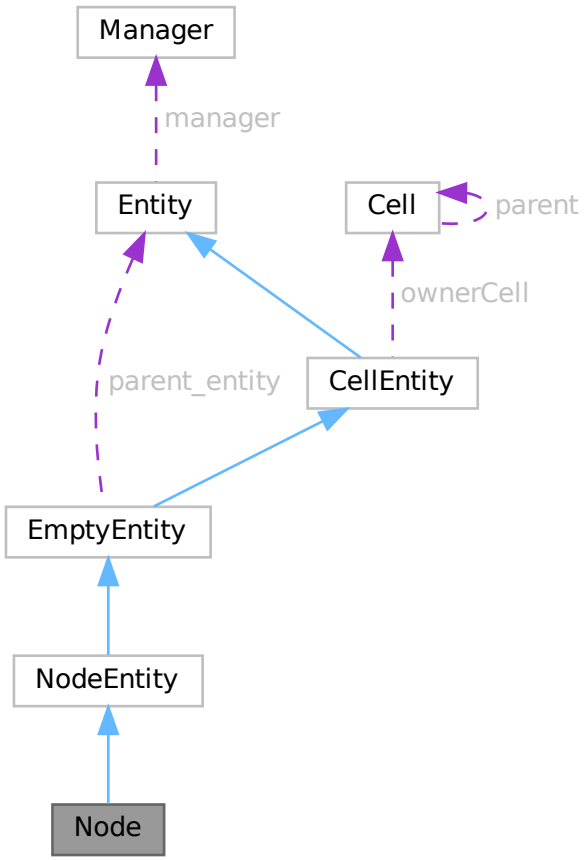
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/AudioEngine/AudioEngine.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/AudioEngine/AudioEngine.cpp

5.65 Node Class Reference

Inheritance diagram for Node:



Collaboration diagram for Node:



Public Member Functions

- **Node** (Manager &mManager)
- void addGroup (Group mGroup) override
- void update (float deltaTime)
- void updatePorts (float deltaTime) override
- void cellUpdate () override
- void addMessage (std::string mMessage) override
- void imgui_print () override
- void imgui_display () override
- void destroy ()
- void addPorts ()
- void removePorts ()

Public Member Functions inherited from NodeEntity

- **NodeEntity** (Manager &mManager)
- void setComponentEntity (NodeComponent *c) override

- void removeEntity () override
- void **addInLink** (LinkEntity *link)
- void **addOutLink** (LinkEntity *link)
- const std::vector< LinkEntity * > & **getInLinks** () const
- const std::vector< LinkEntity * > & **getOutLinks** () const

Public Member Functions inherited from EmptyEntity

- **EmptyEntity** (Manager &mManager)
- void setComponentEntity (Component *c) override
- Entity * getParentEntity () override
- void setParentEntity (Entity *pEntity) override
- void **removeFromCell** ()

Public Member Functions inherited from CellEntity

- **CellEntity** (Manager &mManager)
- void **setOwnerCell** (Cell *cell)
- Cell * getOwnerCell () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- bool **hasGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()

Additional Inherited Members

Public Attributes inherited from CellEntity

- Cell * **ownerCell** = nullptr

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

Protected Attributes inherited from NodeEntity

- std::vector< LinkEntity * > **inLinks**
- std::vector< LinkEntity * > **outLinks**

Protected Attributes inherited from EmptyEntity

- Entity * **parent_entity** = nullptr

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

5.65.1 Member Function Documentation

5.65.1.1 addGroup()

```
void Node::addGroup (
            Group mGroup )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.65.1.2 addMessage()

```
void Node::addMessage (
            std::string mMessage )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.65.1.3 addPorts()

```
void Node::addPorts ( )  [inline], [virtual]
```

Reimplemented from NodeEntity.

5.65.1.4 cellUpdate()

```
void Node::cellUpdate ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.65.1.5 destroy()

```
void Node::destroy ( )  [inline], [virtual]
```

Reimplemented from Entity.

5.65.1.6 imgui_display()

```
void Node::imgui_display ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.65.1.7 imgui_print()

```
void Node::imgui_print ( )  [inline], [override], [virtual]
```

Reimplemented from Entity.

5.65.1.8 removePorts()

```
void Node::removePorts ( )  [inline], [virtual]
```

Reimplemented from NodeEntity.

5.65.1.9 update()

```
void Node::update (
            float deltaTime )  [inline], [virtual]
```

Reimplemented from Entity.

5.65.1.10 updatePorts()

```
void Node::updatePorts (
            float deltaTime )  [inline], [override], [virtual]
```

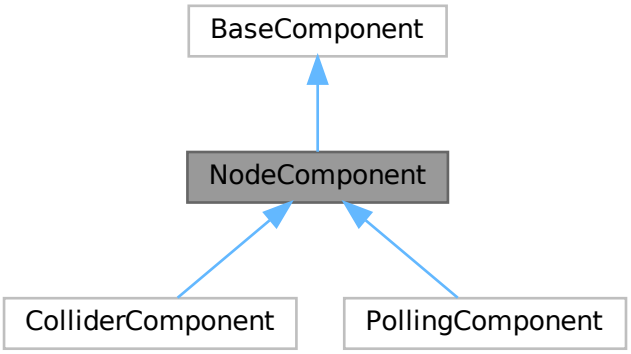
Reimplemented from NodeEntity.

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

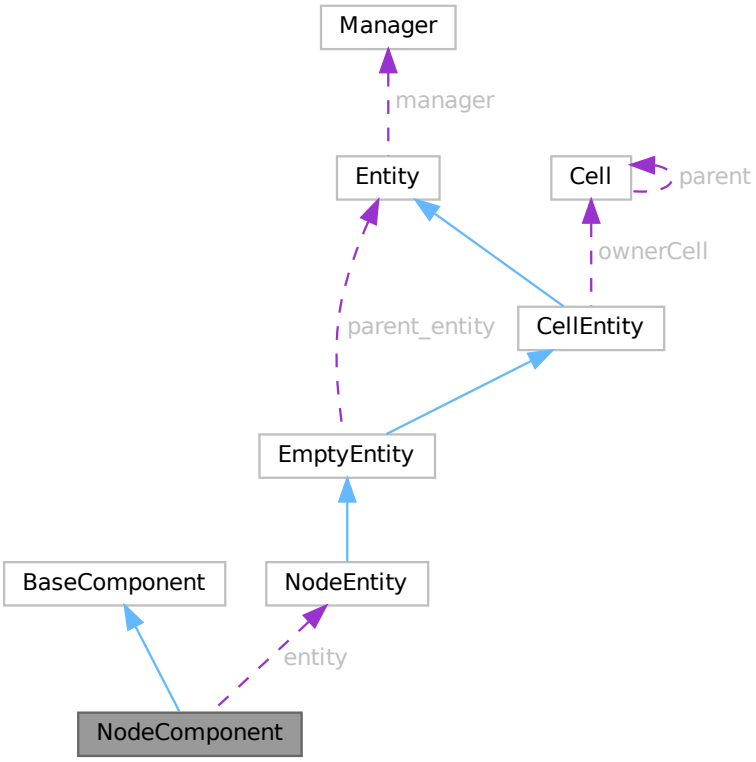
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSEntityTypes.h

5.66 NodeComponent Class Reference

Inheritance diagram for NodeComponent:



Collaboration diagram for NodeComponent:



Public Attributes

- NodeEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

Additional Inherited Members

Public Member Functions inherited from BaseComponent

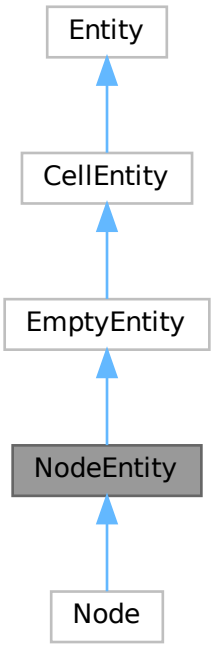
- virtual void **init** ()
- virtual void **update** (float deltaTime)
- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

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

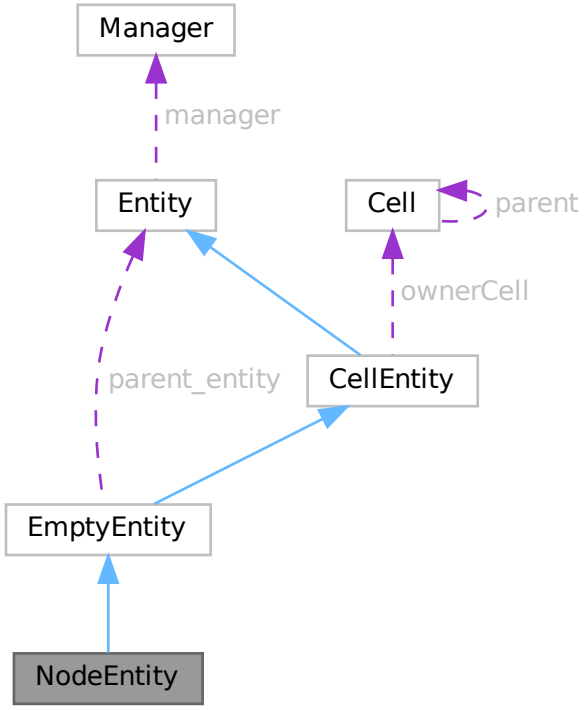
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECS.h

5.67 NodeEntity Class Reference

Inheritance diagram for NodeEntity:



Collaboration diagram for NodeEntity:



Public Member Functions

- **NodeEntity** (Manager &mManager)
- void setComponentEntity (NodeComponent *c) override
- void removeEntity () override
- void **addInLink** (LinkEntity *link)
- void **addOutLink** (LinkEntity *link)
- const std::vector< LinkEntity * > & **getInLinks** () const
- const std::vector< LinkEntity * > & **getOutLinks** () const
- virtual void **addPorts** ()
- virtual void **removePorts** ()
- virtual void **updatePorts** (float deltaTime)

Public Member Functions inherited from EmptyEntity

- **EmptyEntity** (Manager &mManager)
- void setComponentEntity (Component *c) override
- Entity * getParentEntity () override
- void setParentEntity (Entity *pEntity) override
- void **removeFromCell** ()

Public Member Functions inherited from CellEntity

- **CellEntity** (Manager &mManager)
- void **setOwnerCell** (Cell *cell)
- Cell * getOwnerCell () const

Public Member Functions inherited from Entity

- void **setId** (unsigned int m_id)
- unsigned int **getId** ()
- void **hide** ()
- void **reveal** ()
- bool **isHidden** ()
- **Entity** (Manager &mManager)
- virtual void **update** (float deltaTime)
- virtual void **cellUpdate** ()
- void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- bool **isActive** ()
- virtual void **destroy** ()
- bool **hasGroup** (Group mGroup)
- virtual void **addGroup** (Group mGroup)
- void **removeGroup** (Group mGroup)
- template<typename T >
  bool **hasComponent** () const
- template<typename T , typename... TArgs>
  T & **addComponent** (TArgs &&... mArgs)

  have addScript function
- template<typename T >
  void **removeComponent** ()
- virtual void **setComponentEntity** (LinkComponent *c)
- template<typename T >
  T & **GetComponent** () const
- bool **hasComponentByName** (const std::string &componentName)
- Manager * **getManager** ()
- virtual void **addMessage** (std::string mMessage)
- virtual void **imgui_print** ()
- virtual void **imgui_display** ()

Protected Attributes

- std::vector< LinkEntity * > **inLinks**
- std::vector< LinkEntity * > **outLinks**

Protected Attributes inherited from EmptyEntity

- Entity * **parent_entity** = nullptr

Protected Attributes inherited from Entity

- std::optional< ComponentArray > **nodeComponentArray**
- std::optional< ComponentBitSet > **nodeComponentBitSet**
- Manager & **manager**

Additional Inherited Members

Public Attributes inherited from CellEntity

- Cell * **ownerCell** = nullptr

Public Attributes inherited from Entity

- std::unordered_map< std::string, EmptyEntity * > **children**
- std::vector< std::unique_ptr< BaseComponent > > **components**

5.67.1 Member Function Documentation

5.67.1.1 removeEntity()

```
void NodeEntity::removeEntity ( )  [inline], [override], [virtual]
```

Reimplemented from EmptyEntity.

5.67.1.2 setComponentEntity()

```
void NodeEntity::setComponentEntity (
            NodeComponent * c )  [inline], [override], [virtual]
```

Reimplemented from Entity.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Core/GECSEntity.h

5.68 NumericStringCompare Struct Reference

Public Member Functions

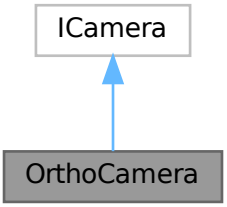
- bool **operator()** (const std::string &a, const std::string &b) const

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

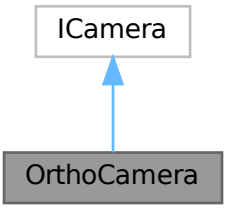
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/JsonParser/JsonParser.h

5.69 OrthoCamera Class Reference

Inheritance diagram for OrthoCamera:



Collaboration diagram for OrthoCamera:



Public Member Functions

- void init () override
- void update () override
- glm::vec2 convertScreenToWorld (glm::vec2 screenCoords) const override
- void setPosition (const glm::vec3 newPosition) override
- void setPosition_X (const float newPosition) override
- void setPosition_Y (const float newPosition) override
- void setPosition_Z (const float newPosition) override
- void setScale (float newScale) override
- glm::vec3 getPosition () const override
- float getScale () const override
- glm::mat4 getCameraMatrix () const override
- glm::ivec2 getCameraDimensions () const override
- SDL_FRect getCameraRect () const override
- void **setCameraMatrix** (glm::mat4 newMatrix)
- bool isPointInCameraView (const glm::vec4 point, float margin)
- bool hasChanged () override
- void makeCameraDirty () override
- void refreshCamera () override

5.69.1 Member Function Documentation

5.69.1.1 convertScreenToWorld()

```
glm::vec2 OrthoCamera::convertScreenToWorld (
            glm::vec2 screenCoords ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.2 getCameraDimensions()

```
glm::ivec2 OrthoCamera::getCameraDimensions ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.3 getCameraMatrix()

```
glm::mat4 OrthoCamera::getCameraMatrix ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.4 getCameraRect()

```
SDL_FRect OrthoCamera::getCameraRect ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.5 getPosition()

```
glm::vec3 OrthoCamera::getPosition ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.6 getScale()

```
float OrthoCamera::getScale ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.7 hasChanged()

```
bool OrthoCamera::hasChanged ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.8 init()

```
void OrthoCamera::init ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.9 isPointInCameraView()

```
bool OrthoCamera::isPointInCameraView (
            const glm::vec4 point,
            float margin )  [inline], [virtual]
```

Implements ICamera.

5.69.1.10 makeCameraDirty()

```
void OrthoCamera::makeCameraDirty ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.11 refreshCamera()

```
void OrthoCamera::refreshCamera ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.12 setPosition()

```
void OrthoCamera::setPosition (
            const glm::vec3 newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.13 setPosition_X()

```
void OrthoCamera::setPosition_X (
            const float newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.14 setPosition_Y()

```
void OrthoCamera::setPosition_Y (
            const float newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.15 setPosition_Z()

```
void OrthoCamera::setPosition_Z (
            const float newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.16 setScale()

```
void OrthoCamera::setScale (
            float newScale )  [inline], [override], [virtual]
```

Implements ICamera.

5.69.1.17 update()

```
void OrthoCamera::update ( )  [inline], [override], [virtual]
```

Implements ICamera.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Camera2.5D/OrthoCamera.h

5.70 PairHash Struct Reference

Public Member Functions

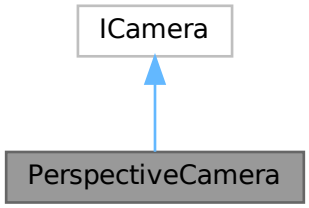
- template<class T1 , class T2 >
  std::size_t **operator()** (const std::pair< T1, T2 > &p) const

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

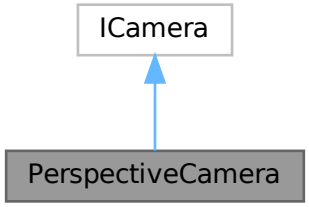
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/Src/AssetManager/AssetManager.h

5.71 PerspectiveCamera Class Reference

Inheritance diagram for PerspectiveCamera:



Collaboration diagram for PerspectiveCamera:



Public Member Functions

- **PerspectiveCamera** (glm::vec3 eye_pos, glm::vec3 aim_pos)
- void init () override
- void update () override
- void **updateCameraOrientation** ()
- void **setOrientation** (glm::vec3 eye, glm::vec3 target, glm::vec3 up)
- glm::vec2 convertScreenToWorld (glm::vec2 screenCoords) const override
- void setPosition (const glm::vec3 newPosition) override
- void setPosition_X (const float newPosition) override
- void setPosition_Y (const float newPosition) override
- void setPosition_Z (const float newPosition) override
- void **movePosition_Hor** (const float step)
- void **movePosition_Vert** (const float step)
- void **movePosition_Forward** (const float step)
- void **setAimPos** (const glm::vec3 newAimPos)
- void **moveAimPos** (glm::vec3 startingAimPos, const glm::vec2 distance)
- glm::vec3 **getEulerAnglesFromDirection** (glm::vec3 direction)

- float **getZFar** ()
- glm::vec3 **getAimPos** ()
- void setScale (float newScale) override
- glm::vec3 getPosition () const override
- float getScale () const override
- glm::mat4 getCameraMatrix () const override
- glm::ivec2 getCameraDimensions () const override
- SDL_FRect getCameraRect () const override
- void **setCameraMatrix** (glm::mat4 newMatrix)
- void **resetCameraPosition** ()
- float **getMinScale** ()
- float **getMaxScale** ()
- bool isPointInCameraView (const glm::vec4 point, float margin)
- bool hasChanged () override
- void makeCameraDirty () override
- void refreshCamera () override
- glm::vec3 **castRayAt** (const glm::vec2 &screenPos)
- glm::vec3 **getPointOnRayAtZ** (const glm::vec3 &rayOrigin, const glm::vec3 &rayDirection, float desiredZ)

Public Attributes

- glm::vec3 **eyePos** { 0,0,0 }
- glm::vec3 **aimPos** { 0,0,0 }
- glm::vec3 **upDir** {0,-1,0}
- float **zFar** = 1000000.0f
- ViewMode **currentViewMode** = ViewMode::Y_UP

5.71.1 Member Function Documentation

5.71.1.1 convertScreenToWorld()

```
glm::vec2 PerspectiveCamera::convertScreenToWorld (
            glm::vec2 screenCoords ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.2 getCameraDimensions()

```
glm::ivec2 PerspectiveCamera::getCameraDimensions ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.3 getCameraMatrix()

```
glm::mat4 PerspectiveCamera::getCameraMatrix ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.4 getCameraRect()

```
SDL_FRect PerspectiveCamera::getCameraRect ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.5 getPosition()

```
glm::vec3 PerspectiveCamera::getPosition ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.6 getScale()

```
float PerspectiveCamera::getScale ( ) const  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.7 hasChanged()

```
bool PerspectiveCamera::hasChanged ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.8 init()

```
void PerspectiveCamera::init ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.9 isPointInCameraView()

```
bool PerspectiveCamera::isPointInCameraView (
            const glm::vec4 point,
            float margin )  [inline], [virtual]
```

Implements ICamera.

5.71.1.10 makeCameraDirty()

```
void PerspectiveCamera::makeCameraDirty ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.11 refreshCamera()

```
void PerspectiveCamera::refreshCamera ( )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.12 setPosition()

```
void PerspectiveCamera::setPosition (
            const glm::vec3 newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.13 setPosition_X()

```
void PerspectiveCamera::setPosition_X (
            const float newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.14 setPosition_Y()

```
void PerspectiveCamera::setPosition_Y (
            const float newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.15 setPosition_Z()

```
void PerspectiveCamera::setPosition_Z (
            const float newPosition )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.16 setScale()

```
void PerspectiveCamera::setScale (
            float newScale )  [inline], [override], [virtual]
```

Implements ICamera.

5.71.1.17 update()

```
void PerspectiveCamera::update ( )  [inline], [override], [virtual]
```

Implements ICamera.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Camera2.5D/PerspectiveCamera.h

5.72 PlaneColorRenderer Class Reference

Public Member Functions

- void **init** ()
- void **begin** ()
- void **end** ()
- void **initColorTriangleBatch** (size_t mSize)
- void **initColorQuadBatch** (size_t mSize)
- void **initColorBoxBatch** (size_t mSize)
- void **initColorSphereBatch** (size_t mSize)
- void **initBatchSize** ()
- void **drawTriangle** (size_t v_index, const glm::vec3 &depth, const glm::vec3 &cpuRotation, const Color &color)
- void **draw** (size_t v_index, const glm::vec2 &rectSize, const glm::vec3 &bodyCenter, const glm::vec3 &mRotation, const Color &color)

  draws are needed to convert the pos and size to vertices
- void **drawBox** (size_t v_index, const glm::vec3 &boxSize, const glm::vec3 &bodyCenter, const glm::vec3 &mRotation, const Color &color)
- void **drawSphere** (size_t v_index, const glm::vec3 &sphereSize, const glm::vec3 &bodyCenter, const glm::vec3 &mRotation, const Color &color)
- void **renderBatch** (GLSLProgram *glsl_program)
- void **dispose** ()

Public Attributes

- std::vector< Position > sphereVertices
- std::vector< GLuint > sphereIndices

5.72.1 Member Data Documentation

5.72.1.1 sphereIndices

```
std::vector<GLuint> PlaneColorRenderer::sphereIndices
```

Initial value:

```
= {

    }
```

5.72.1.2 sphereVertices

```
std::vector<Position> PlaneColorRenderer::sphereVertices
```

Initial value:

```
= {

    }
```

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/PlaneColorRenderer/PlaneColorRenderer.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/PlaneColorRenderer/PlaneColorRenderer.cpp

5.73 PlaneModelRenderer Class Reference

Public Member Functions

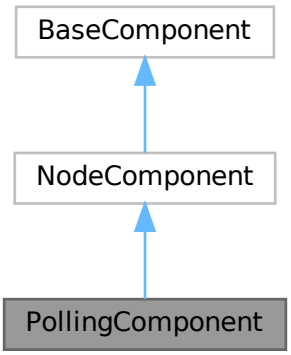
- void **init** ()
- void **begin** ()
- void **end** ()
- void **initTextureQuadBatch** (size_t mSize)
- void **initBatchSize** ()
- void **drawTriangle** (size_t v_index, const glm::vec3 &triangleOffset, const glm::vec3 &mRotation, const glm::vec2 &uv1, const glm::vec2 &uv2, const glm::vec2 &uv3, GLuint texture)
- void **draw** (size_t v_index, const glm::vec2 &rectSize, const glm::vec3 &bodyCenter, const glm::vec3 &mRotation, const glm::vec4 &uvRect, GLuint texture)
- void **renderBatch** (GLSLProgram *glsl_program)
- void **dispose** ()

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

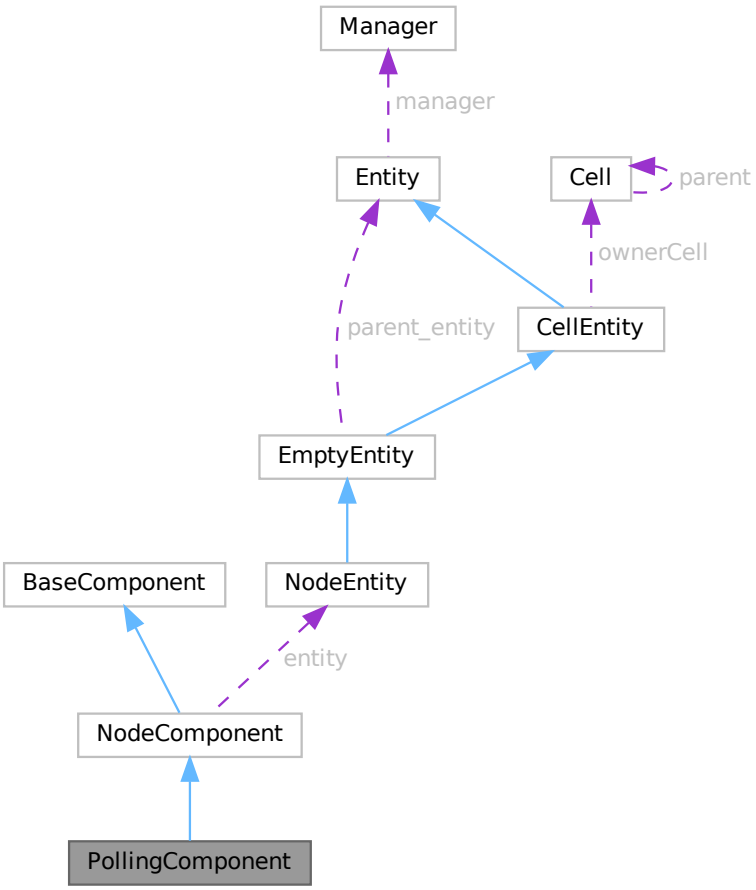
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/PlaneModelRenderer/PlaneModelRenderer.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/PlaneModelRenderer/PlaneModelRenderer.cpp

5.74 PollingComponent Class Reference

Inheritance diagram for PollingComponent:



Collaboration diagram for PollingComponent:



Public Member Functions

- void **StartPolling** (const std::string &file, float delayInSeconds)
- void update (float deltaTime) override
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **init** ()
- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Additional Inherited Members

Public Attributes inherited from NodeComponent

- NodeEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.74.1 Member Function Documentation

5.74.1.1 GetComponentName()

```
std::string PollingComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.74.1.2 update()

```
void PollingComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

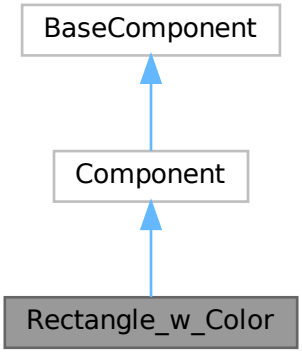
Reimplemented from BaseComponent.

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

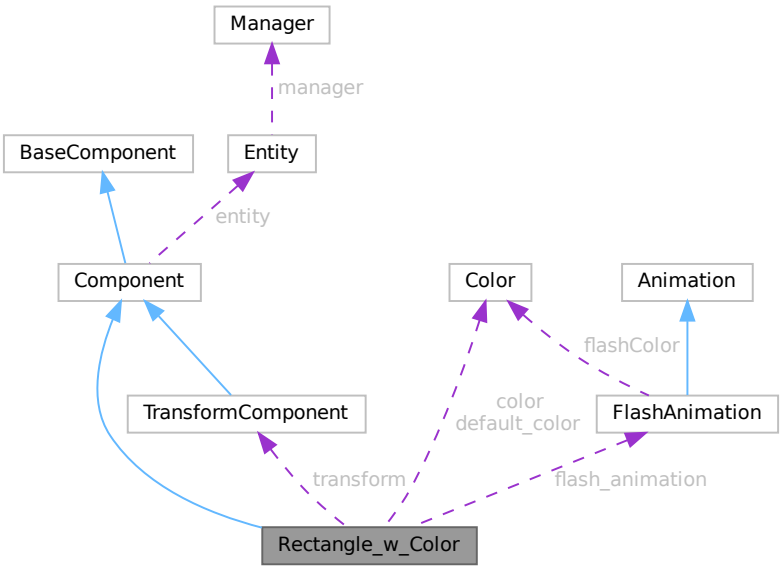
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Node/Util/Polling↩ Component.h

5.75 Rectangle_w_Color Class Reference

Inheritance diagram for Rectangle_w_Color:



Collaboration diagram for Rectangle_w_Color:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void **setColor** (Color clr)
- void **SetFlashAnimation** (int idX, int idY, size_t fr, float sp, const Animation::animType type, const std::vector< float > &flashTimes, Color flashC, int reps=0)
- void **setFlashFrame** ()
- std::string GetComponentName () override
- void showGUI () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)

Public Attributes

- Color **default_color** = { 255, 255, 255, 255 }
- Color **color** = { 255, 255, 255, 255 }
- TransformComponent * **transform** = nullptr
- FlashAnimation **flash_animation**

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.75.1 Member Function Documentation

5.75.1.1 draw()

```
void Rectangle_w_Color::draw (
            size_t v_index,
            PlaneColorRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.75.1.2 GetComponentName()

```
std::string Rectangle_w_Color::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.75.1.3 init()

```
void Rectangle_w_Color::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.75.1.4 showGUI()

```
void Rectangle_w_Color::showGUI ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.75.1.5 update()

```
void Rectangle_w_Color::update (
            float deltaTime )  [inline], [override], [virtual]
```

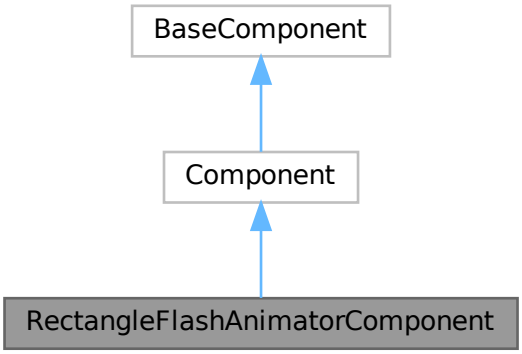
Reimplemented from BaseComponent.

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

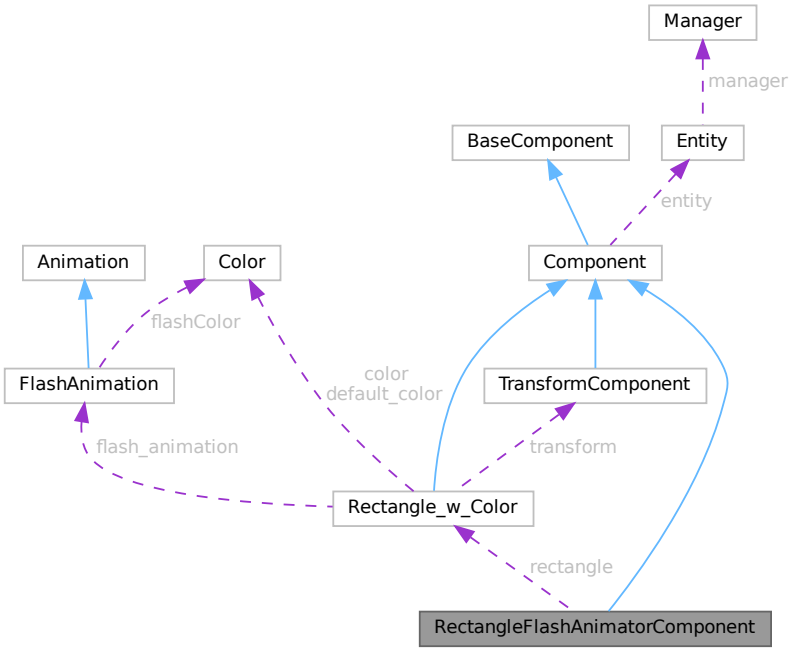
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Basic/Rectangle_w_Color.h

5.76 RectangleFlashAnimatorComponent Class Reference

Inheritance diagram for RectangleFlashAnimatorComponent:



Collaboration diagram for RectangleFlashAnimatorComponent:



Public Member Functions

- void init () override

- void update (float deltaTime) override
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- void **Play** (const std::string &animName, int reps=0)
- void **resetAnimation** ()
- std::string **getPlayName** ()

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual std::string **GetComponentName** ()
- virtual void **showGUI** ()

Public Attributes

- Rectangle_w_Color * **rectangle** = nullptr

  also we use MovingAnimator instead of simple Animator so that entities use less memory and we use it to entities that have triggers that change their animation
- std::string **animationName** = ""
- timestamp **resumeTime** = 0

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.76.1 Member Function Documentation

5.76.1.1 draw()

```
void RectangleFlashAnimatorComponent::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.76.1.2 init()

```
void RectangleFlashAnimatorComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.76.1.3 update()

```
void RectangleFlashAnimatorComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Animators/Rectangle↩ Animators/RectangleFlashAnimatorComponent.h

5.77 RenderBatch Class Reference

Public Member Functions

- **RenderBatch** (GLuint Offset, GLuint NumIndices, glm::vec3 CenterPos, GLuint Texture)

Public Attributes

- GLuint **offset**
- GLuint **numIndices**
- glm::vec3 **centerPos** = glm::vec3(0)
- GLuint **texture**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/PlaneModelRenderer/Plane↩ ModelRenderer.h

5.78 RenderLineBatch Class Reference

Public Member Functions

- **RenderLineBatch** (GLuint Offset, GLuint NumIndices)

Public Attributes

- GLuint **offset** = 0
- GLuint **numIndices** = 0

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LineRenderer/Line↩ Renderer.h

5.79 ResourceManager Class Reference

Public Member Functions

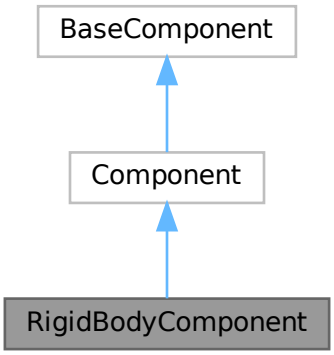
- void **setupShader** (GLSLProgram &shaderProgram, ICamera &camera)
- void **addGLSLProgram** (std::string programName)
- GLSLProgram * **getGLSLProgram** (std::string id)
- void **disposeGLSLPrograms** ()

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

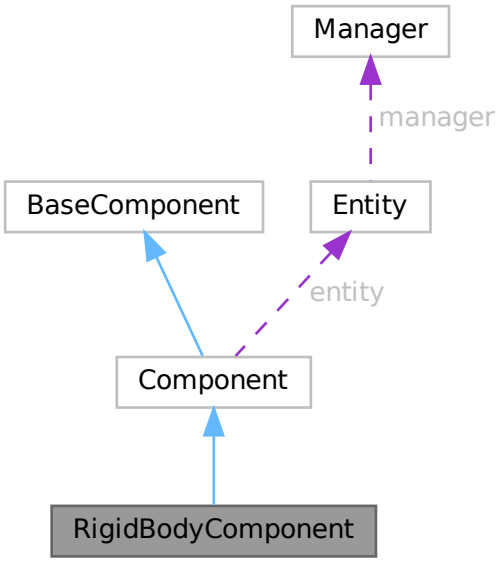
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/ResourceManager/ResourceManager.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/ResourceManager/ResourceManager.cpp

5.80 RigidBodyComponent Class Reference

Inheritance diagram for RigidBodyComponent:



Collaboration diagram for RigidBodyComponent:



Public Member Functions

- **RigidBodyComponent** (float acc, float maxg)
- void init () override
- void update (float deltaTime) override
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- float **GravityForce** = 1.0f
- float **accelGravity** = 0.045f
- float **maxGravity** = 3.f
- bool **onGround** = false
- bool **justjumped** = false

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.80.1 Member Function Documentation

5.80.1.1 GetComponentName()

```
std::string RigidBodyComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.80.1.2 init()

```
void RigidBodyComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.80.1.3 update()

```
void RigidBodyComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

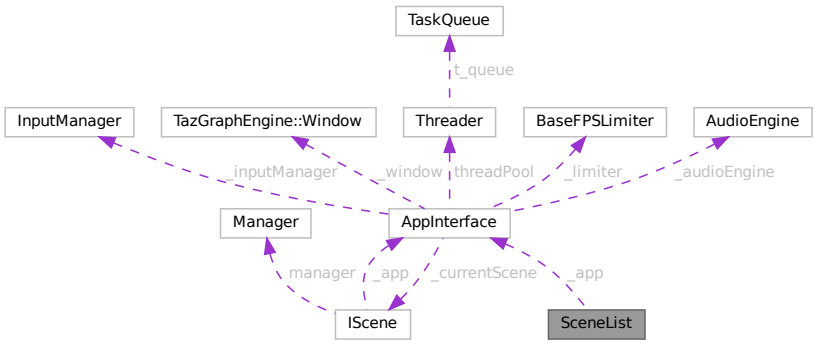
Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Util/RigidBodyComponent.h

5.81 SceneList Class Reference

Collaboration diagram for SceneList:



Public Member Functions

- **SceneList** (AppInterface *app)
- IScene * **moveNext** ()
- IScene * **movePrevious** ()
- void **setScene** (int nextScene)
- void **addScene** (IScene *newScene)
- void **addScene** (std::string managerName, IScene *newScene)
- void **destroy** ()
- IScene * **getCurrent** ()

Protected Attributes

- AppInterface * **_app** = nullptr
- std::vector< IScene * > **_scenes**
- int **_currentSceneIndex** = -1

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GraphScreen/SceneList.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GraphScreen/SceneList.cpp

5.82 SoundEffect Class Reference

Public Member Functions

- void **play** (int loops=0)

Friends

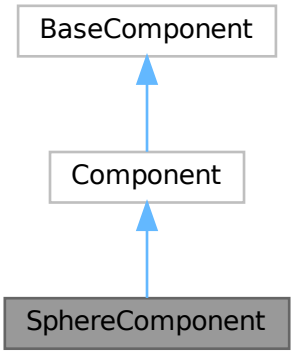
- class **AudioEngine**

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

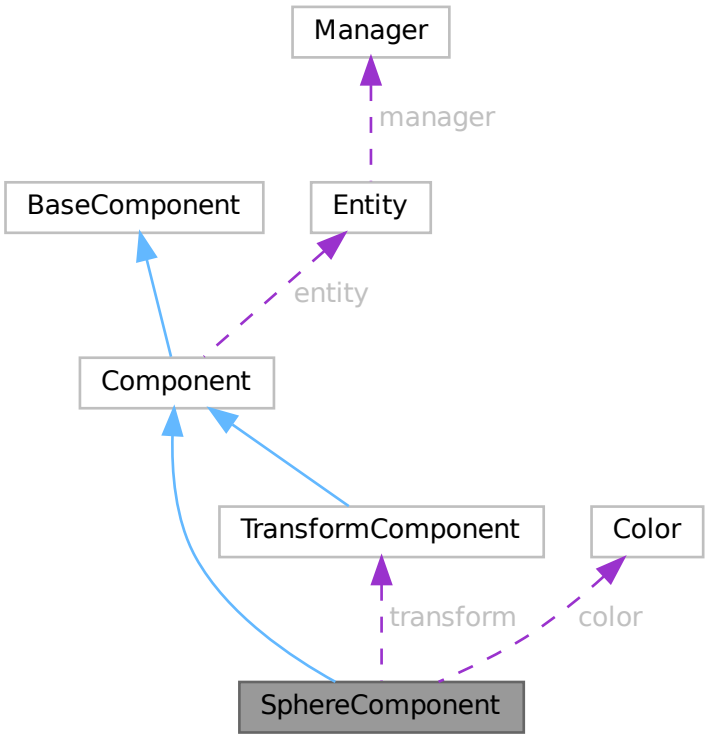
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/AudioEngine/AudioEngine.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/AudioEngine/AudioEngine.cpp

5.83 SphereComponent Class Reference

Inheritance diagram for SphereComponent:



Collaboration diagram for SphereComponent:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- void draw (size_t v_index, LightRenderer &batch, TazGraphEngine::Window &window)
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- Color **color** = { 255, 255, 255, 255 }
- TransformComponent * **transform** = nullptr

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.83.1 Member Function Documentation

5.83.1.1 draw() [1/2]

```
void SphereComponent::draw (
            size_t v_index,
            LightRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.83.1.2 draw() [2/2]

```
void SphereComponent::draw (
            size_t v_index,
            PlaneColorRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.83.1.3 GetComponentName()

```
std::string SphereComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.83.1.4 init()

```
void SphereComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.83.1.5 update()

```
void SphereComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

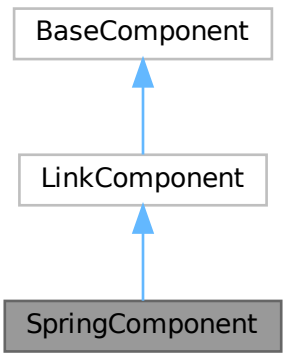
Reimplemented from BaseComponent.

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

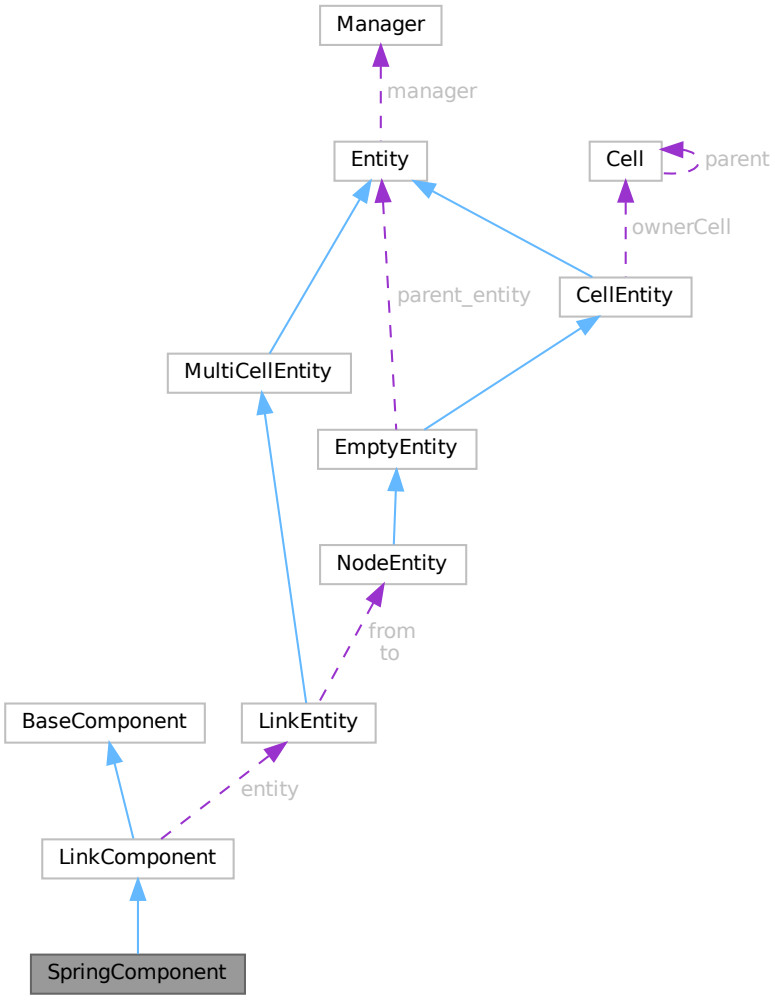
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Basic/SphereComponent.h

5.84 SpringComponent Class Reference

Inheritance diagram for SpringComponent:



Collaboration diagram for SpringComponent:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, LineRenderer &batch, TazGraphEngine::Window &window)
- std::string GetComponentName () override
- void showGUI () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)

Additional Inherited Members

Public Attributes inherited from LinkComponent

- LinkEntity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.84.1 Member Function Documentation

5.84.1.1 draw()

```
void SpringComponent::draw (
            size_t v_index,
            LineRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.84.1.2 GetComponentName()

```
std::string SpringComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.84.1.3 init()

```
void SpringComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.84.1.4 showGUI()

```
void SpringComponent::showGUI ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.84.1.5 update()

```
void SpringComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

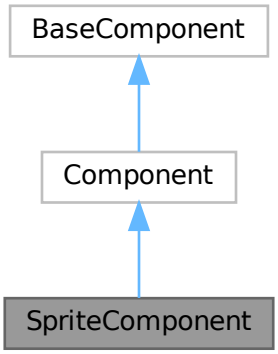
Reimplemented from BaseComponent.

