

R-Type - Engine

Generated by Doxygen 1.9.1

| | |
|--|----------|
| 1 Engine | 1 |
| 1.1 Compilation | 1 |
| 1.1.1 Linux | 1 |
| 2 Hierarchical Index | 3 |
| 2.1 Class Hierarchy | 3 |
| 3 Class Index | 5 |
| 3.1 Class List | 5 |
| 4 Class Documentation | 7 |
| 4.1 Archetypes Class Reference | 7 |
| 4.2 AudioTest Class Reference | 7 |
| 4.3 Color Class Reference | 8 |
| 4.4 Components Class Reference | 9 |
| 4.4.1 Detailed Description | 10 |
| 4.4.2 Constructor & Destructor Documentation | 10 |
| 4.4.2.1 Components() | 10 |
| 4.4.2.2 ~Components() | 10 |
| 4.4.3 Member Function Documentation | 11 |
| 4.4.3.1 init() | 11 |
| 4.5 DrawableComponent Class Reference | 11 |
| 4.5.1 Detailed Description | 12 |
| 4.5.2 Constructor & Destructor Documentation | 12 |
| 4.5.2.1 ~DrawableComponent() | 12 |
| 4.5.3 Member Function Documentation | 12 |
| 4.5.3.1 draw() | 12 |
| 4.6 Entity Class Reference | 13 |
| 4.6.1 Detailed Description | 15 |
| 4.6.2 Constructor & Destructor Documentation | 15 |
| 4.6.2.1 Entity() [1/2] | 15 |
| 4.6.2.2 Entity() [2/2] | 15 |
| 4.6.2.3 ~Entity() | 16 |
| 4.6.3 Member Function Documentation | 16 |
| 4.6.3.1 addComponent() | 16 |
| 4.6.3.2 addDrawable() | 16 |
| 4.6.3.3 drawEntity() | 17 |
| 4.6.3.4 getComponent() | 17 |
| 4.6.3.5 getComponentArrays() | 18 |
| 4.6.3.6 getComponentBitset() | 18 |
| 4.6.3.7 getComponentTypeID() | 18 |
| 4.6.3.8 getDrawableComponents() | 19 |
| 4.6.3.9 getName() | 19 |

| | |
|---|----|
| 4.6.3.10 initEntity() | 19 |
| 4.6.3.11 setName() | 20 |
| 4.7 EntityManager Class Reference | 21 |
| 4.7.1 Constructor & Destructor Documentation | 22 |
| 4.7.1.1 EntityManager() | 22 |
| 4.7.1.2 ~EntityManager() | 22 |
| 4.7.2 Member Function Documentation | 23 |
| 4.7.2.1 addEntity() | 23 |
| 4.7.2.2 getEntities() | 23 |
| 4.7.2.3 getEntity() | 23 |
| 4.7.2.4 getEntityMap() | 24 |
| 4.7.2.5 initEntityManager() | 24 |
| 4.8 EntityManagerTest Class Reference | 25 |
| 4.9 EntityTest Class Reference | 26 |
| 4.10 EventEngine Class Reference | 27 |
| 4.10.1 Detailed Description | 27 |
| 4.10.2 Constructor & Destructor Documentation | 28 |
| 4.10.2.1 EventEngine() | 28 |
| 4.10.2.2 ~EventEngine() | 28 |
| 4.10.3 Member Function Documentation | 28 |
| 4.10.3.1 addKeyPressed() | 28 |
| 4.10.3.2 addMouseButtonPressed() | 29 |
| 4.10.3.3 addMouseMoved() | 29 |
| 4.10.3.4 getEvent() | 30 |
| 4.10.3.5 getKeyPressedMap() | 30 |
| 4.10.3.6 getMouseButtonPressedMap() | 30 |
| 4.10.3.7 getMouseMovedMap() | 31 |
| 4.10.3.8 init() | 31 |
| 4.11 EventTest Class Reference | 31 |
| 4.12 GameEngine Class Reference | 32 |
| 4.12.1 Detailed Description | 35 |
| 4.12.2 Constructor & Destructor Documentation | 36 |
| 4.12.2.1 GameEngine() [1/2] | 36 |
| 4.12.2.2 GameEngine() [2/2] | 36 |
| 4.12.2.3 ~GameEngine() | 37 |
| 4.12.3 Member Function Documentation | 37 |
| 4.12.3.1 addWorld() | 37 |
| 4.12.3.2 eventGameEngine() | 37 |
| 4.12.3.3 getCurrentWorld() | 38 |
| 4.12.3.4 getEventEngine() | 38 |
| 4.12.3.5 getFilesRessources() | 38 |
| 4.12.3.6 getMapMusic() | 39 |