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

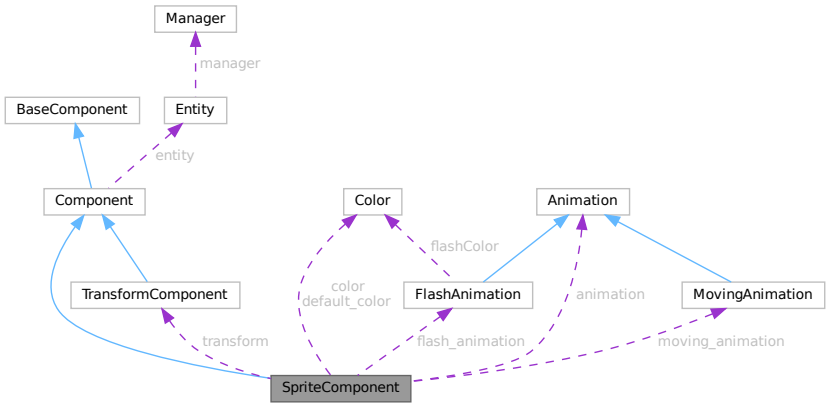
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Link/Basic/Spring↩ Component.h

5.85 SpriteComponent Class Reference

Inheritance diagram for SpriteComponent:



Collaboration diagram for SpriteComponent:



Public Member Functions

- **SpriteComponent** (std::string id)
- **SpriteComponent** (Color clr)
- **SpriteComponent** (std::string id, bool isMainMenu)
- void **setTex** (std::string id)
- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- void **SetAnimation** (int idX, int idY, size_t fr, float sp, const Animation::animType type, int reps=0)

- void **SetMovingAnimation** (int idX, int idY, size_t fr, float sp, const Animation::animType type, const std::vector< glm::vec2 > &_positions, const std::vector< int > &_zIndices, const std::vector< int > &_rotations, int reps=0)
- void **SetFlashAnimation** (int idX, int idY, size_t fr, float sp, const Animation::animType type, const std::vector< float > &flashTimes, Color flashC, int reps=0)
- void **setCurrFrame** ()
- void **setMoveFrame** ()
- void **setSpecificMoveFrame** ()
- void **setFlashFrame** ()
- void **DestroyTex** ()
- void **DestroyGlTex** ()
- std::string GetComponentName () override
- void showGUI () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)

Public Attributes

- std::string **texture_name** = ""
- Color **default_color** = { 255, 255, 255, 255 }
- Color **color** = { 255, 255, 255, 255 }
- TransformComponent * **transform** = nullptr
- SDL_FRect **srcRect** = {0,0,0,0}
- Animation **animation**
- MovingAnimation **moving_animation**
- FlashAnimation **flash_animation**
- SDL_RendererFlip **spriteFlip** = SDL_FLIP_NONE

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.85.1 Member Function Documentation

5.85.1.1 draw()

```
void SpriteComponent::draw (
            size_t v_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.85.1.2 GetComponentName()

```
std::string SpriteComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.85.1.3 init()

```
void SpriteComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.85.1.4 showGUI()

```
void SpriteComponent::showGUI ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.85.1.5 update()

```
void SpriteComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

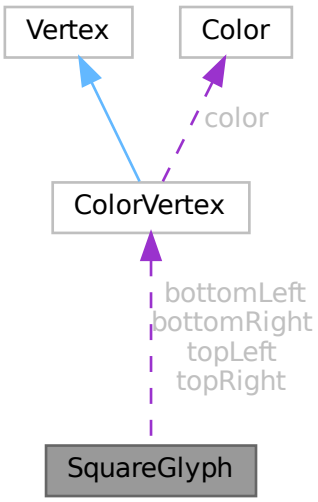
Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Basic/SpriteComponent.h

5.86 SquareGlyph Class Reference

Collaboration diagram for SquareGlyph:



Public Member Functions

- **SquareGlyph** (const glm::vec4 &destRect, const Color &color, float angle, float mdepth)

Public Attributes

- ColorVertex **topLeft**
- ColorVertex **bottomLeft**
- ColorVertex **bottomRight**
- ColorVertex **topRight**

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Renderers/LineRenderer/Line↩ Renderer.h

5.87 TaskQueue Struct Reference

Public Member Functions

- void **addTask** (std::function< void()> &&callback)
- bool **getTask** (std::function< void()> &task)
- void **waitUntilDone** () const
- void **completeTask** ()

Public Attributes

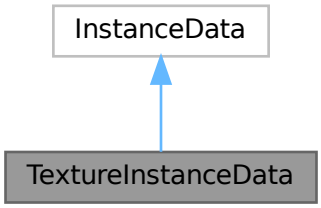
- std::deque< std::function< void()> > **tasks**
- std::mutex **mutex_**
- std::condition_variable **taskCondition**
- std::atomic< int > **remaining_tasks** = 0
- bool **shuttingDown** = false

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

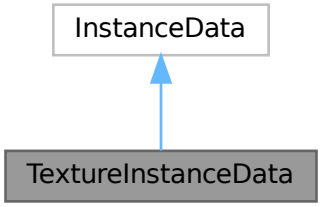
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Threader/Threader.h

5.88 TextureInstanceData Struct Reference

Inheritance diagram for TextureInstanceData:



Collaboration diagram for TextureInstanceData:



Public Member Functions

- **TextureInstanceData** (glm::vec3 mSize, Position mBodyCenter, Rotation mRotation, GLuint Texture)
- **TextureInstanceData** (glm::vec2 mSize, Position mBodyCenter, Rotation mRotation, GLuint Texture)

Public Member Functions inherited from InstanceData

- **InstanceData** (glm::vec3 mSize, Position mBodyCenter, Rotation mRotation)
- **InstanceData** (glm::vec2 mSize, Position mBodyCenter, Rotation mRotation)

Public Attributes

- GLuint **texture** = 0
- UV **uv** = glm::vec2(0.0f)

Public Attributes inherited from InstanceData

- Size **size** = glm::vec3(0.0f)
- Position **bodyCenter** = glm::vec3(0.0f)
- Rotation **rotation** = glm::vec3(0.0f)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.89 TextureManager Class Reference

Public Member Functions

- void **Add_GLTexture** (std::string id, const char *path)
- const GLTexture * **Get_GLTexture** (std::string id)
- std::vector< std::string > **Get_GLTextureNames** () const

Static Public Member Functions

- static TextureManager & **getInstance** ()
- static bool **readFileToBuffer** (const char *filePath, std::vector< unsigned char > &buffer)
- static GLTexture * **loadPNG** (const char *filePath)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/TextureManager/TextureManager.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/TextureManager/TextureManager.↩ cpp

5.90 TextureMeshRenderer Struct Reference

Public Attributes

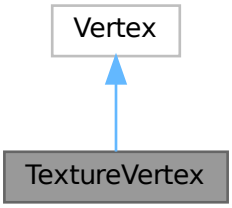
- size_t **meshIndices** = 0
- std::vector< TextureInstanceData > **instances**
- GLuint **vao**
- GLuint **vbo**
- GLuint **ibo**

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

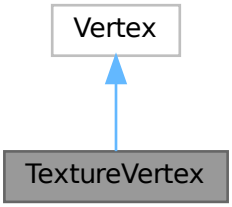
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GLSLProgram.h

5.91 TextureVertex Struct Reference

Inheritance diagram for TextureVertex:



Collaboration diagram for TextureVertex:



Public Member Functions

- void **setUV** (UV m_uv)

Public Member Functions inherited from Vertex

- void **setPosition** (Position m_position)

Public Attributes

- UV **uv** = UV(0)

Public Attributes inherited from Vertex

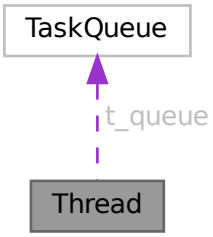
- Position **position** = Position(0)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Vertex.h

5.92 Thread Struct Reference

Collaboration diagram for Thread:



Public Member Functions

- **Thread** (TaskQueue &task_queue_, int id_)
- void **run** ()
- void **stop** ()

Public Attributes

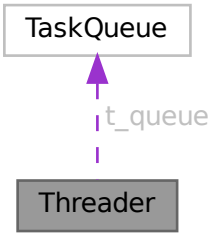
- int **id** = 0
- std::thread **cur_thread**
- std::function< void()> **task** = nullptr
- bool **running** = true
- TaskQueue * **t_queue** = nullptr

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Threader/Threader.h

5.93 Threader Struct Reference

Collaboration diagram for Threader:



Public Member Functions

- **Threader** (int num_threads_)
- void **parallel** (int num_obj, std::function< void(int start, int end)> &&callback)

Public Attributes

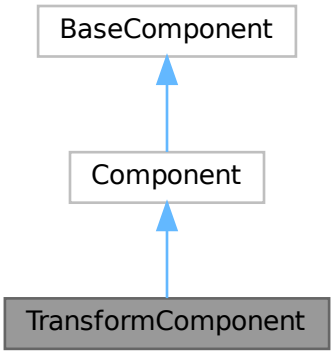
- TaskQueue **t_queue**
- int **num_threads** = 1
- std::vector< Thread > **threads**

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

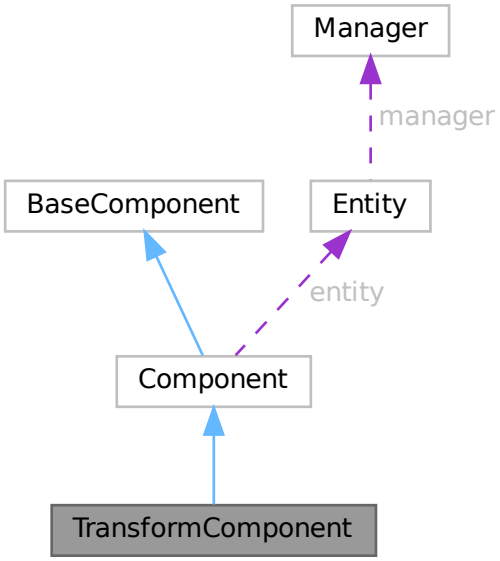
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Threader/Threader.h

5.94 TransformComponent Class Reference

Inheritance diagram for TransformComponent:



Collaboration diagram for TransformComponent:



Public Member Functions

- **TransformComponent** (float sc)
- **TransformComponent** (glm::vec2 m_position)
- **TransformComponent** (glm::vec3 m_position)
- **TransformComponent** (glm::vec2 m_position, layer layer, glm::vec2 m_size, float sc)
- **TransformComponent** (glm::vec2 m_position, layer layer, glm::vec2 size, float sc, int sp)
- **TransformComponent** (glm::vec2 m_position, layer layer, glm::vec3 m_size, float sc)
- **TransformComponent** (glm::vec2 m_position, layer layer, glm::vec3 size, float sc, int sp)
- **TransformComponent** (glm::vec3 m_position, glm::vec3 m_size, float sc)
- void init () override
- void update (float deltaTime) override
- glm::vec3 **getCenterTransform** ()
- glm::vec3 **getSizeCenter** ()
- glm::vec3 **getPosition** ()
- void **setPosition_X** (float newPosition_X)
- void **setPosition_Y** (float newPosition_Y)
- glm::vec3 **getVelocity** ()
- void **setVelocity_X** (float newVelocity_X)
- void **setVelocity_Y** (float newVelocity_Y)
- void **setRotation** (glm::vec3 m_rotation)
- std::string GetComponentName () override
- void showGUI () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)

Public Attributes

- std::string **temp_lineParsed** = ""
- glm::vec3 **velocity** = glm::vec3(0)
- glm::vec3 **rotation** = { 0.0f,0.0f,0.0f }
- glm::vec3 **position** = glm::vec3(0)
- glm::vec3 **size** = glm::vec3(0)
- glm::vec3 **last_position** = glm::vec3(0)
- glm::vec3 **last_size** = glm::vec3(0)
- glm::vec3 **last_velocity** = glm::vec3(0)
- glm::vec3 **bodyCenter** = { 0.0f,0.0f,0.0f }
- float **scale** = 1
- int **speed** = 1

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.94.1 Member Function Documentation

5.94.1.1 GetComponentName()

```
std::string TransformComponent::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.94.1.2 init()

```
void TransformComponent::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.94.1.3 showGUI()

```
void TransformComponent::showGUI ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.94.1.4 update()

```
void TransformComponent::update (
            float deltaTime )  [inline], [override], [virtual]
```

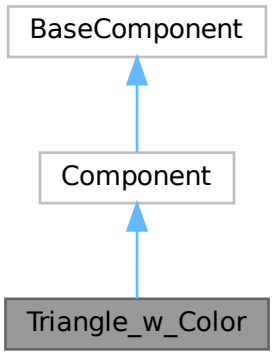
Reimplemented from BaseComponent.

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

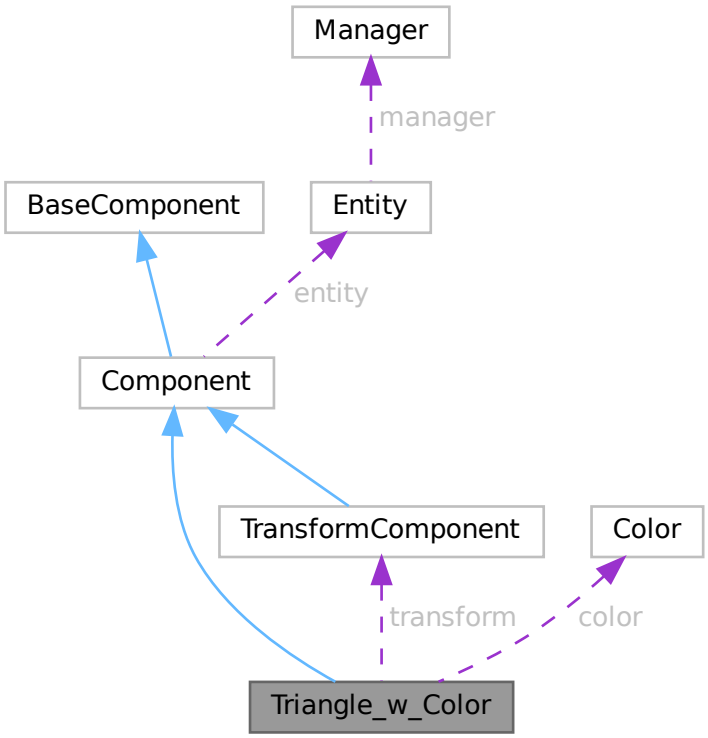
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Basic/TransformComponent.h

5.95 Triangle_w_Color Class Reference

Inheritance diagram for Triangle_w_Color:



Collaboration diagram for Triangle_w_Color:



Public Member Functions

- void init () override
- void update (float deltaTime) override
- void draw (size_t v_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- Color **color** = { 255, 255, 255, 255 }
- glm::vec2 **uv1** = glm::vec2(0)
- glm::vec2 **uv2** = glm::vec2(0)
- glm::vec2 **uv3** = glm::vec2(0)
- TransformComponent * **transform** = nullptr

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.95.1 Member Function Documentation

5.95.1.1 draw()

```
void Triangle_w_Color::draw (
            size_t v_index,
            PlaneColorRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [virtual]
```

Reimplemented from BaseComponent.

5.95.1.2 GetComponentName()

```
std::string Triangle_w_Color::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.95.1.3 init()

```
void Triangle_w_Color::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.95.1.4 update()

```
void Triangle_w_Color::update (
            float deltaTime )  [inline], [override], [virtual]
```

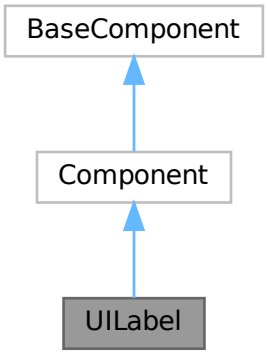
Reimplemented from BaseComponent.

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

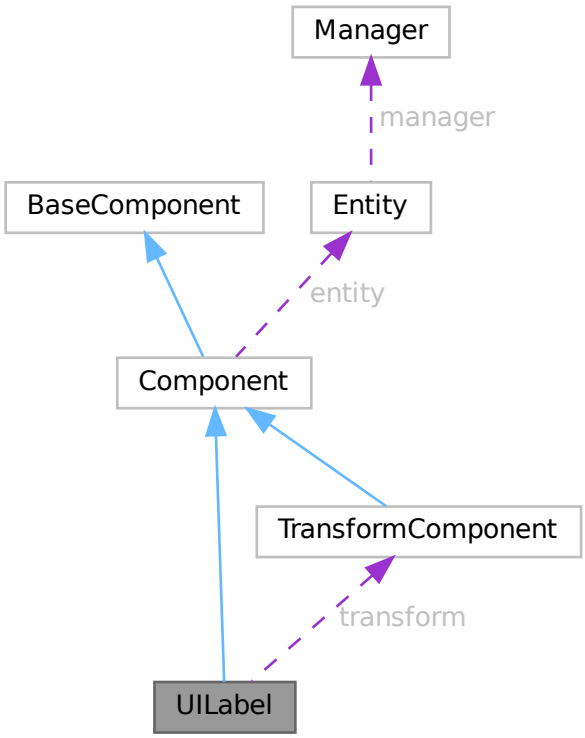
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Basic/Triangle_w_Color.h

5.96 UILabel Class Reference

Inheritance diagram for UILabel:



Collaboration diagram for UILabel:



Public Member Functions

- **UILabel** (Manager *manager, std::string lab, std::string fontFam)
- void init () override
- void update (float deltaTime) override
- void draw (size_t e_index, PlaneModelRenderer &batch, TazGraphEngine::Window &window) override
- void **setLetters** (std::string lab)
- std::string GetComponentName () override

Public Member Functions inherited from BaseComponent

- virtual void **draw** (size_t e_index, LineRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, PlaneColorRenderer &batch, TazGraphEngine::Window &window)
- virtual void **draw** (size_t e_index, LightRenderer &batch, TazGraphEngine::Window &window)
- virtual void **showGUI** ()

Public Attributes

- TransformComponent * **transform** = nullptr

Public Attributes inherited from Component

- Entity * **entity** = nullptr

Public Attributes inherited from BaseComponent

- ComponentID **id** = 0u

5.96.1 Member Function Documentation

5.96.1.1 draw()

```
void UILabel::draw (
            size_t e_index,
            PlaneModelRenderer & batch,
            TazGraphEngine::Window & window )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.96.1.2 GetComponentName()

```
std::string UILabel::GetComponentName ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.96.1.3 init()

```
void UILabel::init ( )  [inline], [override], [virtual]
```

Reimplemented from BaseComponent.

5.96.1.4 update()

```
void UILabel::update (
            float deltaTime )  [inline], [override], [virtual]
```

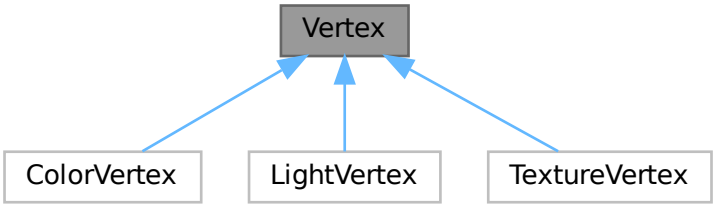
Reimplemented from BaseComponent.

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/GECS/Components/Empty/↩ Util/UILabel.h

5.97 Vertex Struct Reference

Inheritance diagram for Vertex:



Public Member Functions

- void **setPosition** (Position m_position)

Public Attributes

- Position **position** = Position(0)

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Vertex.h

5.98 TazGraphEngine::Window Class Reference

Public Member Functions

- int **create** (std::string windowName, int screenWidth, int screenHeight, float scale, unsigned int currentFlags)
- void **swapBuffer** ()
- void **setScreenWidth** (int width)
- int **getScreenWidth** ()
- void **setScreenHeight** (int height)
- int **getScreenHeight** ()
- void **setScale** (float scale)
- float **getScale** ()

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

- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Window/Window.h
- /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraphEngine/Window/Window.cpp

Chapter 6

File Documentation

6.1 App.h

```
00001 #pragma once
00002
00003 #include "GraphScreen/AppInterface.h"
00004 #include "../Graph.h"
00005 #include "../MainMenuScreen/MainMenuScreen.h"
00006
00007 class App : public AppInterface
00008 {
00009 public:
00010     App(int threadCount);
00011     ~App();
00012
00013     // Called on initialization
00014     virtual void onInit() override;
00015     // For adding all screens
00016     virtual void addScenes() override;
00017     // Called when exiting
00018     virtual void onExit() override;
00019 private:
00020
00021     std::unique_ptr<Graph> _graphplayScreen = nullptr;
00022     std::unique_ptr<MainMenuScreen> _mainMenuScreen = nullptr;
00023
00024     //std::unique_ptr<EditorScreen> m_editorScreen = nullptr;
00025 };
00026
```

6.2 AssetManager.h

```
00001 #pragma once
00002
00003 #include <string>
00004 #include "TextureManager/TextureManager.h"
00005 #include "GECS/Core/GECSManager.h"
00006 #include <SDL2/SDL_ttf.h>
00007 #include "../Graph.h"
00008
00009 struct PairHash {
00010     template <class T1, class T2>
00011     std::size_t operator()(const std::pair<T1, T2>& p) const {
00012         auto hash1 = std::hash<T1>{}(p.first);
00013         auto hash2 = std::hash<T2>{}(p.second);
00014         return hash1 ^ (hash2 « 1);
00015     }
00016 };
00017 class AssetManager //this class created when we added projectiles, based on this class other
    components changed
00018 {                   //it just replaces the paths of textures with names
00019 public:
00020
00021     SDL_Color black = { 0, 0 ,0 ,255 };
00022     SDL_Color white = { 255, 255 ,255 ,255 };
00023     SDL_Color red = { 255, 0 ,0 ,255 };
00024     SDL_Color green = { 0, 255 ,0 ,255 };
```

```
00025
00026     AssetManager(Manager* man, InputManager& inputManager, TazGraphEngine::Window& window);
00027     ~AssetManager();
00028
00029     //graphobjects
00030     void CreateWorldMap(Entity& worldMap);
00031     void CreateGroup(Entity& groupNode, glm::vec3 centerGroup, float groupNodeSize, Grid::Level
      m_level);
00032
00033     void CreateGroupLink(Entity& groupLink, Grid::Level m_level);
00034
00035     void createGroupLayout(Grid::Level m_level);
00036
00037     void ungroupLayout(Grid::Level m_level);
00038
00039     Manager* manager;
00040 private:
00041     InputManager& _inputManager;
00042     TazGraphEngine::Window& _window;
00043 };
```

6.3 CustomFunctions.h

```
00001 #pragma once
00002
00003 #include "GECS/Components.h"
00004 #include "GECS/UtilComponents.h"
00005
00006 class CustomFunctions {
00007 public:
00008
00009     bool isScriptResultsOpen = false;
00010     int activatedScriptShown = 0;
00011
00012     void renderUI(Manager& manager, std::vector<std::pair<Entity*, glm::vec3>>& m_selectedEntities);
00013
00014     void default_renderUI();
00015
00016     void CalculateDegree(Manager& manager, std::vector<std::pair<Entity*, glm::vec3>&
      m_selectedEntities);
00017     void CalculateSignals();
00018     void CalculateHeatMap();
00019     void DrawCandlestickChart();
00020 };
```

6.4 EditorIMGUI.h

```
00001 #pragma once
00002
00003 #include "../Map/Map.h"
00004 #include <algorithm>
00005 #include <vector>
00006 #include <string>
00007 #include <filesystem>
00008 #include "ImGuiInterface/ImGuiInterface.h"
00009 #include "imguiComboAutoselect/imgui_combo_autoselect.h"
00010 #include "BaseFPSLimiter/BaseFPSLimiter.h"
00011
00012 #include "CustomFunctions/CustomFunctions.h"
00013 #include "./EditorLayoutUtils.h"
00014
00015 namespace fs = std::filesystem;
00016
00017 // it is to provide the ImgUI functions for the whole project
00018 class EditorIMGUI : public ImGuiInterface {
00019 private:
00020     std::vector<std::string> _fileNames;
00021     char  _newFileName[126] = "";
00022     std::vector<std::string> _pollingFileNames;
00023     ImGui::ComboAutoSelectData _data;
00024     int _currentOrientationIndex = 0;
00025
00026     bool _filesLoaded = false;
00027
00028     bool _isSaving = false;
00029     bool _isStartingNew = false;
00030     bool _isLoading = false;
00031     bool _goingBack = false;
00032
```

```
00033     CustomFunctions _customFunctions;
00034
00035     // Note: Switch this to true to enable dockspace
00036     bool _dockingEnabled = true;
00037     int _lastEntityDisplayed = 0;
00038 public:
00039     float cameraRotationZ = 0;
00040     int newNodesCount = 0;
00041     int newLinksCount = 0;
00042
00043     float interpolation = 0.0f;
00044     float interpolation_speed = 0.01f;
00045     bool interpolation_running = false;
00046
00047
00048     bool isMouseInSecondColumn = false;
00049
00050
00051
00052     int last_activeLayout = 0;
00053     int activeLayout = 0;
00054
00055     EditorIMGUI();
00056
00057     ~EditorIMGUI();
00058
00059     bool isSaving();
00060     void setNewMap(bool startingNew);
00061     bool isStartingNew();
00062     bool isLoading();
00063     void setLoading(bool loading);
00064     bool isGoingBack();
00065     void SetGoingBack(bool goingBack);
00066
00067     void updateFileNamesInAssets();
00068
00069     void updatePollingFileNamesInAssets();
00070
00071     bool* getDockspaceRef();
00072     void MenuBar();
00073
00074     bool isMouseOnWidget(const std::string& widgetName);
00075     void LeftColumnUIElement(bool& renderDebug, bool& clusterLayout, glm::vec2 mouseCoords, glm::vec2
      mouseCoords2, Manager& manager, Entity* selectedEntity, float(&backgroundColor)[4], int cell_size);
00076     void RightColumnUIElement(Manager& manager, float* nodeRadius);
00077     void FPSCounter(const BaseFPSLimiter& baseFPSLimiter);
00078     void ReloadAccessibleFiles();
00079     void SavingUI(Map* map);
00080     void NewMapUI();
00081     char* LoadingUI();
00082     void MainMenuUI(std::function<void()> onStartSimulator, std::function<void()> onLoadSimulator,
      std::function<void()> onExitSimulator);
00083     void ShowAllEntities(Manager& manager, float& m_nodeRadius);
00084     void availableFunctions();
00085     void SceneViewport(uint32_t textureId, ImVec2& storedWindowPos, ImVec2& storedWindowSize);
00086     void scriptResultsVisualization(Manager& manager, std::vector<std::pair<Entity*, glm::vec3>>&
      m_selectedEntities);
00087     std::string SceneTabs(const std::vector<std::string>& graphNames, std::string& currentActive);
00088     void ShowFunctionExecutionResults();
00089     void updateIsMouseInSecondColumn();
00090     void ShowEntityComponents(glm::vec2 mousePos, Entity* displayedEntity, Manager& manager);
00091     void ShowSceneControl(glm::vec2 mousePos, Manager& manager);
00092     void StartPollingComponent(Entity* entity, const std::string& fileName);
00093 };
```

6.5 EditorLayoutUtils.h

```
00001 #pragma once
00002
00003 #include "../Map/Map.h"
00004
00005 namespace EditorLayoutUtils {
00006
00007     void rotateCamera(float& cameraRotationZ); // Declaration only
00008
00009 }
```

6.6 ScriptComponents.h

```
00001 #pragma once
```

```
00002
00003 #include "Scripts/MainMenuBackground.h"
```

6.7 MainMenuBackground.h

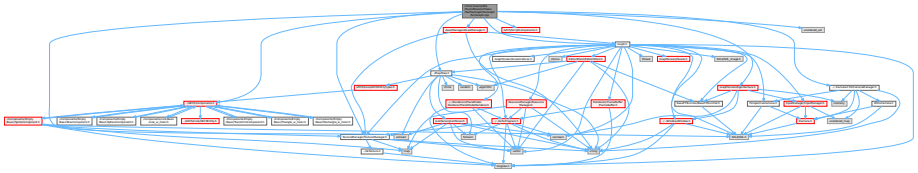
```
00001 #pragma once
00002
00003 #include "GECS/Animators/AnimatorComponent.h"
00004
00005
00006 class MainMenuBackground : public Component
00007 {
00008 private:
00009     TransformComponent* transform = nullptr;
00010     SpriteComponent* sprite = nullptr;
00011
00012     float elapsedTime = 0.0f;
00013 public: // it is like it has init that creates Animator Component since it inherits it
00014
00015
00016     MainMenuBackground()
00017     {
00018
00019     }
00020
00021     ~MainMenuBackground() {
00022
00023     }
00024
00025     void init() override {
00026         std::shared_ptr<PerspectiveCamera> main_camera2D =
      std::dynamic_pointer_cast<PerspectiveCamera>(CameraManager::getInstance().getCamera("mainMenu_main"));
00027
00028         if (!entity->hasComponent<TransformComponent>()) {
00029             entity->addComponent<TransformComponent>(
00030                 glm::vec2(
00031                     -TextureManager::getInstance().Get_GLTexture("graphnetwork")->width / 2,
00032                     -TextureManager::getInstance().Get_GLTexture("graphnetwork")->height / 2
00033                 ), Layer::action,
00034                 glm::ivec2(
00035                     TextureManager::getInstance().Get_GLTexture("graphnetwork")->width,
00036                     TextureManager::getInstance().Get_GLTexture("graphnetwork")->height
00037                 ),
00038                 1.0f);
00039         }
00040         if (!entity->hasComponent<SpriteComponent>()) {
00041             entity->addComponent<SpriteComponent>("graphnetwork", true);
00042         }
00043         transform = &entity->GetComponent<TransformComponent>();
00044         sprite = &entity->GetComponent<SpriteComponent>();
00045
00046     }
00047
00048     void update(float deltaTime) override {
00049
00050         elapsedTime += deltaTime; // Update the accumulated elapsed time
00051         float amplitude = 100.0f; // Maximum displacement along the Y axis
00052         float frequency = 0.002f; // How fast the object moves up and down
00053
00054         // Calculate the new Y position
00055         float newY = amplitude * sin(frequency * elapsedTime);
00056
00057         transform->position.z = newY;
00058     }
00059
00060     std::string GetComponentName() override {
00061         return "MainMenuBackground";
00062     }
00063 };
```

6.8 /mnt/c/Users/lefte/Mujin/MastersThesis/Taz/TazGraph/TazGraph/↩ Src/Graph.cpp File Reference

```
#include "Graph.h"
#include "TextureManager/TextureManager.h"
```

```
#include "Camera2.5D/CameraManager.h"
#include "Map/Map.h"
#include "GECS/Components.h"
#include "GECS/ScriptComponents.h"
#include "AssetManager/AssetManager.h"
#include <sstream>
#include "GraphScreen/AppInterface.h"
#include <unordered_set>
```

Include dependency graph for Graph.cpp:



Functions

- glm::vec2 **convertScreenToWorld** (glm::vec2 screenCoords)

Variables

- glm::vec3 **pointAtZ0**
- glm::vec3 **pointAtO**
- float **nodeRadius** = 1.0f

6.9 Graph.h

```
00001 #ifndef GRAPH_H
00002 #define GRAPH_H
00003
00004 #include <GraphScreen/IScene.h>
00005 #include <SDL2/SDL.h>
00006 #include <SDL2/SDL_image.h>
00007 #include <GL/glew.h>
00008 #include "GLSLProgram.h"
00009 #include "ResourceManager/ResourceManager.h"
00010 #undef main
00011 #include <iostream>
00012 #include <vector>
00013 #include "Camera2.5D/PerspectiveCamera.h"
00014 #include <Renderers/FrameBuffer/Framebuffer.h>
00015 #include "Renderers/PlaneModelRenderer/PlaneModelRenderer.h"
00016 #include "InputManager/InputManager.h"
00017 #include "BaseFPSLimiter/BaseFPSLimiter.h"
00018 //#include "SpriteFont/SpriteFont.h"
00019 #include "Window/Window.h"
00020 #include "TextureManager/TextureManager.h"
00021
00022
00023 #include "GraphScreen/ScreenIndices.h"
00024
00025 #include "EditorIMGUI/EditorIMGUI.h"
00026
00027 #include <chrono>
00028 #include <thread>
00029
00030 class Map;
00031 class AssetManager;
00032 class SceneManager;
00033 class ColliderComponent;
00034 class TransformComponent;
00035
00036
00037
```

```
00038 #define ON_HOVER 0
00039 #define CTRLD_LEFT_CLICK -1
00040
00041 class Graph : public IScene {
00042
00043 public:
00044     Graph(TazGraphEngine::Window* window);
00045     ~Graph();
00046
00047
00048     virtual int getNextSceneIndex() const override;
00049
00050     virtual int getPreviousSceneIndex() const override;
00051
00052     virtual void build() override;
00053
00054     virtual void destroy() override;
00055
00056     virtual void onEntry() override;
00057
00058     virtual void onExit() override;
00059
00060     virtual void update(float deltaTime) override;
00061
00062     virtual void draw() override;
00063
00064     virtual void BeginRender() override;
00065     virtual void updateUI() override;
00066     virtual void EndRender() override;
00067
00068
00070     void renderBatch(const std::vector<LinkEntity*>& entities, LineRenderer& batch);
00071     void renderBatch(const std::vector<EmptyEntity*>& entities, PlaneColorRenderer& batch);
00072     void renderBatch(const std::vector<NodeEntity*>& entities, PlaneColorRenderer& batch);
00073     void renderBatch(const std::vector<EmptyEntity*>& entities, PlaneModelRenderer& batch);
00074     void renderBatch(const std::vector<NodeEntity*>& entities, PlaneModelRenderer& batch);
00075     void renderBatch(const std::vector<EmptyEntity*>& entities, LightRenderer& batch);
00076     void drawHUD(const std::vector<NodeEntity*>& entities);
00077
00079
00080     Map* map = nullptr;
00081     //std::unique_ptr<Grid> grid;
00082
00083     static TazGraphEngine::Window* _window;
00084
00085
00086 private:
00087     float _backgroundColor[4] = { 0.407f,0.384f,0.356f, 1.0f };
00088
00089     std::vector<Cell*> traversedCellsFromRay(glm::vec3 rayOrigin,
00090         glm::vec3 rayDirection,
00091         float maxDistance);
00092
00093     void selectEntityFromRay(glm::vec3 rayOrigin, glm::vec3 rayDirection, int activateMode);
00094
00095     bool setManager(std::string m_managerName);
00096
00097     void checkInput();
00098     bool onPauseGraph();
00099
00100     PlaneModelRenderer _PlaneModelRenderer;
00101     PlaneModelRenderer _hudPlaneModelRenderer;
00102     PlaneColorRenderer _PlaneColorRenderer;
00103     LineRenderer _LineRenderer;
00104     LightRenderer _LightRenderer;
00105
00106     AssetManager* _assetsManager = nullptr;
00107
00108     ResourceManager _resourceManager;
00109
00110     std::vector<std::pair<Entity*, glm::vec3» _selectedEntities;
00111     Entity* _displayedEntity = nullptr;
00112     bool _sceneManagerActive = false;
00113     Entity* _onHoverEntity = nullptr;
00114
00115     int _nextSceneIndex = SCENE_INDEX_GRAPHPLAY;
00116     int _prevSceneIndex = SCENE_INDEX_MAIN_MENU;
00117
00118     const float SCALE_SPEED = 0.1f;
00119     bool _firstLoop = true;
00120
00121     EditorIMGUI _editorImgui;
00122
00123     bool _showSaveWindow = false;
00124
00125     Framebuffer _framebuffer;
00126
```

```
00127     unsigned int _rectVAO = 0, _rectVBO = 0;
00128
00129     unsigned int _FBO = 0;
00130     unsigned int _framebufferTexture = 0;
00131     unsigned int _RBO = 0;
00132
00133     ImVec2 _windowPos;
00134     ImVec2 _windowSize;
00135
00136     glm::vec2 _sceneMousePosition = {0.f,0.f};
00137     glm::vec2 _savedMainViewportMousePosition = { 0.f,0.f };
00138
00139 };
00140
00141
00142 #endif
```

6.10 MainMenuScreen.h

```
00001 #pragma once
00002
00003 #include "GraphScreen/IScene.h"
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL_image.h>
00006 #include <GL/glew.h>
00007 #include "GLSLProgram.h"
00008 #include "ResourceManager/ResourceManager.h"
00009 #undef main
00010 #include <iostream>
00011 #include <vector>
00012 #include <functional>
00013 #include "Camera2.5D/PerspectiveCamera.h"
00014 #include "Renderers/PlaneModelRenderer/PlaneModelRenderer.h"
00015 #include "InputManager/InputManager.h"
00016 #include "BaseFPSLimiter/BaseFPSLimiter.h"
00017 //#include "../SpriteFont/SpriteFont.h"
00018 #include "Window/Window.h"
00019 #include "TextureManager/TextureManager.h"
00020
00021 #include "GraphScreen/ScreenIndices.h"
00022
00023 #include "../EditorIMGUI/EditorIMGUI.h"
00024
00025 class AssetManager;
00026
00027 class MainMenuScreen : public IScene {
00028 public:
00029     MainMenuScreen(TazGraphEngine::Window* window);
00030     ~MainMenuScreen();
00031
00032     virtual int getNextSceneIndex() const override;
00033
00034     virtual int getPreviousSceneIndex() const override;
00035
00036     virtual void build() override;
00037
00038     virtual void destroy() override;
00039
00040     virtual void onEntry() override;
00041
00042     virtual void onExit() override;
00043
00044     virtual void update(float deltaTime) override;
00045
00046     virtual void draw() override;
00047
00048     virtual void BeginRender() override;
00049     virtual void updateUI() override;
00050     virtual void EndRender() override;
00051
00052
00053     void renderBatch(const std::vector<EmptyEntity*>& entities);
00054
00055 private:
00056     float _backgroundColor[4] = { 0.8f, 0.8f, 0.8f, 1.0f };
00057
00058     AssetManager* _assetsManager;
00059
00060     void checkInput();
00061     bool onStartSimulator();
00062     bool onResumeSimulator();
00063     bool onLoadSimulator();
00064     void onExitSimulator();
```

```
00065
00066     TazGraphEngine::Window* _window;
00067
00068     PlaneModelRenderer _PlaneModelRenderer;
00069
00070     ResourceManager _resourceManager;
00071
00072     int _nextSceneIndex = SCENE_INDEX_GRAPHPLAY;
00073     int _prevSceneIndex = SCENE_INDEX_GRAPHPLAY;
00074
00075     EditorIMGUI _editorImgui;
00076 };
```

6.11 Map.h

```
00001 #pragma once
00002 #include <string>
00003 #include <fstream>
00004 #include <sstream>
00005
00006 #include "JsonParser/JsonParser.h"
00007 #include "GECS/Core/GECSEntityTypes.h"
00008
00009 #include <algorithm>
00010 #include <random>
00011 #include <ctime>
00012
00013 class Map
00014 {
00015 public:
00016
00017     Map(Manager& m_manager, int ms, int ns);
00018     ~Map();
00019
00020     void saveMapAsText(const char* fileName);
00021     void ProcessFile(std::ifstream& mapFile, void(Map::* addNodeFunction)(Entity&, glm::vec3
      mPosition), void(Map::* addLinkFunction)(Entity&));
00022     void ProcessPythonFile(std::ifstream& mapFile, void(Map::* addNodeFunction)(Entity&, glm::vec3
      mPosition), void(Map::* addLinkFunction)(Entity&));
00023     void loadTextMap(const char* fileName);
00024
00025     void loadPythonMap(const char* fileName);
00026
00027     void AddDefaultNode(Entity& node, glm::vec3 mPosition);
00028     void AddTreeNode(Entity& node, glm::vec3 mPosition);
00029     void AddDefaultLink(Entity& node);
00030     void AddTreeLink(Entity& link);
00031
00032     Manager* manager;
00033 private:
00034     int mapScale;
00035     int nodeSize;
00036     int scaledSize;
00037 };
```

6.12 AABB.h

```
00001 #pragma once
00002 #include <glm/glm.hpp>
00003 #include <SDL2/SDL_rect.h>
00004 #include <type_traits>
00005
00006 template<typename RectType>
00007 inline bool checkCollision(const RectType& recA, const RectType& recB) {
00008     static_assert(std::is_same<RectType, SDL_Rect>::value || std::is_same<RectType, SDL_FRect>::value,
00009         "checkCollision: RectType must be either SDL_Rect or SDL_FRect");
00010
00011     if (recA.x >= recB.x + recB.w || recA.x + recA.w <= recB.x ||
00012         recA.y >= recB.y + recB.h || recA.y + recA.h <= recB.y) {
00013         return false; // no collision
00014     }
00015     return true;
00016 }
00017
00018 inline bool checkCollision(const SDL_Rect& recA, const SDL_FRect& recB) {
00019     SDL_FRect convertedA = { static_cast<float>(recA.x), static_cast<float>(recA.y),
00020                              static_cast<float>(recA.w), static_cast<float>(recA.h) };
00021     return checkCollision(convertedA, recB);
00022 }
```

```
00023
00024 inline bool checkCollision(const SDL_FRect& recA, const SDL_Rect& recB) {
00025     return checkCollision(recB, recA);
00026 }
00027
00028 inline bool checkCollision3D(const glm::vec3& centerA, const glm::vec3& halfSizeA,
00029     const glm::vec3& centerB, const glm::vec3& halfSizeB,
00030     float padding = 0.0f) {
00031
00032     glm::vec3 paddedHalfSizeA = halfSizeA + glm::vec3(padding);
00033     glm::vec3 paddedHalfSizeB = halfSizeB + glm::vec3(padding);
00034
00035
00036     return !(centerA.x + paddedHalfSizeA.x <= centerB.x - paddedHalfSizeB.x ||
00037         centerA.x - paddedHalfSizeA.x >= centerB.x + paddedHalfSizeB.x ||
00038         centerA.y + paddedHalfSizeA.y <= centerB.y - paddedHalfSizeB.y ||
00039         centerA.y - paddedHalfSizeA.y >= centerB.y + paddedHalfSizeB.y ||
00040         centerA.z + paddedHalfSizeA.z <= centerB.z - paddedHalfSizeB.z ||
00041         centerA.z - paddedHalfSizeA.z >= centerB.z + paddedHalfSizeB.z);
00042 }
00043
00044 inline float pointLineDistance(glm::vec2 point, glm::vec2 lineStartPoint, glm::vec2 lineEndPoint) {
00045     float num = std::abs((lineEndPoint.y - lineStartPoint.y) * point.x - (lineEndPoint.x -
      lineStartPoint.x) * point.y + lineEndPoint.x * lineStartPoint.y - lineEndPoint.y * lineStartPoint.x);
00046     float den = std::sqrt((lineEndPoint.y - lineStartPoint.y) * (lineEndPoint.y - lineStartPoint.y) +
      (lineEndPoint.x - lineStartPoint.x) * (lineEndPoint.x - lineStartPoint.x));
00047     return num / den;
00048 }
00049
00050 inline bool rayIntersectsRectangle(const glm::vec3& rayOrigin, const glm::vec3& rayDirection,
00051     const glm::vec3& planePoint, const glm::vec3& planeNormal,
00052     float xMin, float xMax, float yMin, float yMax) {
00053     // Step 1: Check if the ray intersects the plane
00054     float denom = glm::dot(planeNormal, rayDirection);
00055
00056     // If denom == 0, the ray is parallel to the plane
00057     if (std::abs(denom) < 1e-6) {
00058         return false; // No intersection
00059     }
00060
00061     // Calculate t
00062     float t = glm::dot(planeNormal, planePoint - rayOrigin) / denom;
00063
00064     // If t < 0, the intersection is behind the ray origin
00065     if (t < 0) {
00066         return false;
00067     }
00068
00069     // Calculate the intersection point
00070     glm::vec3 intersectionPoint = rayOrigin + t * rayDirection;
00071
00072     // Step 2: Check if the intersection point lies within the rectangle bounds
00073     if (intersectionPoint.x >= xMin && intersectionPoint.x <= xMax &&
00074         intersectionPoint.y >= yMin && intersectionPoint.y <= yMax) {
00075         return true; // Intersection point is within the rectangle
00076     }
00077
00078     return false; // Intersection point is outside the rectangle
00079 }
00080
00081
00082 inline bool rayIntersectsSphere(
00083     const glm::vec3& rayOrigin,
00084     const glm::vec3& rayDirection,
00085     const glm::vec3& sphereCenter,
00086     float radius,
00087     glm::vec3& t
00088 ) {
00089     glm::vec3 oc = rayOrigin - sphereCenter;
00090
00091     float A = glm::dot(rayDirection, rayDirection);
00092     float B = 2.0f * glm::dot(oc, rayDirection);
00093     float C = glm::dot(oc, oc) - radius * radius;
00094
00095     float discriminant = B * B - 4 * A * C;
00096
00097     if (discriminant < 0) {
00098         return false;  // No intersection
00099     }
00100
00101     // Compute the closest valid t value
00102     float sqrtDiscriminant = sqrt(discriminant);
00103     float t0 = (-B - sqrtDiscriminant) / (2.0f * A);
00104     float t1 = (-B + sqrtDiscriminant) / (2.0f * A);
00105
00106     if (t0 >= 0) {
00107         t = rayOrigin + t0 * rayDirection;  // Closest intersection point
```

```
00108         return true;
00109     }
00110     else if (t1 >= 0) {
00111         t = rayOrigin + t1 * rayDirection;  // Intersection behind the ray origin
00112         return true;
00113     }
00114
00115     return false;
00116 }
00117
00118 inline bool sphereIntersectsBox(
00119     const glm::vec3& sphereCenter, float sphereRadius,
00120     const glm::vec3& boxMin, const glm::vec3& boxMax,
00121     glm::vec3& intersectionPoint)
00122 {
00123     glm::vec3 closestPoint = glm::clamp(sphereCenter, boxMin, boxMax);
00124
00125     intersectionPoint = closestPoint;
00126
00127     glm::vec3 diff = closestPoint - sphereCenter;
00128     float distanceSquared = glm::dot(diff, diff);
00129
00130     return distanceSquared <= (sphereRadius * sphereRadius);
00131 }
00132
00133 inline bool rayIntersectsBox(
00134     const glm::vec3& rayOrigin,
00135     const glm::vec3& rayDirection,
00136     const glm::vec3& boxMin, const glm::vec3& boxMax,
00137     glm::vec3& intersectionPoint,
00138     float m_maxT
00139 ) {
00140
00141     float sphereRad = glm::distance(boxMin, boxMax)/ 2.0f;
00142
00143
00144
00145
00146     for (float t = 0.0f; t < m_maxT; t += sphereRad) {
00147         glm::vec3 samplePoint = rayOrigin + t * rayDirection;
00148
00149         if (sphereIntersectsBox(
00150             samplePoint, sphereRad,
00151             boxMin, boxMax,
00152             intersectionPoint
00153         )) {
00154             return true;
00155         }
00156     }
00157     return false;
00158
00159 }
00160
00161
00162 inline bool rayIntersectsLineSegment(
00163     const glm::vec3& rayOrigin,
00164     const glm::vec3& rayDirection,
00165     const glm::vec3& segmentStart,
00166     const glm::vec3& segmentEnd,
00167     glm::vec3& intersectionPoint,
00168     float m_minT,
00169     float m_maxT,
00170     float m_sphereRad
00171 ) {
00172     for (float t = m_minT; t < m_maxT; t += m_sphereRad) {
00173         m_sphereRad += 0.005f;
00174         glm::vec3 samplePoint = rayOrigin + t * rayDirection;
00175
00176         glm::vec3 lineLength = segmentEnd - segmentStart;
00177
00178         glm::vec3 lineDir = glm::normalize(lineLength);
00179         glm::vec3 t_temp(0.0f);
00180
00181         if (rayIntersectsSphere(segmentStart, lineDir, samplePoint, m_sphereRad, t_temp)) {
00182             if(glm::distance(segmentStart, t_temp) < glm::distance(segmentEnd, segmentStart))
00183                 return true;
00184         }
00185     }
00186     return false;
00187 }
00188
00189 inline bool checkCircleLineCollision(glm::vec2 center, int circleRadius, glm::vec2 lineStartPoint,
      glm::vec2 lineEndPoint) {
00190     float dist = pointLineDistance(center, lineStartPoint, lineEndPoint);
00191
00192     if (dist <= circleRadius) {
00193         float dx = lineEndPoint.x - lineStartPoint.x;
```

```
00194         float dy = lineEndPoint.y - lineStartPoint.y;
00195         float t = ((center.x - lineStartPoint.x) * dx + (center.y - lineStartPoint.y) * dy) / (dx * dx
      + dy * dy);
00196 
00197         t = std::max(0.0f, std::min(1.0f, t));
00198 
00199         float closestX = lineStartPoint.x + t * dx;
00200         float closestY = lineStartPoint.y + t * dy;
00201 
00202         float distanceToCircle = std::sqrt((closestX - center.x) * (closestX - center.x) +
00203             (closestY - center.y) * (closestY - center.y));
00204         return distanceToCircle <= circleRadius;
00205     }
00206 
00207     return false;
00208 }
```