| | |
|---|----|
| 4.12.3.7 getMapTexture() | 39 |
| 4.12.3.8 getWindow() | 39 |
| 4.12.3.9 getWorld() | 40 |
| 4.12.3.10 getWorldMap() | 40 |
| 4.12.3.11 initialize() | 40 |
| 4.12.3.12 initializeAllFiles() | 41 |
| 4.12.3.13 initializeMusic() | 41 |
| 4.12.3.14 initializeMusicFunction() | 42 |
| 4.12.3.15 initializeSound() | 42 |
| 4.12.3.16 initializeSoundFunction() | 42 |
| 4.12.3.17 initializeSpriteFunction() | 43 |
| 4.12.3.18 initializeTexture() | 43 |
| 4.12.3.19 initializeWorldMap() | 43 |
| 4.12.3.20 isWindowOpen() | 44 |
| 4.12.3.21 renderGameEngine() | 44 |
| 4.12.3.22 run() [1/2] | 44 |
| 4.12.3.23 run() [2/2] | 45 |
| 4.12.3.24 setCurrentWorld() | 45 |
| 4.12.3.25 setWindow() | 45 |
| 4.12.3.26 updateGameEngine() | 46 |
| 4.13 GameEngineTest Class Reference | 46 |
| 4.14 Music Class Reference | 48 |
| 4.15 Rect< T > Class Template Reference | 49 |
| 4.15.1 Detailed Description | 50 |
| 4.15.2 Constructor & Destructor Documentation | 50 |
| 4.15.2.1 Rect() | 50 |
| 4.15.2.2 ~Rect() | 51 |
| 4.15.3 Member Function Documentation | 51 |
| 4.15.3.1 contains() | 51 |
| 4.15.3.2 getHeight() | 52 |
| 4.15.3.3 getLeft() | 52 |
| 4.15.3.4 getRect() | 53 |
| 4.15.3.5 getTop() | 53 |
| 4.15.3.6 getWidth() | 53 |
| 4.16 Script Class Reference | 54 |
| 4.17 Sound Class Reference | 54 |
| 4.17.1 Detailed Description | 56 |
| 4.17.2 Constructor & Destructor Documentation | 56 |
| 4.17.2.1 Sound() [1/2] | 56 |
| 4.17.2.2 Sound() [2/2] | 56 |
| 4.17.2.3 ~Sound() | 56 |
| 4.17.3 Member Function Documentation | 57 |

| | | |
|-----------|--|----|
| 4.17.3.1 | getSound() | 57 |
| 4.17.3.2 | getSoundBuffer() | 57 |
| 4.17.3.3 | getVolume() | 57 |
| 4.17.3.4 | isPlaying() | 59 |
| 4.17.3.5 | loadSoundBuffer() | 59 |
| 4.17.3.6 | pause() | 59 |
| 4.17.3.7 | play() | 60 |
| 4.17.3.8 | setLoop() | 60 |
| 4.17.3.9 | setSound() | 60 |
| 4.17.3.10 | setSoundBuffer() | 61 |
| 4.17.3.11 | setVolume() | 61 |
| 4.17.3.12 | stop() | 61 |
| 4.18 | Sprite Class Reference | 62 |
| 4.18.1 | Detailed Description | 64 |
| 4.18.2 | Constructor & Destructor Documentation | 64 |
| 4.18.2.1 | Sprite() [1/2] | 64 |
| 4.18.2.2 | Sprite() [2/2] | 65 |
| 4.18.2.3 | ~Sprite() | 65 |
| 4.18.3 | Member Function Documentation | 65 |
| 4.18.3.1 | applyDeferredSprite() | 66 |
| 4.18.3.2 | createSprite() [1/3] | 66 |
| 4.18.3.3 | createSprite() [2/3] | 66 |
| 4.18.3.4 | createSprite() [3/3] | 67 |
| 4.18.3.5 | draw() | 67 |
| 4.18.3.6 | getBit() | 67 |
| 4.18.3.7 | getSprite() | 68 |
| 4.18.3.8 | getTexture() | 68 |
| 4.18.3.9 | initSprite() | 68 |
| 4.18.3.10 | isTextureLoaded() | 69 |
| 4.18.3.11 | setDeferredSprite() | 69 |
| 4.18.3.12 | setPosition() [1/2] | 69 |
| 4.18.3.13 | setPosition() [2/2] | 70 |
| 4.18.3.14 | setRotation() [1/2] | 70 |
| 4.18.3.15 | setRotation() [2/2] | 70 |
| 4.18.3.16 | setScale() [1/2] | 71 |
| 4.18.3.17 | setScale() [2/2] | 71 |
| 4.18.3.18 | setSprite() [1/2] | 71 |
| 4.18.3.19 | setSprite() [2/2] | 72 |
| 4.18.3.20 | setTexture() | 72 |
| 4.18.3.21 | setTransformSprite() [1/2] | 73 |
| 4.18.3.22 | setTransformSprite() [2/2] | 73 |
| 4.19 | SpriteTest Class Reference | 74 |