6.13 AudioEngine.h

```
00001 #pragma once
00002 
00003 #include <SDL2/SDL_mixer.h>
00004 #include "../ConsoleLogger.h"
00005 
00006 #include <string>
00007 #include <map>
00008 
00009 
00010     class SoundEffect {
00011     public:
00012         friend class AudioEngine;
00013 
00014         //@param loops: if loops == -1 --> loop forever
00015         // else play it loops+1 times
00016         void play(int loops = 0);
00017     private:
00018         Mix_Chunk* _chunk = nullptr;
00019     };
00020 
00021     class Music {
00022     public:
00023         friend class AudioEngine;
00024 
00025         //@param loops: if loops == -1 --> loop forever
00026         // else play it loops times
00027         void play(int loops = 1);
00028 
00029         static void pause();
00030         static void stop();
00031         static void resume();
00032     private:
00033         Mix_Music* _music = nullptr;
00034     };
00035 
00036 
00037     class AudioEngine {
00038     public:
00039         AudioEngine();
00040         ~AudioEngine();
00041 
00042         void init();
00043         void destroy();
00044 
00045         SoundEffect loadSoundEffect(const std::string& filePath);
00046         Music loadMusic(const std::string& filePath);
00047     private:
00048 
00049         std::map<std::string, Mix_Chunk*> _effectMap;
00050         std::map<std::string, Mix_Music*> _musicMap;
00051 
00052         bool _isInitialized = false;
00053     };
00054 
```

6.14 BaseFPSLimiter.h

```
00001 #pragma once
00002 
00003 #include <SDL2/SDL.h>
```

```
00004
00005 class BaseFPSLimiter {
00006 public:
00007     static constexpr int fps_history_count = 100;
00008
00009     BaseFPSLimiter();
00010
00011     void init(float maxFPS);
00012
00013     void setMaxFPS(float maxFPS);
00014
00015     void begin();
00016
00017     //end will return the current FPS
00018     float end();
00019
00020     float fpsHistory[fps_history_count] = { 0 };
00021     int fpsHistoryIndx = 0;
00022
00023     float fps;
00024     float maxFPS;
00025     Uint32 frameTime;
00026     Uint32 startTicks;
00027
00028     void setHistoryValue(float currentFPS);
00029
00030 private:
00031     void calculateFPS();
00032
00033 };
```

6.15 CameraManager.h

```
00001 #pragma once
00002
00003 #include <unordered_map>
00004 #include <memory>
00005 #include <string>
00006 #include "ICamera.h"
00007 #include "PerspectiveCamera.h"
00008 #include "OrthoCamera.h"
00009
00010 class CameraManager {
00011 public:
00012     // Gets the single instance of CameraManager (singleton)
00013     static CameraManager& getInstance() {
00014         static CameraManager instance; // Guaranteed to be destroyed. Instantiated on first use.
00015         return instance;
00016     }
00017
00018     CameraManager() {}
00019
00020     void addCamera(const std::string& name, std::shared_ptr<ICamera> camera) {
00021         cameras[name] = camera;
00022     }
00023
00024     std::shared_ptr<ICamera> getCamera(const std::string& name) {
00025         auto it = cameras.find(name);
00026         if (it != cameras.end()) {
00027             return it->second;
00028         }
00029         return nullptr; // or a default camera2D.worldLocation if you prefer
00030     }
00031
00032     void initializeCameras() {
00033         auto mainCamera = std::make_shared<PerspectiveCamera>();
00034         // Configure your mainCamera as needed
00035         CameraManager::getInstance().addCamera("main", mainCamera);
00036
00037         auto hudCamera = std::make_shared<OrthoCamera>();
00038         // Configure your hudCamera as needed, usually orthographic with fixed positioning
00039         CameraManager::getInstance().addCamera("hud", hudCamera);
00040
00041         auto mainMenu_mainCamera = std::make_shared<PerspectiveCamera>();
00042         // Configure your mainCamera as needed
00043         CameraManager::getInstance().addCamera("mainMenu_main", mainMenu_mainCamera);
00044
00045         auto mainMenu_hudCamera = std::make_shared<OrthoCamera>();
00046         // Configure your hudCamera as needed, usually orthographic with fixed positioning
00047         CameraManager::getInstance().addCamera("mainMenu_hud", mainMenu_hudCamera);
00048     }
00049
00050 private:
```

```
00051     std::unordered_map<std::string, std::shared_ptr<ICamera» cameras;
00052 };
```

6.16 ICamera.h

```
00001 #pragma once
00002
00003 #define GLM_ENABLE_EXPERIMENTAL
00004 #include <glm/glm.hpp>
00005 #include <glm/gtc/matrix_transform.hpp>
00006 #include <glm/gtx/rotate_vector.hpp>
00007 #include <SDL2/SDL.h>
00008
00009 class ICamera {
00010 public:
00011     virtual ~ICamera() = default;
00012
00013     // Initializes the camera2D.worldLocation with the screen's width and height
00014     virtual void init() = 0;
00015
00016     // Updates the camera2D.worldLocation's matrix if there have been any changes
00017     virtual void update() = 0;
00018
00019     // Converts screen coordinates to world coordinates
00020     virtual glm::vec2 convertScreenToWorld(glm::vec2 screenCoords) const = 0;
00021
00022     // Returns the dimensions of the camera2D.worldLocation's view
00023     virtual glm::ivec2 getCameraDimensions() const = 0;
00024
00025     // Returns the SDL_Rect representing the camera2D.worldLocation's viewport
00026     virtual SDL_FRect getCameraRect() const = 0;
00027
00028     // Additional methods to expose camera2D.worldLocation properties as needed
00029     virtual glm::vec3 getPosition() const = 0;
00030     virtual void setPosition(const glm::vec3 newPosition) = 0;
00031     virtual void setPosition_X(const float newPosition) = 0;
00032     virtual void setPosition_Y(const float newPosition) = 0;
00033     virtual void setPosition_Z(const float newPosition) = 0;
00034     virtual float getScale() const = 0;
00035     virtual glm::mat4 getCameraMatrix() const = 0;
00036     virtual void setScale(float scale) = 0;
00037
00038     virtual bool isPointInCameraView(const glm::vec4 point, float margin) = 0;
00039     virtual void makeCameraDirty() = 0;
00040     virtual bool hasChanged() = 0;
00041     virtual void refreshCamera() = 0;
00042 };
```

6.17 OrthoCamera.h

```
00001 #pragma once
00002 #include <SDL2/SDL.h>
00003 #include "ICamera.h"
00004
00005 class OrthoCamera : public ICamera {
00006 public:
00007     OrthoCamera() : _position(0.0f),
00008         _cameraMatrix(1.0f),    //I
00009         _projectionMatrix(1.0f),        //I
00010         _viewMatrix(1.0f),
00011         _scale(1.0f),
00012         _cameraChange(true),
00013         _screenWidth(800),
00014         _screenHeight(640)
00015     {
00016
00017     }
00018     ~OrthoCamera()
00019     {
00020
00021     }
00022
00023     void init() override {
00024
00025         _projectionMatrix = glm::ortho(0.0f, (float)_screenWidth, (float)_screenHeight, 0.0f);
00026     }
00027
00028     void update() override {
00029
```

```
00030         if (_cameraChange) {
00031
00032             _cameraMatrix = glm::mat4(1.0f);
00033
00034             glm::vec3 scale(_scale, _scale, 1.0f);
00035             _cameraMatrix = glm::scale(_cameraMatrix, scale);
00036
00037
00038             glm::vec3 translate(-_position.x, -_position.y, 0.0f);
00039             _cameraMatrix = glm::translate(_cameraMatrix, translate); //if glm ortho = -1,1,-1,1 then
      1 horizontal with -400,-320 to bottom-left
00040
00041             _cameraMatrix = _projectionMatrix * _viewMatrix * _cameraMatrix;
00042
00043             //_cameraMatrix = glm::scale(_cameraMatrix, scale);
00044             _cameraChange = false;
00045         }
00046     }
00047
00048     glm::vec2 convertScreenToWorld(glm::vec2 screenCoords) const override {
00049         //Make 0 the center
00050         screenCoords -= glm::vec2(_screenWidth / 2, _screenHeight / 2);
00051         //Scale coordinates
00052         screenCoords /= _scale;
00053         screenCoords += glm::vec2(_screenWidth / 2, _screenHeight / 2);
00054         //Translate with the camera2D.worldLocation position
00055         screenCoords.x += _position.x;
00056         screenCoords.y += _position.y;
00057
00058
00059         return screenCoords;
00060     }
00061
00062     //setters
00063     void setPosition(const glm::vec3 newPosition) override {
00064         _position = newPosition;
00065         _cameraChange = true;
00066     }
00067
00068     void setPosition_X(const float newPosition) override {
00069         _position.x = newPosition;
00070         _cameraChange = true;
00071     }
00072
00073     void setPosition_Y(const float newPosition) override {
00074         _position.y = newPosition;
00075         _cameraChange = true;
00076     }
00077
00078     void setPosition_Z(const float newPosition) override {
00079         _position.z = newPosition;
00080         _cameraChange = true;
00081     }
00082
00083     void setScale(float newScale) override {
00084         _scale = newScale;
00085         _cameraChange = true;
00086     }
00087
00088     //getters
00089     glm::vec3 getPosition() const override {
00090         return _position;
00091     }
00092
00093     float getScale() const override {
00094         return _scale;
00095     }
00096
00097     glm::mat4 getCameraMatrix() const override {
00098         return _cameraMatrix;
00099     }
00100
00101     glm::ivec2 getCameraDimensions() const override {
00102         glm::vec2 cameraDimensions = { _screenWidth, _screenHeight };
00103         return cameraDimensions;
00104     }
00105
00106     SDL_FRect getCameraRect() const override {
00107         float cameraWidth = getCameraDimensions().x / getScale();
00108         float cameraHeight = getCameraDimensions().y / getScale();
00109
00110         float cameraX = _position.x - cameraWidth / 2.0f + getCameraDimensions().x / 2;
00111         float cameraY = _position.y - cameraHeight / 2.0f + getCameraDimensions().y / 2;
00112
00113         SDL_FRect cameraRect = { cameraX , cameraY , cameraWidth, cameraHeight };
00114         return cameraRect;
00115     }
```

```
00116
00117     void setCameraMatrix(glm::mat4 newMatrix) {
00118         _cameraChange = true;
00119     }
00120
00121     bool isPointInCameraView(const glm::vec4 point, float margin)
00122     {
00123         glm::mat4 vpMatrix = _cameraMatrix;
00124
00125         glm::vec4 clipSpacePos = vpMatrix * point;
00126
00127         if (clipSpacePos.w != 0.0f) {
00128             clipSpacePos.x /= clipSpacePos.w;
00129             clipSpacePos.y /= clipSpacePos.w;
00130             clipSpacePos.z /= clipSpacePos.w;
00131         }
00132
00133         // 0.2f is the margin
00134         if (clipSpacePos.x < -1.0f - margin || clipSpacePos.x > 1.0f + margin) return false;
00135         if (clipSpacePos.y < -1.0f - margin || clipSpacePos.y > 1.0f + margin) return false;
00136         if (clipSpacePos.z < -margin || clipSpacePos.z > 1.0f + margin) return false;
00137
00138         return true;
00139     }
00140
00141     bool hasChanged() override {
00142         return _cameraChange;
00143     }
00144
00145     void makeCameraDirty() override {
00146         _cameraChange = true;
00147     }
00148
00149     void refreshCamera() override {
00150         _cameraChange = false;
00151     }
00152
00153 private:
00154     int _screenWidth, _screenHeight;
00155     float _scale;
00156     bool _cameraChange;
00157
00158     glm::vec3 _position;
00159     glm::mat4 _projectionMatrix; // changed once in init
00160     glm::mat4 _viewMatrix;
00161     glm::mat4 _cameraMatrix;
00162 };
```

6.18 PerspectiveCamera.h

```
00001 #pragma once
00002 #include <SDL2/SDL.h>
00003 #include "ICamera.h"
00004
00005 enum class ViewMode {
00006     Y_UP,
00007     Z_UP
00008 };
00009
00010
00011 class PerspectiveCamera : public ICamera{
00012 public:
00013     glm::vec3 eyePos{ 0,0,0 };
00014     glm::vec3 aimPos{ 0,0,0 };
00015     glm::vec3 upDir{0,-1,0};
00016     float zFar = 1000000.0f;
00017
00018     ViewMode currentViewMode = ViewMode::Y_UP;
00019
00020     PerspectiveCamera() : _position(0.0f, 0.0f),
00021         _cameraMatrix(1.0f),    //I
00022         _projectionMatrix(1.0f),        //I
00023         _viewMatrix(1.0f),
00024         _scale(1.0f),
00025         _cameraChange(true),
00026         _screenWidth(800),
00027         _screenHeight(640)
00028     {
00029         eyePos = glm::vec3(0.f, 0.f, -770.0f);
00030         aimPos = glm::vec3(0.f, 0.f, 0.f);
00031     }
00032
00033     PerspectiveCamera(glm::vec3 eye_pos, glm::vec3 aim_pos) : PerspectiveCamera()
```

```
00034     {
00035         eyePos = eye_pos;
00036         aimPos = aim_pos;
00037     }
00038 
00039     ~PerspectiveCamera()
00040     {
00041 
00042     }
00043 
00044     void init() override {
00045         _projectionMatrix = glm::perspective(glm::radians(45.0f), (float)_screenWidth /
      (float)_screenHeight, 0.1f, zFar); //left, right, top, bottom
00046         updateCameraOrientation();
00047 
00048         _cameraMatrix = glm::mat4(1.0f);
00049 
00050         glm::vec3 scale(_scale, _scale, 1.0f);
00051         _cameraMatrix = glm::scale(_cameraMatrix, scale);
00052 
00053 
00054         glm::vec3 translate(-_position.x, -_position.y, 0.0f);
00055         _cameraMatrix = glm::translate(_cameraMatrix, translate); //if glm ortho = -1,1,-1,1 then 1
      horizontal with -400,-320 to bottom-left
00056 
00057         _cameraMatrix = _projectionMatrix * _viewMatrix * _cameraMatrix;
00058 
00059     }
00060 
00061     void update() override {
00062         if (_cameraChange) {
00063             updateCameraOrientation();
00064 
00065             _cameraMatrix = glm::mat4(1.0f);
00066 
00067 
00068             glm::vec3 translate(-_position.x, -_position.y, 0.0f);
00069             _cameraMatrix = glm::translate(_cameraMatrix, translate); //if glm ortho = -1,1,-1,1 then
      1 horizontal with -400,-320 to bottom-left
00070 
00071             glm::vec3 scale(_scale, _scale, 1.0f);
00072             _cameraMatrix = glm::scale(_cameraMatrix, scale);
00073 
00074 
00075             _cameraMatrix = _projectionMatrix * _viewMatrix * _cameraMatrix;
00076 
00077         }
00078 
00079     }
00080 
00081     void updateCameraOrientation() {
00082         if (currentViewMode == ViewMode::Y_UP) {
00083             upDir = glm::vec3(0.0f, -1.0f, 0.0f);
00084 
00085             setOrientation(
00086                 eyePos, aimPos, upDir
00087             );
00088         }
00089         else {
00090             upDir = glm::vec3(0.0f, 0.0f, -1.0f);
00091 
00092             setOrientation(
00093                 eyePos, aimPos, upDir
00094             );
00095         }
00096     }
00097 
00098     void setOrientation(glm::vec3 eye, glm::vec3 target, glm::vec3 up) {
00099         _viewMatrix = glm::lookAt(eye, target, up);
00100     }
00101 
00102     glm::vec2 convertScreenToWorld(glm::vec2 screenCoords) const override {
00103         SDL_FRect cameraRect = getCameraRect();
00104 
00105         glm::vec2 worldCoords;
00106 
00107         worldCoords = screenCoords;
00108         worldCoords /= _scale;
00109 
00110         worldCoords.x = worldCoords.x + cameraRect.x;
00111         worldCoords.y = worldCoords.y + cameraRect.y;
00112 
00113 
00114         return worldCoords;
00115     }
00116 
00117     //setters
```

```
00118     void setPosition(const glm::vec3 newPosition) override {
00119         eyePos = newPosition;
00120         _cameraChange = true;
00121     }
00122
00123     void setPosition_X(const float newPosition) override {
00124         eyePos.x = newPosition;
00125         _cameraChange = true;
00126     }
00127
00128     void setPosition_Y(const float newPosition) override {
00129         eyePos.y = newPosition;
00130         _cameraChange = true;
00131     }
00132
00133     void setPosition_Z(const float newPosition) override {
00134         eyePos.z = newPosition;
00135         _cameraChange = true;
00136     }
00137
00138     void movePosition_Hor(const float step) {
00139         glm::vec3 direction = glm::normalize(aimPos - eyePos);  // Get movement direction
00140
00141         // Calculate the right vector (perpendicular to direction and up)
00142         glm::vec3 right = glm::normalize(glm::cross(direction, upDir));
00143
00144         // Move the camera horizontally along the right vector
00145         eyePos += right * step;
00146         aimPos += right * step;
00147         _cameraChange = true;
00148     }
00149     void movePosition_Vert(const float step) {
00150         glm::vec3 direction = glm::normalize(aimPos - eyePos);  // Get movement direction
00151
00152         // Move the camera horizontally along the right vector
00153         eyePos += upDir * step;
00154         aimPos += upDir * step;
00155         _cameraChange = true;
00156     }
00157
00158     void movePosition_Forward(const float step) {
00159         glm::vec3 direction = glm::normalize(aimPos - eyePos);
00160         eyePos += direction * step;
00161         aimPos += direction * step;
00162         _cameraChange = true;
00163     }
00164
00165     void setAimPos(const glm::vec3 newAimPos) {
00166         aimPos = newAimPos;
00167         _cameraChange = true;
00168     }
00169
00170     void moveAimPos(glm::vec3 startingAimPos, const glm::vec2 distance) {
00171         aimPos = startingAimPos;
00172         const float sensitivity = 0.0001f;
00173
00174         float yaw = distance.x * sensitivity;
00175         float pitch = distance.y * sensitivity;
00176
00177         glm::vec3 direction = glm::normalize(aimPos - eyePos);
00178
00179         direction = glm::rotate(direction, yaw, upDir);
00180
00181         glm::vec3 right = glm::normalize(glm::cross(direction, upDir));
00182
00183         direction = glm::rotate(direction, pitch, right);
00184
00185         // Update the aimPos based on the new direction
00186         aimPos = eyePos + direction;
00187         _cameraChange = true;
00188     }
00189
00190     glm::vec3 getEulerAnglesFromDirection(glm::vec3 direction) {
00191         float yaw = glm::atan(direction.x, direction.z);
00192         float pitch = glm::asin(-direction.y);
00193         float roll = 0.0f;
00194
00195         return glm::vec3(glm::degrees(pitch), glm::degrees(yaw), glm::degrees(roll));
00196     }
00197
00198
00199     float getZFar() {
00200         return zFar;
00201     }
00202
00203     glm::vec3 getAimPos() {
00204         return aimPos;
```

```
00205     }
00206
00207     void setScale(float newScale) override {
00208         _scale = newScale;
00209         _cameraChange = true;
00210     }
00211
00212     //getters
00213     glm::vec3 getPosition() const override {
00214         return eyePos;
00215     }
00216
00217     float getScale() const override {
00218         return _scale;
00219     }
00220
00221     glm::mat4 getCameraMatrix() const override {
00222         return _cameraMatrix;
00223     }
00224
00225     glm::ivec2 getCameraDimensions() const override {
00226         glm::vec2 cameraDimensions = { _screenWidth, _screenHeight };
00227         return cameraDimensions;
00228     }
00229
00230     SDL_FRect getCameraRect() const override {
00231         float cameraWidth = getCameraDimensions().x / getScale();
00232         float cameraHeight = getCameraDimensions().y / getScale();
00233
00234         float cameraX = _position.x - cameraWidth / 2.0f ;
00235         float cameraY = _position.y - cameraHeight / 2.0f ;
00236
00237         SDL_FRect cameraRect = { cameraX , cameraY , cameraWidth, cameraHeight };
00238         return cameraRect;
00239     }
00240
00241     void setCameraMatrix(glm::mat4 newMatrix) {
00242         _cameraChange = true;
00243     }
00244     void resetCameraPosition() {
00245
00246         _position = glm::vec2(0.0f,0.0f);
00247         _scale = 1.0f;
00248
00249         eyePos = glm::vec3(0.f, 0.f, -770.0f);
00250         aimPos = glm::vec3( 0,0,0 );
00251
00252         currentViewMode = ViewMode::Y_UP;
00253         upDir = glm::vec3( 0,-1,0 );
00254
00255         init();
00256
00257         _cameraChange = true;
00258     }
00259     float getMinScale() {
00260         return _minScale;
00261     }
00262
00263     float getMaxScale() {
00264         return _maxScale;
00265     }
00266
00267
00268     bool isPointInCameraView(const glm::vec4 point, float margin)
00269     {
00270         glm::mat4 vpMatrix = _cameraMatrix;
00271
00272         glm::vec4 clipSpacePos = vpMatrix * point;
00273
00274         if (clipSpacePos.w != 0.0f) {
00275             clipSpacePos.x /= clipSpacePos.w;
00276             clipSpacePos.y /= clipSpacePos.w;
00277             clipSpacePos.z /= clipSpacePos.w;
00278         }
00279
00280         // 0.2f is the margin
00281         if (clipSpacePos.x < -1.0f - margin || clipSpacePos.x > 1.0f + margin) return false;
00282         if (clipSpacePos.y < -1.0f - margin || clipSpacePos.y > 1.0f + margin) return false;
00283         if (clipSpacePos.z < -margin || clipSpacePos.z > 1.0f + margin) return false;
00284
00285         return true;
00286     }
00287
00288     bool hasChanged() override {
00289         return _cameraChange;
00290     }
00291
```

```
00292     void makeCameraDirty() override {
00293         _cameraChange = true;
00294     }
00295 
00296     void refreshCamera() override {
00297         _cameraChange = false;
00298     }
00299 
00300     // Function to cast a ray from screen coordinates into world space
00301     glm::vec3 castRayAt(const glm::vec2& screenPos) {
00302         // Convert screen position to normalized device coordinates (NDC)
00303         float x = (2.0f * screenPos.x) / _screenWidth - 1.0f;
00304         float y = 1.0f - (2.0f * screenPos.y) / _screenHeight;
00305         glm::vec4 clipCoords = glm::vec4(x, y, -1.0f, 1.0f);
00306 
00307         // Convert to eye space
00308         glm::vec4 eyeCoords = glm::inverse(_projectionMatrix) * clipCoords;
00309         eyeCoords = glm::vec4(eyeCoords.x, eyeCoords.y, -1.0f, 0.0f);
00310 
00311         // Convert to world space
00312         glm::vec3 worldRay = glm::vec3(glm::inverse(_viewMatrix) * eyeCoords);
00313         worldRay = glm::normalize(worldRay);
00314 
00315         return worldRay;
00316     }
00317     glm::vec3 getPointOnRayAtZ(const glm::vec3& rayOrigin, const glm::vec3& rayDirection, float
    desiredZ) {
00318         // Check if the ray is parallel to the z-plane (no intersection)
00319         if (rayDirection.z == 0.0f) {
00320             // Ray is parallel to the plane, no intersection
00321             return glm::vec3(std::numeric_limits<float>::infinity()); // Return invalid point
00322         }
00323 
00324         // Calculate t for the desired z value
00325         float t = (desiredZ - rayOrigin.z) / rayDirection.z;
00326 
00327         // Calculate the point on the ray
00328         glm::vec3 pointOnRay = rayOrigin + t * rayDirection;
00329 
00330         return pointOnRay;
00331     }
00332 
00333 private:
00334     int _screenWidth, _screenHeight;
00335     float _minScale = 0.1f, _maxScale = 5.0f;
00336     float _scale; // decreases when zoom-out
00337     bool _cameraChange;
00338 
00339     glm::vec2 _position;
00340     glm::mat4 _projectionMatrix; // changed once in init
00341     glm::mat4 _viewMatrix;
00342     glm::mat4 _cameraMatrix;
00343 };
```

6.19 ConsoleLogger.h

```
00001 #pragma once
00002 
00003 #include <iostream>
00004 #include <string>
00005 #include <ctime>
00006 
00007 namespace TazGraphEngine {
00008     class ConsoleLogger {
00009     public:
00010         static void log(const std::string& message) {
00011             printCurrentTime();
00012             std::cout « " " « message « std::endl;
00013         }
00014 
00015         static void error(const std::string& errorMessage) {
00016             printCurrentTime();
00017             std::cerr « " [ERROR] " « errorMessage « std::endl;
00018         }
00019 
00020     private:
00021         static void printCurrentTime() {
00022             std::time_t now = std::time(nullptr);
00023             struct tm timeInfo;
00024 
00025             #if defined(_WIN32) || defined(_WIN64)
00026                 localtime_s(&timeInfo, &now);
00027             #else
```

```
00028                 localtime_r(&now, &timeInfo);
00029             #endif
00030
00031             char timestamp[20]; // Sufficient space for the timestamp
00032             strftime(timestamp, sizeof(timestamp), "[%Y-%m-%d %H:%M:%S]", &timeInfo);
00033
00034             std::cout << timestamp;
00035         }
00036     };
00037 }
```

6.20 DataManager.h

```
00001 #pragma once
00002
00003 #include <map>
00004 #include <memory>
00005 #include <string>
00006
00007 class DataManager {
00008 public:
00009     // Gets the single instance of CameraManager (singleton)
00010     static DataManager& getInstance() {
00011         static DataManager instance; // Guaranteed to be destroyed. Instantiated on first use.
00012         return instance;
00013     }
00014
00015     DataManager() {}
00016
00017     std::string mapToLoad;
00018 };
```

6.21 Animation.h

```
00001 #pragma once
00002
00003 #include <string>
00004
00005 struct Animation //todo for now just add a bool hasFinished (useful for scripts) and much later may
      need onStop(when game paused)
00006 {                   // todo also we might need to have setReps/getReps and replace "play_once" with
      "play_n_times" where we pass 1 more value to call
00007     typedef enum {
00008         ANIMTYPE_NONE = 0, // just one animation
00009         ANIMTYPE_PLAY_N_TIMES = 1, // just iterates over the images one time. it holds the final image
      when finished.
00010         ANIMTYPE_LOOPED = 2, // going over the images again and again.
00011         ANIMTYPE_BACK_FORTH = 3  // iterate from index=0 to maxframe and back again. keeps holding the
      first image afterwards.
00012     } animType;
00013
00014     int indexX = 0; // initial position
00015     int indexY = 0;
00016     size_t total_frames = 0;
00017     float speed = 1.0f;
00018     animType type = animType::ANIMTYPE_NONE;
00019     int reps = 0;
00020
00021     int frame_times_played = 0;
00022     int cur_frame_index = 0;
00023     float cur_frame_index_f = 0;
00024     int times_played = 0;
00025
00026     int flow_direction = 1;
00027
00028     bool finished = false;
00029
00030     Animation()
00031     {
00032
00033     }
00034
00035     Animation(int ix, int iy , size_t f, float s, const std::string _type, int _reps = 0) // Animation
      frames look the next number of frames from the index
00036     {
00037         indexX = ix;
00038         indexY = iy;
00039         total_frames = f;
00040         speed = s;
```

```
00041
00042         type = _type == "play_n_times" ? ANIMTYPE_PLAY_N_TIMES :
00043             _type == "back_forth" ? ANIMTYPE_BACK_FORTH :
00044             _type == "looped" ? ANIMTYPE_LOOPED :
00045             ANIMTYPE_NONE;
00046         reps = _reps;
00047     }
00048
00049     Animation(int ix, int iy, size_t f, float s, const animType _type, int _reps = 0) // Animation
      frames look the next number of frames from the index
00050     {
00051         indexX = ix;
00052         indexY = iy;
00053         total_frames = f;
00054         speed = s;
00055
00056         type = _type;
00057         reps = _reps;
00058     }
00059
00060     void advanceFrame(float deltaTime) {
00061         unsigned short prev_frame_index = cur_frame_index;
00062
00063         switch (type) {
00064         case Animation::animType::ANIMTYPE_LOOPED:
00065         case Animation::animType::ANIMTYPE_PLAY_N_TIMES:
00066             cur_frame_index_f += speed * deltaTime;
00067             cur_frame_index = static_cast<unsigned short>(cur_frame_index_f);
00068
00069             // Check if the frame index has changed
00070             if (prev_frame_index != cur_frame_index) {
00071                 frame_times_played = 1;
00072             }
00073             else {
00074                 frame_times_played++;
00075             }
00076
00077             if (cur_frame_index > total_frames - 1) //essentially when we see that now we reach a
      frame out of total frames we reset it
00078             {
00079                 resetFrameIndex();
00080                 times_played++;
00081                 if (reps && times_played >= reps) {
00082                     finished = true;
00083                 }
00084             }
00085             break;
00086
00087         case Animation::animType::ANIMTYPE_BACK_FORTH:
00088             if (flow_direction == 1) {
00089                 cur_frame_index_f += speed * deltaTime;
00090
00091                 if (cur_frame_index_f > total_frames) {
00092                     cur_frame_index_f -= speed;
00093                     flow_direction = -1;
00094                 }
00095                 cur_frame_index = static_cast<unsigned short>(cur_frame_index_f);
00096             }
00097             else if (flow_direction == -1) {
00098                 if (cur_frame_index > 0) {
00099                     cur_frame_index_f -= speed * deltaTime;
00100                     cur_frame_index = static_cast<unsigned short>(cur_frame_index_f);
00101                 }
00102                 else {
00103                     times_played++;
00104                     flow_direction = 1;
00105                     resetFrameIndex();
00106                     if (reps && times_played >= reps) {
00107                         finished = true;
00108                     }
00109                 }
00110             }
00111             // Check if the frame index has changed
00112             if (prev_frame_index != cur_frame_index) {
00113                 frame_times_played = 1;
00114             }
00115             else {
00116                 frame_times_played++;
00117             }
00118             break;
00119
00120         case Animation::animType::ANIMTYPE_NONE:
00121             break;
00122         }
00123     }
00124
00125     void resetFrameIndex() {
```

```
00126         cur_frame_index = 0;
00127         cur_frame_index_f = 0;
00128         frame_times_played = 0;
00129     }
00130
00131     bool hasFinished() {
00132         return finished;
00133     }
00134 };
```

6.22 AnimatorComponent.h

```
00001 #pragma once
00002
00003 #include "../Components.h"
00004 #include <map>
00005 #include "Animation.h"
00006 #include "AnimatorManager.h"
00007 #include <functional>
00008
00009 typedef uint32_t timestamp;
00010
00011
00012 class AnimatorComponent : public Component //Animator -> Sprite -> Transform
00013 {
00014 public:
00015
00016     // onAction is the same thing as Play()
00017     // onFinish and onStart are for when we free Animator and when we initialize it
00018     // difference between this and from the lectures is that in lectures it uses seperate animator for
      each animation
00019
00020     SpriteComponent* sprite = nullptr;
00021     std::string textureid;
00022     std::string animationName = "";
00023     timestamp resumeTime = 0;
00024
00025     //std::map<const char*, Animation> animations; //Animator Manager
00026
00027     AnimatorComponent()
00028     {
00029
00030     }
00031
00032     AnimatorComponent(std::string id)
00033     {
00034         textureid = id;
00035     }
00036
00037     ~AnimatorComponent()
00038     {
00039
00040     }
00041
00042     void init() override
00043     {
00044         if (!entity->hasComponent<SpriteComponent>())
00045         {
00046             entity->addComponent<SpriteComponent>(textureid);
00047         }
00048         sprite = &entity->GetComponent<SpriteComponent>();
00049
00050         Play("P1Idle"); //onStart
00051         sprite->setTex(textureid);
00052     }
00053
00054     void update(float deltaTime) override //onAction
00055     {
00056         if (sprite->animation.hasFinished()) { // playing again animation
00057             sprite->animation.finished = false;
00058             sprite->animation.times_played = 0;
00059             resetAnimation();
00060         }
00061
00062         sprite->animation.advanceFrame(deltaTime);
00063         sprite->setCurrFrame();
00064     }
00065
00066     void draw(size_t e_index, PlaneModelRenderer&  batch, TazGraphEngine::Window& window) override
00067     {
00068         //sprite->draw(batch);
00069     }
00070
```

```
00071     void Play(std::string animName, int reps = 0)
00072     {
00073         AnimatorManager& animManager = AnimatorManager::getInstance();
00074         animationName = animName;
00075         sprite->SetAnimation(animManager.animations[animName].indexX,
      animManager.animations[animName].indexY,
00076             animManager.animations[animName].total_frames, animManager.animations[animName].speed,
00077             animManager.animations[animName].type,
00078             reps ? reps : animManager.animations[animName].reps );
00079     }
00080
00081     void resetAnimation() {
00082         AnimatorManager& animManager = AnimatorManager::getInstance();
00083         animationName = "P1Idle";
00084         sprite->SetAnimation(
00085             animManager.animations[animationName].indexX,
      animManager.animations[animationName].indexY,
00086             animManager.animations[animationName].total_frames,
      animManager.animations[animationName].speed,
00087             animManager.animations[animationName].type
00088         );
00089     }
00090
00091     std::string getPlayName()
00092     {
00093         return animationName;
00094     }
00095
00096     void DestroyTex()
00097     {
00098         sprite->DestroyTex();
00099     }
00100
00101 };
```

6.23 AnimatorManager.h

```
00001 #pragma once
00002
00003 #include <map>
00004 #include "Animation.h"
00005 #include "MovingAnimation.h"
00006 #include "FlashAnimation.h"
00007
00008
00009 struct AnimatorManager
00010 {
00011 public:
00012     std::map<std::string, Animation> animations;
00013     std::map<std::string, MovingAnimation> moving_animations;
00014     std::map<std::string, FlashAnimation> flash_animations;
00015
00016     AnimatorManager()
00017     {}
00018
00019     static AnimatorManager& getInstance() {
00020         static AnimatorManager instance;
00021         return instance;
00022     }
00023
00024     void InitializeAnimators()
00025     {
00026         Animation defaultAnimation = Animation(6, 2, 1, 0.04f, "looped");
00027
00028         animations.emplace("Default", defaultAnimation);
00029
00030         MovingAnimation defaultMoveAnimation = MovingAnimation(0, 0, 0, 0.0, "looped", 0, 0);
00031
00032         moving_animations.emplace("Default", defaultMoveAnimation);
00033
00034         FlashAnimation defaultFlashAnimation = FlashAnimation(0, 0, 3, 0.0f, "looped", { 0.2f, 1.0f,
      0.2f, 1.0f }, { 255,255,255,255 });
00035         FlashAnimation lineTransferFlashAnimation = FlashAnimation(0, 0, 3, 0.01f, "play_n_times", {
      0.01f, 0.01f, 0.00f, 0.01f }, { 255,255,255,255 }, 1);
00036         FlashAnimation rectangleInterpolationFlashAnimation = FlashAnimation(0, 0, 3, 0.01f,
      "back_forth", { 0.01f, 0.01f, 1.00f, 0.01f }, { 255,0,0,255 }, 0);
00037
00038         flash_animations.emplace("Default", defaultFlashAnimation);
00039         flash_animations.emplace("LineTransfer", lineTransferFlashAnimation);
00040         flash_animations.emplace("RectInterpolation", rectangleInterpolationFlashAnimation);
00041
00042
00043     }
00044 };
```

6.24 FlashAnimation.h

```
00001 #pragma once
00002
00003 #include "Animation.h"
00004
00005 #define NUM_LOOP_STATES 3
00006 #define NUM_BACK_FORTH_STATES 4
00007
00008
00009 struct FlashAnimation : public Animation //todo moving animation can be moving sprite with of without
      transform
00010 {
00011     enum class FlashState {
00012         FLASH_OUT,
00013         EASE_IN,
00014         FLASH_IN,
00015         EASE_OUT
00016     };
00017
00018     float interpolation_a = 0.0f;
00019     std::map<FlashState, float> speeds;
00020
00021     FlashState currentSpeedIndex = FlashState::FLASH_OUT;
00022     Color flashColor = Color(255,255,255,255);
00023
00024     FlashAnimation() : Animation()
00025     {
00026
00027     }
00028     // ix,iy is initial position (destX, destY), f is total frames to move, s is the speed to move
      frames, type as in animation, dx,dy distance to move
00029     FlashAnimation(int ix, int iy, size_t f, float s, const std::string _type,
00030         const std::vector<float>& flashTimes, Color flashC, int _reps = 0) : Animation(ix, iy, f, s,
      _type) // Animation frames look the next number of frames from the index
00031     {
00032         speeds[FlashState::FLASH_OUT] = flashTimes[0];
00033         speeds[FlashState::EASE_IN] = flashTimes[1];
00034         speeds[FlashState::FLASH_IN] = flashTimes[2];
00035         speeds[FlashState::EASE_OUT] = flashTimes[3];
00036         flashColor = flashC;
00037         reps = _reps;
00038     }
00039
00040     FlashAnimation(int ix, int iy, size_t f, float s, const animType _type,
00041         const std::vector<float>& flashTimes, Color flashC, int _reps = 0) : Animation(ix, iy, f, s,
      _type) // Animation frames look the next number of frames from the index
00042     {
00043         speeds[FlashState::FLASH_OUT] = flashTimes[0];
00044         speeds[FlashState::EASE_IN] = flashTimes[1];
00045         speeds[FlashState::FLASH_IN] = flashTimes[2];
00046         speeds[FlashState::EASE_OUT] = flashTimes[3];
00047         flashColor = flashC;
00048         reps = _reps;
00049     }
00050
00051     void advanceFrame(float deltaTime) {
00052         unsigned short prev_frame_index = cur_frame_index;
00053
00054         speed = speeds[currentSpeedIndex];
00055         switch (currentSpeedIndex) {
00056         case FlashState::FLASH_OUT:
00057             interpolation_a = 0;
00058             break;
00059         case FlashState::EASE_IN:
00060             // Using simple subtraction to find the fractional part
00061             interpolation_a = cur_frame_index_f - (int)cur_frame_index_f;
00062             break;
00063         case FlashState::FLASH_IN:
00064             interpolation_a = 1;
00065             break;
00066         case FlashState::EASE_OUT:
00067             // Using simple subtraction and inversion for the fractional part
00068             interpolation_a = 1 - (cur_frame_index_f - (int)cur_frame_index_f);
00069             break;
00070         }
00071         switch (type) {
00072         case Animation::animType::ANIMTYPE_BACK_FORTH:
00073
00074             cur_frame_index_f += speed * deltaTime;
00075             cur_frame_index = static_cast<unsigned short>(cur_frame_index_f);
00076
00077             if (cur_frame_index >= NUM_BACK_FORTH_STATES) {
00078                 times_played++;
00079                 resetFrameIndex();
00080                 if (reps && times_played >= reps) {
00081                     finished = true;
```

```
00082                 }
00083             }
00084             // Check if the frame index has changed
00085             if (prev_frame_index != cur_frame_index) {
00086                 frame_times_played = 1;
00087                 if ((static_cast<int>(currentSpeedIndex) + 1) % NUM_BACK_FORTH_STATES <
      NUM_BACK_FORTH_STATES)
00088                     currentSpeedIndex = static_cast<FlashState>((static_cast<int>(currentSpeedIndex) +
      1) % speeds.size());
00089             }
00090             else {
00091                 frame_times_played++;
00092             }
00093             break;
00094         case Animation::animType::ANIMTYPE_LOOPED:
00095         case Animation::animType::ANIMTYPE_PLAY_N_TIMES:
00096             cur_frame_index_f += speed * deltaTime;
00097             cur_frame_index = static_cast<unsigned short>(cur_frame_index_f);
00098 
00099             // Check if the frame index has changed
00100             if (prev_frame_index != cur_frame_index) {
00101                 frame_times_played = 1;
00102                 if( static_cast<int>(currentSpeedIndex) % NUM_LOOP_STATES < NUM_LOOP_STATES)
00103                     currentSpeedIndex = static_cast<FlashState>(static_cast<int>(currentSpeedIndex) %
      NUM_LOOP_STATES);
00104             }
00105             else {
00106                 frame_times_played++;
00107             }
00108 
00109             if (cur_frame_index >= NUM_LOOP_STATES) //essentially when we see that now we reach a
      frame out of total frames we reset it
00110             {
00111                 resetFrameIndex();
00112                 times_played++;
00113                 if (reps && times_played >= reps) {
00114                     finished = true;
00115                 }
00116             }
00117             break;
00118 
00119         case Animation::animType::ANIMTYPE_NONE:
00120             break;
00121         }
00122     }
00123 
00124     std::vector<float> getSpeedsAsVector() const {
00125         std::vector<float> speedsVector(4);  // Create a vector of size 4
00126 
00127         // Ensure the vector fills according to the FlashState order
00128         speedsVector[0] = speeds.at(FlashState::FLASH_OUT);  // Use at() for direct access with bounds
      checking
00129         speedsVector[1] = speeds.at(FlashState::EASE_IN);
00130         speedsVector[2] = speeds.at(FlashState::FLASH_IN);
00131         speedsVector[3] = speeds.at(FlashState::EASE_OUT);
00132 
00133         return speedsVector;
00134     }
00135 
00136 };
```

6.25 FlashAnimatorComponent.h

```
00001 #pragma once
00002 
00003 
00004 #include "../Components.h"
00005 #include <map>
00006 #include "Animation.h"
00007 #include "AnimatorManager.h"
00008 
00009 class FlashAnimatorComponent : public AnimatorComponent //Animator -> Sprite -> Transform
00010 {
00011 public:
00012     //std::map<const char*, Animation> animations; //Animator Manager
00013 
00014     FlashAnimatorComponent() : AnimatorComponent()
00015     {
00016 
00017     }
00018 
00019     FlashAnimatorComponent(std::string id) : AnimatorComponent(id)
00020     {
```

```
00021
00022     }
00023
00024     ~FlashAnimatorComponent()
00025     {
00026
00027     }
00028
00029     void init() override
00030     {
00031         if (!entity->hasComponent<SpriteComponent>())
00032         {
00033             entity->addComponent<SpriteComponent>(textureid);
00034         }
00035         sprite = &entity->GetComponent<SpriteComponent>();
00036
00037         Play("Default");
00038         sprite->setTex(textureid);
00039     }
00040
00041     void update(float deltaTime) override
00042     {
00043         if (sprite->flash_animation.hasFinished()) { // playing again animation
00044             sprite->flash_animation.finished = false;
00045             sprite->flash_animation.times_played = 0;
00046             resetAnimation();
00047         }
00048
00049         sprite->flash_animation.advanceFrame(deltaTime);
00050         sprite->setFlashFrame();
00051     }
00052
00053     void draw(size_t e_index, PlaneModelRenderer&  batch, TazGraphEngine::Window& window) override
00054     {
00055         //sprite->draw(batch);
00056     }
00057
00058     void Play(std::string animName, int reps = 0)
00059     {
00060         AnimatorManager& animManager = AnimatorManager::getInstance();
00061         animationName = animName;
00062         sprite->SetFlashAnimation(
00063             animManager.flash_animations[animationName].indexX,
      animManager.flash_animations[animationName].indexY,
00064             animManager.flash_animations[animationName].total_frames,
      animManager.flash_animations[animationName].speed,
00065             animManager.flash_animations[animationName].type,
00066             animManager.flash_animations[animationName].getSpeedsAsVector(),
00067             animManager.flash_animations[animationName].flashColor,
00068             reps ? reps : animManager.flash_animations[animationName].reps
00069         );
00070     }
00071
00072     void resetAnimation() {
00073         AnimatorManager& animManager = AnimatorManager::getInstance();
00074         animationName = "Default";
00075         sprite->SetFlashAnimation(
00076             animManager.flash_animations[animationName].indexX,
      animManager.flash_animations[animationName].indexY,
00077             animManager.flash_animations[animationName].total_frames,
      animManager.flash_animations[animationName].speed,
00078             animManager.flash_animations[animationName].type,
00079             animManager.flash_animations[animationName].getSpeedsAsVector(),
00080             animManager.flash_animations[animationName].flashColor);
00081     }
00082
00083     std::string getPlayName()
00084     {
00085         return animationName;
00086     }
00087
00088     void DestroyTex()
00089     {
00090         sprite->DestroyTex();
00091     }
00092
00093 };
```

6.26 LineFlashAnimatorComponent.h

```
00001 #pragma once
00002
00003
```

```
00004 #include "../../Components.h"
00005 #include <map>
00006 #include "../Animation.h"
00007 #include "../AnimatorManager.h"
00008
00009
00010 typedef uint32_t timestamp;
00011
00012
00013 class LineFlashAnimatorComponent : public LinkComponent //Animator -> Sprite -> Transform
00014 {
00015 public:
00016
00017     Line_w_Color* line = nullptr;
00018     std::string animationName = "";
00019     timestamp resumeTime = 0;
00020
00021     LineFlashAnimatorComponent()
00022     {
00023
00024     }
00025
00026     ~LineFlashAnimatorComponent()
00027     {
00028
00029     }
00030
00031     void init() override
00032     {
00033         if (!entity->hasComponent<Line_w_Color>())
00034         {
00035             entity->addComponent<Line_w_Color>();
00036         }
00037         line = &entity->GetComponent<Line_w_Color>();
00038
00039         Play("Default");
00040     }
00041
00042     void update(float deltaTime) override
00043     {
00044         if (animationName == "Default") {
00045             return;
00046         }
00047         if (line->flash_animation.hasFinished()) { // playing again animation
00048             line->flash_animation.finished = false;
00049             line->flash_animation.times_played = 0;
00050             resetAnimation();
00051         }
00052
00053         line->flash_animation.advanceFrame(deltaTime);
00054         line->setFlashFrame();
00055     }
00056
00057     void draw(size_t e_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window) override
00058     {
00059         //sprite->draw(batch);
00060     }
00061
00062     void Play(const char* animName, int reps = 0)
00063     {
00064         AnimatorManager& animManager = AnimatorManager::getInstance();
00065         animationName = animName;
00066         line->SetFlashAnimation(
00067             animManager.flash_animations[animationName].indexX,
      animManager.flash_animations[animationName].indexY,
00068             animManager.flash_animations[animationName].total_frames,
      animManager.flash_animations[animationName].speed,
00069             animManager.flash_animations[animationName].type,
00070             animManager.flash_animations[animationName].getSpeedsAsVector(),
00071             animManager.flash_animations[animationName].flashColor,
00072             reps ? reps : animManager.flash_animations[animationName].reps
00073         );
00074     }
00075
00076     void resetAnimation() {
00077         AnimatorManager& animManager = AnimatorManager::getInstance();
00078         animationName = "Default";
00079         line->SetFlashAnimation(
00080             animManager.flash_animations[animationName].indexX,
      animManager.flash_animations[animationName].indexY,
00081             animManager.flash_animations[animationName].total_frames,
      animManager.flash_animations[animationName].speed,
00082             animManager.flash_animations[animationName].type,
00083             animManager.flash_animations[animationName].getSpeedsAsVector(),
00084             animManager.flash_animations[animationName].flashColor);
00085     }
00086
```

```
00087     std::string getPlayName()
00088     {
00089         return animationName;
00090     }
00091
00092 };
```