| | |
|---|----|
| 4.20 TestWorld Class Reference | 75 |
| 4.21 Text Class Reference | 76 |
| 4.21.1 Member Function Documentation | 77 |
| 4.21.1.1 draw() | 77 |
| 4.22 toSFML Class Reference | 78 |
| 4.22.1 Detailed Description | 78 |
| 4.22.2 Constructor & Destructor Documentation | 78 |
| 4.22.2.1 toSFML() | 78 |
| 4.22.2.2 ~toSFML() | 79 |
| 4.22.3 Member Function Documentation | 79 |
| 4.22.3.1 toSFMLRect() | 79 |
| 4.23 Transform Class Reference | 80 |
| 4.23.1 Detailed Description | 81 |
| 4.23.2 Constructor & Destructor Documentation | 81 |
| 4.23.2.1 Transform() | 81 |
| 4.23.2.2 ~Transform() | 82 |
| 4.23.3 Member Function Documentation | 82 |
| 4.23.3.1 getBit() | 82 |
| 4.23.3.2 getPosition() | 82 |
| 4.23.3.3 getRotation() | 83 |
| 4.23.3.4 getScale() | 83 |
| 4.23.3.5 getTransformStruct() | 83 |
| 4.23.3.6 init() | 84 |
| 4.23.3.7 setTransform() | 84 |
| 4.23.3.8 setTransformPosition() | 85 |
| 4.23.3.9 setTransformRotation() | 85 |
| 4.23.3.10 setTransformScale() | 85 |
| 4.24 TransformTest Class Reference | 86 |
| 4.25 Vector2< T > Class Template Reference | 87 |
| 4.25.1 Detailed Description | 87 |
| 4.25.2 Constructor & Destructor Documentation | 87 |
| 4.25.2.1 Vector2() | 87 |
| 4.25.2.2 ~Vector2() | 88 |
| 4.25.3 Member Function Documentation | 88 |
| 4.25.3.1 getVector2Struct() | 88 |
| 4.25.3.2 getX() | 89 |
| 4.25.3.3 getY() | 89 |
| 4.26 World Class Reference | 89 |
| 4.26.1 Detailed Description | 91 |
| 4.26.2 Constructor & Destructor Documentation | 91 |
| 4.26.2.1 World() | 91 |
| 4.26.2.2 ~World() | 92 |

| | |
|--|-----------|
| 4.26.3 Member Function Documentation | 92 |
| 4.26.3.1 addEntityManager() | 92 |
| 4.26.3.2 createEntities() | 92 |
| 4.26.3.3 getEntityManager() | 93 |
| 4.26.3.4 getEntityManagerMap() | 93 |
| 4.26.3.5 getNameWorld() | 93 |
| 4.26.3.6 initWorld() | 94 |
| 4.26.3.7 setNameWorld() | 94 |
| Index | 95 |

Chapter 1

Engine

1.1 Compilation

1.1.1 Linux

Use the following command to compile the engine:

```
cmake -Bbuild  
make -Cbuild
```

Use the following command to compile the engine and its tests:

```
cmake -Bbuild -DBUILD_TESTS=ON  
make -Cbuild
```

Use the following command for create the package (.tgz or .zip) after compile:

```
cd build  
cpack
```


Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

| | |
|-----------------------------|----|
| Archetypes | 7 |
| Color | 8 |
| Components | 9 |
| Entity | 13 |
| EntityManager | 21 |
| World | 89 |
| GameEngine | 32 |
| Music | 48 |
| Sound | 54 |
| Transform | 80 |
| Rect< T > | 49 |
| Sprite | 62 |
| Text | 76 |
| DrawableComponent | 11 |
| Sprite | 62 |
| Text | 76 |
| EventEngine | 27 |
| GameEngine | 32 |
| Script | 54 |
| testing::Test | |
| AudioTest | 7 |
| EntityManagerTest | 25 |
| EntityTest | 26 |
| EventTest | 31 |
| GameEngineTest | 46 |
| SpriteTest | 74 |
| TestWorld | 75 |
| TransformTest | 86 |
| toSFML | 78 |
| Sprite | 62 |
| Vector2< T > | 87 |
| Vector2< float > | 87 |

Chapter 3

Class Index

3.1 Class List

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

| | |
|--|----|
| Archetypes | 7 |
| AudioTest | 7 |
| Color | 8 |
| Components | |
| Components class: Components is a class that represents a component in the game | 9 |
| DrawableComponent | |
| DrawableComponent class: DrawableComponent is a class that represents a drawable component in the game | 11 |
| Entity | |
| Entity class: Entity is a class that represents an entity in the game | 13 |
| EntityManager | 21 |
| EntityManagerTest | 25 |
| EntityTest | 26 |
| EventEngine | |
| EventEngine class: EventEngine is a class that represents the event engine of the game | 27 |
| EventTest | 31 |
| GameEngine | |
| GameEngine class: GameEngine is a class that represents the game engine | 32 |
| GameEngineTest | 46 |
| Music | 48 |
| Rect< T > | |
| Rect class: Rect is a class that represents a rectangle | 49 |
| Script | 54 |
| Sound | |
| Sound class: Sound is a class that represents the audio properties of a Component | 54 |
| Sprite | |
| Sprite class: Sprite is a class that represents the rendering properties of a Component | 62 |
| SpriteTest | 74 |
| TestWorld | 75 |
| Text | 76 |
| toSFML | |
| toSFML class: toSFML is a class that convert some class into SFML class | 78 |
| Transform | |
| Transform class: Transform is a class that represents the transform of a Component | 80 |
| TransformTest | 86 |

| | |
|---|--------------------|
| Vector2< T > | |
| Vector class: Vector is a class that represents a vector in 2 dimensions | 87 |
| World | |
| World class: World is a class that represents the world of the game | 89 |

Chapter 4

Class Documentation

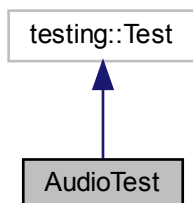
4.1 Archetypes Class Reference

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

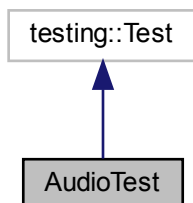
- `src/Archetype/include/Archetypes.h`

4.2 AudioTest Class Reference

Inheritance diagram for AudioTest:



Collaboration diagram for AudioTest:



Protected Member Functions

- void **SetUp** () override
- void **TearDown** () override

Protected Attributes

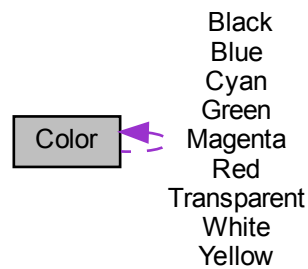
- Audio **audio**

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

- tests/Components/all_components/TestAudio.cpp

4.3 Color Class Reference

Collaboration diagram for Color:



Public Member Functions

- int **getRed** () const
- int **getGreen** () const
- int **getBlue** () const
- int **getAlpha** () const
- void **setRed** (int newRed)
- void **setGreen** (int newGreen)
- void **setBlue** (int newBlue)
- void **setAlpha** (int newAlpha)
- **operator sf::Color** () const

Static Public Member Functions

- static **Color** **fromSFMLColor** (const sf::Color &sfColor)

Static Public Attributes

- static const **Color Black** = Color::fromSFMLColor(sf::Color::Black)
- static const **Color White** = Color::fromSFMLColor(sf::Color::White)
- static const **Color Red** = Color::fromSFMLColor(sf::Color::Red)
- static const **Color Green** = Color::fromSFMLColor(sf::Color::Green)
- static const **Color Blue** = Color::fromSFMLColor(sf::Color::Blue)
- static const **Color Yellow** = Color::fromSFMLColor(sf::Color::Yellow)
- static const **Color Magenta** = Color::fromSFMLColor(sf::Color::Magenta)
- static const **Color Cyan** = Color::fromSFMLColor(sf::Color::Cyan)
- static const **Color Transparent** = Color::fromSFMLColor(sf::Color::Transparent)