6.27 MovingAnimation.h

```
00001 #pragma once
00002
00003 #include "Animation.h"
00004 #include <vector>
00005 #include <glm/glm.hpp>
00006 #include <stdexcept>
00007
00008 class MovingAnimation : public Animation //todo moving animation can be moving sprite with of without
      transform
00009 {
00010 public:
00011     // have a vector for the positions
00012     // when given only dx and dy then add that to the same vector as first element
00013     // sprite will check bool if it is about continious
00014     std::vector<glm::vec2> positions;  // Stores the positions for the animation
00015     std::vector<int> zIndices;         // zIndex for each position
00016     std::vector<int> rotations;
00017
00018     MovingAnimation() : Animation()
00019     {
00020
00021     }
00022     // ix,iy is initial position (destX, destY), f is total frames to move, s is the speed to move
      frames, type as in animation, dx,dy distance to move
00023     MovingAnimation(int ix, int iy, int f, float s, const std::string _type, int dx, int dy, int _reps
      = 0) : Animation(ix, iy, f, s, _type) // Animation frames look the next number of frames from the
      index
00024     {
00025         positions.clear();
00026         zIndices.clear();
00027         rotations.clear();
00028
00029         glm::vec2 distanceXY(dx, dy);
00030         positions.push_back(distanceXY);
00031         zIndices.push_back(0);
00032         rotations.push_back(0);
00033
00034         reps = _reps;
00035     }
00036
00037     MovingAnimation(int ix, int iy, int f, float s, const animType _type, int dx, int dy, int _reps =
      0) : Animation(ix, iy, f, s, _type) // Animation frames look the next number of frames from the index
00038     {
00039         positions.clear();
00040         zIndices.clear();
00041         rotations.clear();
00042
00043         glm::vec2 distanceXY(dx, dy);
00044         positions.push_back(distanceXY);
00045         zIndices.push_back(0);
00046         rotations.push_back(0);
00047
00048         reps = _reps;
00049     }
00050
00051     MovingAnimation(
00052         int ix, int iy, int f, float s, const std::string _type, const std::vector<glm::vec2>&
      _positions, const std::vector<int>& _zIndices, const std::vector<int>& _rotations, int _reps = 0) //
      Animation frames look the next number of frames from the index
00053     {
00054         positions.clear();
00055         zIndices.clear();
00056         rotations.clear();
00057
00058         positions = _positions;
00059         zIndices = _zIndices;
00060         rotations = _rotations;
00061
00062         Animation(ix, iy, positions.size(), s, _type);
00063
00064         reps = _reps;
00065         validateVectors();
00066     }
00067
```

```
00068     MovingAnimation(
00069         int ix, int iy, size_t f, float s, const animType _type, const std::vector<glm::vec2>&
      _positions, const std::vector<int>& _zIndices, const std::vector<int>& _rotations, int _reps = 0)  //
      Animation frames look the next number of frames from the index
00070     {
00071         positions.clear();
00072         zIndices.clear();
00073         rotations.clear();
00074 
00075         positions = _positions;
00076         zIndices = _zIndices;
00077         rotations = _rotations;
00078 
00079         Animation(ix, iy, positions.size(), s, _type);
00080 
00081         reps = _reps;
00082         validateVectors();
00083     }
00084 
00085 private:
00086     void validateVectors() const {
00087         // Ensure all vectors are of the same length to prevent indexing errors
00088         if (positions.size() != zIndices.size() || positions.size() != rotations.size()) {
00089             throw std::invalid_argument("All vectors (positions, zIndices, rotations) must have the
      same length.");
00090         }
00091     }
00092 };
```

6.28 MovingAnimatorComponent.h

```
00001 #pragma once
00002 
00003 
00004 #include "../Components.h"
00005 #include <map>
00006 #include "Animation.h"
00007 #include "AnimatorManager.h"
00008 // TODO: in comparison to AnimatorComponent, here we also have access to sprite so create functions
      like SetDx,SetDy,SetRepsOfMove,SetDelayOfMove, IsForeverRepeatedMove
00009 
00010 class MovingAnimatorComponent : public AnimatorComponent //Animator -> Sprite -> Transform
00011 {
00012 public:
00013     //std::map<const char*, Animation> animations; //Animator Manager
00014 
00015     MovingAnimatorComponent() : AnimatorComponent()
00016     {
00017 
00018     }
00019 
00020     MovingAnimatorComponent(std::string id) : AnimatorComponent(id)
00021     {
00022 
00023     }
00024 
00025     ~MovingAnimatorComponent()
00026     {
00027 
00028     }
00029 
00030     void init() override
00031     {
00032         if (!entity->hasComponent<SpriteComponent>())
00033         {
00034             entity->addComponent<SpriteComponent>(textureid);
00035         }
00036         sprite = &entity->GetComponent<SpriteComponent>();
00037 
00038         Play("Default");
00039         sprite->setTex(textureid);
00040     }
00041 
00042     void update(float deltaTime) override
00043     {
00044         if (sprite->moving_animation.hasFinished()) { // playing again animation
00045             sprite->moving_animation.finished = false;
00046             sprite->moving_animation.times_played = 0;
00047             resetAnimation();
00048         }
00049 
00050         sprite->transform->setPosition_X(sprite->transform->getPosition().x); //make player move with
      the camera, being stable in centre, except on edges
```

```
00051         sprite->transform->setPosition_Y(sprite->transform->getPosition().y);
00052
00053         sprite->moving_animation.advanceFrame(deltaTime);
00054         if (sprite->moving_animation.positions.size() == 1) {
00055             sprite->setMoveFrame();
00056         }
00057         else {
00058             sprite->setSpecificMoveFrame();
00059         }
00060     }
00061
00062     void draw(size_t e_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window) override
00063     {
00064         //sprite->draw();
00065     }
00066
00067     void Play(const char* animName, int reps = 0)
00068     {
00069         AnimatorManager& animManager = AnimatorManager::getInstance();
00070         animationName = animName;
00071         sprite->SetMovingAnimation(
00072             animManager.moving_animations[animationName].indexX,
      animManager.moving_animations[animationName].indexY,
00073             animManager.moving_animations[animationName].total_frames,
      animManager.moving_animations[animationName].speed,
00074             animManager.moving_animations[animationName].type,
00075             animManager.moving_animations[animationName].positions,
      animManager.moving_animations[animationName].zIndices,
      animManager.moving_animations[animationName].rotations, // here needs to be vector
00076             reps ? reps : animManager.moving_animations[animationName].reps
00077         );
00078     }
00079
00080     void resetAnimation() {
00081         AnimatorManager& animManager = AnimatorManager::getInstance();
00082         animationName = "Default";
00083         sprite->SetMovingAnimation(
00084             animManager.moving_animations[animationName].indexX,
      animManager.moving_animations[animationName].indexY,
00085             animManager.moving_animations[animationName].total_frames,
      animManager.moving_animations[animationName].speed,
00086             animManager.moving_animations[animationName].type,
00087             animManager.moving_animations[animationName].positions,
      animManager.moving_animations[animationName].zIndices,
      animManager.moving_animations[animationName].rotations);
00088     }
00089
00090     std::string getPlayName()
00091     {
00092         return animationName;
00093     }
00094
00095     void DestroyTex()
00096     {
00097         sprite->DestroyTex();
00098     }
00099
00100 };
```

6.29 RectangleFlashAnimatorComponent.h

```
00001 #pragma once
00002
00003
00004 #include "../../Components.h"
00005 #include <map>
00006 #include "../Animation.h"
00007 #include "../AnimatorManager.h"
00008
00009
00010 typedef uint32_t timestamp;
00011
00012
00013 class RectangleFlashAnimatorComponent : public Component //Animator -> Sprite -> Transform
00014 {
00015 public:
00016
00017     Rectangle_w_Color* rectangle = nullptr;
00018     std::string animationName = "";
00019     timestamp resumeTime = 0;
00020
00021     RectangleFlashAnimatorComponent()
00022     {
```

```
00023
00024     }
00025
00026     ~RectangleFlashAnimatorComponent()
00027     {
00028
00029     }
00030
00031     void init() override
00032     {
00033         if (!entity->hasComponent<Rectangle_w_Color>())
00034         {
00035             entity->addComponent<Rectangle_w_Color>();
00036         }
00037         rectangle = &entity->GetComponent<Rectangle_w_Color>();
00038
00039         Play("Default");
00040     }
00041
00042     void update(float deltaTime) override
00043     {
00044         if (animationName == "Default") {
00045             return;
00046         }
00047         if (rectangle->flash_animation.hasFinished()) { // playing again animation
00048             rectangle->flash_animation.finished = false;
00049             rectangle->flash_animation.times_played = 0;
00050             resetAnimation();
00051         }
00052
00053         rectangle->flash_animation.advanceFrame(deltaTime);
00054         rectangle->setFlashFrame();
00055     }
00056
00057     void draw(size_t e_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window) override
00058     {
00059         //sprite->draw(batch);
00060     }
00061
00062     void Play(const std::string& animName, int reps = 0)
00063     {
00064         AnimatorManager& animManager = AnimatorManager::getInstance();
00065         animationName = animName;
00066         FlashAnimation& flash_animation = animManager.flash_animations[animationName];
00067         rectangle->SetFlashAnimation(
00068             flash_animation.indexX, flash_animation.indexY,
00069             flash_animation.total_frames, flash_animation.speed,
00070             flash_animation.type,
00071             flash_animation.getSpeedsAsVector(),
00072             flash_animation.flashColor,
00073             reps ? reps : flash_animation.reps
00074         );
00075     }
00076
00077     void resetAnimation() {
00078         AnimatorManager& animManager = AnimatorManager::getInstance();
00079         animationName = "Default";
00080         rectangle->SetFlashAnimation(
00081             animManager.flash_animations[animationName].indexX,
      animManager.flash_animations[animationName].indexY,
00082             animManager.flash_animations[animationName].total_frames,
      animManager.flash_animations[animationName].speed,
00083             animManager.flash_animations[animationName].type,
00084             animManager.flash_animations[animationName].getSpeedsAsVector(),
00085             animManager.flash_animations[animationName].flashColor);
00086     }
00087
00088     std::string getPlayName()
00089     {
00090         return animationName;
00091     }
00092
00093 };
```

6.30 Components.h

```
00001 #pragma once
00002
00003 #include "./Core/GECSEntity.h"
00004
00005 #include "./Components/Empty/Basic/TransformComponent.h"
00006 #include "./Components/Empty/Basic/SpriteComponent.h"
00007 #include "./Components/Empty/Basic/Triangle_w_Color.h"
```

```
00008 #include "./Components/Empty/Basic/Rectangle_w_Color.h"
00009 #include "./Components/Empty/Basic/BoxComponent.h"
00010 #include "./Components/Empty/Basic/SphereComponent.h"
00011 #include "./Components/Link/Basic/Line_w_Color.h"
00012 #include "./Components/Link/Basic/SpringComponent.h"
00013
00014
00015 #include "./Animators/AnimatorComponent.h"
00016 #include "./Animators/MovingAnimatorComponent.h"
00017 #include "./Animators/FlashAnimatorComponent.h"
00018
00019 #include "./Animators/LineAnimators/LineFlashAnimatorComponent.h"
00020 #include "./Animators/RectangleAnimators/RectangleFlashAnimatorComponent.h"
00021
00022
00023
00024
```

6.31 BoxComponent.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
00004
00005 class BoxComponent : public Component
00006 {
00007 public:
00008     Color color = { 255, 255, 255, 255 };
00009
00010     SDL_Rect destRect;
00011     TransformComponent* transform = nullptr;
00012
00013     float temp_rotation = 0.0f;
00014
00015     BoxComponent()
00016     {
00017
00018     }
00019
00020
00021     ~BoxComponent() {
00022
00023     }
00024
00025     void init() override {
00026         transform = &entity->GetComponent<TransformComponent>();
00027
00028         destRect.w = transform->size.x * transform->scale;
00029         destRect.h = transform->size.y * transform->scale;
00030     }
00031
00032     void update(float deltaTime) override {
00033         destRect.x = static_cast<int>(transform->getPosition().x); //make player move with the camera,
      being stable in centre, except on edges
00034         destRect.y = static_cast<int>(transform->getPosition().y);
00035         destRect.w = transform->size.x * transform->scale;
00036         destRect.h = transform->size.y * transform->scale;
00037
00038         temp_rotation += 0.1f;
00039     }
00040
00041     void draw(size_t v_index, PlaneColorRenderer& batch, TazGraphEngine::Window& window ) {
00042         batch.drawBox(v_index, transform->size, transform->bodyCenter, transform->rotation,color);
00043     }
00044
00045     void draw(size_t v_index, LightRenderer& batch, TazGraphEngine::Window& window) {
00046         batch.drawBox(v_index, transform->size, transform->bodyCenter, transform->rotation, color);
00047     }
00048
00049     std::string GetComponentName() override {
00050         return "BoxComponent";
00051     }
00052 };
```

6.32 Rectangle_w_Color.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
```

```
00004
00005 class Rectangle_w_Color : public Component
00006 {
00007 public:
00008     Color default_color = { 255, 255, 255, 255 };
00009
00010     Color color = { 255, 255, 255, 255 };
00011
00012     TransformComponent* transform = nullptr;
00013
00014     FlashAnimation flash_animation;
00015
00016     Rectangle_w_Color()
00017     {
00018
00019     }
00020
00021
00022     ~Rectangle_w_Color() {
00023
00024     }
00025
00026     void init() override {
00027         transform = &entity->GetComponent<TransformComponent>();
00028     }
00029
00030     void update(float deltaTime) override {
00031     }
00032
00033     //void draw(size_t v_index, PlaneModelRenderer&  batch, TazGraphEngine::Window& window) {
00034     //  glm::vec3 pos((float)destRect.x, (float)destRect.y, transform->getPosition().z);
00035     //  glm::vec2 size = glm::vec2((float)destRect.w, (float)destRect.h);
00036     //  batch.draw(v_index, pos, size, transform->rotation, glm::vec4(-1.0f, -1.0f, 2.0f, 2.0f), 0,
    transform->bodyCenter, color); // 0 is for texture
00037     //}
00038
00039     void draw(size_t v_index, PlaneColorRenderer& batch, TazGraphEngine::Window& window) {
00040         glm::vec2 size((float)transform->size.x, (float)transform->size.y);
00041         batch.draw(v_index, size, transform->bodyCenter, transform->rotation, color);
00042     }
00043
00044     void setColor(Color clr) {
00045         default_color = clr;
00046         color = clr;
00047     }
00048
00049     void SetFlashAnimation(int idX, int idY, size_t fr, float sp, const Animation::animType type,
    const std::vector<float>& flashTimes, Color flashC, int reps = 0)
00050     {
00051         flash_animation = FlashAnimation(idX, idY, fr, sp, type, flashTimes, flashC, reps);
00052     }
00053
00054     void setFlashFrame() {
00055         float t = this->flash_animation.interpolation_a;
00056         this->color = Color::fromVec4(glm::mix(default_color.toVec4(),
    this->flash_animation.flashColor.toVec4(), t));
00057
00058
00059     }
00060
00061
00062     std::string GetComponentName() override {
00063         return "Rectangle_w_Color";
00064     }
00065
00066     void showGUI() override {
00067         ImGui::Separator();
00068
00069         ImVec4 a_color = ImVec4(color.r / 255.0f, color.g / 255.0f, color.b / 255.0f, color.a /
    255.0f);
00070         if(ImGui::ColorPicker4("Color", (float*)&a_color)) {
00071             Color newColor = {
00072                         (GLubyte)(a_color.x * 255),
00073                         (GLubyte)(a_color.y * 255),
00074                         (GLubyte)(a_color.z * 255),
00075                         (GLubyte)(a_color.w * 255)
00076             };
00077
00078             color = newColor;
00079         }
00080
00081     }
00082 };
```

6.33 SphereComponent.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
00004
00005 class SphereComponent : public Component
00006 {
00007 public:
00008     Color color = { 255, 255, 255, 255 };
00009
00010     TransformComponent* transform = nullptr;
00011
00012     SphereComponent()
00013     {
00014
00015     }
00016
00017
00018     ~SphereComponent() {
00019
00020     }
00021
00022     void init() override {
00023         transform = &entity->GetComponent<TransformComponent>();
00024     }
00025
00026     void update(float deltaTime) override {
00027     }
00028
00029     void draw(size_t v_index, PlaneColorRenderer& batch, TazGraphEngine::Window& window) {
00030         batch.drawSphere(v_index, transform->size, transform->bodyCenter, transform->rotation, color);
00031     }
00032
00033     void draw(size_t v_index, LightRenderer& batch, TazGraphEngine::Window& window) {
00034         batch.drawSphere(v_index, transform->size, transform->bodyCenter, transform->rotation, color);
00035     }
00036
00037     std::string GetComponentName() override {
00038         return "SphereComponent";
00039     }
00040 };
```

6.34 SpriteComponent.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
00004 #include <SDL2/SDL.h>
00005 #include "GL/glew.h"
00006 #include "../../../../TextureManager/TextureManager.h"
00007 #include "../../../Animators/Animation.h"
00008 #include "../../../Animators/MovingAnimation.h"
00009 #include "../../../Animators/FlashAnimation.h"
00010 #include <map>
00011 #include "../../../../Vertex.h"
00012 #include <cstddef>
00013
00014
00015 // TODO: (extra): can add states for different states (0 for full solid tile or 1 for no solid
00016 class SpriteComponent : public Component //sprite -> transform
00017 {
00018 private:
00019     const GLTexture *gl_texture = nullptr;
00020     bool _isMainMenu = false;
00021
00022 public:
00023     std::string texture_name = "";
00024     Color default_color = { 255, 255, 255, 255 };
00025     Color color = { 255, 255, 255, 255 };
00026
00027     TransformComponent* transform = nullptr;
00028     SDL_FRect srcRect = {0,0,0,0};
00029
00030     Animation animation;
00031     MovingAnimation moving_animation;
00032     FlashAnimation flash_animation;
00033
00034     SDL_RendererFlip spriteFlip = SDL_FLIP_NONE;
00035
00036     SpriteComponent() = default;
00037     SpriteComponent(std::string id)
00038     {
```

```
00039         setTex(id);
00040     }
00041     SpriteComponent(Color clr)
00042     {
00043         default_color = clr;
00044         color = clr;
00045     }
00046 
00047     SpriteComponent(std::string id, bool isMainMenu)
00048     {
00049         _isMainMenu = isMainMenu;
00050         setTex(id);
00051     }
00052 
00053     ~SpriteComponent()
00054     {
00055     }
00056 
00057     void setTex(std::string id) //this function is used to change texture of a sprite
00058     {
00059         texture_name = id;
00060         gl_texture = TextureManager::getInstance().Get_GLTexture(id);
00061         srcRect.w = (float)gl_texture->width;
00062         srcRect.h = (float)gl_texture->height;
00063     }
00064 
00065     void init() override
00066     {
00067         if (!entity->hasComponent<TransformComponent>())
00068         {
00069             entity->addComponent<TransformComponent>();
00070         }
00071         transform = &entity->GetComponent<TransformComponent>();
00072 
00073         srcRect.x = srcRect.y = 0;
00074         srcRect.w = transform->size.x;
00075         srcRect.h = transform->size.y;
00076 
00077     }
00078 
00079     void update(float deltaTime) override
00080     {
00081     }
00082 
00083     void draw(size_t v_index, PlaneModelRenderer&  batch, TazGraphEngine::Window& window)
00084     {
00085         if (gl_texture == NULL)
00086         {
00087             return;
00088         }
00089 
00090         float screenScale = window.getScale();
00091 
00092         glm::vec3 pos(
00093             transform->getPosition().x * screenScale,
00094             transform->getPosition().y * screenScale,
00095             transform->getPosition().z);
00096 
00097         glm::vec2 size(
00098             transform->size.x * transform->scale * screenScale,
00099             transform->size.y * transform->scale * screenScale);
00100 
00101         float srcUVposX = spriteFlip == SDL_FLIP_HORIZONTAL ?
00102             (srcRect.x + srcRect.w) / gl_texture->width :
00103             srcRect.x / gl_texture->width;
00104         float srcUVposY = (srcRect.y) / gl_texture->height;
00105 
00106         float srcUVw = spriteFlip == SDL_FLIP_HORIZONTAL ?
00107             -srcRect.w / gl_texture->width :
00108             srcRect.w / gl_texture->width;
00109         float srcUVh = srcRect.h / gl_texture->height;
00110 
00111         glm::vec4 uv(srcUVposX, srcUVposY, srcUVw, srcUVh);
00112 
00113         batch.draw(v_index, size, transform->bodyCenter, transform->rotation, uv, gl_texture->id);
00114 
00115         glDisableVertexAttribArray(0);
00116         glDisableVertexAttribArray(1);
00117         glDisableVertexAttribArray(2);
00118 
00119         glBindBuffer(GL_ARRAY_BUFFER, 0);
00120 
00121         /*glBindTexture(GL_TEXTURE_2D, 0);*/
00122     }
00123 
00124     void SetAnimation(int idX, int idY, size_t fr, float sp, const Animation::animType type, int reps
      = 0)
```

```
00125     {
00126         animation = Animation(idX, idY, fr, sp, type, reps);
00127     }
00128
00129     void SetMovingAnimation(int idX, int idY, size_t fr, float sp, const Animation::animType type,
      const std::vector<glm::vec2>& _positions, const std::vector<int>& _zIndices, const std::vector<int>&
      _rotations, int reps = 0)
00130     {
00131         moving_animation = MovingAnimation(idX, idY, fr, sp, type, _positions, _zIndices, _rotations,
      reps); // dx,dy needs to be vector, if yes then dont need int dx dy
00132     }
00133
00134     void SetFlashAnimation(int idX, int idY, size_t fr, float sp, const Animation::animType type,
      const std::vector<float>& flashTimes, Color flashC, int reps = 0)
00135     {
00136         flash_animation = FlashAnimation(idX, idY, fr, sp, type, flashTimes, flashC, reps);
00137     }
00138
00139     void setCurrFrame() {
00140         this->srcRect.x = (this->animation.indexX * this->transform->size.x) /* init */ + (
      this->srcRect.w * animation.cur_frame_index/* curframe from total frams */);
00141         this->srcRect.y = this->animation.indexY * this->transform->size.y;
00142     }
00143
00144     void setMoveFrame() {
00145
00146         this->transform->setPosition_X(((this->transform->getPosition().x) +
      this->moving_animation.indexX) /* init */ + (this->moving_animation.positions[0].x *
      moving_animation.cur_frame_index));
00147         this->transform->setPosition_Y(((this->transform->getPosition().y) +
      this->moving_animation.indexY) + (this->moving_animation.positions[0].y *
      moving_animation.cur_frame_index));
00148     }
00149
00150     void setSpecificMoveFrame() {
00151
00152         this->transform->setPosition_X(((this->transform->getPosition().x) +
      this->moving_animation.indexX) /* init */ +
      (this->moving_animation.positions[moving_animation.cur_frame_index].x));
00153         this->transform->setPosition_Y(((this->transform->getPosition().y) +
      this->moving_animation.indexY) +
      (this->moving_animation.positions[moving_animation.cur_frame_index].y));
00154     }
00155
00156     void setFlashFrame() {
00157         this->color = this->flash_animation.flashColor * this->flash_animation.interpolation_a
00158             + default_color * (1 - this->flash_animation.interpolation_a);
00159     }
00160
00161     void DestroyTex()
00162     {
00163         //TextureManager::DestroyTexture(texture);
00164         gl_texture = nullptr;
00165     }
00166     void DestroyGlTex()
00167     {
00168         gl_texture = NULL;
00169     }
00170
00171     std::string GetComponentName() override {
00172         return "SpriteComponent";
00173     }
00174
00175     void showGUI() override {
00176         ImGui::Separator();
00177
00178         // Get the list of texture names
00179         std::vector<std::string> textureNames = TextureManager::getInstance().Get_GLTextureNames();
00180
00181         static int currentItem = 0;
00182         if (!textureNames.empty()) {
00183             std::vector<const char*> items;
00184             for (const std::string& name : textureNames)
00185                 items.push_back(name.c_str());
00186
00187             if (ImGui::Combo("Textures", &currentItem, items.data(), (int)items.size())) {
00188                 if (!textureNames[currentItem].empty()) {
00189                     setTex(textureNames[currentItem]);
00190                 }
00191             }
00192         }
00193     };
00194 };
```

6.35 TransformComponent.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
00004
00005 class TransformComponent : public Component //transform as in graphics, we have rotation and scale
00006 {
00007 public:
00008     std::string temp_lineParsed = "";
00009
00010     glm::vec3 velocity = glm::vec3(0);
00011     glm::vec3 rotation = { 0.0f,0.0f,0.0f };
00012     glm::vec3 position = glm::vec3(0);
00013     glm::vec3 size = glm::vec3(0);
00014
00015     glm::vec3 last_position = glm::vec3(0);
00016     glm::vec3 last_size = glm::vec3(0);
00017     glm::vec3 last_velocity = glm::vec3(0);
00018
00019
00020     glm::vec3 bodyCenter = { 0.0f,0.0f,0.0f };
00021
00022     float scale = 1;
00023
00024     int speed = 1;
00025
00026     TransformComponent()
00027     {
00028     }
00029
00030     TransformComponent(float sc)
00031     {
00032         scale = sc;
00033     }
00034
00035     TransformComponent(glm::vec2 m_position)
00036     {
00037         position.x = m_position.x;
00038         position.y = m_position.y;
00039         bodyCenter = position + (size / 2.0f);
00040     }
00041
00042     TransformComponent(glm::vec3 m_position)
00043     {
00044         position = m_position;
00045         bodyCenter = position + (size / 2.0f);
00046     }
00047
00048     TransformComponent(glm::vec2 m_position, layer layer, glm::vec2 m_size, float sc) :
      TransformComponent(m_position) {
00049         position = { m_position.x, m_position.y, getLayerDepth(layer) };
00050         size = { m_size.x, m_size.y, 0.0f };
00051         scale = sc;
00052         bodyCenter = position + (size / 2.0f);
00053     }
00054
00055     TransformComponent(glm::vec2 m_position, layer layer, glm::vec2 size, float sc, int sp) :
      TransformComponent(m_position, layer, size, sc)
00056     {
00057         speed = sp;
00058     }
00059
00060     TransformComponent(glm::vec2 m_position, layer layer , glm::vec3 m_size, float sc) :
      TransformComponent(m_position){
00061         size = m_size;
00062         scale = sc;
00063     }
00064
00065     TransformComponent(glm::vec2 m_position, layer layer, glm::vec3 size, float sc, int sp) :
      TransformComponent(m_position, layer, size, sc)
00066     {
00067         speed = sp;
00068     }
00069
00070     TransformComponent(glm::vec3 m_position, glm::vec3 m_size, float sc) :
      TransformComponent(m_position) {
00071         size = m_size;
00072         scale = sc;
00073     }
00074
00075     void init() override
00076     {
00077     }
00078     void update(float deltaTime) override
00079     {
00080
```

```
00081         if (entity->getParentEntity() && dynamic_cast<NodeEntity*>(entity->getParentEntity())) {
00082             Entity* parent = entity->getParentEntity();
00083             TransformComponent* parentTR = &parent->GetComponent<TransformComponent>();
00084             if (
00085                 parentTR->position == parentTR->last_position
00086                 && parentTR->size == parentTR->last_size
00087                 && parentTR->velocity == parentTR->last_velocity
00088                 ) {
00089                 return;
00090             }
00091 
00092             bodyCenter = parentTR->getCenterTransform() + position;
00093 
00094         }
00095         else {
00096             bodyCenter = position + (size / 2.0f);
00097         }
00098 
00099         if (position == last_position && size == last_size && velocity == last_velocity) {
00100             return;
00101         }
00102 
00103         entity->cellUpdate();
00104 
00105         last_position = position;
00106         last_size = size;
00107         last_velocity = velocity;
00108 
00109         position.x += velocity.x * speed * deltaTime;
00110         position.y += velocity.y * speed * deltaTime;
00111 
00112         velocity *= 0.98f;
00113 
00114 
00115         //todo dont update the children on every iteration
00116         // todo do this for component when needed
00117 
00118     }
00119 
00120     glm::vec3 getCenterTransform()
00121     {
00122         return bodyCenter;
00123     }
00124 
00125     glm::vec3 getSizeCenter() {
00126         return glm::vec3(size.x * scale / 2, size.y * scale / 2, size.z * scale / 2);
00127     }
00128 
00129     glm::vec3 getPosition() {
00130         return position;
00131     }
00132 
00133     void setPosition_X(float newPosition_X) {
00134         position.x = newPosition_X;
00135     }
00136     void setPosition_Y(float newPosition_Y) {
00137         position.y = newPosition_Y;
00138     }
00139 
00140     glm::vec3 getVelocity() {
00141         return velocity;
00142     }
00143 
00144     void setVelocity_X(float newVelocity_X) {
00145         velocity.x = newVelocity_X;
00146     }
00147     void setVelocity_Y(float newVelocity_Y) {
00148         velocity.y = newVelocity_Y;
00149     }
00150 
00151     void setRotation(glm::vec3 m_rotation) {
00152         rotation = m_rotation;
00153     }
00154 
00155     std::string GetComponentName() override {
00156         return "TransformComponent";
00157     }
00158 
00159     void showGUI() override {
00160         ImGui::Separator();
00161 
00162         ImGui::Text("Line: %s", temp_lineParsed.c_str());
00163 
00164         // Position Controls
00165         ImGui::Text("Position:");
00166         ImGui::SliderFloat3("##position", &position.x, -1000.0f, 1000.0f);
00167 
```

```
00168         // Size Controls
00169         ImGui::Text("Size:");
00170         ImGui::SliderFloat3("##size", &size.x, 1.0f, 100.0f);
00171 
00172         // Rotation Controls
00173         ImGui::Text("Rotation:");
00174         ImGui::SliderFloat3("##rotation", glm::value_ptr(rotation), -180.0f, 180.0f);
00175 
00176         ImGui::Text("Scale:");
00177         ImGui::SliderFloat("##scale", &scale, 0.1f, 10.0f);
00178 
00179         // Speed Control
00180         ImGui::Text("Speed:");
00181         ImGui::InputInt("##speed", &speed);
00182     };
00183 };
```

6.36 Triangle_w_Color.h

```
00001 #pragma once
00002 
00003 #include "../../../Components.h"
00004 
00005 class Triangle_w_Color : public Component
00006 {
00007 public:
00008     Color color = { 255, 255, 255, 255 };
00009 
00010     glm::vec2 uv1 = glm::vec2(0), uv2 = glm::vec2(0), uv3 = glm::vec2(0);
00011 
00012     TransformComponent* transform = nullptr;
00013 
00014     Triangle_w_Color()
00015     {
00016 
00017     }
00018 
00019 
00020     ~Triangle_w_Color() {
00021 
00022     }
00023 
00024     void init() override {
00025         transform = &entity->GetComponent<TransformComponent>();
00026     }
00027 
00028     void update(float deltaTime) override {
00029 
00030         //transform->setRotation(transform->getRotation() + 0.1f);
00031     }
00032 
00033     /*void draw(size_t v_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window) {
00034         float tempScreenScale = window.getScale();
00035 
00036         batch.drawTriangle(
00037             v_index,
00038             transform->getPosition(),
00039             transform->rotation,
00040             uv1, uv2, uv3,
00041             0, color
00042         );
00043     }*/
00044 
00045     void draw(size_t v_index, PlaneColorRenderer& batch, TazGraphEngine::Window& window) {
00046         float tempScreenScale = window.getScale();
00047 
00048         batch.drawTriangle(
00049             v_index,
00050             transform->bodyCenter,
00051             transform->rotation, color
00052         );
00053     }
00054     std::string GetComponentName() override {
00055         return "Triangle_w_Color";
00056     }
00057 };
```

6.37 ButtonComponent.h

```
00001 #pragma once
```

```
00002
00003 #include "../../../Components.h"
00004 #include "../../../UtilComponents.h"
00005 #include <functional>  // For std::function
00006
00007 class ButtonComponent : public Component {
00008 public:
00009     enum class ButtonState {
00010         NORMAL,
00011         HOVERED,
00012         PRESSED
00013     };
00014
00015 private:
00016     ButtonState _state = ButtonState::NORMAL;
00017     std::function<void()> _onClick; // Callback function for click action
00018
00019     std::string buttonLabel = "";
00020     glm::vec2 bDimensions{0,0};
00021     Color bBackground{255,255,255,255};
00022
00023     Entity* uiLabel = nullptr;
00024     Entity* buttonBackground = nullptr;
00025 public:
00026     ButtonComponent()
00027         : _state(ButtonState::NORMAL),
00028         _onClick([]() {}) // Default empty function
00029     {
00030     }
00031     ButtonComponent(std::function<void()> onClick)
00032         : _state(ButtonState::NORMAL), _onClick(onClick) {}
00033
00034     ButtonComponent(std::function<void()> onClick, std::string button_label, glm::vec2 b_dimensions,
      Color b_background)
00035         : _state(ButtonState::NORMAL),
00036         _onClick(onClick),
00037         buttonLabel(button_label),
00038         bDimensions(b_dimensions),
00039         bBackground(b_background){
00040            // create UI label with the string given
00041            // set the button background
00042     }
00043
00044     void init() override {
00045         if (buttonLabel.length() > 0) {
00046             // need different shader to render text so it has to be different
00047             uiLabel = &entity->getManager()->addEntity<Node>();
00048
00049             uiLabel->addComponent<TransformComponent>(glm::vec2(0, 0), Layer::action, glm::ivec2(32,
      32), 1);
00050             uiLabel->addComponent<UILabel>(uiLabel->getManager(), buttonLabel, "arial");
00051
00052             uiLabel->setParentEntity(entity);
00053
00054             uiLabel->addGroup(Manager::buttonLabels);
00055         }
00056         if (bDimensions != glm::vec2(0.0f, 0.0f)) {
00057             buttonBackground = &entity->getManager()->addEntity<Node>();
00058
00059             buttonBackground->addComponent<TransformComponent>(glm::vec2(0, 0), Layer::action,
      bDimensions, 1);
00060             buttonBackground->addComponent<Rectangle_w_Color>();
00061
00062             buttonBackground->GetComponent<Rectangle_w_Color>().color = bBackground; // Grey color
00063             buttonBackground->setParentEntity(entity);
00064
00065             buttonBackground->addGroup(Manager::panelBackground);
00066         }
00067     }
00068
00069     void setOnClick(std::function<void()> newOnClick) {
00070         _onClick = newOnClick;
00071     }
00072
00073     void setState(ButtonState state) {
00074         _state = state;
00075     }
00076
00077     void update(float deltaTime) override {
00078         // Update the button state based on user input
00079         // Change the texture or appearance based on the state
00080         // ...
00081
00082         if (_state == ButtonState::PRESSED && _onClick) {
00083             _onClick(); // Call the click action callback
00084         }
00085         setState(ButtonState::NORMAL); // Reset the state
```

```
00086     }
00087
00088     // Other methods for setting textures, handling mouse events, etc.
00089
00090     std::string GetComponentName() override {
00091         return "ButtonComponent";
00092     }
00093 };
```

6.38 ColliderComponent.h

```
00001 #pragma once
00002 #include <string>
00003 #include <unordered_set>
00004 #include <SDL2/SDL.h>
00005 #include <glm/glm.hpp>
00006 #include "../../../Components.h"
00007 #include "../../../UtilComponents.h"
00008 #include "../../../../TextureManager/TextureManager.h"
00009
00010
00011 class ColliderComponent : public NodeComponent //collider -> transform
00012 {
00013 private:
00014     Manager* _manager = nullptr;
00015     float _collisionPadding = 100.0f;
00016     std::unordered_set<Group> _groupChecks;
00017     float repulsion_strength = 100.0f;
00018 public:
00019     glm::vec3 box_collider = glm::vec3(0.0f);
00020
00021     TransformComponent* transform = nullptr;
00022
00023     ColliderComponent()
00024     {
00025     }
00026
00027     // todo instead have offset
00028     ColliderComponent(Manager* manager, glm::vec3 boxCollider_size)
00029     {
00030         _manager = manager;
00031         box_collider = boxCollider_size;
00032     }
00033
00034     void init() override
00035     {
00036         if (!entity->hasComponent<TransformComponent>()) //todo: problem: having transform on top left
      grid, not every collider its own
00037         {
00038             entity->addComponent<TransformComponent>();
00039         }
00040         transform = &entity->GetComponent<TransformComponent>();
00041
00042     }
00043
00044     void update(float deltaTime) override
00045     {
00046
00047
00048     }
00049
00050     void collisionPhysics() {
00051         glm::vec3 nodePosition = transform->bodyCenter;
00052         glm::vec3 nodeHalfSize = 0.5f * transform->size;
00053
00054         for (Group group : _groupChecks) {
00055             const auto& adjacentEntities = _manager->adjacentEntities(entity, group);
00056
00057             for (Entity* other : adjacentEntities) {
00058
00059                 auto areEntitiesLinked = [&](NodeEntity* main, Entity* other)
00060                     {
00061                         for (auto& i : main->getOutLinks()) {
00062                             if (i->getToNode() == other) {
00063                                 return true;
00064                             }
00065                         }
00066                         return false;
00067                     };
00068
00069                 if (areEntitiesLinked(entity, other)) {
00070                     //continue;
00071                 }
```

```
00072
00073                 glm::vec3 otherPosition = other->GetComponent<TransformComponent>().bodyCenter;
00074                 glm::vec3 otherHalfSize = 0.5f * other->GetComponent<TransformComponent>().size;
00075
00076                 glm::vec3 delta = nodePosition - otherPosition;
00077
00078                 float dist = std::max(length(delta), 1e-4f);
00079                 glm::vec3 repulsion = repulsion_strength * normalize(delta) / (dist * dist);
00080
00081                 transform->velocity += repulsion;
00082                 other->GetComponent<TransformComponent>().velocity -= repulsion;
00083
00084             }
00085         }
00086     }
00087
00088
00089     void draw(size_t e_index, PlaneModelRenderer&  batch, TazGraphEngine::Window& window) override
00090     {
00091     }
00092
00093     std::string GetComponentName() override {
00094         return "ColliderComponent";
00095     }
00096
00097     void addCollisionGroup(Group g) {
00098         _groupChecks.insert(g);
00099     }
00100
00101     void removeCollisionGroup(Group g) {
00102         _groupChecks.erase(g);
00103     }
00104 };
```

6.39 GridComponent.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
00004
00005 #define GRID_EMPTY_TILE 0x0000
00006 #define GRID_SOLID_TILE 0x0001
00007 #define GRID_ROWS 4
00008 #define GRID_COLUMNS 4
00009 #define GRID_ELEMENT_WIDTH 8
00010 #define GRID_ELEMENT_HEIGHT 8
00011 #define TILE_NUM_GRID_ELEMENTS (GRID_ROWS * GRID_COLUMNS) //16
00012 #define MAX_MAP_TILE_HEIGHT 20
00013 #define MAX_MAP_WIDTH_TILES 100
00014 #define MAX_MAP_GRID_HEIGHT (MAX_MAP_TILE_HEIGHT * GRID_ROWS) //80
00015 #define MAX_MAP_GRID_WIDTH (MAX_MAP_TILE_HEIGHT * GRID_COLUMNS) //100
00016
00017 constexpr size_t GRID_CELL_NUM = 16;
00018
00019 class GridComponent : public Component //GridComp --> ColliderComp
00020 {
00021 private:
00022     std::bitset<GRID_CELL_NUM> bitset ;
00023 public:
00024     glm::ivec2 position;
00025     int scaledTile;
00026     ColliderComponent* collider = nullptr;
00027
00028     GridComponent() = default;
00029
00030     ~GridComponent()
00031     {
00032
00033     }
00034
00035     GridComponent(int xpos, int ypos, int tscaled, std::bitset<GRID_CELL_NUM> collider_bitSet =
      std::bitset<GRID_CELL_NUM>())
00036     {
00037         position.x = xpos;
00038         position.y = ypos;
00039
00040         scaledTile = tscaled;
00041
00042         bitset = collider_bitSet.set();
00043     }
00044
00045     void init() override
00046     {
```

```
00047         auto flippedBitset = ~bitset;
00048
00049         if (flippedBitset.none()) {
00050             //entity->addComponent<ColliderComponent>((position.x), (position.y), GRID_ELEMENT_WIDTH *
      GRID_COLUMNS);
00051         }
00052         else {
00053             glm::ivec2 gridPos;
00054
00055             for (auto gridindex = 0; gridindex < TILE_NUM_GRID_ELEMENTS; gridindex++)
00056             { //SetGridTileBlock
00057                 if (bitset[gridindex]) {
00058                     gridPos.x = (gridindex % GRID_COLUMNS) * GRID_ELEMENT_WIDTH;
00059                     gridPos.y = (int)(gridindex / GRID_ROWS) * GRID_ELEMENT_HEIGHT;
00060
00061                     //entity->addComponent<ColliderComponent>((position.x + gridPos.x), (position.y +
      gridPos.y), GRID_ELEMENT_WIDTH);
00062
00063                 }
00064             }
00065         }
00066
00067     }
00068
00069     void update(float deltaTime) override
00070     {
00071
00072     }
00073
00074     std::string GetComponentName() override {
00075         return "GridComponent";
00076     }
00077 };
```

6.40 KeyboardControllerComponent.h

```
00001 #pragma once
00002
00003
00004 #include "../../../Components.h"
00005
00006 #if defined(_WIN32) || defined(_WIN64)
00007     #include <Windows.h>
00008     #include <MMSystem.h>
00009     #pragma comment(lib, "winmm.lib") // Link to Windows multimedia library
00010 #elif defined(__linux__) || defined(__unix__)
00011     #include <unistd.h> // Common Unix/Linux header
00012 #endif
00013
00014 #include "../../../../InputManager/InputManager.h"
00015
00016 constexpr float walkingSpeed = 3.5f, runningSpeed = 8.5f, jumpingSpeed = 3.0f;
00017
00018
00019 class KeyboardControllerComponent : public Component
00020 {
00021 public: //TODO: maybe have variables as private
00022     InputManager* _inputManager = nullptr;
00023
00024     TransformComponent* transform = nullptr;
00025     AnimatorComponent* animator = nullptr;
00026     RigidBodyComponent* rigidbody = nullptr;
00027     SpriteComponent* sprite = nullptr;
00028
00029     SDL_KeyCode walkUpKey = SDLK_UNKNOWN, walkLeftKey = SDLK_UNKNOWN, walkRightKey = SDLK_UNKNOWN,
      walkDownKey = SDLK_UNKNOWN;
00030
00031
00032     KeyboardControllerComponent()
00033     {
00034
00035     }
00036
00037     KeyboardControllerComponent(
00038         InputManager* inputManager,
00039         SDL_KeyCode walkUpKey,
00040         SDL_KeyCode walkLeftKey,
00041         SDL_KeyCode walkRightKey,
00042         SDL_KeyCode walkDownKey
00043         )
00044         : _inputManager(inputManager),
00045         walkUpKey(walkUpKey),
00046         walkLeftKey(walkLeftKey),
```

```
00047         walkRightKey(walkRightKey),
00048         walkDownKey(walkDownKey)
00049     {
00050
00051     }
00052
00053     ~KeyboardControllerComponent()
00054     {
00055
00056     }
00057
00058     void init() override
00059     {
00060         transform = &entity->GetComponent<TransformComponent>();
00061         animator = &entity->GetComponent<AnimatorComponent>();
00062         sprite = &entity->GetComponent<SpriteComponent>();
00063     }
00064
00065     void update(float deltaTime) override
00066     {
00067
00068         if (_inputManager->isKeyDown(walkLeftKey)) {
00069             transform->setVelocity_X(-runningSpeed);
00070         }
00071         if (_inputManager->isKeyDown(walkRightKey)) {
00072             transform->setVelocity_X(runningSpeed);
00073         }
00074         if (!_inputManager->isKeyDown(walkRightKey) && !_inputManager->isKeyDown(walkLeftKey)) {
00075             transform->setVelocity_X(0);
00076         }
00077         if (_inputManager->isKeyDown(walkUpKey)) {
00078             transform->setVelocity_Y(-runningSpeed);
00079         }
00080         if (_inputManager->isKeyDown(walkDownKey)) {
00081             transform->setVelocity_Y(runningSpeed);
00082         }
00083         if (!_inputManager->isKeyDown(walkUpKey) && !_inputManager->isKeyDown(walkDownKey)) {
00084             transform->setVelocity_Y(0);
00085         }
00086     }
00087
00088     std::string GetComponentName() override {
00089         return "KeyboardControllerComponent";
00090     }
00091 };
```

6.41 RigidBodyComponent.h

```
00001 #pragma once
00002 #include "../../../Components.h"
00003
00004 class RigidBodyComponent : public Component
00005 {
00006 private:
00007     TransformComponent* transform = nullptr;
00008 public:
00009     float GravityForce = 1.0f;
00010     float accelGravity = 0.045f;
00011     float maxGravity = 3.f;
00012     bool onGround = false;
00013     bool justjumped = false;
00014
00015     RigidBodyComponent() = default;
00016
00017     RigidBodyComponent(float acc, float maxg)
00018     {
00019         accelGravity = acc;
00020         maxGravity = maxg;
00021     }
00022
00023     ~RigidBodyComponent()
00024     {
00025
00026     }
00027
00028     void init() override
00029     {
00030         transform = &entity->GetComponent<TransformComponent>();
00031     }
00032
00033     void update(float deltaTime) override
00034     {
00035         if (onGround && !justjumped)
```

```
00036         {
00037             GravityForce = 1.0f;
00038             transform->setVelocity_Y(GravityForce);
00039             GravityForce = 0.0f;
00040         }
00041         else
00042         {
00043             justjumped = false;
00044             GravityForce += accelGravity * deltaTime;
00045             transform->setVelocity_Y(transform->getVelocity().y + GravityForce * deltaTime);
00046             if (transform->getVelocity().y > maxGravity)
00047             {
00048                 transform->setVelocity_Y(maxGravity);
00049             }
00050         }
00051     }
00052
00053     std::string GetComponentName() override {
00054         return "RigidBodyComponent";
00055     }
00056 };
```