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

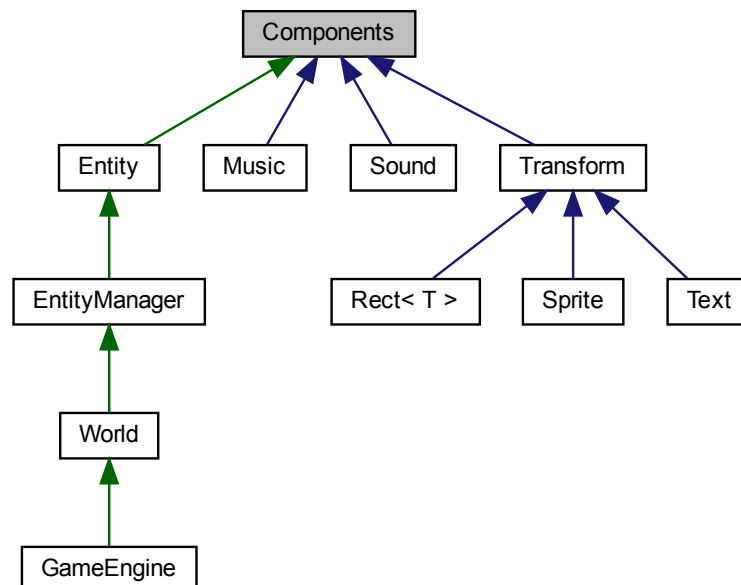
- src/Other/include/Color.h
- src/Other/Color.cpp

4.4 Components Class Reference

Components class: **Components** is a class that represents a component in the game.

```
#include <Components.h>
```

Inheritance diagram for Components:



Public Member Functions

- [Components](#) ()=default
Default [Components](#) constructor.
- virtual [~Components](#) ()=default
[Components](#) destructor.
- virtual bool [init](#) ()
[init\(\)](#): Initialize the component
- virtual void **update** (sf::Time timeDelta)=0

4.4.1 Detailed Description

[Components](#) class: [Components](#) is a class that represents a component in the game.

[Components](#) are the building blocks of the game. They are attached to entities and define their behavior.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Components()

```
Components::Components ( ) [default]
```

Default [Components](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.4.2.2 ~Components()

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

[Components](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.4.3 Member Function Documentation

4.4.3.1 init()

```
virtual bool Components::init ( ) [inline], [virtual]
```

[init\(\)](#): Initialize the component

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: true if the component is initialized, false otherwise

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

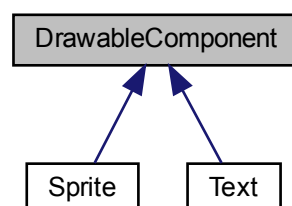
- `src/Components/include/Components.h`

4.5 DrawableComponent Class Reference

[DrawableComponent](#) class: [DrawableComponent](#) is a class that represents a drawable component in the game.

```
#include <DrawableComponent.h>
```

Inheritance diagram for DrawableComponent:



Public Member Functions

- virtual [~DrawableComponent](#) ()=default
Default [DrawableComponent](#) constructor.
- virtual void [draw](#) (sf::RenderWindow &window) const =0
[draw\(\)](#): Draw the component

4.5.1 Detailed Description

[DrawableComponent](#) class: [DrawableComponent](#) is a class that represents a drawable component in the game.

DrawableComponents are components that can be drawn on the screen.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ~DrawableComponent()

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

Default [DrawableComponent](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.5.3 Member Function Documentation

4.5.3.1 draw()

```
virtual void DrawableComponent::draw (
    sf::RenderWindow & window ) const [pure virtual]
```

[draw\(\)](#): Draw the component

Parameters

| | |
|---------------|---------------------------------|
| <i>window</i> | Window to draw the component on |
|---------------|---------------------------------|

Returns

void

Implemented in [Text](#), and [Sprite](#).