6.42 UILabel.h

```
00001 #pragma once
00002
00003 #include "../../../../PNG_Letters.h"
00004 #include "../../../Components.h"
00005 #include "../../../UtilComponents.h"
00006
00007 class Manager;
00008
00009 class UILabel : public Component
00010 {
00011 private:
00012     Manager* _manager;
00013     std::vector<Entity*> letters;
00014     std::string label;
00015     std::string fontFamily;
00016 public:
00017     TransformComponent* transform = nullptr;
00018
00019     UILabel() = default;
00020     UILabel(Manager* manager, std::string lab, std::string fontFam)
00021     {
00022         _manager = manager;
00023         fontFamily = fontFam;
00024         label = lab;
00025     }
00026
00027     ~UILabel() {
00028
00029     }
00030
00031     void init() override {
00032         //create entities for each letter
00033         if (!entity->hasComponent<TransformComponent>())
00034         {
00035             entity->addComponent<TransformComponent>();
00036         }
00037         transform = &entity->GetComponent<TransformComponent>();
00038         setLetters(label);
00039     }
00040
00041     void update(float deltaTime) override {
00042         //if string changes then for all the labels that have been created,
00043         //find the ones that are for that string and delete them?
00044
00045         float previousCharX = 0;
00046
00047         for (auto& l : letters) {
00048             l->GetComponent<TransformComponent>().setPosition_X(transform->getPosition().x +
      previousCharX) ;
00049             l->GetComponent<TransformComponent>().setPosition_Y(transform->getPosition().y);
00050             previousCharX += l->GetComponent<TransformComponent>().size.x;
00051         }
00052
00053         if (previousCharX > transform->size.x) {
00054             transform->size.x = previousCharX;
00055         }
00056     }
00057
00058     void draw(size_t e_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window) override {
```

```
00059         //draw each letter
00060         for (auto& l : letters) {
00061             l->draw(e_index, batch, window);
00062         }
00063     }
00064
00065     void setLetters(std::string lab) {
00066         letters.clear();
00067         label = lab;
00068         for (char c : label) {
00069             // todo make this as an entity
00070             //auto& label(_manager->addEntity());
00071             //SDL_Rect charRect = getLetterRect(c);
00072             //label.addComponent<TransformComponent>(transform->getPosition(), Manager::actionLayer,
00073             //  glm::ivec2(charRect.w, charRect.h),//!set the dest.w/h from the table and then also
      set src.x/y/w/h. dest.x/y is based on previous letter and original label position
00074             //  1);
00075             //label.addComponent<SpriteComponent>(fontFamily);
00076             //label.GetComponent<SpriteComponent>().srcRect.x = charRect.x;
00077             //label.GetComponent<SpriteComponent>().srcRect.y = charRect.y;
00078
00079             //letters.push_back(&label);
00080
00081         }
00082     }
00083
00084     std::string GetComponentName() override {
00085         return "UILabel";
00086     }
00087 };
00088
```

6.43 Line_w_Color.h

```
00001 #pragma once
00002
00003 #include "../../../Components.h"
00004
00005 class Line_w_Color : public LinkComponent
00006 {
00007 public:
00008     std::string temp_lineParsed = "";
00009
00010     Color default_src_color = { 255, 255, 255, 255 };
00011     Color src_color = { 255, 255, 255, 255 };
00012
00013     Color default_dest_color = { 255, 255, 255, 255 };
00014     Color dest_color = { 255, 255, 255, 255 };
00015
00016     FlashAnimation flash_animation;
00017
00018     Line_w_Color()
00019     {
00020
00021     }
00022
00023     ~Line_w_Color() {
00024
00025     }
00026
00027     void init() override {
00028     }
00029
00030     void update(float deltaTime) override {
00031     }
00032
00033     void draw(size_t v_index, LineRenderer& batch, TazGraphEngine::Window& window) {
00034         //float tempScreenScale = window.getScale();
00035
00036         glm::vec3 fromNodeCenter =
      entity->getFromNode()->GetComponent<TransformComponent>().getCenterTransform();
00037         glm::vec3 toNodeCenter =
      entity->getToNode()->GetComponent<TransformComponent>().getCenterTransform();
00038
00039         batch.drawLine(v_index, fromNodeCenter, toNodeCenter, src_color, dest_color);
00040     }
00041
00042     void drawWithPorts(size_t v_index, LineRenderer& batch, TazGraphEngine::Window& window) {
00043         //float tempScreenScale = window.getScale();
00044
00045         glm::vec3 fromPortCenter =
      entity->getFromNode()->children[entity->fromPort]->GetComponent<TransformComponent>().getCenterTransform();
00046         glm::vec3 toPortCenter =
      entity->getToNode()->children[entity->toPort]->GetComponent<TransformComponent>().getCenterTransform();
```

```
00047
00048         batch.drawLine(v_index, fromPortCenter, toPortCenter, src_color, dest_color);
00049     }
00050
00051     void setSrcColor(Color clr) {
00052         default_src_color = clr;
00053         src_color = clr;
00054     }
00055
00056     void setDestColor(Color clr) {
00057         default_dest_color = clr;
00058         dest_color = clr;
00059     }
00060
00061     void SetFlashAnimation(int idX, int idY, size_t fr, float sp, const Animation::animType type,
      const std::vector<float>& flashTimes, Color flashC, int reps = 0)
00062     {
00063         flash_animation = FlashAnimation(idX, idY, fr, sp, type, flashTimes, flashC, reps);
00064     }
00065
00066     void setFlashFrame() {
00067         float t = this->flash_animation.interpolation_a;
00068
00069         if (t < 0.33f) {
00070             this->src_color = Color::fromVec4(glm::mix(default_src_color.toVec4(),
      this->flash_animation.flashColor.toVec4(), 3 * t));
00071         }
00072         else if (t < 0.66f) {
00073             this->src_color = Color::fromVec4(glm::mix(this->flash_animation.flashColor.toVec4(),
      default_src_color.toVec4(), std::min((3 * (t - 0.33f)), 1.0f)));
00074             this->dest_color = Color::fromVec4(glm::mix(default_dest_color.toVec4(),
      this->flash_animation.flashColor.toVec4(), 3 * (t - 0.33f)));
00075         }
00076         else {
00077             this->dest_color = Color::fromVec4(glm::mix(this->flash_animation.flashColor.toVec4(),
      default_dest_color.toVec4(), std::min((3 * (t - 0.66f)), 1.0f)));
00078         }
00079         // Smooth transition using lerp (linear interpolation)
00080
00081     }
00082
00083     std::string GetComponentName() override {
00084         return "Line_w_Color";
00085     }
00086
00087     void showGUI() override {
00088         ImGui::Separator();
00089
00090         ImGui::Text("Line: %s", temp_lineParsed.c_str());
00091
00092         ImVec4 a_color = ImVec4(src_color.r / 255.0f, src_color.g / 255.0f, src_color.b / 255.0f,
      src_color.a / 255.0f);
00093         if (ImGui::ColorPicker4("Color Line Src", (float*)&a_color)) {
00094             Color newColor = {
00095                         (GLubyte)(a_color.x * 255),
00096                         (GLubyte)(a_color.y * 255),
00097                         (GLubyte)(a_color.z * 255),
00098                         (GLubyte)(a_color.w * 255)
00099             };
00100
00101             setSrcColor(newColor);
00102         }
00103
00104
00105         ImVec4 b_color = ImVec4(dest_color.r / 255.0f, dest_color.g / 255.0f, dest_color.b / 255.0f,
      dest_color.a / 255.0f);
00106         if (ImGui::ColorPicker4("Color Line Dest", (float*)&b_color)) {
00107             Color newColor = {
00108                         (GLubyte)(b_color.x * 255),
00109                         (GLubyte)(b_color.y * 255),
00110                         (GLubyte)(b_color.z * 255),
00111                         (GLubyte)(b_color.w * 255)
00112             };
00113
00114             setDestColor(newColor);
00115         }
00116     }
00117 };
```

6.44 SpringComponent.h

```
00001 #pragma once
00002
```

```
00003 #include "../../../Components.h"
00004
00005 class SpringComponent : public LinkComponent
00006 {
00007 private:
00008     int deltaThreshold = 300;
00009     float springStrength = 0.00002f;
00010 public:
00011
00012     SpringComponent()
00013     {
00014
00015     }
00016
00017     ~SpringComponent() {
00018
00019     }
00020
00021     void init() override {
00022     }
00023
00024     void update(float deltaTime) override {
00025         NodeEntity* a = entity->getFromNode();
00026         NodeEntity* b = entity->getToNode();
00027
00028         glm::vec3 posA = a->GetComponent<TransformComponent>().bodyCenter;
00029         glm::vec3 posB = b->GetComponent<TransformComponent>().bodyCenter;
00030
00031         glm::vec3 delta = posB - posA;
00032         glm::vec3 attraction = delta * springStrength;
00033
00034         if (glm::length(delta) > deltaThreshold) {
00035             a->GetComponent<TransformComponent>().velocity += attraction;
00036             b->GetComponent<TransformComponent>().velocity -= attraction;
00037         }
00038
00039         /*springStrength -= 0.001f;
00040         if (springStrength < 0) {
00041             entity->removeComponent<SpringComponent>();
00042         }*/
00043     }
00044
00045     void draw(size_t v_index, LineRenderer& batch, TazGraphEngine::Window& window) {
00046     }
00047
00048     std::string GetComponentName() override {
00049         return "SpringComponent";
00050     }
00051
00052     void showGUI() override {
00053         ImGui::Separator();
00054
00055         ImGui::Text("SpringComponent");
00056
00057     }
00058 };
```

6.45 PollingComponent.h

```
00001 #pragma once
00002 #include "../../../Components.h"
00003
00004 class PollingComponent : public NodeComponent {
00005 public:
00006     void StartPolling(const std::string& file, float delayInSeconds) {
00007         pollingFile = file;
00008         timer = delayInSeconds;
00009         pollingActive = true;
00010     }
00011
00012     void update(float deltaTime) override {
00013         if (!pollingActive) return;
00014
00015         timer -= deltaTime / 10.0f;
00016         if (timer <= 0.0f) {
00017             SendMessageToOtherNodes(pollingFile);
00018             pollingActive = false; // Stop polling after sending the message
00019         }
00020     }
00021
00022     std::string GetComponentName() override {
00023         return "PollingComponent";
00024     }
```

```
00025
00026 private:
00027     void SendMessageToOtherNodes(const std::string& file) {
00028         std::cout « "Polling complete. Sending message from file: " « file « std::endl;
00029
00030        // get outLinks of Node, fine the node with the id, and send a message saying "Hello World"
00031        // keep log in the nodes where it shows the messages received
00032         if (!entity->getOutLinks().empty()) {
00033             for (auto& link : entity->getOutLinks()) {
00034                 link->getToNode()->addMessage("Hello World");
00035
00036
00037                 link->GetComponent<LineFlashAnimatorComponent>().Play("LineTransfer");
00038             }
00039         }
00040         // entity->makeAnimation(0.01f)
00041
00042     }
00043
00044     std::string pollingFile;
00045     float timer = 0.0f;
00046     bool pollingActive = false;
00047 };
```

6.46 CellEntity.h

```
00001 #pragma once
00002
00003 #include "GECS.h"
00004
00005 struct Cell {
00006     std::vector<EmptyEntity*> emptyEntities;
00007     std::vector<NodeEntity*> nodes;
00008     std::vector<LinkEntity*> links;
00009
00010     glm::vec3 boundingBox_origin = glm::vec3(0); // Starting point (minimum corner) of the cell
00011     glm::vec3 boundingBox_size = glm::vec3(0);
00012     glm::vec3 boundingBox_center = glm::vec3(0);
00013
00014     Cell* parent = nullptr;
00015     std::vector<Cell*> children;
00016
00017     template <typename T>
00018     std::vector<T*>& getEntityList() {
00019         if constexpr (std::is_same_v<T, NodeEntity>) {
00020             return nodes;
00021         }
00022         else if constexpr (std::is_same_v<T, EmptyEntity>) {
00023             return emptyEntities;
00024         }
00025         else if constexpr (std::is_same_v<T, LinkEntity>) {
00026             return links;
00027         }
00028         else {
00029             static_assert(sizeof(T) == 0, "Unsupported entity type.");
00030         }
00031     }
00032 };
00033
00034
00035 class CellEntity : public Entity {
00036 public:
00037     Cell* ownerCell = nullptr;
00038
00039
00040     CellEntity(Manager& mManager) : Entity(mManager) {
00041
00042     }
00043
00044     void setOwnerCell(Cell* cell) {
00045         this->ownerCell = cell;
00046     }
00047
00048
00049     Cell* getOwnerCell() const { return ownerCell; }
00050
00051 };
00052
00053 class MultiCellEntity : public Entity {
00054 public:
00055     std::vector<Cell*> ownerCells = {};
00056
00057     MultiCellEntity(Manager& mManager) : Entity(mManager) {
```

```
00058
00059     }
00060
00061     void setOwnerCells(std::vector<Cell*> cells) {
00062         this->ownerCells = cells;
00063     }
00064
00065     Cell* getOwnerCells() const { return ownerCells[0]; }
00066
00067 };
```

6.47 GECS.h

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <vector>
00005 #include <memory>
00006 #include <algorithm>
00007 #include <bitset>
00008 #include <array>
00009 #include <unordered_map>
00010
00011 #include <SDL2/SDL.h>
00012 #include "../../Renderers/PlaneModelRenderer/PlaneModelRenderer.h"
00013 #include "../../Renderers/LineRenderer/LineRenderer.h"
00014 #include "../../Renderers/PlaneColorRenderer/PlaneColorRenderer.h"
00015 #include "../../Renderers/LightRenderer/LightRenderer.h"
00016 #include "../../Camera2.5D/CameraManager.h"
00017 #include "../../Window/Window.h"
00018 #include <optional>
00019
00020 #define CULLING_OFFSET 100
00021
00022 class BaseComponent;
00023 class Entity;
00024 class EmptyEntity;
00025 class NodeEntity;
00026 class LinkEntity;
00027
00028 class Manager;
00029 class Window;
00030 struct Cell;
00031
00032 using ComponentID = std::size_t;
00033 using Group = std::size_t;
00034
00035 using layer = std::size_t;
00036
00037 namespace Layer {
00038     enum layerIndexes : std::size_t
00039     {
00040         action,
00041         menubackground
00042     };
00043 }
00044
00045 const std::unordered_map<layer, float> layerNames = {
00046         {Layer::action, 0.0f},
00047         {Layer::menubackground, -100.0f}
00048
00049 };
00050
00051 inline float getLayerDepth(layer mLayer) {
00052     return layerNames.at(mLayer);
00053 }
00054
00055
00056 inline ComponentID getNewComponentTypeID()
00057 {
00058     static ComponentID lastID = 0u;
00059     return lastID++;
00060 }
00061
00062 inline ComponentID getNewNodeComponentTypeID()
00063 {
00064     static ComponentID lastID_nodeC = 0u;
00065     return lastID_nodeC++;
00066 }
00067
00068 inline ComponentID getNewLinkComponentTypeID()
00069 {
00070     static ComponentID lastID_linkC = 0u;
```

```
00071     return lastID_linkC++;
00072 }
00073 
00074 
00075 template <typename T> inline ComponentID GetComponentTypeID() noexcept
00076 {
00077     static ComponentID typeID = getNewComponentTypeID(); // typeID is unique for each function type T
      and only initialized once.
00078     return typeID;
00079 }
00080 
00081 template <typename T> inline ComponentID GetNodeComponentTypeID() noexcept
00082 {
00083     static ComponentID typeID = getNewNodeComponentTypeID(); // typeID is unique for each function
      type T and only initialized once.
00084     return typeID;
00085 }
00086 
00087 template <typename T> inline ComponentID GetLinkComponentTypeID() noexcept
00088 {
00089     static ComponentID typeID = getNewLinkComponentTypeID(); // typeID is unique for each function
      type T and only initialized once.
00090     return typeID;
00091 }
00092 
00093 constexpr std::size_t maxComponents = 16;
00094 constexpr std::size_t maxGroups = 16;
00095 
00096 using ComponentBitSet = std::bitset<maxComponents>;
00097 using GroupBitSet = std::bitset<maxGroups>;
00098 
00099 using ComponentArray = std::array<BaseComponent*, maxComponents>;
00100 
00101 class BaseComponent
00102 {
00103 public:
00104 
00105     ComponentID id = 0u;
00106 
00107     virtual void init() {}
00108     virtual void update(float deltaTime) {}
00109     virtual void draw(size_t e_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window) {}
00110     virtual void draw(size_t e_index, LineRenderer& batch, TazGraphEngine::Window& window) {}
00111     virtual void draw(size_t e_index, PlaneColorRenderer& batch, TazGraphEngine::Window& window) {}
00112     virtual void draw(size_t e_index, LightRenderer& batch, TazGraphEngine::Window& window) {}
00113 
00114     virtual std::string GetComponentName() { return ""; };
00115 
00116     virtual void showGUI() {
00117         ImGui::Text("MyComponent Properties:");
00118     };
00119 
00120     virtual ~BaseComponent() {}
00121 };
00122 
00123 class Component : public BaseComponent {
00124 public:
00125     Entity* entity = nullptr;
00126 };
00127 
00128 class NodeComponent : public BaseComponent {
00129 public:
00130     NodeEntity* entity = nullptr;
00131 };
00132 
00133 class LinkComponent : public BaseComponent {
00134 public:
00135     LinkEntity* entity = nullptr;
00136 };
00137 
00138 
00139 class Entity
00140 {
00141 private:
00142     unsigned int id = 0;
00143 
00144     bool active = true; // false if about to delete
00145     bool hidden = false; // true if not do updates
00146     ComponentArray componentArray = {};//create 2 arrays, this is for the fast access
00147 
00148     ComponentBitSet componentBitSet;
00149     GroupBitSet groupBitSet;
00150 
00151 protected:
00152     std::optional<ComponentArray> nodeComponentArray;
00153     std::optional<ComponentBitSet> nodeComponentBitSet;
00154 
```

```
00155     Manager& manager;
00156 public:
00157     std::unordered_map<std::string,EmptyEntity*> children;
00158 
00159     void setId(unsigned int m_id) { id = m_id; }
00160     unsigned int getId() { return id; }
00161 
00162     void hide() {
00163         hidden = true;
00164     }
00165 
00166     void reveal() {
00167         hidden = false;
00168     }
00169 
00170     bool isHidden() {
00171         return hidden;
00172     }
00173 
00174     std::vector<std::unique_ptr<BaseComponent» components; //create 2 arrays, this is for the
      concurrent access
00175 
00176     Entity(Manager& mManager) : manager(mManager) {}
00177     virtual ~Entity() {}
00178 
00179     virtual void update(float deltaTime)
00180     {
00181 
00182         for (auto& c : components) {
00183             c->update(deltaTime); // start from which was added first
00184         }
00185     }
00186 
00187     virtual void cellUpdate() {};
00188 
00189     virtual Cell* getOwnerCell() const { return nullptr; };
00190 
00191     void draw(size_t e_index, PlaneModelRenderer& batch, TazGraphEngine::Window& window)
00192     {
00193         for (auto& c : components) {
00194             c->draw(e_index, batch, window);
00195         }
00196     }
00197     void draw(size_t e_index, LineRenderer& batch, TazGraphEngine::Window& window)
00198     {
00199         for (auto& c : components) {
00200             c->draw(e_index, batch, window);
00201         }
00202     }
00203     void draw(size_t e_index, PlaneColorRenderer& batch, TazGraphEngine::Window& window)
00204     {
00205         for (auto& c : components) {
00206             c->draw(e_index, batch, window);
00207         }
00208     }
00209     void draw(size_t e_index, LightRenderer& batch, TazGraphEngine::Window& window)
00210     {
00211         for (auto& c : components) {
00212             c->draw(e_index, batch, window);
00213         }
00214     }
00215     bool isActive() { return active; }
00216     virtual void destroy() { active = false;
00217     } // destroy happens relative to the group referencing
00218 
00219     bool hasGroup(Group mGroup)
00220     {
00221         return groupBitSet[mGroup];
00222     }
00223 
00224     virtual void addGroup(Group mGroup);
00225     void removeGroup(Group mGroup);
00226 
00227     template <typename T> bool hasComponent() const
00228     {
00229         if constexpr (std::is_base_of_v<LinkComponent, T>) {
00230             return this && componentBitSet[GetLinkComponentTypeID<T>()];
00231         }
00232         else if constexpr (std::is_base_of_v<NodeComponent, T>) {
00233             return this && nodeComponentBitSet.has_value() &&
      (*nodeComponentBitSet)[GetNodeComponentTypeID<T>()];
00234         }
00235         return this && componentBitSet[GetComponentTypeID<T>()];
00236     }
00238     template <typename T, typename... TArgs>
00239     T& addComponent(TArgs&&... mArgs)
00240     {
```

```
00241         T* c(new T(std::forward<TArgs>(mArgs)...));
00242         if constexpr (std::is_base_of_v<LinkComponent, T>) {
00243             std::unique_ptr<LinkComponent> uPtr{ c };
00244             components.emplace_back(std::move(uPtr));
00245 
00246             setComponentEntity(c);
00247             componentArray[GetLinkComponentTypeID<T>()] = c;
00248             componentBitSet[GetLinkComponentTypeID<T>()] = true;
00249 
00250             c->id = GetLinkComponentTypeID<T>();
00251 
00252             c->init();
00253             return *c;
00254         }
00255         else if constexpr (std::is_base_of_v<NodeComponent, T>) {
00256             std::unique_ptr<NodeComponent> uPtr{ c };
00257             components.emplace_back(std::move(uPtr));
00258 
00259             setComponentEntity(c);
00260             (*nodeComponentArray)[GetNodeComponentTypeID<T>()] = c;
00261             (*nodeComponentBitSet)[GetNodeComponentTypeID<T>()] = true;
00262 
00263             c->id = GetNodeComponentTypeID<T>();
00264 
00265             c->init();
00266             return *c;
00267         }
00268         else {
00269             std::unique_ptr<Component> uPtr{ c };
00270             components.emplace_back(std::move(uPtr));
00271 
00272             setComponentEntity(c);
00273             componentArray[GetComponentTypeID<T>()] = c;
00274             componentBitSet[GetComponentTypeID<T>()] = true;
00275 
00276             c->id = GetComponentTypeID<T>();
00277 
00278             c->init();
00279             return *c;
00280         }
00281 
00282 
00283     }
00284 
00285     template <typename T>
00286     void removeComponent()
00287     {
00288         if constexpr (std::is_base_of_v<LinkComponent, T>)
00289         {
00290             size_t id = GetLinkComponentTypeID<T>();
00291             auto it = std::remove_if(components.begin(), components.end(),
00292                 [id](const std::unique_ptr<BaseComponent>& comp) {
00293                     return typeid(*comp).hash_code() == typeid(T).hash_code();
00294                 });
00295 
00296             if (it != components.end())
00297             {
00298                 components.erase(it, components.end());
00299                 componentArray[id] = nullptr;
00300                 componentBitSet[id] = false;
00301             }
00302         }
00303         else if constexpr (std::is_base_of_v<NodeComponent, T>)
00304         {
00305             size_t id = GetNodeComponentTypeID<T>();
00306             auto it = std::remove_if(components.begin(), components.end(),
00307                 [id](const std::unique_ptr<BaseComponent>& comp) {
00308                     return typeid(*comp).hash_code() == typeid(T).hash_code();
00309                 });
00310 
00311             if (it != components.end())
00312             {
00313                 components.erase(it, components.end());
00314                 (*nodeComponentArray)[id] = nullptr;
00315                 (*nodeComponentBitSet)[id] = false;
00316             }
00317         }
00318         else
00319         {
00320             size_t id = GetComponentTypeID<T>();
00321             auto it = std::remove_if(components.begin(), components.end(),
00322                 [id](const std::unique_ptr<BaseComponent>& comp) {
00323                     return typeid(*comp).hash_code() == typeid(T).hash_code();
00324                 });
00325 
00326             if (it != components.end())
00327             {
```

```
00328                 components.erase(it, components.end());
00329                 componentArray[id] = nullptr;
00330                 componentBitSet[id] = false;
00331             }
00332         }
00333     }
00334
00335     virtual void setComponentEntity(Component* c) {
00336
00337     }
00338     virtual void setComponentEntity(NodeComponent* c) {
00339
00340     }
00341     virtual void setComponentEntity(LinkComponent* c) {
00342
00343     }
00344
00345     template<typename T> T& GetComponent() const
00346     {
00347         if constexpr (std::is_base_of_v<LinkComponent, T>) {
00348             auto ptr(componentArray[GetLinkComponentTypeID<T>()]);
00349             return *static_cast<T*>(ptr);
00350         }
00351         else if constexpr (std::is_base_of_v<NodeComponent, T>) {
00352             auto ptr((*nodeComponentArray)[GetNodeComponentTypeID<T>()]);
00353             return *static_cast<T*>(ptr);
00354         }
00355         else {
00356             auto ptr(componentArray[GetComponentTypeID<T>()]);
00357             return *static_cast<T*>(ptr);
00358         }
00359     }
00360
00361     bool hasComponentByName(const std::string& componentName) {
00362         for (auto& component : components) {
00363             if (component &&
00364                 component->GetComponentName() == componentName) {
00365                 return true;
00366             }
00367         }
00368         return false;
00369     }
00370
00371
00372
00373     // for when wanting to add new entities from components
00374     Manager* getManager() {
00375         return &manager;
00376     }
00377
00378     virtual void addMessage(std::string mMessage) {}
00379
00380
00381     virtual Entity* getParentEntity() {
00382         return nullptr;
00383     }
00384
00385     virtual void setParentEntity(Entity* pEntity) {}
00386
00387     virtual void imgui_print() {}
00388
00389     virtual void imgui_display() {}
00390
00391     virtual void removeEntity() {}
00392 };
00393
```

6.48 GECSEntity.h

```
00001 #pragma once
00002
00003 #include "CellEntity.h"
00004
00005 class LinkEntity;
00006
00007 class EmptyEntity : public CellEntity {
00008 protected:
00009     Entity* parent_entity = nullptr;
00010 public:
00011
00012     EmptyEntity(Manager& mManager) : CellEntity(mManager) {}
00013
00014     void setComponentEntity(Component* c) override {
```

```
00015         c->entity = this;
00016     }
00017
00018     Entity* getParentEntity() override {
00019         return parent_entity;
00020     }
00021
00022     void setParentEntity(Entity* pEntity) override {
00023         parent_entity = pEntity;
00024     }
00025
00026     void removeFromCell() {
00027         if (this->ownerCell) {
00028             removeEntity();
00029             this->ownerCell = nullptr;
00030         }
00031     }
00032
00033     void removeEntity() override{
00034         ownerCell->emptyEntities.erase(
00035             std::remove(this->ownerCell->emptyEntities.begin(), this->ownerCell->emptyEntities.end(),
00036                 this),
00037             this->ownerCell->emptyEntities.end());
00038     }
00039
00040
00041 };
00042
00043 class NodeEntity : public EmptyEntity {
00044 protected:
00045     std::vector<LinkEntity*> inLinks;
00046     std::vector<LinkEntity*> outLinks;
00047 public:
00048
00049     NodeEntity(Manager& mManager) : EmptyEntity(mManager) {
00050         nodeComponentArray.emplace();
00051         nodeComponentBitSet.emplace();
00052     }
00053     void setComponentEntity(NodeComponent* c) override {
00054         c->entity = this;
00055
00056     }
00057
00058     void removeEntity() override {
00059         ownerCell->nodes.erase(
00060             std::remove(this->ownerCell->nodes.begin(), this->ownerCell->nodes.end(),
00061                 this),
00062             this->ownerCell->nodes.end());
00063     }
00064
00065     void addInLink(LinkEntity* link) {
00066         inLinks.push_back(link);
00067     }
00068
00069     void addOutLink(LinkEntity* link) {
00070         outLinks.push_back(link);
00071     }
00072
00073     const std::vector<LinkEntity*>& getInLinks() const {
00074         return inLinks;
00075     }
00076
00077     const std::vector<LinkEntity*>& getOutLinks() const {
00078         return outLinks;
00079     }
00080
00081     virtual void addPorts() {}
00082
00083     virtual void removePorts() {}
00084
00085     virtual void updatePorts(float deltaTime) {}
00086
00087 };
00088
00089
00090 class LinkEntity : public MultiCellEntity {
00091 protected:
00092     unsigned int fromId = 0;
00093     unsigned int toId = 0;
00094
00095     NodeEntity* from = nullptr;
00096     NodeEntity* to = nullptr;
00097
00098 public:
00099     std::string fromPort;
00100     std::string toPort;
00101
```

```
00102     LinkEntity(Manager& mManager) : MultiCellEntity(mManager) {
00103     }
00104
00105     LinkEntity(Manager& mManager, unsigned int mfromId, unsigned int mtoId)
00106         : MultiCellEntity(mManager), fromId(mfromId), toId(mtoId) {
00107     }
00108
00109     LinkEntity(Manager& mManager, NodeEntity* mfrom, NodeEntity* mto)
00110         : MultiCellEntity(mManager), from(mfrom), to(mto) {
00111     }
00112
00113     void setComponentEntity(LinkComponent* c) override {
00114         c->entity = this;
00115     }
00116
00117     void removeFromCells() {
00118         removeEntity();
00119         ownerCells.clear();
00120     }
00121
00122     void removeEntity() override {
00123         for (auto cell : ownerCells) {
00124             cell->links.erase(std::remove(cell->links.begin(), cell->links.end(),
00125                 this),
00126                 cell->links.end());
00127         }
00128     }
00129
00130     NodeEntity* getFromNode() const {
00131         return from;
00132     }
00133
00134     NodeEntity* getToNode() const {
00135         return to;
00136     }
00137
00138     EmptyEntity* getFromPort() {
00139         return from->children[fromPort];
00140     }
00141
00142     EmptyEntity* getToPort() {
00143         return to->children[toPort];
00144     }
00145
00146     virtual void updateLinkToPorts() {}
00147
00148     virtual void updateLinkToNodes() {}
00149
00150     virtual void updateArrowHeads() {}
00151
00152     virtual void addArrowHead() {}
00153
00154     virtual void removeArrowHead() {}
00155 };
```

6.49 GECSEntityTypes.h

```
00001 #pragma once
00002
00003 #include "GECSManager.h"
00004 #include "../Components.h"
00005
00006
00007 class Empty : public EmptyEntity {
00008
00009 public:
00010
00011     Empty(Manager& mManager) : EmptyEntity(mManager) {
00012
00013     }
00014
00015     void addGroup(Group mGroup) override {
00016         Entity::addGroup(mGroup);
00017         manager.AddToGroup(this, mGroup);
00018     }
00019
00020     virtual ~Empty() {
00021
00022     }
00023
00024     void update(float deltaTime)
00025     {
00026         //cellUpdate();
```

```
00027
00028         Entity::update(deltaTime);
00029     }
00030
00031     void cellUpdate() override {
00032         if (this->ownerCell) {
00033             Cell* newCell = manager.grid->getCell(*this, manager.grid->getGridLevel());
00034             if (newCell != this->ownerCell) {
00035                 // Need to shift the entity
00036                 removeEntity();
00037                 manager.grid->addEmpty(this, newCell);
00038             }
00039         }
00040     }
00041
00042
00043
00044     void imgui_print() override {
00045         glm::vec2 position = this->GetComponent<TransformComponent>().getPosition();  // Make sure
      Entity class has a getPosition method
00046         ImGui::Text("Position: (%.2f, %.2f)", position.x, position.y);
00047     }
00048
00049     void destroy() {
00050         Entity::destroy();
00051         manager.aboutTo_updateActiveEntities(); // cant have it at destroy in baseclass
00052         // may need to also update Visible Entities
00053     }
00054 };
00055
00056 class Node: public NodeEntity {
00057 private:
00058
00059
00060     std::vector<std::string> messageLog;
00061 public:
00062
00063     Node(Manager& mManager) : NodeEntity(mManager) {
00064
00065
00066     }
00067
00068     void addGroup(Group mGroup) override {
00069         Entity::addGroup(mGroup);
00070         manager.AddToGroup(this, mGroup);
00071     }
00072
00073     virtual ~Node() {
00074
00075     }
00076
00077     void update(float deltaTime)
00078     {
00079         //cellUpdate();
00080
00081         Entity::update(deltaTime);
00082     }
00083
00084     void updatePorts(float deltaTime) override {
00085         // first set the new position of port based on parent size
00086         // then set its bodyPosition through the transformComponent
00087
00088         TransformComponent* tr = &GetComponent<TransformComponent>();
00089
00090         glm::vec3 m_position = glm::vec3(-tr->size.x / 2, 0.0f, 0.0f);
00091
00092         if (children["leftPort"]) {
00093             children["leftPort"]->GetComponent<TransformComponent>().position = m_position;
00094             children["leftPort"]->update(deltaTime);
00095         }
00096
00097         m_position = glm::vec3(tr->size.x / 2, 0.0f, 0.0f);
00098
00099         if (children["rightPort"]) {
00100             children["rightPort"]->GetComponent<TransformComponent>().position = m_position;
00101             children["rightPort"]->update(deltaTime);
00102         }
00103
00104         m_position = glm::vec3(0.0f, -tr->size.y / 2.0f, 0.0f);
00105
00106         if (children["topPort"]) {
00107             children["topPort"]->GetComponent<TransformComponent>().position = m_position;
00108             children["topPort"]->update(deltaTime);
00109         }
00110
00111         m_position = glm::vec3(0.0f, tr->size.y / 2.0f, 0.0f);
00112
```

```
00113         if (children["bottomPort"]) {
00114             children["bottomPort"]->GetComponent<TransformComponent>().position = m_position;
00115             children["bottomPort"]->update(deltaTime);
00116         }
00117     }
00118 
00119     void cellUpdate() override{
00120         if (this->ownerCell) {
00121             updatePorts(0.0f);
00122             //this->GetComponent<TransformComponent>().update(0.0f);
00123             Cell* newCell = manager.grid->getCell(*this, manager.grid->getGridLevel());
00124             if (newCell != this->ownerCell) {
00125                 std::scoped_lock lock(manager.movedNodesMutex);
00126                 removeEntity();
00127                 manager.grid->addNode(this, newCell);
00128 
00129                 manager.movedNodes.push_back(this);
00130             }
00131             for (auto& link : inLinks) {
00132                 link->updateArrowHeads();
00133             }
00134             for (auto& link : outLinks) {
00135                 link->updateArrowHeads();
00136             }
00137         }
00138     }
00139 
00140     void addMessage(std::string mMessage) override{
00141         messageLog.push_back(mMessage);
00142     }
00143 
00144     void imgui_print() override {
00145         glm::vec2 position = this->GetComponent<TransformComponent>().getPosition();  // Make sure
      Entity class has a getPosition method
00146         ImGui::Text("Position: (%.2f, %.2f)", position.x, position.y);
00147 
00148 
00149         if (ImGui::BeginTable("GroupsTable", 1, ImGuiTableFlags_Borders | ImGuiTableFlags_RowBg)) {
00150             ImGui::TableSetupColumn("Message Log", ImGuiTableColumnFlags_WidthStretch);
00151 
00152             ImGui::TableHeadersRow();
00153             for (auto str : messageLog) {
00154                 ImGui::TableNextRow();
00155                 ImGui::TableSetColumnIndex(0);
00156                 ImGui::Text("%s", str.c_str());
00157             }
00158         }
00159         ImGui::EndTable();
00160 
00161 
00162     }
00163 
00164     void imgui_display() override {
00165         ImGui::Text("Display Info Here Node");
00166     }
00167 
00168     void destroy() {
00169         Entity::destroy();
00170         manager.aboutTo_updateActiveEntities();
00171     }
00172 
00173     void addPorts() {
00174         TransformComponent* tr = &GetComponent<TransformComponent>();
00175 
00176         auto& leftPort = getManager()->addEntityNoId<Empty>();
00177         glm::vec3 m_position = glm::vec3(-tr->size.x / 2, 0.0f, 0.0f);
00178         leftPort.addComponent<TransformComponent>(m_position, glm::vec3(0), 1.0f);
00179         children["leftPort"] = &leftPort;
00180         children["leftPort"]->setParentEntity(this);
00181         children["leftPort"]->GetComponent<TransformComponent>().bodyCenter = tr->bodyCenter +
      m_position;
00182 
00183         auto& rightPort = getManager()->addEntityNoId<Empty>();
00184         m_position = glm::vec3(tr->size.x / 2 , 0.0f, 0.0f);
00185         rightPort.addComponent<TransformComponent>(m_position, glm::vec3(0), 1.0f);
00186         children["rightPort"] = &rightPort;
00187         children["rightPort"]->setParentEntity(this);
00188         children["rightPort"]->GetComponent<TransformComponent>().bodyCenter = tr->bodyCenter +
      m_position;
00189 
00190         // Initialize top port
00191         auto& topPort = getManager()->addEntityNoId<Empty>();
00192         m_position = glm::vec3(0.0f, -tr->size.y / 2.0f, 0.0f);
00193         topPort.addComponent<TransformComponent>(m_position, glm::vec3(0), 1.0f);
00194         children["topPort"] = &topPort;
00195         children["topPort"]->setParentEntity(this);
00196         children["topPort"]->GetComponent<TransformComponent>().bodyCenter = tr->bodyCenter +
```

```
    m_position;
00197
00198         // Initialize bottom port
00199         auto& bottomPort = getManager()->addEntityNoId<Empty>();
00200         m_position = glm::vec3(0.0f, tr->size.y / 2.0f, 0.0f);
00201         bottomPort.addComponent<TransformComponent>(m_position, glm::vec3(0), 1.0f);
00202         children["bottomPort"] = &bottomPort;
00203         children["bottomPort"]->setParentEntity(this);
00204         children["bottomPort"]->GetComponent<TransformComponent>().bodyCenter = tr->bodyCenter +
    m_position;
00205
00206     }
00207
00208     void removePorts() {
00209         for (auto portName : { "leftPort", "rightPort", "topPort", "bottomPort" }) {
00210             if (children[portName]) {
00211                 children.erase(portName);
00212             }
00213         }
00214     }
00215 };
00216
00217
00218 class Link : public LinkEntity {
00219 private:
00220
00221
00222
00223 public:
00224
00225
00226     Color color = {};
00227
00228     Link(Manager& mManager) : LinkEntity(mManager) {
00229     }
00230
00231     Link(Manager& mManager, unsigned int mfromId, unsigned int mtoId)
00232         : LinkEntity(mManager, mfromId, mtoId)
00233     {
00234         from = dynamic_cast<NodeEntity*>(mManager.getEntityFromId(fromId));
00235         from->addOutLink(this);
00236         to = dynamic_cast<NodeEntity*>(mManager.getEntityFromId(toId));
00237         to->addInLink(this);
00238     }
00239
00240     Link(Manager& mManager, Entity* mfrom, Entity* mto)
00241         : LinkEntity(mManager,
00242             dynamic_cast<NodeEntity*>(mfrom), // it is node but cant see it due to getParentEntity
00243             dynamic_cast<NodeEntity*>(mto))
00244     {
00245         fromId = from->getId();
00246         toId = to->getId();
00247     }
00248
00249     Link(Manager& mManager, NodeEntity* mfrom, NodeEntity* mto)
00250         : LinkEntity(mManager,
00251             mfrom,
00252             mto)
00253     {
00254         fromId = from->getId();
00255         toId = to->getId();
00256     }
00257
00258     void addGroup(Group mGroup) override {
00259         Entity::addGroup(mGroup);
00260         manager.AddLinkToGroup(this, mGroup);
00261     }
00262
00263     virtual ~Link() {
00264
00265     }
00266
00267     void update(float deltaTime) override
00268     {
00269         Entity::update(deltaTime);
00270
00271
00272     }
00273
00274     void cellUpdate() override {
00275         // if cell(or position) of fromNode or cell(or position) of toNode is different than
00276         // the saved cells in ownerCells then update it
00277         if (!ownerCells.empty()) {
00278             auto level = manager.grid->getGridLevel();
00279             const auto& fromCell = manager.grid->getCell(*getFromNode(), level);
00280             const auto& toCell = manager.grid->getCell(*getToNode(), level);
00281
```

```
00282             const auto& ownerFront = ownerCells.front();
00283             const auto& ownerBack = ownerCells.back();
00284 
00285             if (fromCell != ownerFront
00286                 || toCell != ownerBack)
00287             {
00288                 removeFromCells();
00289 
00290                 manager.grid->addLink(this, manager.grid->getGridLevel());
00291             }
00292         }
00293 
00294     }
00295 
00296     void updateArrowHeads() override {
00297         if (children["ArrowHead"]) {
00298             TransformComponent* tr = &children["ArrowHead"]->GetComponent<TransformComponent>();
00299 
00300             children["ArrowHead"]->update(0.0f);
00301 
00302             // set position of arrowHead
00303             TransformComponent* ch_tr = &to->children[toPort]->GetComponent<TransformComponent>();
00304 
00305             TransformComponent* toPortTR = ch_tr;
00306             TransformComponent* fromPortTR =
      &from->children[fromPort]->GetComponent<TransformComponent>();
00307 
00308             glm::vec3 direction = toPortTR->getCenterTransform() - fromPortTR->getCenterTransform();
00309 
00310             glm::vec3 unitDirection = glm::normalize(direction);
00311             float offset = 10.0f;
00312 
00313             glm::vec3 arrowHeadPos = toPortTR->getCenterTransform() - unitDirection * offset;
00314 
00315             // Calculate the angle in radians, and convert it to degrees
00316             float angleRadians = -atan2(direction.y, direction.x);
00317             float angleDegrees = glm::degrees(angleRadians);
00318 
00319             glm::ivec3 arrowSize(10, 20, 0);
00320             glm::vec3 farrowSize(10.0f, 20.0f, 0.0f);
00321 
00322             glm::vec3 newArrowHeadPosition = arrowHeadPos - (farrowSize / 2.0f);
00323             children["ArrowHead"]->GetComponent<TransformComponent>().position = newArrowHeadPosition;
00324 
00325             children["ArrowHead"]->GetComponent<TransformComponent>().setRotation(glm::vec3(0.0f,
      0.0f, angleRadians + glm::half_pi<float>()));
00326         }
00327     }
00328 
00329     void updateLinkToPorts() override{
00330         TransformComponent* toTR = &to->GetComponent<TransformComponent>();
00331         TransformComponent* fromTR = &from->GetComponent<TransformComponent>();
00332 
00333         fromPort = getBestPortForConnection(fromTR->getCenterTransform(), toTR->getCenterTransform());
00334         toPort = getBestPortForConnection(toTR->getCenterTransform(), fromTR->getCenterTransform());
00335     }
00336 
00337     void addArrowHead() override {
00338         TransformComponent* toTR = &to->GetComponent<TransformComponent>();
00339         TransformComponent* fromTR = &from->GetComponent<TransformComponent>();
00340 
00341         fromPort = getBestPortForConnection(fromTR->getCenterTransform(), toTR->getCenterTransform());
00342         toPort = getBestPortForConnection(toTR->getCenterTransform(), fromTR->getCenterTransform());
00343 
00344         TransformComponent* toPortTR = &to->children[toPort]->GetComponent<TransformComponent>();
00345         TransformComponent* fromPortTR =
      &from->children[fromPort]->GetComponent<TransformComponent>();
00346 
00347         glm::vec3 direction = toPortTR->getCenterTransform() - fromPortTR->getCenterTransform();
00348 
00349         glm::vec3 unitDirection = glm::normalize(direction);
00350         float offset = 10.0f;
00351 
00352         glm::vec3 arrowHeadPos = toPortTR->getCenterTransform() - unitDirection * offset;
00353 
00354         auto& temp_arrowHead = getManager()->addEntityNoId<Empty>();
00355 
00356         // Calculate the angle in radians, and convert it to degrees
00357         float angleRadians = -atan2(direction.y, direction.x);
00358         float angleDegrees = glm::degrees(angleRadians);
00359 
00360         glm::vec3 farrowSize(10.0f, 20.0f, 0.0f);
00361 
00362         temp_arrowHead.addComponent<TransformComponent>(arrowHeadPos - (farrowSize /2.0f), farrowSize,
      1);
00363         temp_arrowHead.addComponent<Triangle_w_Color>();
00364         temp_arrowHead.GetComponent<Triangle_w_Color>().color = Color(0, 0, 0, 255);
```

```
00365
00366         temp_arrowHead.GetComponent<TransformComponent>().setRotation(glm::vec3(0.0f, 0.0f,
      angleRadians + glm::half_pi<float>()));
00367
00368         temp_arrowHead.addGroup(Manager::groupArrowHeads_0);
00369
00370         temp_arrowHead.setParentEntity(this);
00371
00372         manager.grid->addEmpty(&temp_arrowHead, manager.grid->getGridLevel());
00373
00374         children["ArrowHead"] = &temp_arrowHead;
00375     }
00376
00377     void removeArrowHead() override {
00378         std::string portName = "ArrowHead";
00379         if (children[portName]) {
00380             children[portName]->removeFromCell();
00381             children[portName]->destroy();
00382             children.erase(portName);
00383         }
00384     }
00385
00386     void updateLinkToNodes() override {
00387         fromPort = "";
00388         toPort = "";
00389     }
00390
00391     std::string getBestPortForConnection(const glm::vec3& fromPos, const glm::vec3& toPos) {
00392         // Simple logic to determine the port based on relative position
00393         float deltaX = toPos.x - fromPos.x;
00394         float deltaY = toPos.y - fromPos.y;
00395
00396         if (abs(deltaX) > abs(deltaY)) {  // Horizontal distance is greater
00397             return deltaX > 0 ? "rightPort" : "leftPort";
00398         }
00399         else {  // Vertical distance is greater
00400             return deltaY > 0 ? "bottomPort" : "topPort";
00401         }
00402     }
00403
00404
00405
00406
00407     void imgui_print() override {
00408         glm::vec2 fromNodePosition =
      this->getFromNode()->GetComponent<TransformComponent>().getCenterTransform();
00409         glm::vec2 toNodePosition =
      this->getToNode()->GetComponent<TransformComponent>().getCenterTransform();
00410
00411         ImGui::Text("From Node Position: (%.2f, %.2f)", fromNodePosition.x, fromNodePosition.y);
00412         ImGui::Text("To Node Position: (%.2f, %.2f)", toNodePosition.x, toNodePosition.y);
00413
00414         ImGui::Text("Bounding boxes of intercepted cells:");
00415
00416         for (auto cell : ownerCells) {
00417             ImGui::Text("- %.2f , %.2f , %.2f", cell->boundingBox_origin.x,
      cell->boundingBox_origin.y, cell->boundingBox_origin.z);
00418         }
00419     }
00420
00421     void imgui_display() override {
00422         ImGui::Text("Display Info Here Link");
00423     }
00424
00425     void destroy() {
00426         Entity::destroy();
00427
00428         if (children["ArrowHead"]) {
00429             children["ArrowHead"]->destroy();
00430         }
00431
00432         manager.aboutTo_updateActiveEntities();
00433     }
00434
00435 };
```

6.50 GECSManager.h

```
00001 #pragma once
00002
00003 #include "GECS.h"
00004 #include "../../Grid/Grid.h"
00005
```

```
00006 #include "../../Threader/Threader.h"
00007 #include <regex>
00008 #include <filesystem>
00009
00010 namespace fs = std::filesystem;
00011
00012
00013 class Manager
00014 {
00015 private:
00016     Threader* _threader = nullptr;
00017     int lastEntityId = 0;
00018     int negativeEntityId = -1;
00019     std::vector<std::unique_ptr<Entity>> entities;
00020
00021     std::array<std::vector<EmptyEntity*>, maxGroups> groupedEmptyEntities;
00022     std::array<std::vector<NodeEntity*>, maxGroups> groupedNodeEntities;
00023     std::array<std::vector<LinkEntity*>, maxGroups> groupedLinkEntities;
00024
00025     std::vector<EmptyEntity*> visible_emptyEntities;
00026     std::vector<NodeEntity*> visible_nodes;
00027     std::vector<LinkEntity*> visible_links;
00028
00029     std::array<std::vector<EmptyEntity*>, maxGroups> visible_groupedEmptyEntities;
00030     std::array<std::vector<NodeEntity*>, maxGroups> visible_groupedNodeEntities;
00031     std::array<std::vector<LinkEntity*>, maxGroups> visible_groupedLinkEntities;
00032
00033     bool _update_active_entities = false;
00034 public:
00035
00036     std::vector<NodeEntity*> movedNodes;
00037     std::mutex movedNodesMutex;
00038
00039     bool arrowheadsEnabled = false;
00040     bool last_arrowheadsEnabled = false;
00041
00042     std::unordered_map<std::string, std::vector<std::string>> componentNames;
00043
00044     std::unique_ptr<Grid> grid;
00045
00046     Manager() {}
00047
00048     ~Manager() { _threader = nullptr;  }
00049
00050     void setThreader(Threader& mthreader) {
00051         _threader = &mthreader;
00052     }
00053
00054     void update(float deltaTime = 1.0f)
00055     {
00057
00058         if (_threader && !_threader->t_queue.shuttingDown) {
00059
00061
00062             //_threader->parallel(visible_emptyEntities.size(), [&](int start, int end) {
00063             //  for (int i = start; i < end; i++) {
00064             //      if (visible_emptyEntities[i] && visible_emptyEntities[i]->isActive()) {
00065             //          visible_emptyEntities[i]->cellUpdate();
00066             //      }
00067             //  }
00068             //  });
00069
00070             //_threader->parallel(visible_nodes.size(), [&](int start, int end) {
00071             //  for (int i = start; i < end; i++) {
00072             //      if (visible_nodes[i] && visible_nodes[i]->isActive()) {
00073             //          visible_nodes[i]->cellUpdate();
00074             //      }
00075             //  }
00076             //  });
00077
00079             //? THIS MAY CAUSE ERRORS, IF REMOVE LINK FROM CELL AND OTHER LINK THAT HAS THAT CELL IN
      SEARCH
00080             //? WILL PUMP IN AN EMPTY ELEMENT OR THE SIZE WILL BE SMALLER FOR THAT LINK TO FIND
      ELEMENT
00081             for (auto& e : movedNodes) {
00082                 for (auto& link : e->getInLinks()) {
00083                     link->cellUpdate();
00084                 }
00085                 for (auto& link : e->getOutLinks()) {
00086                     link->cellUpdate();
00087                 }
00088             }
00089
00090             _threader->parallel(movedNodes.size(), [&](int start, int end) {
00091                 for (int i = start; i < end; i++) {
00092                     for (auto& link : movedNodes[i]->getInLinks()) {
00093                         link->updateLinkToPorts();
```

```
00094                     }
00095                 }
00096                 });
00097
00098             _threader->parallel(movedNodes.size(), [&](int start, int end) {
00099                 for (int i = start; i < end; i++) {
00100                     for (auto& link : movedNodes[i]->getOutLinks()) {
00101                         link->updateLinkToPorts();
00102                     }
00103                 }
00104                 });
00105
00106             movedNodes.clear();
00107
00109             _threader->parallel(visible_emptyEntities.size(), [&](int start, int end) {
00110                 for (int i = start; i < end; i++) {
00111                     if (visible_emptyEntities[i] && visible_emptyEntities[i]->isActive()) {
00112                         visible_emptyEntities[i]->update(deltaTime);
00113                     }
00114                 }
00115                 });
00116
00117
00118             _threader->parallel(visible_nodes.size(), [&](int start, int end) {
00119                 for (int i = start; i < end; i++) {
00120                     if (visible_nodes[i] && visible_nodes[i]->isActive()) {
00121                         visible_nodes[i]->update(deltaTime);
00122                     }
00123                 }
00124
00125                 });
00126
00127
00128             _threader->parallel(visible_links.size(), [&](int start, int end) {
00129
00130                 for (int i = start; i < end; i++) {
00131                     if (visible_links[i] && visible_links[i]->isActive()) {
00132                         visible_links[i]->update(deltaTime);
00133                     }
00134                 }
00135                 });
00136
00137         }
00138
00140         else {
00141
00143
00144             for (auto& e : movedNodes) {
00145                 for (auto& link : e->getInLinks()) {
00146                     link->cellUpdate();
00147                 }
00148                 for (auto& link : e->getOutLinks()) {
00149                     link->cellUpdate();
00150                 }
00151             }
00152
00153             for (auto& e : movedNodes) {
00154                 for (auto& link : e->getInLinks()) {
00155                     link->updateLinkToPorts();
00156                 }
00157             }
00158
00159             for (auto& e : movedNodes) {
00160                 for (auto& link : e->getOutLinks()) {
00161                     link->updateLinkToPorts();
00162                 }
00163             }
00164
00165             movedNodes.clear();
00166
00167             for (auto& e : visible_emptyEntities) {
00168                 if (!e || !e->isActive()) continue;
00169
00170                 e->update(deltaTime);
00171             }
00172
00173             if (arrowheadsEnabled) {
00174                 for (auto& e : visible_nodes) {
00175                     if (!e || !e->isActive()) continue;
00176
00177                     e->update(deltaTime);
00178
00179                 }
00180             }
00181
00182
00183             for (auto& e : visible_links) {
```

```
00184                 if (!e || !e->isActive()) continue;
00185
00186                 e->update(deltaTime);
00187             }
00188         }
00189     }
00190
00191     // update fully will update all nodes and links in the world
00192     void updateFully(float deltaTime = 1.0f)
00193     {
00194         // the links are updating once since after first update we check wether the nodes are aligned
      with the ownerCells
00195         for (auto& e : entities) {
00196             if (!e || !e->isActive()) continue;
00197
00198             e->update(deltaTime);
00199         }
00200     }
00201
00202     void refresh(ICamera* camera = nullptr)
00203     {
00204
00205         if (grid && (camera->hasChanged() || grid->gridLevelChanged())) {
00206             bool interceptedCellsChanged = grid->setIntersectedCameraCells(*camera);
00207
00208             if (interceptedCellsChanged) {
00209                 aboutTo_updateActiveEntities();
00210             }
00211             camera->refreshCamera();
00212         }
00213
00214         if (_update_active_entities) {
00215             _update_active_entities = false;
00216
00217             updateActiveEntities();
00218             updateVisibleEntities();
00219         }
00220
00221     }
00222
00223     void aboutTo_updateActiveEntities() {
00224         _update_active_entities = true;
00225     }
00226
00227     void updateActiveEntities();
00228
00229     void updateVisibleEntities();
00230
00231     void AddToGroup(EmptyEntity* mEntity, Group mGroup)
00232     {
00233         groupedEmptyEntities[mGroup].emplace_back(mEntity);
00234     }
00235
00236     void AddToGroup(NodeEntity* mEntity, Group mGroup)
00237     {
00238         groupedNodeEntities[mGroup].emplace_back(mEntity);
00239     }
00240
00241     void AddLinkToGroup(LinkEntity* mEntity, Group mGroup)
00242     {
00243         groupedLinkEntities[mGroup].emplace_back(mEntity);
00244     }
00245
00246     const std::vector<std::unique_ptr<Entity>>& getEntities() const {
00247         return entities;
00248     }
00249
00250     template <typename T>
00251     std::vector<T*> getVisible() {
00252         if constexpr (std::is_same_v<T, EmptyEntity>) {
00253             return visible_emptyEntities;
00254         }
00255         else if constexpr (std::is_same_v<T, NodeEntity>) {
00256             return visible_nodes;
00257         }
00258         else if constexpr (std::is_same_v<T, LinkEntity>) {
00259             return visible_links;
00260         }
00261         else {
00262             static_assert(sizeof(T) == 0, "Unsupported entity type.");
00263         }
00264     }
00265
00266     template <typename T>
00267     std::vector<T*>& getVisibleGroup(Group mGroup) {
00268         if constexpr (std::is_same_v<T, EmptyEntity>) {
00269             return visible_groupedEmptyEntities[mGroup];
```

```
00270         }
00271         else if constexpr (std::is_same_v<T, NodeEntity>) {
00272             return visible_groupedNodeEntities[mGroup];
00273         }
00274         else if constexpr (std::is_same_v<T, LinkEntity>) {
00275             return visible_groupedLinkEntities[mGroup];
00276         }
00277         else {
00278             static_assert(sizeof(T) == 0, "Unsupported entity type.");
00279         }
00280     }
00281 
00282     template <typename T>
00283     std::vector<T*>& getGroup(Group mGroup) {
00284         if constexpr (std::is_same_v<T, EmptyEntity>) {
00285             return groupedEmptyEntities[mGroup];
00286         }
00287         else if constexpr (std::is_same_v<T, NodeEntity>) {
00288             return groupedNodeEntities[mGroup];
00289         }
00290         else if constexpr (std::is_same_v<T, LinkEntity>) {
00291             return groupedLinkEntities[mGroup];
00292         }
00293         else {
00294             static_assert(sizeof(T) == 0, "Unsupported entity type.");
00295         }
00296     }
00297 
00298     template <typename T, typename... TArgs>
00299     T& addEntityNoId(TArgs&&... mArgs)
00300     {
00301         T* e(new T(*this, std::forward<TArgs>(mArgs)...));
00302         e->setId(negativeEntityId--);
00303         std::unique_ptr<T> uPtr{ e };
00304         entities.emplace_back(std::move(uPtr));
00305 
00306         return *e;
00307     }
00308 
00309     template <typename T, typename... TArgs>
00310     T& addEntity(TArgs&&... mArgs)
00311     {
00312         T* e(new T(*this, std::forward<TArgs>(mArgs)...));
00313         e->setId(lastEntityId++);
00314         std::unique_ptr<T> uPtr{ e };
00315         entities.emplace_back(std::move(uPtr));
00316 
00317         return *e;
00318     }
00319 
00320     void resetEntityId() {
00321         lastEntityId = 0;
00322     }
00323 
00324     Entity* getEntityFromId(unsigned int mId) {
00325         for (auto& entity : entities) {
00326             if (entity->getId() == mId && entity->isActive()) {
00327                 return &*entity;
00328             }
00329         }
00330         return nullptr;
00331     }
00332 
00333     void clearAllEntities() {
00334         for (auto& group : groupedNodeEntities) {
00335             group.clear();
00336         }
00337         for (auto& group : groupedLinkEntities) {
00338             group.clear();
00339         }
00340         entities.clear();
00341     }
00342 
00343     void removeAllEntites() {
00344         for (std::size_t group = Manager::groupBackgroundLayer; group != Manager::buttonLabels;
      group++) {
00345             removeAllEntitiesFromGroup(group);
00346             removeAllEntitiesFromLinkGroup(group);
00347         }
00348     }
00349 
00350     void removeAllEntitiesFromGroup(Group mGroup) {
00351         auto& entitiesInGroup = groupedNodeEntities[mGroup];
00352 
00353         for (Entity* entity : entitiesInGroup) {
00354             entity->destroy();
00355         }
```

```
00356     }
00357     void removeAllEntitiesFromLinkGroup(Group mGroup) {
00358         auto& entitiesInGroup = groupedLinkEntities[mGroup];
00359 
00360         for (Entity* entity : entitiesInGroup) {
00361             entity->destroy();
00362         }
00363     }
00364 
00365     std::vector<Entity*> adjacentEntities(Entity* mainEntity, Group group) {
00366         std::vector<Entity*> nearbyEntities;
00367 
00368         auto adjacentCells = grid->getAdjacentCells(*mainEntity, grid->getGridLevel());
00369 
00370         for (Cell* adjCell : adjacentCells) {
00371             for (auto& neighbor : adjCell->nodes) {
00372                 if (neighbor->hasGroup(group) && (neighbor != mainEntity) ) {
00373                     nearbyEntities.push_back(neighbor);
00374                 }
00375             }
00376         }
00377 
00378         return nearbyEntities;
00379     }
00380 
00381     enum groupLabels : std::size_t //todo should add groups at end for some reason
00382     {
00383         //back
00384         groupBackgroundLayer,
00385         panelBackground,
00386 
00387         //action
00388         groupLinks_0,
00389         groupGroupLinks_0,
00390         groupGroupLinks_1,
00391 
00392         groupArrowHeads_0,
00393 
00394         groupNodes_0,
00395         groupGroupNodes_0,
00396         groupGroupNodes_1,
00397         groupColliders,
00398 
00399         groupEmpties,
00400         groupSphereEmpties,
00401 
00402         groupRenderSprites,
00403 
00404         //fore
00405         buttonLabels,
00406     };
00407 
00408     const std::unordered_map<Group, std::string> groupNames = {
00409         {groupBackgroundLayer, "groupBackgroundLayer" },
00410         {panelBackground, "panelBackground"},
00411 
00412         //action
00413         { groupLinks_0,"groupLinks_0" },
00414         {groupGroupLinks_0, "groupGroupLinks_0"},
00415         {groupGroupLinks_1, "groupGroupLinks_1"},
00416 
00417         {groupArrowHeads_0, "groupArrowHeads_0"},
00418 
00419         { groupNodes_0,"groupNodes_0" },
00420         { groupGroupNodes_0, "groupGroupNodes_0"},
00421         { groupGroupNodes_1, "groupGroupNodes_1"},
00422 
00423         { groupEmpties,"groupEmpties" },
00424         { groupSphereEmpties,"groupSphereEmpties" },
00425 
00426         { groupColliders,"groupColliders" },
00427         { groupRenderSprites,"groupRenderSprites" },
00428 
00429 
00430         //fore
00431         { buttonLabels,"buttonLabels" },
00432     };
00433 
00434     std::string getGroupName(Group mGroup) const;
00435 
00436     void scanComponentNames(const std::string& folderPath);
00437 
00438     void setComponentNames();
00439 };
```

6.51 GECSUtil.h

```
00001 #pragma once
00002
00003 #include "./GECSManager.h"
00004 #include <unordered_map>
00005
00006 // Map of component names to functions for adding components
00007 static const std::unordered_map<std::string, std::function<void(Entity*)>> addComponentMap = {
00008     //{"TransformComponent", [](Entity* entity) { entity->addComponent<TransformComponent>(); }},
00009     {"SpriteComponent", [](Entity* entity)
      {entity->addGroup(Manager::groupRenderSprites);entity->addComponent<SpriteComponent>(); }},
00010     {"ColliderComponent", [](Entity* entity) { entity->addComponent<ColliderComponent>(); }},
00011     {"Triangle_w_Color", [](Entity* entity) { entity->addComponent<Triangle_w_Color>(); }},
00012     {"Rectangle_w_Color", [](Entity* entity) { entity->addComponent<Rectangle_w_Color>(); }},
00013     {"Line_w_Color", [](Entity* entity) { entity->addComponent<Line_w_Color>(); }},
00014     {"SpringComponent", [](Entity* entity) { entity->addComponent<SpringComponent>(); }},
00015     //{"BoxComponent", [](Entity* entity) { entity->addComponent<BoxComponent>(); }},
00016     {"AnimatorComponent", [](Entity* entity) { entity->addComponent<AnimatorComponent>(); }},
00017     {"MovingAnimatorComponent", [](Entity* entity) { entity->addComponent<MovingAnimatorComponent>();
      }},
00018     {"FlashAnimatorComponent", [](Entity* entity) { entity->addComponent<FlashAnimatorComponent>();
      }},
00019     {"LineFlashAnimatorComponent", [](Entity* entity) {
      entity->addComponent<LineFlashAnimatorComponent>(); }},
00020     {"RectangleFlashAnimatorComponent", [](Entity* entity) {
      entity->addComponent<RectangleFlashAnimatorComponent>(); }},
00021     {"UILabel", [](Entity* entity) { entity->addComponent<UILabel>(); }},
00022     {"ButtonComponent", [](Entity* entity) { entity->addComponent<ButtonComponent>(); }},
00023     {"RigidBodyComponent", [](Entity* entity) { entity->addComponent<RigidBodyComponent>(); }},
00024     {"KeyboardControllerComponent", [](Entity* entity) {
      entity->addComponent<KeyboardControllerComponent>(); }},
00025     {"GridComponent", [](Entity* entity) { entity->addComponent<GridComponent>(); }},
00026     {"PollingComponent", [](Entity* entity) { entity->addComponent<PollingComponent>(); }},
00027 };
00028
00029 // Map of component names to functions for removing components
00030 static const std::unordered_map<std::string, std::function<void(Entity*)>> removeComponentMap = {
00031     //{"TransformComponent", [](Entity* entity) { entity->removeComponent<TransformComponent>(); }},
00032     {"SpriteComponent", [](Entity* entity) { entity->removeGroup(Manager::groupRenderSprites);
      entity->removeComponent<SpriteComponent>(); }},
00033     {"ColliderComponent", [](Entity* entity) { entity->removeComponent<ColliderComponent>(); }},
00034     {"Triangle_w_Color", [](Entity* entity) { entity->removeComponent<Triangle_w_Color>(); }},
00035     {"Rectangle_w_Color", [](Entity* entity) { entity->removeComponent<Rectangle_w_Color>(); }},
00036     {"Line_w_Color", [](Entity* entity) { entity->removeComponent<Line_w_Color>(); }},
00037     {"SpringComponent", [](Entity* entity) { entity->removeComponent<SpringComponent>(); }},
00038     {"AnimatorComponent", [](Entity* entity) { entity->removeComponent<AnimatorComponent>(); }},
00039     {"MovingAnimatorComponent", [](Entity* entity) {
      entity->removeComponent<MovingAnimatorComponent>(); }},
00040     {"FlashAnimatorComponent", [](Entity* entity) { entity->removeComponent<FlashAnimatorComponent>();
      }},
00041     {"LineFlashAnimatorComponent", [](Entity* entity) {
      entity->removeComponent<LineFlashAnimatorComponent>(); }},
00042     {"RectangleFlashAnimatorComponent", [](Entity* entity) {
      entity->removeComponent<RectangleFlashAnimatorComponent>(); }},
00043     {"UILabel", [](Entity* entity) { entity->removeComponent<UILabel>(); }},
00044     {"ButtonComponent", [](Entity* entity) { entity->removeComponent<ButtonComponent>(); }},
00045     {"RigidBodyComponent", [](Entity* entity) { entity->removeComponent<RigidBodyComponent>(); }},
00046     {"KeyboardControllerComponent", [](Entity* entity) {
      entity->removeComponent<KeyboardControllerComponent>(); }},
00047     {"GridComponent", [](Entity* entity) { entity->removeComponent<GridComponent>(); }},
00048     {"PollingComponent", [](Entity* entity) { entity->removeComponent<PollingComponent>(); }},
00049 };
00050
00051 // Map of component names to functions for getting components
00052 static const std::unordered_map<std::string, std::function<BaseComponent* (Entity*)>> getComponentMap =
      {
00053     {"TransformComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<TransformComponent>(); }},
00054     {"SpriteComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<SpriteComponent>(); }},
00055     {"ColliderComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<ColliderComponent>(); }},
00056     {"Triangle_w_Color", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<Triangle_w_Color>(); }},
00057     {"Rectangle_w_Color", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<Rectangle_w_Color>(); }},
00058     {"Line_w_Color", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<Line_w_Color>(); }},
00059     {"SpringComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<SpringComponent>(); }},
00060     {"AnimatorComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<AnimatorComponent>(); }},
00061     {"MovingAnimatorComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<MovingAnimatorComponent>(); }},
00062     {"FlashAnimatorComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<FlashAnimatorComponent>(); }},
```

```
00063     {"LineFlashAnimatorComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<LineFlashAnimatorComponent>(); }},
00064     {"RectangleFlashAnimatorComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<RectangleFlashAnimatorComponent>(); }},
00065     {"UILabel", [](Entity* entity) -> BaseComponent* { return &entity->GetComponent<UILabel>(); }},
00066     {"ButtonComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<ButtonComponent>(); }},
00067     {"RigidBodyComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<RigidBodyComponent>(); }},
00068     {"KeyboardControllerComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<KeyboardControllerComponent>(); }},
00069     {"GridComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<GridComponent>(); }},
00070     {"PollingComponent", [](Entity* entity) -> BaseComponent* { return
      &entity->GetComponent<PollingComponent>(); }},
00071 };
00072
00073
00074 // Function to add a component by name
00075 void AddComponentByName(const std::string& componentName, Entity* entity) {
00076     auto it = addComponentMap.find(componentName);
00077     if (it != addComponentMap.end()) {
00078         it->second(entity);
00079     }
00080
00081     entity->getManager()->aboutTo_updateActiveEntities();
00082 }
00083
00084 // Function to remove a component by name
00085 void RemoveComponentByName(const std::string& componentName, Entity* entity) {
00086     auto it = removeComponentMap.find(componentName);
00087     if (it != removeComponentMap.end()) {
00088         it->second(entity);
00089     }
00090
00091     entity->getManager()->aboutTo_updateActiveEntities();
00092 }
00093
00094 // Function to get a component by name
00095 BaseComponent* getComponentByName(const std::string& componentName, Entity* entity) {
00096     auto it = getComponentMap.find(componentName);
00097     if (it != getComponentMap.end()) {
00098         return it->second(entity);
00099     }
00100     return nullptr;
00101 }
```

6.52 UtilComponents.h

```
00001 #pragma once
00002
00003 #include "./Core/GECSEntityTypes.h"
00004
00005 #include "./Components/Empty/Util/ColliderComponent.h"
00006 #include "./Components/Empty/Util/UILabel.h"
00007 #include "./Components/Empty/Util/ButtonComponent.h"
00008 #include "./Components/Empty/Util/RigidBodyComponent.h"
00009 #include "./Components/Empty/Util/KeyboardControllerComponent.h"
00010 #include "./Components/Empty/Util/GridComponent.h"
00011 #include "./Components/Node/Util/PollingComponent.h"
```

6.53 GLSLProgram.h

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <string>
00005 #include "GL/glew.h"
00006
00007 #include <vector>
00008 #include <fstream>
00009
00010 #include "./Vertex.h"
00011
00012 #include "TextureManager/TextureManager.h"
00013 #include "ConsoleLogger.h"
00014
00015 #define LINE_OFFSET 2
00016 #define SQUARE_OFFSET 4
```

```
00017
00018 #define TRIANGLE_VERTICES 3
00019 #define QUAD_INDICES 6
00020 #define BOX_OFFSET 8
00021
00022 #define TOTAL_MESHES 4
00023
00024 #define TRIANGLE_MESH_IDX 0
00025 #define RECTANGLE_MESH_IDX 1
00026 #define BOX_MESH_IDX 2
00027 #define SPHERE_MESH_IDX 3
00028
00029 constexpr int INDICES_LINE_OFFSET = LINE_OFFSET;
00030 constexpr int INDICES_SQUARE_OFFSET = 2 * SQUARE_OFFSET;
00031 constexpr int INDICES_BOX_OFFSET = 3 * BOX_OFFSET;
00032
00033 constexpr int ARRAY_BOX_OFFSET = 36;
00034
00035
00036 static Position triangleVertices[3] = {
00037     {  0.0f,  0.5f, 0.0f }, // Top
00038     { -0.5f, -0.5f, 0.0f }, // Bottom Left
00039     {  0.5f, -0.5f, 0.0f }  // Bottom Right
00040 };
00041
00042 static GLuint triangleIndices[3] = {
00043     0, 1, 2
00044 };
00045
00046 static Position quadVertices[4] = {
00047     {   -0.5f,  0.5f,   0.0f    },
00048     {   -0.5f,  -0.5f,  0.0f    },
00049     {   0.5f,   -0.5f,  0.0f    },
00050     {   0.5f,   0.5f,   0.0f    }
00051 };
00052
00053 static UV uv_quadVertices[4] = {
00054     {0.0f, 0.0f},
00055     {1.0f, 0.0f},
00056     {1.0f, 1.0f},
00057     {0.0f, 1.0f}
00058 };
00059
00060 static TextureVertex tex_quadVertices[4] = {
00061     { glm::vec3(-0.5f,  0.5f, 0.0f),  glm::vec2(0.0f, 1.0f) }, // top-left
00062     { glm::vec3(-0.5f, -0.5f, 0.0f),  glm::vec2(0.0f, 0.0f) }, // bottom-left
00063     { glm::vec3(0.5f, -0.5f, 0.0f),  glm::vec2(1.0f, 0.0f) }, // bottom-right
00064     { glm::vec3(0.5f,  0.5f, 0.0f),  glm::vec2(1.0f, 1.0f) }  // top-right
00065 };
00066
00067 static GLuint quadIndices[6] = {
00068     0, 1, 2,
00069     2, 3, 0
00070 };
00071
00072 static Position cubeVertices[8] = {
00073     { -0.5f, -0.5f, -0.5f },
00074     {  0.5f, -0.5f, -0.5f },
00075     {  0.5f,  0.5f, -0.5f },
00076     { -0.5f,  0.5f, -0.5f },
00077     { -0.5f, -0.5f,  0.5f },
00078     {  0.5f, -0.5f,  0.5f },
00079     {  0.5f,  0.5f,  0.5f },
00080     { -0.5f,  0.5f,  0.5f }
00081 };
00082
00083 static LightVertex light_cubeVertices[24] = {
00084     // Front face
00085     { glm::vec3(-0.5f, -0.5f, -0.5f), glm::vec3(0.0f, 0.0f, -1.0f) },
00086     { glm::vec3(0.5f, -0.5f, -0.5f),  glm::vec3(0.0f, 0.0f, -1.0f) },
00087     { glm::vec3(0.5f, 0.5f, -0.5f),   glm::vec3(0.0f, 0.0f, -1.0f) },
00088     { glm::vec3(-0.5f, 0.5f, -0.5f),  glm::vec3(0.0f, 0.0f, -1.0f) },
00089
00090     // Back face
00091     { glm::vec3(-0.5f, -0.5f, 0.5f),  glm::vec3(0.0f, 0.0f, 1.0f) },
00092     { glm::vec3(0.5f, -0.5f, 0.5f),   glm::vec3(0.0f, 0.0f, 1.0f) },
00093     { glm::vec3(0.5f, 0.5f, 0.5f),    glm::vec3(0.0f, 0.0f, 1.0f) },
00094     { glm::vec3(-0.5f, 0.5f, 0.5f),   glm::vec3(0.0f, 0.0f, 1.0f) },
00095
00096     // Left face
00097     { glm::vec3(-0.5f, -0.5f, -0.5f), glm::vec3(-1.0f, 0.0f, 0.0f) },
00098     { glm::vec3(-0.5f, -0.5f, 0.5f),  glm::vec3(-1.0f, 0.0f, 0.0f) },
00099     { glm::vec3(-0.5f, 0.5f, 0.5f),   glm::vec3(-1.0f, 0.0f, 0.0f) },
00100     { glm::vec3(-0.5f, 0.5f, -0.5f),  glm::vec3(-1.0f, 0.0f, 0.0f) },
00101
00102     // Right face
00103     { glm::vec3(0.5f, -0.5f, -0.5f),  glm::vec3(1.0f, 0.0f, 0.0f) },
```

```
00104     { glm::vec3(0.5f, -0.5f, 0.5f),   glm::vec3(1.0f, 0.0f, 0.0f) },
00105     { glm::vec3(0.5f, 0.5f, 0.5f),    glm::vec3(1.0f, 0.0f, 0.0f) },
00106     { glm::vec3(0.5f, 0.5f, -0.5f),   glm::vec3(1.0f, 0.0f, 0.0f) },
00107
00108     // Bottom face
00109     { glm::vec3(-0.5f, -0.5f, -0.5f), glm::vec3(0.0f, -1.0f, 0.0f) },
00110     { glm::vec3(0.5f, -0.5f, -0.5f),  glm::vec3(0.0f, -1.0f, 0.0f) },
00111     { glm::vec3(0.5f, -0.5f, 0.5f),   glm::vec3(0.0f, -1.0f, 0.0f) },
00112     { glm::vec3(-0.5f, -0.5f, 0.5f),  glm::vec3(0.0f, -1.0f, 0.0f) },
00113
00114     // Top face
00115     { glm::vec3(-0.5f, 0.5f, -0.5f),  glm::vec3(0.0f, 1.0f, 0.0f) },
00116     { glm::vec3(0.5f, 0.5f, -0.5f),   glm::vec3(0.0f, 1.0f, 0.0f) },
00117     { glm::vec3(0.5f, 0.5f, 0.5f),    glm::vec3(0.0f, 1.0f, 0.0f) },
00118     { glm::vec3(-0.5f, 0.5f, 0.5f),   glm::vec3(0.0f, 1.0f, 0.0f) }
00119 };
00120
00121 static GLuint cubeIndices[36] = {
00122      0, 1, 2,
00123     2, 3, 0,
00124
00125     // Back face (vertices 4-7)
00126     4, 5, 6,
00127     6, 7, 4,
00128
00129     // Left face (vertices 8-11)
00130     8, 9,10,
00131     10,11,8,
00132
00133     // Right face (vertices 12-15)
00134     12,13,14,
00135     14,15,12,
00136
00137     // Bottom face (vertices 16-19)
00138     16,17,18,
00139     18,19,16,
00140
00141     // Top face (vertices 20-23)
00142     20,21,22,
00143     22,23,20
00144 };
00145
00146
00147 static void generateSphereMesh(std::vector<Position>& vertices, std::vector<GLuint>& indices,
00148     float radius = 1.0f, unsigned int sectorCount = 36, unsigned int stackCount = 18) {
00149     const float PI = 3.14159265359f;
00150
00151     vertices.clear();
00152     indices.clear();
00153
00154     for (unsigned int i = 0; i <= stackCount; ++i) {
00155         float stackAngle = PI / 2.0f - i * PI / stackCount;  // from pi/2 to -pi/2
00156         float xy = radius * cosf(stackAngle);
00157         float z = radius * sinf(stackAngle);
00158
00159         for (unsigned int j = 0; j <= sectorCount; ++j) {
00160             float sectorAngle = j * 2.0f * PI / sectorCount;
00161
00162             float x = xy * cosf(sectorAngle);
00163             float y = xy * sinf(sectorAngle);
00164             glm::vec3 pos(x, y, z);
00165
00166             vertices.push_back(pos);
00167         }
00168     }
00169
00170     for (unsigned int i = 0; i < stackCount; ++i) {
00171         unsigned int k1 = i * (sectorCount + 1);
00172         unsigned int k2 = k1 + sectorCount + 1;
00173
00174         for (unsigned int j = 0; j < sectorCount; ++j, ++k1, ++k2) {
00175             if (i != 0) {
00176                 indices.push_back(k1);
00177                 indices.push_back(k2);
00178                 indices.push_back(k1 + 1);
00179             }
00180             if (i != (stackCount - 1)) {
00181                 indices.push_back(k1 + 1);
00182                 indices.push_back(k2);
00183                 indices.push_back(k2 + 1);
00184             }
00185         }
00186     }
00187 }
00188
00189 static void generateSphereMesh(std::vector<LightVertex>& vertices, std::vector<GLuint>& indices,
00190     float radius = 1.0f, unsigned int sectorCount = 36, unsigned int stackCount = 18) {
```

```
00191     const float PI = 3.14159265359f;
00192
00193     vertices.clear();
00194     indices.clear();
00195
00196     for (unsigned int i = 0; i <= stackCount; ++i) {
00197         float stackAngle = PI / 2.0f - i * PI / stackCount;  // from pi/2 to -pi/2
00198         float xy = radius * cosf(stackAngle);
00199         float z = radius * sinf(stackAngle);
00200
00201         for (unsigned int j = 0; j <= sectorCount; ++j) {
00202             float sectorAngle = j * 2.0f * PI / sectorCount;
00203
00204             float x = xy * cosf(sectorAngle);
00205             float y = xy * sinf(sectorAngle);
00206             glm::vec3 pos(x, y, z);
00207
00208             glm::vec3 normal = glm::normalize(pos);
00209             vertices.push_back({ pos, normal });
00210         }
00211     }
00212
00213     for (unsigned int i = 0; i < stackCount; ++i) {
00214         unsigned int k1 = i * (sectorCount + 1);
00215         unsigned int k2 = k1 + sectorCount + 1;
00216
00217         for (unsigned int j = 0; j < sectorCount; ++j, ++k1, ++k2) {
00218             if (i != 0) {
00219                 indices.push_back(k1);
00220                 indices.push_back(k2);
00221                 indices.push_back(k1 + 1);
00222             }
00223             if (i != (stackCount - 1)) {
00224                 indices.push_back(k1 + 1);
00225                 indices.push_back(k2);
00226                 indices.push_back(k2 + 1);
00227             }
00228         }
00229     }
00230 }
00231
00232 struct InstanceData {
00233
00234     InstanceData() {}
00235     InstanceData(glm::vec3 mSize, Position mBodyCenter, Rotation mRotation) :
00236         size(mSize),
00237         bodyCenter(mBodyCenter),
00238         rotation(mRotation)
00239     {
00240     }
00241
00242     InstanceData(glm::vec2 mSize, Position mBodyCenter, Rotation mRotation) :
00243         size(glm::vec3(mSize, 0.0f)),
00244         bodyCenter(mBodyCenter),
00245         rotation(mRotation)
00246     {
00247     }
00248
00249     ~InstanceData() {}
00250
00251     Size size = glm::vec3(0.0f);
00252     Position bodyCenter = glm::vec3(0.0f);
00253     Rotation rotation = glm::vec3(0.0f);
00254 };
00255
00256
00257 struct ColorInstanceData : InstanceData {
00258
00259     ColorInstanceData() {}
00260     ColorInstanceData(glm::vec3 mSize, Position mBodyCenter, Rotation mRotation, Color mColor) :
      InstanceData(mSize, mBodyCenter, mRotation), color(mColor) {
00261     }
00262     ColorInstanceData(glm::vec2 mSize, Position mBodyCenter, Rotation mRotation, Color mColor) :
      InstanceData(mSize, mBodyCenter, mRotation), color(mColor) {
00263     }
00264
00265     ~ColorInstanceData() {};
00266
00267     Color color = Color(255, 255, 255, 255);
00268 };
00269
00270 struct TextureInstanceData : InstanceData {
00271
00272     TextureInstanceData() {}
00273     TextureInstanceData(glm::vec3 mSize, Position mBodyCenter, Rotation mRotation, GLuint Texture) :
      InstanceData(mSize, mBodyCenter, mRotation), texture(Texture) {
00274     }
```

```
00275     TextureInstanceData(glm::vec2 mSize, Position mBodyCenter, Rotation mRotation, GLuint Texture) :
      InstanceData(mSize, mBodyCenter, mRotation), texture(Texture) {
00276     }
00277
00278     ~TextureInstanceData() {};
00279
00280     GLuint texture = 0;
00281     UV uv = glm::vec2(0.0f);
00282 };
00283
00284 struct MeshRenderer {
00285     size_t meshIndices = 0;
00286
00287     std::vector<InstanceData> instances;
00288
00289     GLuint vao;
00290
00291     GLuint vbo; //for static draws
00292     GLuint ibo;
00293 };
00294
00295 struct ColorMeshRenderer {
00296     size_t meshIndices = 0;
00297
00298     std::vector<ColorInstanceData> instances;
00299
00300     GLuint vao;
00301
00302     GLuint vbo; //for static draws
00303     GLuint ibo;
00304 };
00305
00306 struct TextureMeshRenderer {
00307     size_t meshIndices = 0;
00308
00309     std::vector<TextureInstanceData> instances;
00310
00311     GLuint vao;
00312
00313     GLuint vbo; //for static draws
00314     GLuint ibo;
00315 };
00316
00317 class GLSLProgram {
00318 public:
00319     GLSLProgram() : _programID(0), _vertexShaderID(0), _fragmentShaderID(0), _numAttributes(0)
00320     {
00321
00322     }
00323
00324     ~GLSLProgram() {
00325
00326     }
00327
00328     void compileShaders(const std::string& vertexShaderFilePath, const std::string&
      fragmentShaderFilePath) {
00329
00330         std::vector<unsigned char> vertSourceVec;
00331         std::vector<unsigned char> fragSourceVec;
00332
00333         TextureManager::readFileToBuffer(vertexShaderFilePath.c_str(), vertSourceVec);
00334         TextureManager::readFileToBuffer(fragmentShaderFilePath.c_str(), fragSourceVec);
00335
00336         std::string vertSource(vertSourceVec.begin(), vertSourceVec.end());
00337         std::string fragSource(fragSourceVec.begin(), fragSourceVec.end());
00338
00339         compileShadersFromSource(vertSource.c_str(), fragSource.c_str());
00340     }
00341
00342     void compileShadersFromSource(const char* vertexSource, const char* fragmentSource) {
00343
00344         // Vertex and fragment shaders are successfully compiled.
00345         // Now time to link them together into a program.
00346         // Get a program object.
00347         _programID = glCreateProgram();
00348
00349         _vertexShaderID = glCreateShader(GL_VERTEX_SHADER);
00350         if (_vertexShaderID == 0) {
00351             TazGraphEngine::ConsoleLogger::error("Vertex Shader Failed to create!");
00352         }
00353
00354         _fragmentShaderID = glCreateShader(GL_FRAGMENT_SHADER);
00355         if (_fragmentShaderID == 0) {
00356             TazGraphEngine::ConsoleLogger::error("Fragment Shader Failed to create!");
00357         }
00358
00359         compileShader(vertexSource, "vertex Shader", _vertexShaderID);
```

```
00360         compileShader(fragmentSource, "fragment Shader", _fragmentShaderID);
00361     }
00362
00363     void linkShaders() {
00364         // Attach our shaders to our program
00365         glAttachShader(_programID, _vertexShaderID);
00366         glAttachShader(_programID, _fragmentShaderID);
00367
00368         // Link our program
00369         glLinkProgram(_programID);
00370
00371         // Note the different functions here: glGetProgram* instead of glGetShader*.
00372         GLint isLinked = 0;
00373         glGetProgramiv(_programID, GL_LINK_STATUS, (int*)&isLinked);
00374         if (isLinked == GL_FALSE)
00375         {
00376             GLint maxLength = 0;
00377             glGetProgramiv(_programID, GL_INFO_LOG_LENGTH, &maxLength);
00378
00379             // The maxLength includes the NULL character
00380             std::vector<GLchar> errorLog(maxLength);
00381             glGetProgramInfoLog(_programID, maxLength, &maxLength, &errorLog[0]);
00382
00383             // We don't need the program anymore.
00384             glDeleteProgram(_programID);
00385             // Don't leak shaders either.
00386             glDeleteShader(_vertexShaderID);
00387             glDeleteShader(_fragmentShaderID);
00388
00389             std::printf("%s\n", &(errorLog[0]));
00390             TazGraphEngine::ConsoleLogger::error("Shaders failed to link");
00391         }
00392
00393         // Always detach shaders after a successful link.
00394         glDetachShader(_programID, _vertexShaderID);
00395         glDetachShader(_programID, _fragmentShaderID);
00396         glDeleteShader(_vertexShaderID);
00397         glDeleteShader(_fragmentShaderID);
00398     }
00399
00400     void addAttribute(const std::string& attributeName) {
00401         glBindAttribLocation(_programID, _numAttributes++, attributeName.c_str());
00402     }
00403
00404     GLint getUniformLocation(const std::string& uniformName) {
00405         GLint location = glGetUniformLocation(_programID, uniformName.c_str());
00406
00407         if (location == GL_INVALID_INDEX) {
00408             TazGraphEngine::ConsoleLogger::error("Uniform " + uniformName + " not found in shader!");
00409         }
00410         return location;
00411     }
00412
00413     void use() {
00414         glUseProgram(_programID);
00415         for (int i = 0; i < _numAttributes; i++) {
00416             glEnableVertexAttribArray(i);
00417         }
00418     }
00419
00420     void unuse() {
00421         glUseProgram(0);
00422         for (int i = 0; i < _numAttributes; i++) {
00423             glDisableVertexAttribArray(i);
00424         }
00425     }
00426
00427     void dispose() {
00428         if(_programID != 0) glDeleteProgram(_programID);
00429     }
00430
00431     GLuint getProgramID() {
00432         return _programID;
00433     }
00434
00435 private:
00436     GLuint _programID;
00437
00438     GLuint _vertexShaderID;
00439     GLuint _fragmentShaderID;
00440
00441     int _numAttributes;
00442
00443     void compileShader(const char* source, const std::string& name, GLuint id) {
00444
00445         glShaderSource(id, 1, &source, nullptr); //1 for number of strings
00446
```

```
00447         glCompileShader(id);
00448
00449         GLint success = 0;
00450         glGetShaderiv(id, GL_COMPILE_STATUS, &success);
00451
00452         if (success == GL_FALSE)
00453         {
00454             GLint maxLength = 0;
00455             glGetShaderiv(id, GL_INFO_LOG_LENGTH, &maxLength);
00456
00457             // The maxLength includes the NULL character
00458             std::vector<GLchar> errorLog(maxLength);
00459             glGetShaderInfoLog(id, maxLength, &maxLength, &errorLog[0]);
00460
00461             // Provide the infolog in whatever manor you deem best.
00462             // Exit with failure.
00463             glDeleteShader(id); // Don't leak the shader.
00464
00465             std::printf("%s\n", &(errorLog[0]));
00466             std::cout << "Shader " + name + " failed to compile" << std::endl;
00467
00468         }
00469     }
00470 };
```

6.54 GLTexture.h

```
00001 #pragma once
00002
00003 #include <GL/glew.h>
00004
00005 struct GLTexture {
00006     GLuint id;
00007     int width;
00008     int height;
00009 };
```

6.55 AppInterface.h

```
00001 #pragma once
00002
00003 #include "../InputManager/InputManager.h"
00004 #include "../Window/Window.h"
00005 #include <memory>
00006 #include <iostream>
00007
00008 #include "SceneList.h"
00009 #include "../BaseFPSLimiter/BaseFPSLimiter.h"
00010
00011 #include "../AudioEngine/AudioEngine.h"
00012
00013 #include "../Threader/Threader.h"
00014
00015
00016 class SceneList;
00017 class IScene;
00018
00019 class AppInterface {
00020 public:
00021     AppInterface(int threadCount);
00022     virtual ~AppInterface();
00023
00024     void run();
00025     void exitSimulator();
00026
00027     virtual void onInit() = 0;
00028     virtual void addScenes() = 0;
00029     virtual void onExit() = 0;
00030
00031     void onSDLEvent(SDL_Event& evnt);
00032
00033     InputManager _inputManager;
00034     TazGraphEngine::Window _window;
00035
00036     BaseFPSLimiter& getFPSLimiter() { return _limiter; }
00037     AudioEngine& getAudioEngine() { return _audioEngine; }
00038
00039     Threader threadPool;
00040
```

```
00041 protected:
00042     virtual void checkInput();
00043     virtual void update(float deltaTime);
00044     virtual void draw();
00045     virtual void updateUI();
00046
00047     bool init();
00048     bool initSystems();
00049
00050     BaseFPSLimiter _limiter;
00051     AudioEngine _audioEngine;
00052
00053     std::unique_ptr<SceneList> _sceneList = nullptr;
00054     IScene* _currentScene = nullptr;
00055     bool _isRunning = false;
00056
00057     const float SCALE_SPEED = 0.1f;
00058 };
```

6.56 IScene.h

```
00001 #pragma once
00002
00003 #include "../Renderers/LineRenderer/LineRenderer.h"
00004 #include "../DataManager/DataManager.h"
00005
00006
00007 #include "../GECS/Core/GECSManager.h"
00008
00009 #define SCENE_INDEX_NO_SCENE -1
00010
00011 class AppInterface;
00012
00013 enum class SceneState {
00014     NONE,
00015     RUNNING,
00016     EXIT_APPLICATION,
00017     CHANGE_NEXT,
00018     CHANGE_PREVIOUS
00019 };
00020
00021 class IScene {
00022 public:
00023     friend class SceneList;
00024     IScene() {
00025
00026     }
00027     virtual ~IScene() {
00028
00029     }
00030
00031     //Returns the index of the next or previous screen when changing screens
00032     virtual int getNextSceneIndex() const = 0;
00033     virtual int getPreviousSceneIndex() const = 0;
00034
00035     //Called at beginning and end of application
00036     virtual void build() = 0;
00037     virtual void destroy() = 0;
00038
00039     //Called when a screen enters and exits focus
00040     virtual void onEntry() = 0;
00041     virtual void onExit() = 0;
00042
00043     virtual void checkInput() = 0;
00044
00045     virtual void update(float deltaTime) = 0;
00046     virtual void draw() = 0;
00047
00048     virtual void BeginRender() = 0;
00049     virtual void updateUI() = 0;
00050     virtual void EndRender() = 0;
00051
00052     int getSceneIndex() const {
00053         return _sceneIndex;
00054     }
00055     void setRunning() {
00056         _currentState = SceneState::RUNNING;
00057     }
00058
00059     SceneState getState() const { return _currentState; }
00060
00061     void setParentApp(AppInterface* app) { _app = app; }
00062
```

```
00063     AppInterface* getApp() const { return _app; }
00064
00065     void setManager(std::string m_managerName) {
00066         if (!m_managerName.empty()) {
00067             auto it = managers.find(m_managerName);
00068             if (it == managers.end()) {
00069                 managers[m_managerName] = new Manager();
00070             }
00071             manager = managers[m_managerName];
00072             managerName = m_managerName;
00073         }
00074     };
00075
00076 protected:
00077     SceneState _currentState = SceneState::NONE;
00078     AppInterface* _app = nullptr;
00079     int _sceneIndex = -1;
00080
00081     std::unordered_map<std::string, Manager*> managers = {
00082     };
00083
00084     Manager* manager = nullptr;
00085     std::string managerName = "";
00086
00087     bool _renderDebug = false;
00088     bool _clusterLayout = false;
00089
00090 };
```

6.57 SceneList.h

```
00001 #pragma once
00002 #include <vector>
00003 #include <string>
00004
00005 class AppInterface;
00006 class IScene;
00007
00008 class SceneList {
00009 public:
00010     SceneList(AppInterface* app);
00011     ~SceneList();
00012
00013     IScene* moveNext();
00014     IScene* movePrevious();
00015
00016     void setScene(int nextScene);
00017     void addScene(IScene* newScene);
00018     void addScene(std::string managerName, IScene* newScene);
00019
00020     void destroy();
00021
00022     IScene* getCurrent();
00023
00024 protected:
00025     AppInterface* _app = nullptr;
00026     std::vector<IScene*> _scenes;
00027     int _currentSceneIndex = -1;
00028 };
```

6.58 ScreenIndices.h

```
00001 #pragma once
00002
00003
00004 const int SCENE_INDEX_MAIN_MENU = 0;
00005 const int SCENE_INDEX_GRAPHPLAY = 1;
```

6.59 Grid.h

```
00001 #pragma once
00002 #include "../GECS/Core/GECSEntity.h"
00003 #include "../GECS/Components.h"
00004
00005 #include "../AABB/AABB.h"
```

```
00006
00007 #include <vector>
00008
00009 #include <cmath>
00010
00011 constexpr int CELL_SIZE = 100;
00012 constexpr int AXIS_CELLS = 80;
00013 constexpr int DEPTH_AXIS_CELLS = 4;
00014 constexpr int ROW_CELL_SIZE = AXIS_CELLS * CELL_SIZE;
00015 constexpr int COLUMN_CELL_SIZE = AXIS_CELLS * CELL_SIZE;
00016 constexpr int DEPTH_CELL_SIZE = DEPTH_AXIS_CELLS * CELL_SIZE;
00017
00018
00019 struct GridLevelData {
00020     float numXCells, numYCells, numZCells;
00021     float startX, endX, startY, endY, startZ, endZ;
00022     float cameraMargin = 0.0f;
00023 };
00024
00025 class Grid {
00026 public:
00027     enum Level {
00028         Basic,
00029         Outer1,
00030         Outer2
00031     };
00032
00033     Grid(int width, int height, int depth, int cellSize);
00034     ~Grid();
00035
00036     void setSize(int cellSize);
00037     void init(int width, int height, int depth, int cellSize);
00038
00039     void createCells(Grid::Level size);
00040
00041     void addLink(LinkEntity* link, Grid::Level m_level);
00042     std::vector<Cell*> getLinkCells(const LinkEntity& link, Grid::Level m_level);
00043     void addLink(LinkEntity* link, std::vector<Cell*> cell);
00044
00045     void addEmpty(EmptyEntity* entity, Grid::Level m_level);
00046
00047     void addNode(NodeEntity* entity, Grid::Level m_level);
00048     void addEmpty(EmptyEntity* entity, Cell* cell);
00049     void addNode(NodeEntity* entity, Cell* cell);
00050
00051     Cell* getCell(int x, int y, int z, Grid::Level m_level);
00052     Cell* getCell(const Entity& position, Grid::Level m_level);
00053     std::vector<Cell*> getAdjacentCells(int x, int y, int z, Grid::Level m_level);
00054     std::vector<Cell*> getAdjacentCells(const Entity& entity, Grid::Level m_level);
00055     std::vector<Cell>& getCells(Grid::Level m_level);
00056     int getCellSize();
00057     int getNumXCells();
00058     int getNumYCells();
00059     int getNumZCells();
00060
00061     bool setIntersectedCameraCells(ICamera& camera);
00062
00063     std::vector<Cell*> getIntersectedCameraCells(ICamera& camera);
00064
00065     template <typename T>
00066     std::vector<T*> getRevealedEntitiesInCameraCells() {
00067         std::vector<T*> result;
00068
00069         if constexpr (std::is_same_v<T, NodeEntity>) {
00070             for (auto& cell : _interceptedCells) {
00071                 for (auto& entity : cell->nodes) {
00072                     if (!entity->isHidden()) {  // Check if the entity is visible
00073                         result.push_back(entity);
00074                     }
00075                 }
00076             }
00077         }
00078         else if constexpr (std::is_same_v<T, EmptyEntity>) {
00079             for (auto& cell : _interceptedCells) {
00080                 for (auto& entity : cell->emptyEntities) {
00081                     if (!entity->isHidden()) {  // Check if the entity is visible
00082                         result.push_back(entity);
00083                     }
00084                 }
00085             }
00086         }
00087         else if constexpr (std::is_same_v<T, LinkEntity>) {
00088             std::map<unsigned int, LinkEntity*> uniqueEntities;
00089
00090             for (auto& cell : _interceptedCells) {
00091                 for (auto& link : cell->links) {
00092                     if (!link->isHidden()) {
```

```
00093                         unsigned int linkId = link->getId();
00094
00095                         if (uniqueEntities.find(linkId) == uniqueEntities.end()) {
00096                             uniqueEntities[linkId] = link;
00097                         }
00098                     }
00099                 }
00100             }
00101             for (auto& entry : uniqueEntities) {
00102                 result.push_back(entry.second);
00103             }
00104         }
00105         else {
00106             static_assert(sizeof(T) == 0, "Unsupported entity type.");
00107         }
00108         return result;
00109
00110     }
00111
00112     template <typename T>
00113     std::vector<T*> getEntitiesInCameraCells() {
00114         std::vector<T*> result;
00115
00116         if constexpr (std::is_same_v<T, NodeEntity>) {
00117             for (auto& cell : _interceptedCells) {
00118                 result.insert(result.end(), cell->nodes.begin(), cell->nodes.end());
00119             }
00120         }
00121         else if constexpr (std::is_same_v<T, EmptyEntity>) {
00122             for (auto& cell : _interceptedCells) {
00123                 result.insert(result.end(), cell->emptyEntities.begin(), cell->emptyEntities.end());
00124             }
00125         }
00126         else {
00127             static_assert(sizeof(T) == 0, "Unsupported entity type.");
00128         }
00129         return result;
00130     }
00131
00132     std::vector<LinkEntity*> getLinksInCameraCells();
00133
00134
00135     bool gridLevelChanged();
00136
00137     Level getGridLevel();
00138     void setGridLevel(Level newLevel);
00139
00140     int getLevelCellScale();
00141
00142     int getLevelCellScale(Level level);
00143 private:
00144     std::vector<Cell*> _interceptedCells;
00145
00146     std::vector<Cell> _cells;
00147     std::vector<Cell> _parentCells;
00148     std::vector<Cell> _superParentCells;
00149
00150     int _cellSize;
00151
00152     int _width;
00153     int _height;
00154     int _depth;
00155
00156     int _numXCells;
00157     int _numYCells;
00158     int _numZCells;
00159
00160     // can change between different scenes/managers
00161     std::map<Level, GridLevelData> gridLevelsData;
00162
00163     std::map<Level, int> gridLevels = {
00164         {Level::Basic,  1},
00165         {Level::Outer1, 2},
00166         {Level::Outer2, 4}
00167     };
00168
00169     Level _level = Level::Basic;
00170     Level _lastLevel = Level::Basic;
00171 };
```

6.60 ImGuiInterface.h

```
00001 #pragma once
```

```
00002
00003 #include <iostream>
00004 #include <string>
00005
00006 #include <imgui.h>
00007 #include <imgui_impl_sdl2.h>
00008 #include <imgui_impl_opengl3.h>
00009 #include <implot.h>
00010 #include <implot_internal.h>
00011 #include "../imguiComboAutoselect/imgui_combo_autoselect.h"
00012
00013 class ImGuiInterface {
00014 public:
00015     ImGuiInterface();
00016
00017     ~ImGuiInterface();
00018
00019     //void SetupImGui();
00020
00021     void BeginRender();
00022
00023     void RenderUI();
00024
00025     void EndRender();
00026 };
```

6.61 InputManager.h

```
00001 #pragma once
00002
00003 #include <unordered_map>
00004 #define GLM_ENABLE_EXPERIMENTAL
00005 #include <glm/glm.hpp>
00006
00007 #include <SDL2/SDL.h>
00008
00009 #include "../Camera2.5D/ICamera.h"
00010 class InputManager {
00011 public:
00012     InputManager();
00013     ~InputManager();
00014
00015     void update();
00016
00017     void pressKey(unsigned int keyID);
00018     void releaseKey(unsigned int keyID);
00019
00020     //returns true if the key is held down
00021     bool isKeyDown(unsigned int keyID);
00022
00023     //returns true if the key was just pressed
00024     bool isKeyPressed(unsigned int keyID);
00025
00026     bool checkMouseCollision(glm::vec2 position, glm::ivec2 tr_size);
00027     void setMouseCoords(float x, float y);
00028
00029     glm::vec2 getMouseCoords() const;
00030
00031     // Panning
00032     void setPanningPoint(glm::vec2 position);
00033     glm::vec2 calculatePanningDelta(glm::vec2 position);
00034
00035     void setObjectRelativePos(glm::vec2 relativeObjectPos);
00036     glm::vec2 getObjectRelativePos();
00037
00038     glm::vec2 convertWindowToCameraCoords(glm::vec2 mousePos,
00039         glm::vec2 viewportSize,
00040         glm::vec2 windowDimensions,
00041         const glm::vec2& windowPos, const glm::vec2& windowSize,
00042         const ICamera& camera);
00043
00044     /*glm::vec2 convertCameraToWindowCoords(glm::vec2 mousePos,
00045         glm::vec2 viewportSize,
00046         glm::vec2 windowDimensions,
00047         const glm::vec2& windowPos, const glm::vec2& windowSize,
00048         const ICamera& camera);*/
00049
00050
00051 private:
00052     bool wasKeyDown(unsigned int keyID);
00053
00054     std::unordered_map<unsigned int, bool> _keyMap;
00055     std::unordered_map<unsigned int, bool> _prevKeyMap;
```

```
00056
00057     glm::vec2 _mouseCoords = glm::vec2(0);
00058
00059     glm::vec2 _panningPoint = glm::vec2(0);
00060
00061     glm::vec2 _relativeObjectPos = glm::vec2(0);
00062 };
```

6.62 JsonParser.h

```
00001 #pragma once
00002
00003 #include <fstream>
00004 #include <iostream>
00005 #include <map>
00006 #include <vector>
00007 #include <string>
00008 #include <sstream>
00009 #include <cctype>
00010 #include <stdexcept>
00011
00012 enum class JsonType {
00013     Object, Array, String, Number, Boolean, Null
00014 };
00015
00016 struct NumericStringCompare {
00017     bool operator()(const std::string& a, const std::string& b) const {
00018         bool a_numeric = isNumeric(a);
00019         bool b_numeric = isNumeric(b);
00020
00021         if (a_numeric && b_numeric) {
00022             return std::stod(a) < std::stod(b);
00023         }
00024         return a < b; // fallback to alphabetical for non-numeric
00025     }
00026
00027 private:
00028     bool isNumeric(const std::string& s) const {
00029         char* end = nullptr;
00030         std::strtod(s.c_str(), &end);
00031         return end != s.c_str() && *end == '\0';
00032     }
00033 };
00034 struct JsonValue {
00035     JsonType type = JsonType::Object;
00036     std::map<std::string, JsonValue, NumericStringCompare> obj;
00037     std::vector<JsonValue> arr;
00038     std::string str = "";
00039     double num = -1;
00040     bool boolean = false;
00041 };
00042
00043 class JsonParser {
00044 public:
00045     explicit JsonParser(const std::string& input) : input(input), pos(0) {}
00046
00047     explicit JsonParser(std::ifstream& file) {
00048         std::stringstream buffer;
00049         buffer « file.rdbuf();
00050         input = buffer.str();
00051         pos = 0;
00052
00053         pos = input.find('{');
00054         input = input.substr(pos);
00055         pos = 0;
00056     }
00057
00058     JsonValue parse() {
00059         JsonValue value = parseValue();
00060         skipWhitespace();
00061         if (!eof()) {
00062             throw std::runtime_error("Unexpected characters at end of input.");
00063         }
00064         return value;
00065     }
00066
00067 private:
00068     std::string input;
00069     size_t pos;
00070
00071     void skipWhitespace() {
00072         while (!eof() && std::isspace(input[pos])) {
00073             pos++;
```

```
00074         }
00075     }
00076
00077     bool eof() const {
00078         return pos >= input.size();
00079     }
00080
00081     char peek() const {
00082         return input[pos];
00083     }
00084
00085     char consume() {
00086         return input[pos++];
00087     }
00088
00089     JsonValue parseValue() {
00090         skipWhitespace();
00091         if (eof()) {
00092             throw std::runtime_error("Unexpected end of input.");
00093         }
00094
00095         char ch = peek();
00096         if (ch == '{') return parseObject();
00097         if (ch == '[') return parseArray();
00098         if (ch == '\"' || ch == '\'') return parseString();
00099         if (std::isdigit(ch) || ch == '-') return parseNumber();
00100         if (ch == 't' || ch == 'T' || ch == 'f' || ch == 'F') return parseBoolean();
00101         if (ch == 'n') return parseNull();
00102
00103         throw std::runtime_error("Unexpected character.");
00104     }
00105
00106     JsonValue parseObject() {
00107         consume(); // '{'
00108         JsonValue obj;
00109         obj.type = JsonType::Object;
00110
00111         skipWhitespace();
00112         if (peek() == '}') {
00113             consume();
00114             return obj;
00115         }
00116
00117         while (true) {
00118             skipWhitespace();
00119             JsonValue key;
00120             if (std::isdigit(peek()) || peek() == '-') {
00121                 key = parseNumber();
00122                 key.type = JsonType::String;
00123                 key.str = std::to_string(key.num);
00124             }
00125             else if (peek() == '\"' || peek() == '\'') {
00126                 key = parseString();
00127             }
00128             skipWhitespace();
00129             if (consume() != ':') {
00130                 throw std::runtime_error("Expected ':' after key in object.");
00131             }
00132             JsonValue value = parseValue();
00133             obj.obj[key.str] = value;
00134
00135             skipWhitespace();
00136             char ch = consume();
00137             if (ch == '}' || ch == '\0') break;
00138             if (ch != ',') {
00139                 throw std::runtime_error("Expected ',' or '}' in object.");
00140             }
00141         }
00142         return obj;
00143     }
00144
00145     JsonValue parseArray() {
00146         consume(); // '['
00147         JsonValue array;
00148         array.type = JsonType::Array;
00149
00150         skipWhitespace();
00151         if (peek() == ']') {
00152             consume();
00153             return array;
00154         }
00155
00156         while (true) {
00157             JsonValue value = parseValue();
00158             array.arr.push_back(value);
00159
00160             skipWhitespace();
```

```
00161             char ch = consume();
00162             if (ch == ']') break;
00163             if (ch != ',') {
00164                 throw std::runtime_error("Expected ',' or ']' in array.");
00165             }
00166         }
00167         return array;
00168     }
00169
00170     JsonValue parseString() {
00171         std::ostringstream result;
00172         consume(); // the initial '"'
00173         while (true) {
00174             if (eof()) throw std::runtime_error("Unexpected end of input during string parsing.");
00175
00176             char ch = consume();
00177             if (ch == '"' || ch == '\"') break; // end of string
00178             if (ch == '\\') { // handle escapes
00179                 if (eof()) throw std::runtime_error("Unexpected end of input during escape
    sequence.");
00180                 char esc = consume();
00181                 switch (esc) {
00182                 case '"': result << '"'; break;
00183                 case '\": result << '\"; break;
00184                 case '\': result << '\'; break;
00185                 case '/':  result << '/'; break;
00186                 case 'b':  result << '\b'; break;
00187                 case 'f':  result << '\f'; break;
00188                 case 'n':  result << '\n'; break;
00189                 case 'r':  result << '\r'; break;
00190                 case 't':  result << '\t'; break;
00191                 default: throw std::runtime_error("Invalid escape sequence.");
00192                 }
00193             }
00194             else {
00195                 result << ch;
00196             }
00197         }
00198
00199         JsonValue strValue;
00200         strValue.type = JsonType::String;
00201         strValue.str = result.str();
00202         return strValue;
00203     }
00204
00205     JsonValue parseNumber() {
00206         std::ostringstream result;
00207         if (peek() == '-') {
00208             result << consume();
00209         }
00210
00211         while (!eof() && (std::isdigit(peek()) || peek() == '.')) {
00212             result << consume();
00213         }
00214         if (!eof() && (peek() == 'e' || peek() == 'E')) {
00215             result << consume();
00216
00217             if (!eof() && (peek() == '+' || peek() == '-')) {
00218                 result << consume();
00219             }
00220
00221             while (!eof() && std::isdigit(peek())) {
00222                 result << consume();
00223             }
00224         }
00225
00226         JsonValue number;
00227         number.type = JsonType::Number;
00228         number.num = std::stod(result.str());
00229         return number;
00230     }
00231
00232     JsonValue parseBoolean() {
00233         std::ostringstream result;
00234         for (int i = 0; i < 4 && !eof(); ++i) { // "true" or "fals"
00235             result << consume();
00236         }
00237
00238         JsonValue boolValue;
00239         boolValue.type = JsonType::Boolean;
00240         std::string res = result.str();
00241         if (res == "true" || res == "True") {
00242             boolValue.boolean = true;
00243         }
00244         else if (res == "false" || res == "False") {
00245             boolValue.boolean = false;
00246         }
```

```
00247         else {
00248             throw std::runtime_error("Invalid value for boolean.");
00249         }
00250         return boolValue;
00251     }
00252
00253     JsonValue parseNull() {
00254         std::string result;
00255         for (int i = 0; i < 4 && !eof(); ++i) { // "null"
00256             result += consume();
00257         }
00258         if (result != "null") {
00259             throw std::runtime_error("Invalid value for null.");
00260         }
00261         JsonValue nullValue;
00262         nullValue.type = JsonType::Null;
00263         return nullValue;
00264     }
00265 };
```

6.63 picoPNG.h

```
00001 #pragma once
00002
00003 #include <vector>
00004
00005 /*
00006 decodePNG: The picoPNG function, decodes a PNG file buffer in memory, into a raw pixel buffer.
00007 out_image: output parameter, this will contain the raw pixels after decoding.
00008   By default the output is 32-bit RGBA color.
00009   The std::vector is automatically resized to the correct size.
00010 image_width: output_parameter, this will contain the width of the image in pixels.
00011 image_height: output_parameter, this will contain the height of the image in pixels.
00012 in_png: pointer to the buffer of the PNG file in memory. To get it from a file on
00013   disk, load it and store it in a memory buffer yourself first.
00014 in_size: size of the input PNG file in bytes.
00015 convert_to_rgba32: optional parameter, true by default.
00016   Set to true to get the output in RGBA 32-bit (8 bit per channel) color format
00017   no matter what color type the original PNG image had. This gives predictable,
00018   useable data from any random input PNG.
00019   Set to false to do no color conversion at all. The result then has the same data
00020   type as the PNG image, which can range from 1 bit to 64 bits per pixel.
00021   Information about the color type or palette colors are not provided. You need
00022   to know this information yourself to be able to use the data so this only
00023   works for trusted PNG files. Use LodePNG instead of picoPNG if you need this information.
00024 return: 0 if success, not 0 if some error occured.
00025 */
00026 int decodePNG(std::vector<unsigned char>& out_image, unsigned long& image_width, unsigned long&
      image_height, const unsigned char* in_png, size_t in_size, bool convert_to_rgba32)
00027 {
00028     // picoPNG version 20101224
00029     // Copyright (c) 2005-2010 Lode Vandevenne
00030     //
00031     // This software is provided 'as-is', without any express or implied
00032     // warranty. In no event will the authors be held liable for any damages
00033     // arising from the use of this software.
00034     //
00035     // Permission is granted to anyone to use this software for any purpose,
00036     // including commercial applications, and to alter it and redistribute it
00037     // freely, subject to the following restrictions:
00038     //
00039     //     1. The origin of this software must not be misrepresented; you must not
00040     //     claim that you wrote the original software. If you use this software
00041     //     in a product, an acknowledgment in the product documentation would be
00042     //     appreciated but is not required.
00043     //     2. Altered source versions must be plainly marked as such, and must not be
00044     //     misrepresented as being the original software.
00045     //     3. This notice may not be removed or altered from any source distribution.
00046
00047     // picoPNG is a PNG decoder in one C++ function of around 500 lines. Use picoPNG for
00048     // programs that need only 1 .cpp file. Since it's a single function, it's very limited,
00049     // it can convert a PNG to raw pixel data either converted to 32-bit RGBA color or
00050     // with no color conversion at all. For anything more complex, another tiny library
00051     // is available: LodePNG (lodepng.c(pp)), which is a single source and header file.
00052     // Apologies for the compact code style, it's to make this tiny.
00053
00054     static const unsigned long LENBASE[29] = {
      3,4,5,6,7,8,9,10,11,13,15,17,19,23,27,31,35,43,51,59,67,83,99,115,131,163,195,227,258 };
00055     static const unsigned long LENEXTRA[29] = { 0,0,0,0,0,0,0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3,
      4, 4, 4,  4,  5,  5,  5,  5,  0 };
00056     static const unsigned long DISTBASE[30] = {
      1,2,3,4,5,7,9,13,17,25,33,49,65,97,129,193,257,385,513,769,1025,1537,2049,3073,4097,6145,8193,12289,16385,24577
      };
```

```
00057     static const unsigned long DISTEXTRA[30] = { 0,0,0,0,1,1,2, 2, 3, 3, 4, 4, 5, 5,  6,  6,  7,  7,
      8,  8,   9,   9,  10,  10,  11,  11,  12,   12,   13,   13 };
00058     static const unsigned long CLCL[19] = { 16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14,
      1, 15 }; //code length code lengths
00059     struct Zlib //nested functions for zlib decompression
00060     {
00061         static unsigned long readBitFromStream(size_t& bitp, const unsigned char* bits) { unsigned
      long result = (bits[bitp » 3] » (bitp & 0x7)) & 1; bitp++; return result; }
00062         static unsigned long readBitsFromStream(size_t& bitp, const unsigned char* bits, size_t nbits)
00063         {
00064             unsigned long result = 0;
00065             for (size_t i = 0; i < nbits; i++) result += (readBitFromStream(bitp, bits)) « i;
00066             return result;
00067         }
00068         struct HuffmanTree
00069         {
00070             int makeFromLengths(const std::vector<unsigned long>& bitlen, unsigned long maxbitlen)
00071             { //make tree given the lengths
00072                 unsigned long numcodes = (unsigned long)(bitlen.size()), treepos = 0, nodefilled = 0;
00073                 std::vector<unsigned long> tree1d(numcodes), blcount(maxbitlen + 1, 0),
      nextcode(maxbitlen + 1, 0);
00074                 for (unsigned long bits = 0; bits < numcodes; bits++) blcount[bitlen[bits]]++; //count
      number of instances of each code length
00075                 for (unsigned long bits = 1; bits <= maxbitlen; bits++) nextcode[bits] =
      (nextcode[bits - 1] + blcount[bits - 1]) « 1;
00076                 for (unsigned long n = 0; n < numcodes; n++) if (bitlen[n] != 0) tree1d[n] =
      nextcode[bitlen[n]]++; //generate all the codes
00077                 tree2d.clear(); tree2d.resize(numcodes * 2, 32767); //32767 here means the tree2d
      isn't filled there yet
00078                 for (unsigned long n = 0; n < numcodes; n++) //the codes
00079                     for (unsigned long i = 0; i < bitlen[n]; i++) //the bits for this code
00080                     {
00081                         unsigned long bit = (tree1d[n] » (bitlen[n] - i - 1)) & 1;
00082                         if (treepos > numcodes - 2) return 55;
00083                         if (tree2d[2 * treepos + bit] == 32767) //not yet filled in
00084                         {
00085                             if (i + 1 == bitlen[n]) { tree2d[2 * treepos + bit] = n; treepos = 0; }
      //last bit
00086                             else { tree2d[2 * treepos + bit] = ++nodefilled + numcodes; treepos =
      nodefilled; } //addresses are encoded as values > numcodes
00087                         }
00088                         else treepos = tree2d[2 * treepos + bit] - numcodes; //subtract numcodes from
      address to get address value
00089                     }
00090                 return 0;
00091             }
00092             int decode(bool& decoded, unsigned long& result, size_t& treepos, unsigned long bit) const
00093             { //Decodes a symbol from the tree
00094                 unsigned long numcodes = (unsigned long)tree2d.size() / 2;
00095                 if (treepos >= numcodes) return 11; //error: you appeared outside the codetree
00096                 result = tree2d[2 * treepos + bit];
00097                 decoded = (result < numcodes);
00098                 treepos = decoded ? 0 : result - numcodes;
00099                 return 0;
00100             }
00101             std::vector<unsigned long> tree2d; //2D representation of a huffman tree: The one
      dimension is "0" or "1", the other contains all nodes and leaves of the tree.
00102         };
00103         struct Inflator
00104         {
00105             int error;
00106             void inflate(std::vector<unsigned char>& out, const std::vector<unsigned char>& in, size_t
      inpos = 0)
00107             {
00108                 size_t bp = 0, pos = 0; //bit pointer and byte pointer
00109                 error = 0;
00110                 unsigned long BFINAL = 0;
00111                 while (!BFINAL && !error)
00112                 {
00113                     if (bp » 3 >= in.size()) { error = 52; return; } //error, bit pointer will jump
      past memory
00114                     BFINAL = readBitFromStream(bp, &in[inpos]);
00115                     unsigned long BTYPE = readBitFromStream(bp, &in[inpos]); BTYPE += 2 *
      readBitFromStream(bp, &in[inpos]);
00116                     if (BTYPE == 3) { error = 20; return; } //error: invalid BTYPE
00117                     else if (BTYPE == 0) inflateNoCompression(out, &in[inpos], bp, pos, in.size());
00118                     else inflateHuffmanBlock(out, &in[inpos], bp, pos, in.size(), BTYPE);
00119                 }
00120                 if (!error) out.resize(pos); //Only now we know the true size of out, resize it to
      that
00121             }
00122             void generateFixedTrees(HuffmanTree& tree, HuffmanTree& treeD) //get the tree of a
      deflated block with fixed tree
00123             {
00124                 std::vector<unsigned long> bitlen(288, 8), bitlenD(32, 5);;
00125                 for (size_t i = 144; i <= 255; i++) bitlen[i] = 9;
00126                 for (size_t i = 256; i <= 279; i++) bitlen[i] = 7;
```

```
00127                 tree.makeFromLengths(bitlen, 15);
00128                 treeD.makeFromLengths(bitlenD, 15);
00129             }
00130             HuffmanTree codetree, codetreeD, codelengthcodetree; //the code tree for Huffman codes,
      dist codes, and code length codes
00131             unsigned long huffmanDecodeSymbol(const unsigned char* in, size_t& bp, const HuffmanTree&
      codetree, size_t inlength)
00132             { //decode a single symbol from given list of bits with given code tree. return value is
      the symbol
00133                 bool decoded; unsigned long ct;
00134                 for (size_t treepos = 0;;)
00135                 {
00136                     if ((bp & 0x07) == 0 && (bp >> 3) > inlength) { error = 10; return 0; } //error:
      end reached without endcode
00137                     error = codetree.decode(decoded, ct, treepos, readBitFromStream(bp, in)); if
      (error) return 0; //stop, an error happened
00138                     if (decoded) return ct;
00139                 }
00140             }
00141             void getTreeInflateDynamic(HuffmanTree& tree, HuffmanTree& treeD, const unsigned char* in,
      size_t& bp, size_t inlength)
00142             { //get the tree of a deflated block with dynamic tree, the tree itself is also Huffman
      compressed with a known tree
00143                 std::vector<unsigned long> bitlen(288, 0), bitlenD(32, 0);
00144                 if (bp >> 3 >= inlength - 2) { error = 49; return; } //the bit pointer is or will go
      past the memory
00145                 size_t HLIT = readBitsFromStream(bp, in, 5) + 257; //number of literal/length codes +
      257
00146                 size_t HDIST = readBitsFromStream(bp, in, 5) + 1; //number of dist codes + 1
00147                 size_t HCLEN = readBitsFromStream(bp, in, 4) + 4; //number of code length codes + 4
00148                 std::vector<unsigned long> codelengthcode(19); //lengths of tree to decode the lengths
      of the dynamic tree
00149                 for (size_t i = 0; i < 19; i++) codelengthcode[CLCL[i]] = (i < HCLEN) ?
      readBitsFromStream(bp, in, 3) : 0;
00150                 error = codelengthcodetree.makeFromLengths(codelengthcode, 7); if (error) return;
00151                 size_t i = 0, replength;
00152                 while (i < HLIT + HDIST)
00153                 {
00154                     unsigned long code = huffmanDecodeSymbol(in, bp, codelengthcodetree, inlength); if
      (error) return;
00155                     if (code <= 15) { if (i < HLIT) bitlen[i++] = code; else bitlenD[i++ - HLIT] =
      code; } //a length code
00156                     else if (code == 16) //repeat previous
00157                     {
00158                         if (bp >> 3 >= inlength) { error = 50; return; } //error, bit pointer jumps
      past memory
00159                         replength = 3 + readBitsFromStream(bp, in, 2);
00160                         unsigned long value; //set value to the previous code
00161                         if ((i - 1) < HLIT) value = bitlen[i - 1];
00162                         else value = bitlenD[i - HLIT - 1];
00163                         for (size_t n = 0; n < replength; n++) //repeat this value in the next lengths
00164                         {
00165                             if (i >= HLIT + HDIST) { error = 13; return; } //error: i is larger than
      the amount of codes
00166                             if (i < HLIT) bitlen[i++] = value; else bitlenD[i++ - HLIT] = value;
00167                         }
00168                     }
00169                     else if (code == 17) //repeat "0" 3-10 times
00170                     {
00171                         if (bp >> 3 >= inlength) { error = 50; return; } //error, bit pointer jumps
      past memory
00172                         replength = 3 + readBitsFromStream(bp, in, 3);
00173                         for (size_t n = 0; n < replength; n++) //repeat this value in the next lengths
00174                         {
00175                             if (i >= HLIT + HDIST) { error = 14; return; } //error: i is larger than
      the amount of codes
00176                             if (i < HLIT) bitlen[i++] = 0; else bitlenD[i++ - HLIT] = 0;
00177                         }
00178                     }
00179                     else if (code == 18) //repeat "0" 11-138 times
00180                     {
00181                         if (bp >> 3 >= inlength) { error = 50; return; } //error, bit pointer jumps
      past memory
00182                         replength = 11 + readBitsFromStream(bp, in, 7);
00183                         for (size_t n = 0; n < replength; n++) //repeat this value in the next lengths
00184                         {
00185                             if (i >= HLIT + HDIST) { error = 15; return; } //error: i is larger than
      the amount of codes
00186                             if (i < HLIT) bitlen[i++] = 0; else bitlenD[i++ - HLIT] = 0;
00187                         }
00188                     }
00189                     else { error = 16; return; } //error: somehow an unexisting code appeared. This
      can never happen.
00190                 }
00191                 if (bitlen[256] == 0) { error = 64; return; } //the length of the end code 256 must be
      larger than 0
00192                 error = tree.makeFromLengths(bitlen, 15); if (error) return; //now we've finally got
```

```
    HLIT and HDIST, so generate the code trees, and the function is done
00193                 error = treeD.makeFromLengths(bitlenD, 15); if (error) return;
00194             }
00195             void inflateHuffmanBlock(std::vector<unsigned char>& out, const unsigned char* in, size_t&
    bp, size_t& pos, size_t inlength, unsigned long btype)
00196             {
00197                 if (btype == 1) { generateFixedTrees(codetree, codetreeD); }
00198                 else if (btype == 2) { getTreeInflateDynamic(codetree, codetreeD, in, bp, inlength);
    if (error) return; }
00199                 for (;;)
00200                 {
00201                     unsigned long code = huffmanDecodeSymbol(in, bp, codetree, inlength); if (error)
    return;
00202                     if (code == 256) return; //end code
00203                     else if (code <= 255) //literal symbol
00204                     {
00205                         if (pos >= out.size()) out.resize((pos + 1) * 2); //reserve more room
00206                         out[pos++] = (unsigned char)(code);
00207                     }
00208                     else if (code >= 257 && code <= 285) //length code
00209                     {
00210                         size_t length = LENBASE[code - 257], numextrabits = LENEXTRA[code - 257];
00211                         if ((bp » 3) >= inlength) { error = 51; return; } //error, bit pointer will
    jump past memory
00212                         length += readBitsFromStream(bp, in, numextrabits);
00213                         unsigned long codeD = huffmanDecodeSymbol(in, bp, codetreeD, inlength); if
    (error) return;
00214                         if (codeD > 29) { error = 18; return; } //error: invalid dist code (30-31 are
    never used)
00215                         unsigned long dist = DISTBASE[codeD], numextrabitsD = DISTEXTRA[codeD];
00216                         if ((bp » 3) >= inlength) { error = 51; return; } //error, bit pointer will
    jump past memory
00217                         dist += readBitsFromStream(bp, in, numextrabitsD);
00218                         size_t start = pos, back = start - dist; //backwards
00219                         if (pos + length >= out.size()) out.resize((pos + length) * 2); //reserve more
    room
00220                         for (size_t i = 0; i < length; i++) { out[pos++] = out[back++]; if (back >=
    start) back = start - dist; }
00221                     }
00222                 }
00223             }
00224             void inflateNoCompression(std::vector<unsigned char>& out, const unsigned char* in,
    size_t& bp, size_t& pos, size_t inlength)
00225             {
00226                 while ((bp & 0x7) != 0) bp++; //go to first boundary of byte
00227                 size_t p = bp / 8;
00228                 if (p >= inlength - 4) { error = 52; return; } //error, bit pointer will jump past
    memory
00229                 unsigned long LEN = in[p] + 256 * in[p + 1], NLEN = in[p + 2] + 256 * in[p + 3]; p +=
    4;
00230                 if (LEN + NLEN != 65535) { error = 21; return; } //error: NLEN is not one's complement
    of LEN
00231                 if (pos + LEN >= out.size()) out.resize(pos + LEN);
00232                 if (p + LEN > inlength) { error = 23; return; } //error: reading outside of in buffer
00233                 for (unsigned long n = 0; n < LEN; n++) out[pos++] = in[p++]; //read LEN bytes of
    literal data
00234                 bp = p * 8;
00235             }
00236         };
00237         int decompress(std::vector<unsigned char>& out, const std::vector<unsigned char>& in)
    //returns error value
00238         {
00239             Inflator inflator;
00240             if (in.size() < 2) { return 53; } //error, size of zlib data too small
00241             if ((in[0] * 256 + in[1]) % 31 != 0) { return 24; } //error: 256 * in[0] + in[1] must be a
    multiple of 31, the FCHECK value is supposed to be made that way
00242             unsigned long CM = in[0] & 15, CINFO = (in[0] » 4) & 15, FDICT = (in[1] » 5) & 1;
00243             if (CM != 8 || CINFO > 7) { return 25; } //error: only compression method 8: inflate with
    sliding window of 32k is supported by the PNG spec
00244             if (FDICT != 0) { return 26; } //error: the specification of PNG says about the zlib
    stream: "The additional flags shall not specify a preset dictionary."
00245             inflator.inflate(out, in, 2);
00246             return inflator.error; //note: adler32 checksum was skipped and ignored
00247         }
00248     };
00249     struct PNG //nested functions for PNG decoding
00250     {
00251         struct Info
00252         {
00253             unsigned long width, height, colorType, bitDepth, compressionMethod, filterMethod,
    interlaceMethod, key_r, key_g, key_b;
00254             bool key_defined; //is a transparent color key given?
00255             std::vector<unsigned char> palette;
00256         } info;
00257         int error;
00258         void decode(std::vector<unsigned char>& out, const unsigned char* in, size_t size, bool
    convert_to_rgba32)
```

```
00259         {
00260             error = 0;
00261             if (size == 0 || in == 0) { error = 48; return; } //the given data is empty
00262             readPngHeader(&in[0], size); if (error) return;
00263             size_t pos = 33; //first byte of the first chunk after the header
00264             std::vector<unsigned char> idat; //the data from idat chunks
00265             bool IEND = false, known_type = true;
00266             info.key_defined = false;
00267             while (!IEND) //loop through the chunks, ignoring unknown chunks and stopping at IEND
      chunk. IDAT data is put at the start of the in buffer
00268             {
00269                 if (pos + 8 >= size) { error = 30; return; } //error: size of the in buffer too small
      to contain next chunk
00270                 size_t chunkLength = read32bitInt(&in[pos]); pos += 4;
00271                 if (chunkLength > 2147483647) { error = 63; return; }
00272                 if (pos + chunkLength >= size) { error = 35; return; } //error: size of the in buffer
      too small to contain next chunk
00273                 if (in[pos + 0] == 'I' && in[pos + 1] == 'D' && in[pos + 2] == 'A' && in[pos + 3] ==
      'T') //IDAT chunk, containing compressed image data
00274                 {
00275                     idat.insert(idat.end(), &in[pos + 4], &in[pos + 4 + chunkLength]);
00276                     pos += (4 + chunkLength);
00277                 }
00278                 else if (in[pos + 0] == 'I' && in[pos + 1] == 'E' && in[pos + 2] == 'N' && in[pos + 3]
      == 'D') { pos += 4; IEND = true; }
00279                 else if (in[pos + 0] == 'P' && in[pos + 1] == 'L' && in[pos + 2] == 'T' && in[pos + 3]
      == 'E') //palette chunk (PLTE)
00280                 {
00281                     pos += 4; //go after the 4 letters
00282                     info.palette.resize(4 * (chunkLength / 3));
00283                     if (info.palette.size() > (4 * 256)) { error = 38; return; } //error: palette too
      big
00284                     for (size_t i = 0; i < info.palette.size(); i += 4)
00285                     {
00286                         for (size_t j = 0; j < 3; j++) info.palette[i + j] = in[pos++]; //RGB
00287                         info.palette[i + 3] = 255; //alpha
00288                     }
00289                 }
00290                 else if (in[pos + 0] == 't' && in[pos + 1] == 'R' && in[pos + 2] == 'N' && in[pos + 3]
      == 'S') //palette transparency chunk (tRNS)
00291                 {
00292                     pos += 4; //go after the 4 letters
00293                     if (info.colorType == 3)
00294                     {
00295                         if (4 * chunkLength > info.palette.size()) { error = 39; return; } //error:
      more alpha values given than there are palette entries
00296                         for (size_t i = 0; i < chunkLength; i++) info.palette[4 * i + 3] = in[pos++];
00297                     }
00298                     else if (info.colorType == 0)
00299                     {
00300                         if (chunkLength != 2) { error = 40; return; } //error: this chunk must be 2
      bytes for greyscale image
00301                         info.key_defined = 1; info.key_r = info.key_g = info.key_b = 256 * in[pos] +
      in[pos + 1]; pos += 2;
00302                     }
00303                     else if (info.colorType == 2)
00304                     {
00305                         if (chunkLength != 6) { error = 41; return; } //error: this chunk must be 6
      bytes for RGB image
00306                         info.key_defined = 1;
00307                         info.key_r = 256 * in[pos] + in[pos + 1]; pos += 2;
00308                         info.key_g = 256 * in[pos] + in[pos + 1]; pos += 2;
00309                         info.key_b = 256 * in[pos] + in[pos + 1]; pos += 2;
00310                     }
00311                     else { error = 42; return; } //error: tRNS chunk not allowed for other color
      models
00312                 }
00313                 else //it's not an implemented chunk type, so ignore it: skip over the data
00314                 {
00315                     if (!(in[pos + 0] & 32)) { error = 69; return; } //error: unknown critical chunk
      (5th bit of first byte of chunk type is 0)
00316                     pos += (chunkLength + 4); //skip 4 letters and uninterpreted data of unimplemented
      chunk
00317                     known_type = false;
00318                 }
00319                 pos += 4; //step over CRC (which is ignored)
00320             }
00321             unsigned long bpp = getBpp(info);
00322             std::vector<unsigned char> scanlines(((info.width * (info.height * bpp + 7)) / 8) +
      info.height); //now the out buffer will be filled
00323             Zlib zlib; //decompress with the Zlib decompressor
00324             error = zlib.decompress(scanlines, idat); if (error) return; //stop if the zlib
      decompressor returned an error
00325             size_t bytewidth = (bpp + 7) / 8, outlength = (info.height * info.width * bpp + 7) / 8;
00326             out.resize(outlength); //time to fill the out buffer
00327             unsigned char* out_ = outlength ? &out[0] : 0; //use a regular pointer to the std::vector
      for faster code if compiled without optimization
```

```
00328             if (info.interlaceMethod == 0) //no interlace, just filter
00329             {
00330                 size_t linestart = 0, linelength = (info.width * bpp + 7) / 8; //length in bytes of a
      scanline, excluding the filtertype byte
00331                 if (bpp >= 8) //byte per byte
00332                     for (unsigned long y = 0; y < info.height; y++)
00333                     {
00334                         unsigned long filterType = scanlines[linestart];
00335                         const unsigned char* prevline = (y == 0) ? 0 : &out_[(y - 1) * info.width *
      bytewidth];
00336                         unFilterScanline(&out_[linestart - y], &scanlines[linestart + 1], prevline,
      bytewidth, filterType, linelength); if (error) return;
00337                         linestart += (1 + linelength); //go to start of next scanline
00338                     }
00339                 else //less than 8 bits per pixel, so fill it up bit per bit
00340                 {
00341                     std::vector<unsigned char> templine((info.width * bpp + 7) >> 3); //only used if
      bpp < 8
00342                     for (size_t y = 0, obp = 0; y < info.height; y++)
00343                     {
00344                         unsigned long filterType = scanlines[linestart];
00345                         const unsigned char* prevline = (y == 0) ? 0 : &out_[(y - 1) * info.width *
      bytewidth];
00346                         unFilterScanline(&templine[0], &scanlines[linestart + 1], prevline, bytewidth,
      filterType, linelength); if (error) return;
00347                         for (size_t bp = 0; bp < info.width * bpp;) setBitOfReversedStream(obp, out_,
      readBitFromReversedStream(bp, &templine[0]));
00348                         linestart += (1 + linelength); //go to start of next scanline
00349                     }
00350                 }
00351             }
00352             else //interlaceMethod is 1 (Adam7)
00353             {
00354                 size_t passw[7] = { (info.width + 7) / 8, (info.width + 3) / 8, (info.width + 3) / 4,
      (info.width + 1) / 4, (info.width + 1) / 2, (info.width + 0) / 2, (info.width + 0) / 1 };
00355                 size_t passh[7] = { (info.height + 7) / 8, (info.height + 7) / 8, (info.height + 3) /
      8, (info.height + 3) / 4, (info.height + 1) / 4, (info.height + 1) / 2, (info.height + 0) / 2 };
00356                 size_t passstart[7] = { 0 };
00357                 size_t pattern[28] = { 0,4,0,2,0,1,0,0,0,4,0,2,0,1,8,8,4,4,2,2,1,8,8,8,4,4,2,2 };
      //values for the adam7 passes
00358                 for (int i = 0; i < 6; i++) passstart[i + 1] = passstart[i] + passh[i] * ((passw[i] ?
      1 : 0) + (passw[i] * bpp + 7) / 8);
00359                 std::vector<unsigned char> scanlineo((info.width * bpp + 7) / 8),
      scanlinen((info.width * bpp + 7) / 8); //"old" and "new" scanline
00360                 for (int i = 0; i < 7; i++)
00361                     adam7Pass(&out_[0], &scanlinen[0], &scanlineo[0], &scanlines[passstart[i]],
      info.width, pattern[i], pattern[i + 7], pattern[i + 14], pattern[i + 21], passw[i], passh[i], bpp);
00362             }
00363             if (convert_to_rgba32 && (info.colorType != 6 || info.bitDepth != 8)) //conversion needed
00364             {
00365                 std::vector<unsigned char> data = out;
00366                 error = convert(out, &data[0], info, info.width, info.height);
00367             }
00368         }
00369         void readPngHeader(const unsigned char* in, size_t inlength) //read the information from the
      header and store it in the Info
00370         {
00371             if (inlength < 29) { error = 27; return; } //error: the data length is smaller than the
      length of the header
00372             if (in[0] != 137 || in[1] != 80 || in[2] != 78 || in[3] != 71 || in[4] != 13 || in[5] !=
      10 || in[6] != 26 || in[7] != 10) { error = 28; return; } //no PNG signature
00373             if (in[12] != 'I' || in[13] != 'H' || in[14] != 'D' || in[15] != 'R') { error = 29;
      return; } //error: it doesn't start with a IHDR chunk!
00374             info.width = read32bitInt(&in[16]); info.height = read32bitInt(&in[20]);
00375             info.bitDepth = in[24]; info.colorType = in[25];
00376             info.compressionMethod = in[26]; if (in[26] != 0) { error = 32; return; } //error: only
      compression method 0 is allowed in the specification
00377             info.filterMethod = in[27]; if (in[27] != 0) { error = 33; return; } //error: only filter
      method 0 is allowed in the specification
00378             info.interlaceMethod = in[28]; if (in[28] > 1) { error = 34; return; } //error: only
      interlace methods 0 and 1 exist in the specification
00379             error = checkColorValidity(info.colorType, info.bitDepth);
00380         }
00381         void unFilterScanline(unsigned char* recon, const unsigned char* scanline, const unsigned
      char* precon, size_t bytewidth, unsigned long filterType, size_t length)
00382         {
00383             switch (filterType)
00384             {
00385             case 0: for (size_t i = 0; i < length; i++) recon[i] = scanline[i]; break;
00386             case 1:
00387                 for (size_t i = 0; i < bytewidth; i++) recon[i] = scanline[i];
00388                 for (size_t i = bytewidth; i < length; i++) recon[i] = scanline[i] + recon[i -
      bytewidth];
00389                 break;
00390             case 2:
00391                 if (precon) for (size_t i = 0; i < length; i++) recon[i] = scanline[i] + precon[i];
00392                 else       for (size_t i = 0; i < length; i++) recon[i] = scanline[i];
```

```
00393                 break;
00394             case 3:
00395                 if (precon)
00396                 {
00397                     for (size_t i = 0; i < bytewidth; i++) recon[i] = scanline[i] + precon[i] / 2;
00398                     for (size_t i = bytewidth; i < length; i++) recon[i] = scanline[i] + ((recon[i -
      bytewidth] + precon[i]) / 2);
00399                 }
00400                 else
00401                 {
00402                     for (size_t i = 0; i < bytewidth; i++) recon[i] = scanline[i];
00403                     for (size_t i = bytewidth; i < length; i++) recon[i] = scanline[i] + recon[i -
      bytewidth] / 2;
00404                 }
00405                 break;
00406             case 4:
00407                 if (precon)
00408                 {
00409                     for (size_t i = 0; i < bytewidth; i++) recon[i] = scanline[i] + paethPredictor(0,
      precon[i], 0);
00410                     for (size_t i = bytewidth; i < length; i++) recon[i] = scanline[i] +
      paethPredictor(recon[i - bytewidth], precon[i], precon[i - bytewidth]);
00411                 }
00412                 else
00413                 {
00414                     for (size_t i = 0; i < bytewidth; i++) recon[i] = scanline[i];
00415                     for (size_t i = bytewidth; i < length; i++) recon[i] = scanline[i] +
      paethPredictor(recon[i - bytewidth], 0, 0);
00416                 }
00417                 break;
00418             default: error = 36; return; //error: unexisting filter type given
00419             }
00420         }
00421         void adam7Pass(unsigned char* out, unsigned char* linen, unsigned char* lineo, const unsigned
      char* in, unsigned long w, size_t passleft, size_t passtop, size_t spacex, size_t spacey, size_t
      passw, size_t passh, unsigned long bpp)
00422         { //filter and reposition the pixels into the output when the image is Adam7 interlaced. This
      function can only do it after the full image is already decoded. The out buffer must have the correct
      allocated memory size already.
00423             if (passw == 0) return;
00424             size_t bytewidth = (bpp + 7) / 8, linelength = 1 + ((bpp * passw + 7) / 8);
00425             for (unsigned long y = 0; y < passh; y++)
00426             {
00427                 unsigned char filterType = in[y * linelength], * prevline = (y == 0) ? 0 : lineo;
00428                 unFilterScanline(linen, &in[y * linelength + 1], prevline, bytewidth, filterType, (w *
      bpp + 7) / 8); if (error) return;
00429                 if (bpp >= 8) for (size_t i = 0; i < passw; i++) for (size_t b = 0; b < bytewidth;
      b++) //b = current byte of this pixel
00430                     out[bytewidth * w * (passtop + spacey * y) + bytewidth * (passleft + spacex * i) +
      b] = linen[bytewidth * i + b];
00431                 else for (size_t i = 0; i < passw; i++)
00432                 {
00433                     size_t obp = bpp * w * (passtop + spacey * y) + bpp * (passleft + spacex * i), bp
      = i * bpp;
00434                     for (size_t b = 0; b < bpp; b++) setBitOfReversedStream(obp, out,
      readBitFromReversedStream(bp, &linen[0]));
00435                 }
00436                 unsigned char* temp = linen; linen = lineo; lineo = temp; //swap the two buffer
      pointers "line old" and "line new"
00437             }
00438         }
00439         static unsigned long readBitFromReversedStream(size_t& bitp, const unsigned char* bits) {
      unsigned long result = (bits[bitp » 3] » (7 - (bitp & 0x7))) & 1; bitp++; return result; }
00440         static unsigned long readBitsFromReversedStream(size_t& bitp, const unsigned char* bits,
      unsigned long nbits)
00441         {
00442             unsigned long result = 0;
00443             for (size_t i = nbits - 1; i < nbits; i--) result += ((readBitFromReversedStream(bitp,
      bits)) « i);
00444             return result;
00445         }
00446         void setBitOfReversedStream(size_t& bitp, unsigned char* bits, unsigned long bit) { bits[bitp
      » 3] |= (bit « (7 - (bitp & 0x7))); bitp++; }
00447         unsigned long read32bitInt(const unsigned char* buffer) { return (buffer[0] « 24) | (buffer[1]
      « 16) | (buffer[2] « 8) | buffer[3]; }
00448         int checkColorValidity(unsigned long colorType, unsigned long bd) //return type is a LodePNG
      error code
00449         {
00450             if ((colorType == 2 || colorType == 4 || colorType == 6)) { if (!(bd == 8 || bd == 16))
      return 37; else return 0; }
00451             else if (colorType == 0) { if (!(bd == 1 || bd == 2 || bd == 4 || bd == 8 || bd == 16))
      return 37; else return 0; }
00452             else if (colorType == 3) { if (!(bd == 1 || bd == 2 || bd == 4 || bd == 8)) return 37;
      else return 0; }
00453             else return 31; //unexisting color type
00454         }
00455         unsigned long getBpp(const Info& info)
```

```
00456         {
00457             if (info.colorType == 2) return (3 * info.bitDepth);
00458             else if (info.colorType >= 4) return (info.colorType - 2) * info.bitDepth;
00459             else return info.bitDepth;
00460         }
00461         int convert(std::vector<unsigned char>& out, const unsigned char* in, Info& infoIn, unsigned
      long w, unsigned long h)
00462         { //converts from any color type to 32-bit. return value = LodePNG error code
00463             size_t numpixels = w * h, bp = 0;
00464             out.resize(numpixels * 4);
00465             unsigned char* out_ = out.empty() ? 0 : &out[0]; //faster if compiled without optimization
00466             if (infoIn.bitDepth == 8 && infoIn.colorType == 0) //greyscale
00467                 for (size_t i = 0; i < numpixels; i++)
00468                 {
00469                     out_[4 * i + 0] = out_[4 * i + 1] = out_[4 * i + 2] = in[i];
00470                     out_[4 * i + 3] = (infoIn.key_defined && in[i] == infoIn.key_r) ? 0 : 255;
00471                 }
00472             else if (infoIn.bitDepth == 8 && infoIn.colorType == 2) //RGB color
00473                 for (size_t i = 0; i < numpixels; i++)
00474                 {
00475                     for (size_t c = 0; c < 3; c++) out_[4 * i + c] = in[3 * i + c];
00476                     out_[4 * i + 3] = (infoIn.key_defined == 1 && in[3 * i + 0] == infoIn.key_r &&
      in[3 * i + 1] == infoIn.key_g && in[3 * i + 2] == infoIn.key_b) ? 0 : 255;
00477                 }
00478             else if (infoIn.bitDepth == 8 && infoIn.colorType == 3) //indexed color (palette)
00479                 for (size_t i = 0; i < numpixels; i++)
00480                 {
00481                     if (4U * in[i] >= infoIn.palette.size()) return 46;
00482                     for (size_t c = 0; c < 4; c++) out_[4 * i + c] = infoIn.palette[4 * in[i] + c];
      //get rgb colors from the palette
00483                 }
00484             else if (infoIn.bitDepth == 8 && infoIn.colorType == 4) //greyscale with alpha
00485                 for (size_t i = 0; i < numpixels; i++)
00486                 {
00487                     out_[4 * i + 0] = out_[4 * i + 1] = out_[4 * i + 2] = in[2 * i + 0];
00488                     out_[4 * i + 3] = in[2 * i + 1];
00489                 }
00490             else if (infoIn.bitDepth == 8 && infoIn.colorType == 6) for (size_t i = 0; i < numpixels;
      i++) for (size_t c = 0; c < 4; c++) out_[4 * i + c] = in[4 * i + c]; //RGB with alpha
00491             else if (infoIn.bitDepth == 16 && infoIn.colorType == 0) //greyscale
00492                 for (size_t i = 0; i < numpixels; i++)
00493                 {
00494                     out_[4 * i + 0] = out_[4 * i + 1] = out_[4 * i + 2] = in[2 * i];
00495                     out_[4 * i + 3] = (infoIn.key_defined && 256U * in[i] + in[i + 1] == infoIn.key_r)
      ? 0 : 255;
00496                 }
00497             else if (infoIn.bitDepth == 16 && infoIn.colorType == 2) //RGB color
00498                 for (size_t i = 0; i < numpixels; i++)
00499                 {
00500                     for (size_t c = 0; c < 3; c++) out_[4 * i + c] = in[6 * i + 2 * c];
00501                     out_[4 * i + 3] = (infoIn.key_defined && 256U * in[6 * i + 0] + in[6 * i + 1] ==
      infoIn.key_r && 256U * in[6 * i + 2] + in[6 * i + 3] == infoIn.key_g && 256U * in[6 * i + 4] + in[6 *
      i + 5] == infoIn.key_b) ? 0 : 255;
00502                 }
00503             else if (infoIn.bitDepth == 16 && infoIn.colorType == 4) //greyscale with alpha
00504                 for (size_t i = 0; i < numpixels; i++)
00505                 {
00506                     out_[4 * i + 0] = out_[4 * i + 1] = out_[4 * i + 2] = in[4 * i]; //most
      significant byte
00507                     out_[4 * i + 3] = in[4 * i + 2];
00508                 }
00509             else if (infoIn.bitDepth == 16 && infoIn.colorType == 6) for (size_t i = 0; i < numpixels;
      i++) for (size_t c = 0; c < 4; c++) out_[4 * i + c] = in[8 * i + 2 * c]; //RGB with alpha
00510             else if (infoIn.bitDepth < 8 && infoIn.colorType == 0) //greyscale
00511                 for (size_t i = 0; i < numpixels; i++)
00512                 {
00513                     unsigned long value = (readBitsFromReversedStream(bp, in, infoIn.bitDepth) * 255)
      / ((1 << infoIn.bitDepth) - 1); //scale value from 0 to 255
00514                     out_[4 * i + 0] = out_[4 * i + 1] = out_[4 * i + 2] = (unsigned char)(value);
00515                     out_[4 * i + 3] = (infoIn.key_defined && value && ((1U << infoIn.bitDepth) - 1U) ==
      infoIn.key_r && ((1U << infoIn.bitDepth) - 1U)) ? 0 : 255;
00516                 }
00517             else if (infoIn.bitDepth < 8 && infoIn.colorType == 3) //palette
00518                 for (size_t i = 0; i < numpixels; i++)
00519                 {
00520                     unsigned long value = readBitsFromReversedStream(bp, in, infoIn.bitDepth);
00521                     if (4 * value >= infoIn.palette.size()) return 47;
00522                     for (size_t c = 0; c < 4; c++) out_[4 * i + c] = infoIn.palette[4 * value + c];
      //get rgb colors from the palette
00523                 }
00524             return 0;
00525         }
00526         unsigned char paethPredictor(short a, short b, short c) //Paeth predicter, used by PNG filter
      type 4
00527         {
00528             short p = a + b - c, pa = p > a ? (p - a) : (a - p), pb = p > b ? (p - b) : (b - p), pc =
      p > c ? (p - c) : (c - p);
```

```
00529             return (unsigned char)((pa <= pb && pa <= pc) ? a : pb <= pc ? b : c);
00530         }
00531     };
00532     PNG decoder; decoder.decode(out_image, in_png, in_size, convert_to_rgba32);
00533     image_width = decoder.info.width; image_height = decoder.info.height;
00534     return decoder.error;
00535 }
```

6.64 PNG_Letters.h

```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004
00005 #define RECT_a SDL_Rect{ 0, 0,  10, 20 };
00006 #define RECT_b  SDL_Rect{ 10, 0, 10 , 20};
00007 #define RECT_c  SDL_Rect{ 20, 0, 10 , 20};
00008 #define RECT_d  SDL_Rect{ 30, 0, 10 , 20};
00009 #define RECT_e  SDL_Rect{ 40, 0, 10 , 20};
00010 #define RECT_f  SDL_Rect{ 50, 0, 5 , 20};
00011 #define RECT_g  SDL_Rect{ 55, 0, 10 , 20};
00012 #define RECT_h  SDL_Rect{ 65, 0, 10 , 20};
00013 #define RECT_i  SDL_Rect{ 75, 0, 3 , 20};
00014 #define RECT_j  SDL_Rect{ 80, 0, 3 , 20};
00015 #define RECT_k  SDL_Rect{ 83, 0, 8 , 20};
00016 #define RECT_l  SDL_Rect{ 91, 0, 5 , 20};
00017 #define RECT_m  SDL_Rect{ 96, 0, 15 , 20};
00018 #define RECT_n  SDL_Rect{ 110, 0, 10 , 20};
00019 #define RECT_o  SDL_Rect{ 120, 0, 10 , 20};
00020 #define RECT_p  SDL_Rect{ 130, 0, 10 , 20};
00021 #define RECT_q  SDL_Rect{ 140, 0, 10 , 20};
00022 #define RECT_r  SDL_Rect{ 150, 0, 7 , 20};
00023 #define RECT_s  SDL_Rect{ 157, 0, 9 , 20};
00024 #define RECT_t  SDL_Rect{ 166, 0, 5 , 20};
00025 #define RECT_u  SDL_Rect{ 171, 0, 10 , 20};
00026 #define RECT_v  SDL_Rect{ 181, 0, 9 , 20};
00027 #define RECT_w  SDL_Rect{ 190, 0, 12 , 20};
00028 #define RECT_x  SDL_Rect{ 202, 0, 10 , 20};
00029 #define RECT_y  SDL_Rect{ 212, 0, 9 , 20};
00030 #define RECT_z  SDL_Rect{ 221, 0, 9 , 20};
00031
00032 #define RECT_A  SDL_Rect{ 0, 20, 12, 20 };
00033 #define RECT_B  SDL_Rect{ 12, 20, 12, 20 };
00034 #define RECT_C  SDL_Rect{ 24, 20, 13, 20 };
00035 #define RECT_D  SDL_Rect{ 37, 20, 12, 20 };
00036 #define RECT_E  SDL_Rect{ 49, 20, 13, 20 };
00037 #define RECT_F  SDL_Rect{ 62, 20, 12, 20 };
00038 #define RECT_G  SDL_Rect{ 74, 20, 13, 20 };
00039 #define RECT_H  SDL_Rect{ 87, 20, 13, 20 };
00040 #define RECT_I  SDL_Rect{ 100, 20, 5, 20 };
00041 #define RECT_J  SDL_Rect{ 105, 20, 9, 20 };
00042 #define RECT_K  SDL_Rect{ 113, 20, 13, 20 };
00043 #define RECT_L  SDL_Rect{ 126, 20, 11, 20 };
00044 #define RECT_M  SDL_Rect{ 137, 20, 14, 20 };
00045 #define RECT_N  SDL_Rect{ 151, 20, 13, 20 };
00046 #define RECT_O  SDL_Rect{ 164, 20, 14, 20 };
00047 #define RECT_P  SDL_Rect{ 178, 20, 12, 20 };
00048 #define RECT_Q  SDL_Rect{ 190, 20, 14, 20 };
00049 #define RECT_R  SDL_Rect{ 204, 20, 14, 20 };
00050 #define RECT_S  SDL_Rect{ 217, 20, 12, 20 };
00051 #define RECT_T  SDL_Rect{ 229, 20, 12, 20 };
00052 #define RECT_U  SDL_Rect{ 241, 20, 12, 20 };
00053 #define RECT_V  SDL_Rect{ 253, 20, 12, 20 };
00054 #define RECT_W  SDL_Rect{ 265, 20, 17, 20 };
00055 #define RECT_X  SDL_Rect{ 282, 20, 12, 20 };
00056 #define RECT_Y  SDL_Rect{ 294, 20, 12, 20 };
00057 #define RECT_Z  SDL_Rect{ 306, 20, 11, 20 };
00058
00059 #define RECT_0  SDL_Rect{ 0, 40,  10,20 };
00060 #define RECT_1  SDL_Rect{ 10, 40, 10 , 20};
00061 #define RECT_2  SDL_Rect{ 20, 40, 10 , 20};
00062 #define RECT_3  SDL_Rect{ 30, 40, 10 , 20};
00063 #define RECT_4  SDL_Rect{ 40, 40, 10 , 20};
00064 #define RECT_5  SDL_Rect{ 50, 40, 10 , 20};
00065 #define RECT_6  SDL_Rect{ 60, 40, 10 , 20};
00066 #define RECT_7  SDL_Rect{ 70, 40, 10 , 20};
00067 #define RECT_8  SDL_Rect{ 80, 40, 10 , 20};
00068 #define RECT_9  SDL_Rect{ 90, 40, 10 , 20};
00069
00070 #define RECT_DOT  SDL_Rect{ 10, 106, 20, 20 };
00071 #define RECT_COLON  SDL_Rect{ 10, 106, 20, 20 };
00072 #define RECT_COMMA  SDL_Rect{ 10, 106, 20, 20 };
00073 #define RECT_SEMICOLON  SDL_Rect{ 10, 106, 20, 20 };
```

```
00074 #define RECT_LEFT_PARENTHESIS  SDL_Rect{ 10, 106, 20, 20 };
00075 #define RECT_ASTERISK  SDL_Rect{ 10, 106, 20, 20 };
00076 #define RECT_EXCLAMATION  SDL_Rect{ 10, 106, 20, 20 };
00077 #define RECT_QUESTION  SDL_Rect{ 10, 106, 20, 20 };
00078 #define RECT_RIGHT_BRACK  SDL_Rect{ 10, 106, 20, 20 };
00079 #define RECT_CARET  SDL_Rect{ 10, 106, 20, 20 };
00080 #define RECT_RIGHT_PARENTHESIS  SDL_Rect{ 10, 106, 20, 20 };
00081 #define RECT_HASHTAG  SDL_Rect{ 10, 106, 20, 20 };
00082 #define RECT_DOLLAR  SDL_Rect{ 10, 106, 20, 20 };
00083 #define RECT_LEFT_BRACK  SDL_Rect{ 10, 106, 20, 20 };
00084 #define RECT_PERCENTAGE  SDL_Rect{ 10, 106, 20, 20 };
00085 #define RECT_AMPERSAND  SDL_Rect{ 10, 106, 20, 20 };
00086 #define RECT_DASH  SDL_Rect{ 10, 106, 20, 20 };
00087 #define RECT_PLUS  SDL_Rect{ 10, 106, 20, 20 };
00088 #define RECT_AT  SDL_Rect{ 10, 106, 20, 20 };
00089
00090 #define RECT_SPACE  SDL_Rect{ 300, 0, 12, 20 };
00091
00092
00093 SDL_Rect getLetterRect(char letter);
```

6.65 Framebuffer.h

```
00001 #pragma once
00002
00003 #include "../../GLSLProgram.h"
00004 #include <GL/glew.h>
00005 #include <glm/glm.hpp>
00006
00007 class Framebuffer
00008 {
00009 private:
00010     float _rectangleVertices[24] =
00011     {
00012         // Coords    // texCoords
00013          1.0f, -1.0f,  1.0f, 0.0f,
00014         -1.0f, -1.0f,  0.0f, 0.0f,
00015         -1.0f,  1.0f,  0.0f, 1.0f,
00016
00017          1.0f,  1.0f,  1.0f, 1.0f,
00018          1.0f, -1.0f,  1.0f, 0.0f,
00019         -1.0f,  1.0f,  0.0f, 1.0f
00020     };
00021
00022     unsigned int _rectVAO, _rectVBO;
00023
00024     unsigned int _FBO;
00025     unsigned int _RBO;
00026 public:
00027     uint32_t _framebufferTexture;
00028
00029     Framebuffer();
00030     ~Framebuffer();
00031
00032     void init(int windowWidth, int windowHeight);
00033
00034     void Bind();
00035     void Unbind();
00036 };
```

6.66 LightRenderer.h

```
00001 #pragma once
00002
00003 #include <GL/glew.h>
00004 #include <glm/glm.hpp>
00005 #include <vector>
00006
00007 #include "../../GLSLProgram.h"
00008
00009
00010 // init --_
00011 //       `-->begin()
00012 //       |
00013 //       | --> draw()
00014 //       | --> draw()
00015 //       |
00016 //       |--> end()
00017 //       `--> renderBatch()
```

```
00018
00019 class LightRenderer {
00020 public:
00021     LightRenderer();
00022     ~LightRenderer();
00023
00024     void init();
00025
00026     void begin();
00027     void end();
00028
00029     void initLightTriangleBatch(size_t mSize);
00030     void initLightQuadBatch(size_t mSize);
00031     void initLightBoxBatch(size_t mSize);
00032     void initLightSphereBatch(size_t mSize);
00033
00034     void initBatchSize();
00035
00036     void drawTriangle(size_t v_index,
00037         const glm::vec3& depth,
00038         const glm::vec3& cpuRotation, const Color& color);
00039
00040     void draw(size_t v_index,
00041         const glm::vec2& rectSize,
00042         const glm::vec3& bodyCenter,
00043         const glm::vec3& mRotation,
00044         const Color& color);
00045
00046     void drawBox(size_t v_index,
00047         const glm::vec3& boxSize,
00048         const glm::vec3& bodyCenter,
00049         const glm::vec3& mRotation,
00050         const Color& color);
00051
00052     void drawSphere(size_t v_index,
00053         const glm::vec3& sphereSize,
00054         const glm::vec3& bodyCenter,
00055         const glm::vec3& mRotation,
00056         const Color& color);
00057
00058
00059     void renderBatch(GLSLProgram* glsl_program);
00060
00061     void dispose();
00062
00063     std::vector<LightVertex> sphereVertices = {
00064         // Generated vertices will go here
00065     };
00066
00067     std::vector<GLuint> sphereIndices = {
00068         // Generated indices will go here
00069     };
00070
00071 private:
00072     void createRenderBatches();
00073     void createInstancesVBO();
00074     void createVertexArray();
00075
00076     GLuint _vboInstances;
00077
00078     size_t _glyphs_size = 0; //actual glyphs
00079     size_t _triangleGlyphs_size = 0; //actual glyphs
00080     size_t _boxGlyphs_size = 0;
00081     size_t _sphereGlyphs_size = 0;
00082
00083     std::vector<ColorMeshRenderer> _meshesArrays;
00084     std::vector<ColorMeshRenderer> _meshesElements;
00085
00086 };
```

6.67 LineRenderer.h

```
00001 #pragma once
00002
00003 #include "../../GLSLProgram.h"
00004 #include <GL/glew.h>
00005 #include <glm/glm.hpp>
00006 #include <vector>
00007
00008 #include "../../Vertex.h"
00009
00010
00011
```

```
00012
00013 class RenderLineBatch {
00014 public:
00015     RenderLineBatch() {
00016
00017     }
00018     RenderLineBatch(GLuint Offset, GLuint NumIndices) : offset(Offset),
00019         numIndices(NumIndices){
00020
00021     }
00022     GLuint offset = 0;
00023     GLuint numIndices = 0;
00024 };
00025
00026 class LineGlyph {
00027
00028 public:
00029     LineGlyph() {};
00030     LineGlyph(const glm::vec3& fromPosition, const glm::vec3& toPosition, const Color& srcColor, const
      Color& destColor)
00031         {
00032
00033         fromV.color = srcColor;
00034         fromV.setPosition(fromPosition);
00035
00036         toV.color = destColor;
00037         toV.setPosition(toPosition);
00038     };
00039
00040     ColorVertex fromV;
00041     ColorVertex toV;
00042 };
00043
00044 class SquareGlyph {
00045     //todo avoid rotating every point individually, takes a lot of time
00046     //todo instead rotate them from transform using a global angle(based on aimpos) that is added to
      all transforms
00047     //todo then rotation in glyph is just taking that transform
00048 public:
00049     SquareGlyph() {};
00050     SquareGlyph(const glm::vec4& destRect, const Color& color, float angle, float mdepth)
00051         {
00052         glm::vec3 atopLeft(destRect.x, destRect.y, mdepth);
00053         glm::vec3 abottomLeft(destRect.x, destRect.y + destRect.w, mdepth);
00054         glm::vec3 abottomRight(destRect.x + destRect.z, destRect.y + destRect.w, mdepth);
00055         glm::vec3 atopRight(destRect.x + destRect.z, destRect.y, mdepth);
00056
00057         topLeft.color = color;
00058         topLeft.setPosition(atopLeft);
00059
00060         bottomLeft.color = color;
00061         bottomLeft.setPosition(abottomLeft);
00062
00063         bottomRight.color = color;
00064         bottomRight.setPosition(abottomRight);
00065
00066         topRight.color = color;
00067         topRight.setPosition(atopRight);
00068
00069     };
00070
00071     ColorVertex topLeft;
00072     ColorVertex bottomLeft;
00073     ColorVertex bottomRight;
00074     ColorVertex topRight;
00075 };
00076
00077 class BoxGlyph {
00078
00079 public:
00080     BoxGlyph() {};
00081     BoxGlyph(const glm::vec3& origin, const glm::vec3& size, const Color& color, float angle)
00082     {
00083
00084         glm::vec3 atopLeft(origin.x, origin.y, origin.z);
00085         glm::vec3 abottomLeft(origin.x, origin.y + size.y, origin.z);
00086         glm::vec3 abottomRight(origin.x + size.x, origin.y + size.y, origin.z);
00087         glm::vec3 atopRight(origin.x + size.x, origin.y, origin.z);
00088
00089         a_topLeft.color = color;
00090         a_topLeft.setPosition(atopLeft);
00091
00092         a_bottomLeft.color = color;
00093         a_bottomLeft.setPosition(abottomLeft);
00094
00095         a_bottomRight.color = color;
00096         a_bottomRight.setPosition(abottomRight);
```

```
00097
00098         a_topRight.color = color;
00099         a_topRight.setPosition(atopRight);
00100
00101         glm::vec3 btopLeft(origin.x, origin.y, origin.z + size.z);
00102         glm::vec3 bbottomLeft(origin.x, origin.y + size.y, origin.z + size.z);
00103         glm::vec3 bbottomRight(origin.x + size.x, origin.y + size.y, origin.z + size.z);
00104         glm::vec3 btopRight(origin.x + size.x, origin.y, origin.z + size.z);
00105
00106         b_topLeft.color = color;
00107         b_topLeft.setPosition(btopLeft);
00108
00109         b_bottomLeft.color = color;
00110         b_bottomLeft.setPosition(bbottomLeft);
00111
00112         b_bottomRight.color = color;
00113         b_bottomRight.setPosition(bbottomRight);
00114
00115         b_topRight.color = color;
00116         b_topRight.setPosition(btopRight);
00117
00118     };
00119
00120     ColorVertex a_topLeft;
00121     ColorVertex a_bottomLeft;
00122     ColorVertex a_bottomRight;
00123     ColorVertex a_topRight;
00124
00125     ColorVertex b_topLeft;
00126     ColorVertex b_bottomLeft;
00127     ColorVertex b_bottomRight;
00128     ColorVertex b_topRight;
00129 };
00130
00131 class LineRenderer {
00132 public:
00133     const char* VERT_SRC = R"(#version 400
00134
00135 in vec3 vertexPosition; //vec3 is array of 3 floats
00136 in vec4 vertexColor;
00137
00138 out vec4 fragmentColor;
00139
00140 uniform mat4 projection;
00141
00142 void main() {
00143     gl_Position = projection * vec4(vertexPosition.xyz, 1.0);
00144
00145     fragmentColor = vertexColor;
00146 })";
00147
00148     const char* FRAG_SRC = R"(#version 400
00149
00150 in vec4 fragmentColor;
00151
00152 out vec4 color; //rgb value
00153
00154 void main() {
00155     color = vec4(fragmentColor.rgb, fragmentColor.a);
00156 })";
00157
00158     LineRenderer();
00159     ~LineRenderer();
00160
00161     void init();
00162     void begin();
00163
00164     void end();
00165
00166     void initBatchLines(size_t msize);
00167
00168     void initBatchSquares(size_t msize);
00169
00170     void initBatchBoxes(size_t msize);
00171
00172     void drawLine(size_t v_index, const glm::vec3 srcPosition, const glm::vec3 destPosition, const
      Color& srcColor, const Color& destColor);
00173     void drawRectangle(size_t v_index, const glm::vec4& destRect, const Color& color, float angle,
      float zIndex =0.0f);
00174     void drawBox(size_t v_index, const glm::vec3& origin, const glm::vec3& size, const Color& color,
      float angle);
00175     void drawCircle(const glm::vec2& center, const Color& color, float radius);
00176
00177     void initBatchSize();
00178     void renderBatch(float lineWidth);
00179
00180     void dispose();
```

```
00181
00182     int box_edgePairs[12][2] = {
00183             {0, 1}, {1, 2}, {2, 3}, {3, 0}, // Bottom face
00184             {4, 5}, {5, 6}, {6, 7}, {7, 4}, // Top face
00185             {0, 4}, {1, 5}, {2, 6}, {3, 7}  // Vertical edges
00186     };
00187
00188 private:
00189     void createRenderBatches();
00190     void createVertexArray();
00191
00192     GLuint _vbo = 0, _vao = 0, _ibo = 0;
00193
00194     std::vector<ColorVertex> _vertices;
00195     std::vector<GLuint> _indices;
00196
00197     size_t _lineGlyphs_size     = 0;
00198     size_t _squareGlyphs_size   = 0;
00199     size_t _boxGlyphs_size      = 0;
00200
00201     size_t _rectangles_verticesOffset = 0;
00202     size_t _lines_verticesOffset = 0;
00203     size_t _boxes_verticesOffset= 0;
00204
00205     size_t _rectangles_indicesOffset = 0;
00206     size_t _lines_indicesOffset = 0;
00207     size_t _boxes_indicesOffset = 0;
00208
00209     std::vector<RenderLineBatch> _renderBatches;
00210 };
```

6.68 PlaneColorRenderer.h

```
00001 #pragma once
00002
00003 #include <GL/glew.h>
00004 #include <glm/glm.hpp>
00005 #include <vector>
00006
00007 #include "../../GLSLProgram.h"
00008
00009
00010 // init --_
00011 //       `-->begin()
00012 //       |
00013 //       | --> draw()
00014 //       | --> draw()
00015 //       |
00016 //       |--> end()
00017 //       `--> renderBatch()
00018
00019 class PlaneColorRenderer {
00020 public:
00021     PlaneColorRenderer();
00022     ~PlaneColorRenderer();
00023
00024     void init();
00025
00026     void begin();
00027     void end();
00028
00029     void initColorTriangleBatch(size_t mSize);
00030     void initColorQuadBatch(size_t mSize);
00031     void initColorBoxBatch(size_t mSize);
00032     void initColorSphereBatch(size_t mSize);
00033
00034     void initBatchSize();
00035
00036     void drawTriangle(size_t v_index,
00037         const glm::vec3& depth,
00038         const glm::vec3& cpuRotation, const Color& color);
00039
00040     void draw(size_t v_index,
00041         const glm::vec2& rectSize,
00042         const glm::vec3& bodyCenter,
00043         const glm::vec3& mRotation,
00044         const Color& color);
00045
00046     void drawBox(size_t v_index,
00047         const glm::vec3& boxSize,
00048         const glm::vec3& bodyCenter,
00049         const glm::vec3& mRotation,
00050         const Color& color);
```

```
00051
00052     void drawSphere(size_t v_index,
00053         const glm::vec3& sphereSize,
00054         const glm::vec3& bodyCenter,
00055         const glm::vec3& mRotation,
00056         const Color& color);
00057
00058     void renderBatch(GLSLProgram* glsl_program);
00059
00060     void dispose();
00061
00062     std::vector<Position> sphereVertices = {
00063         // Generated vertices will go here
00064     };
00065
00066     std::vector<GLuint> sphereIndices = {
00067         // Generated indices will go here
00068     };
00069
00070 private:
00071     void createRenderBatches();
00072     void createInstancesVBO();
00073     void createVertexArray();
00074
00075     GLuint _vboInstances;
00076
00077     size_t _glyphs_size = 0; //actual glyphs
00078     size_t _triangleGlyphs_size = 0; //actual glyphs
00079     size_t _boxGlyphs_size = 0;
00080     size_t _sphereGlyphs_size = 0;
00081
00082     std::vector<ColorMeshRenderer> _meshesArrays;
00083     std::vector<ColorMeshRenderer> _meshesElements;
00084
00085 };
```

6.69 PlaneModelRenderer.h

```
00001 #pragma once
00002
00003 #include <GL/glew.h>
00004 #define GLM_ENABLE_EXPERIMENTAL
00005 #include <glm/glm.hpp>
00006 #include <vector>
00007
00008 #include "../../Vertex.h"
00009 #include "../../GLSLProgram.h"
00010
00011 class RenderBatch {
00012 public:
00013     RenderBatch(GLuint Offset, GLuint NumIndices, glm::vec3 CenterPos, GLuint Texture) :
      offset(Offset),
00014         numIndices(NumIndices),
00015         centerPos(CenterPos),
00016         texture(Texture) {
00017
00018     }
00019     GLuint offset;
00020     GLuint numIndices;
00021     glm::vec3 centerPos = glm::vec3(0);
00022     GLuint texture;
00023 };
00024
00025
00026 class Glyph {
00027
00028 public:
00029     Glyph() {};
00030     Glyph(const glm::vec2& rectSize,
00031         const glm::vec3& mRotation,
00032         const glm::vec4& uvRect,
00033         GLuint texture, float Depth)
00034         : texture(texture) {
00035
00036         float halfW = rectSize.x / 2.0f;
00037         float halfH = rectSize.y / 2.0f;
00038
00039         Position positions[4] = {
00040        {-halfW, -halfH, 0.0f},  // topLeft
00041        {-halfW,  halfH, 0.0f},  // bottomLeft
00042        { halfW,  halfH, 0.0f},  // bottomRight
00043        { halfW, -halfH, 0.0f}   // topRight
00044         };
```

```
00045
00046         glm::vec2 uv_topLeft = glm::vec2(uvRect.x, uvRect.y);
00047         glm::vec2 uv_bottomLeft = glm::vec2(uvRect.x, uvRect.y + uvRect.w);
00048         glm::vec2 uv_bottomRight = glm::vec2(uvRect.x + uvRect.z, uvRect.y + uvRect.w);
00049         glm::vec2 uv_topRight = glm::vec2(uvRect.x + uvRect.z, uvRect.y);
00050
00051         topLeft.setPosition(positions[0]);
00052         topLeft.setUV(uv_topLeft); // Use bottom y for top
00053
00054         bottomLeft.setPosition(positions[1]);
00055         bottomLeft.setUV(uv_bottomLeft); // Use top y for bottom
00056
00057         bottomRight.setPosition(positions[2]);
00058         bottomRight.setUV(uv_bottomRight); // Use top y for bottom
00059
00060         topRight.setPosition(positions[3]);
00061         topRight.setUV(uv_topRight); // Use bottom y for top
00062     };
00063
00064     GLuint texture = 0;
00065
00066     TextureVertex topLeft;
00067     TextureVertex bottomLeft;
00068     TextureVertex topRight;
00069     TextureVertex bottomRight;
00070 };
00071
00072 // init --_
00073 //       `-->begin()
00074 //       |
00075 //       | --> draw()
00076 //       | --> draw()
00077 //       |
00078 //       |--> end()
00079 //       `--> renderBatch()
00080
00081 class PlaneModelRenderer {
00082 public:
00083     PlaneModelRenderer();
00084     ~PlaneModelRenderer();
00085
00086     void init();
00087
00088     void begin();
00089     void end();
00090
00091     void initTextureQuadBatch(size_t mSize);
00092
00093     void initBatchSize();
00094
00095     void drawTriangle(
00096         size_t v_index,
00097         const glm::vec3& triangleOffset,
00098         const glm::vec3& mRotation,
00099         const glm::vec2& uv1, const glm::vec2& uv2, const glm::vec2& uv3,
00100         GLuint texture);
00101
00102     void draw(size_t v_index,
00103         const glm::vec2& rectSize,
00104         const glm::vec3& bodyCenter,
00105         const glm::vec3& mRotation,
00106         const glm::vec4& uvRect,
00107         GLuint texture
00108         );
00109
00110     void renderBatch(GLSLProgram* glsl_program);
00111
00112     void dispose();
00113 private:
00114     void createRenderBatches();
00115     void createInstancesVBO();
00116     void createVertexArray();
00117
00118     GLuint _vboInstances;
00119
00120     size_t _glyphs_size = 0; //actual glyphs
00121
00122     std::vector<TextureMeshRenderer> _meshesElements;
00123
00124 };
```

6.70 ResourceManager.h

```
00001 #pragma once
```

```
00002
00003 #include <map>
00004
00005 #include <vector>
00006 #include <string>
00007
00008 #include "../GLSLProgram.h"
00009 #include "../GECS/Core/GECS.h"
00010
00011 class ResourceManager
00012 {
00013 public:
00015
00016     void setupShader(GLSLProgram& shaderProgram, ICamera& camera);
00017
00018     void addGLSLProgram(std::string programName);
00019     GLSLProgram* getGLSLProgram(std::string id);
00020
00021     void disposeGLSLPrograms();
00022 private:
00023     std::map<std::string, GLSLProgram*> glsl_programs;
00024 };
00025
```

6.71 TextureManager.h

```
00001 #pragma once
00002
00003 #include <map>
00004
00005 #include <vector>
00006 #include "../GLTexture.h"
00007 #include <string>
00008
00009 class TextureManager {
00010
00011 public:
00012     static TextureManager& getInstance();
00013     //OPENGL functions
00014     static bool readFileToBuffer(const char* filePath, std::vector <unsigned char>& buffer);
00015     static GLTexture* loadPNG(const char* filePath);
00016     //texture management
00017     void Add_GLTexture(std::string id, const char* path);
00018     const GLTexture* Get_GLTexture(std::string id);
00019     std::vector<std::string> Get_GLTextureNames() const;
00020
00021 private:
00022     std::map<std::string, const GLTexture*> gl_textures;
00023 };
```

6.72 Threader.h

```
00001 #pragma once
00002 #include <functional>
00003 #include <queue>
00004 #include <vector>
00005 #include <thread>
00006 #include <mutex>
00007 #include <atomic>
00008 #include <condition_variable>
00009
00010 struct TaskQueue {
00011     std::deque<std::function<void()» tasks;
00012     std::mutex                       mutex_;
00013     std::condition_variable taskCondition;
00014     std::atomic<int>                 remaining_tasks = 0;
00015     bool shuttingDown = false;
00016     void addTask(std::function<void()>&& callback) {
00017         {
00018             std::lock_guard<std::mutex> lock(mutex_);
00019             tasks.emplace_back(std::move(callback));
00020         }
00021         remaining_tasks++;
00022         taskCondition.notify_one();
00023     }
00024
00025     bool getTask(std::function<void()>& task) {
00026         std::unique_lock<std::mutex> lock(mutex_);
00027         taskCondition.wait(lock, [this] { return !tasks.empty() || shuttingDown; });
```

```
00028
00029         if (shuttingDown || tasks.empty()) return false;  // Shouldn't happen, but just in case
00030
00031         task = std::move(tasks.front());
00032         tasks.pop_front();
00033         return true;
00034     }
00035
00036     void waitUntilDone() const {
00037         while (remaining_tasks > 0) {
00038             std::this_thread::yield();
00039         }
00040     }
00041
00042     void completeTask() {
00043         remaining_tasks--;
00044     }
00045 };
00046
00047 struct Thread {
00048     int                   id = 0;
00049     std::thread           cur_thread;
00050     std::function<void()> task = nullptr;
00051     bool                  running = true;
00052     TaskQueue* t_queue = nullptr;
00053
00054     Thread(TaskQueue& task_queue_, int id_)
00055         : id{ id_ }
00056         , t_queue{ &task_queue_ }
00057     {
00058         cur_thread = std::thread([this]() {
00059             run();
00060             });
00061     }
00062
00063     void run() {
00064         while (running) {
00065             std::function<void()> task;
00066             if (t_queue->getTask(task)) {
00067                 task();
00068                 t_queue->completeTask();
00069             }
00070         }
00071     }
00072
00073     void stop() {
00074         running = false;
00075         t_queue->shuttingDown = true;
00076         t_queue->taskCondition.notify_all();
00077         if (cur_thread.joinable()) {
00078         cur_thread.join();
00079     }
00080     }
00081
00082 };
00083
00084 struct Threader {
00085     TaskQueue t_queue;
00086     int num_threads = 1;
00087     std::vector<Thread> threads;
00088
00089     Threader(int num_threads_)
00090         : num_threads{ num_threads_ }
00091     {
00092         threads.reserve(num_threads_);
00093         for (int i = 0; i < num_threads_; i++)
00094             threads.emplace_back(t_queue, i);
00095     }
00096
00097     void parallel(int num_obj, std::function<void(int start, int end)>&& callback) {
00098         if (num_obj == 0) return;
00099         int slice_size = num_obj / num_threads;
00100         for (int i = 0; i < num_threads; i++) {
00101             int start = i * slice_size;
00102             int end = start + slice_size;
00103             t_queue.addTask([start, end, &callback]() { callback(start, end);});
00104         }
00105         if (slice_size * num_threads < num_obj) {
00106             int start = slice_size * num_threads;
00107             callback(start, num_obj);
00108         }
00109         //todo this may be done only at specific times
00110         t_queue.waitUntilDone();
00111     }
00112 };
```

6.73 Vertex.h

```
00001 #pragma once
00002
00003 #define GLM_ENABLE_EXPERIMENTAL
00004 #include "GL/glew.h"
00005 #include "glm/glm.hpp"
00006 #include "glm/gtc/matrix_transform.hpp"
00007 #include "glm/gtc/type_ptr.hpp"
00008
00009 static glm::mat4 getRotationMatrix(glm::vec3 newRot) {
00010     float radX = glm::radians(newRot.x);
00011     float radY = glm::radians(newRot.y);
00012     float radZ = glm::radians(newRot.z);
00013
00014     glm::mat4 rotationX = glm::rotate(glm::mat4(1.0f), radX, glm::vec3(1.0f, 0.0f, 0.0f));
00015     glm::mat4 rotationY = glm::rotate(glm::mat4(1.0f), radY, glm::vec3(0.0f, 1.0f, 0.0f));
00016     glm::mat4 rotationZ = glm::rotate(glm::mat4(1.0f), radZ, glm::vec3(0.0f, 0.0f, 1.0f));
00017
00018     glm::mat4 rotationMatrix = rotationZ * rotationY * rotationX;
00019     return rotationMatrix;
00020 }
00021
00022 static glm::vec2 rotatePoint(float x, float y, float centerX, float centerY, float radians) {
00023     float dx = x - centerX;
00024     float dy = y - centerY;
00025     return glm::vec2(
00026         centerX + dx * cos(radians) - dy * sin(radians),
00027         centerY + dx * sin(radians) + dy * cos(radians)
00028     );
00029 };
00030 static glm::vec2 rotatePoint(float x, float y, float radians) {
00031     return rotatePoint(x,y, 0.0f, 0.0f, radians);
00032 };
00033 /*  glm::vec3 arotatedTopLeft = rotatePoint(atopLeft.x, atopLeft.y, atopLeft.z, centerX, centerY,
      centerZ, 0, 0, 0);
00034         glm::vec3 arotatedBottomLeft = rotatePoint(abottomLeft.x, abottomLeft.y, abottomLeft.z,
      centerX, centerY, centerZ, 0, 0, 0);
00035         glm::vec3 arotatedBottomRight = rotatePoint(abottomRight.x, abottomRight.y, abottomRight.z,
      centerX, centerY, centerZ, 0, 0, 0);
00036         glm::vec3 arotatedTopRight = rotatePoint(atopRight.x, atopRight.y, atopRight.z, centerX,
      centerY, centerZ, 0, 0, 0);*/
00037
00038
00039 using Position  = glm::vec3;
00040 using Size      = glm::vec3;
00041 using Rotation  = glm::vec3;
00042
00043 using Normal    = glm::vec3;
00044
00045 using UV        = glm::vec2;
00046
00047 struct Color {
00048     Color() : r(0), g(0), b(0), a(0) {}
00049     Color(GLubyte R, GLubyte G, GLubyte B, GLubyte A) : r(R), g(G), b(B), a(A) {}
00050
00051     glm::vec4 toVec4() const {
00052         return glm::vec4(r, g, b, a) / 255.0f; // Normalize 0-255 to 0-1
00053     }
00054
00055     static Color fromVec4(const glm::vec4& v) {
00056         return Color(
00057             static_cast<GLubyte>(v.r * 255.0f),
00058             static_cast<GLubyte>(v.g * 255.0f),
00059             static_cast<GLubyte>(v.b * 255.0f),
00060             static_cast<GLubyte>(v.a * 255.0f)
00061         );
00062     }
00063
00064     GLubyte r;
00065     GLubyte g;
00066     GLubyte b;
00067     GLubyte a;
00068
00069     Color operator*(float scalar) const {
00070         return Color(static_cast<GLubyte>(r * scalar),
00071             static_cast<GLubyte>(g * scalar),
00072             static_cast<GLubyte>(b * scalar),
00073             static_cast<GLubyte>(a * scalar));
00074     }
00075
00076     // Operator overload for color addition
00077     Color operator+(const Color& other) const {
00078         return Color(static_cast<GLubyte>(r + other.r),
00079             static_cast<GLubyte>(g + other.g),
00080             static_cast<GLubyte>(b + other.b),
00081             static_cast<GLubyte>(a + other.a));
```

```
00082     }
00083
00084     bool operator==(const Color& other) const {
00085         return r == other.r && g == other.g && b == other.b && a == other.a;
00086     }
00087 };
00088
00089
00090 struct Vertex {
00091     Position position = Position(0);
00092
00093     inline void setPosition(Position m_position) {
00094         position = m_position;
00095     }
00096 };
00097
00098 struct ColorVertex : Vertex { //instead of using the general Vertex that has also info about texture
00099     // we use this where we want just color
00100     //todo different instanceVBO for the centers
00101     //Position centerMesh;
00102     Color color = Color();
00103
00104     void setColor(GLubyte r, GLubyte g, GLubyte b, GLubyte a) {
00105         color.r = r;
00106         color.g = g;
00107         color.b = b;
00108         color.a = a;
00109     }
00110 };
00111
00112 struct LightVertex : Vertex {
00113     Normal normal = Normal();
00114
00115 };
00116
00117 struct TextureVertex  : Vertex{
00118     // UV texture coordinates
00119     UV uv = UV(0);
00120
00121     inline void setUV(UV m_uv) {
00122         uv = m_uv;
00123     }
00124 };
```

6.74 Window.h

```
00001 #pragma once
00002
00003 #include <SDL2/SDL.h>
00004 #include <GL/glew.h>
00005 #include <string>
00006
00007 #include <imgui.h>
00008 #include <imgui_impl_sdl2.h>
00009 #include <imgui_impl_opengl3.h>
00010 #include <implot.h>
00011 #include <implot_internal.h>
00012
00013 namespace TazGraphEngine {
00014
00015     enum WindowFlags { INVISIBLE = 0x1, VISIBLE = 0x2, FULLSCREEN = 0x4, BORDERLESS = 0x8 };
00016
00017     class Window
00018     {
00019     public:
00020         Window();
00021         ~Window();
00022
00023         int create(std::string windowName, int screenWidth, int screenHeight, float scale, unsigned
      int currentFlags);
00024
00025         void swapBuffer();
00026
00027         void setScreenWidth(int width) { _screenWidth = width; }
00028         int getScreenWidth() { return _screenWidth; }
00029         void setScreenHeight(int height) { _screenHeight = height; }
00030         int getScreenHeight() { return _screenHeight; }
00031         void setScale(float scale) { _scale = scale; }
00032         float getScale() { return _scale; }
00033     private:
00034         SDL_Window* _sdlWindow = nullptr;
00035         int _screenWidth = 0, _screenHeight = 0;
00036         float _scale = 0.0f;
```

```
00037     };
00038
00039 }
```