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

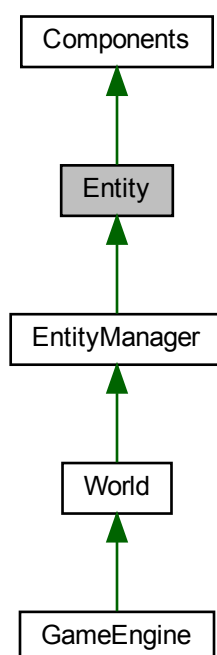
- `src/Components/include/DrawableComponent.h`

4.6 Entity Class Reference

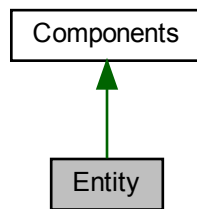
[Entity](#) class: [Entity](#) is a class that represents an entity in the game.

```
#include <entity.h>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



Public Member Functions

- [Entity](#) ()=default
Default [Entity](#) constructor.
- [Entity](#) (std::string nameEntity, [Archetypes](#) newArchetype=[Archetypes](#)())
[Entity](#) constructor.
- [~Entity](#) () override=default
[Entity](#) destructor.
- bool [initEntity](#) ()
[init\(\)](#): Initialize the entity
- std::string [getName](#) () const
[genName\(\)](#): Get the name of the entity
- void **update** (sf::Time deltaTime) override
- void [setName](#) (std::string newName)
[setName\(\)](#): Set the name of the entity
- void [addDrawable](#) ([Components](#) *component)
[addDrawable\(\)](#): Add a drawable component to the entity
- void [drawEntity](#) (sf::RenderWindow &window)
[drawEntity\(\)](#): Draw the entities
- template<typename T , typename... TArgs>
T & [addComponent](#) (TArgs &&... args)
[addComponent\(\)](#): Add a component to the entity
- template<typename T >
T & [getComponent](#) ()
[getComponent\(\)](#): Get a component from the entity
- template<typename T >
std::size_t [getComponentTypeID](#) () noexcept
[getComponentTypeID\(\)](#): Get the ID of a component
- std::bitset< 6 > [getComponentBitset](#) () const
[getComponentBitset\(\)](#): Get the bitset of the components
- std::vector< [DrawableComponent](#) * > [getDrawableComponents](#) () const
[getDrawableComponents\(\)](#): Get the drawable components of the entity
- std::array< [Components](#) *, 6 > [getComponentArrays](#) () const
[getComponentArrays\(\)](#): Get the array of components

Additional Inherited Members

4.6.1 Detailed Description

[Entity](#) class: [Entity](#) is a class that represents an entity in the game.

The [Entity](#) class manages components associated with the entity.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Entity() [1/2]

```
Entity::Entity ( ) [default]
```

Default [Entity](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.6.2.2 Entity() [2/2]

```
Entity::Entity (
    std::string nameEntity,
    Archetypes newArchetype = Archetypes() ) [inline], [explicit]
```

[Entity](#) constructor.

Parameters

| | |
|---------------------|---|
| <i>nameEntity</i> | name of the entity |
| <i>newArchetype</i> | archetype of the entity (optional, default = new archetype) |

Returns

void

4.6.2.3 ~Entity()

```
Entity::~~Entity ( ) [override], [default]
```

[Entity](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.6.3 Member Function Documentation

4.6.3.1 addComponent()

```
template<typename T , typename... TArgs>
template Text & Entity::addComponent< Text > (
    TArgs &&... args )
```

[addComponent\(\)](#): Add a component to the entity

Template Parameters

| | |
|--------------|--|
| <i>T</i> | Type of the component |
| <i>TArgs</i> | Variadic template for component constructor arguments. |

Parameters

| | |
|-------------|----------------------------|
| <i>args</i> | arguments of the component |
|-------------|----------------------------|

Returns

T&: reference of the component

4.6.3.2 addDrawable()

```
void Entity::addDrawable (
    Components * component )
```

[addDrawable\(\)](#): Add a drawable component to the entity

Parameters

| | |
|------------------|------------------|
| <i>component</i> | component to add |
|------------------|------------------|

Returns

void

4.6.3.3 drawEntity()

```
void Entity::drawEntity (
    sf::RenderWindow & window )
```

[drawEntity\(\)](#): Draw the entities

Parameters

| | |
|---------------|-------------------------------------|
| <i>window</i> | window where the entities are drawn |
|---------------|-------------------------------------|

Returns

void

4.6.3.4 GetComponent()

```
template<typename T >
template Text & Entity::GetComponent< Text > ( )
```

[GetComponent\(\)](#): Get a component from the entity

Template Parameters

| | |
|----------|-----------------------|
| <i>T</i> | Type of the component |
|----------|-----------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

T&: reference of the component

4.6.3.5 GetComponentArrays()

```
std::array<Components*, 6> Entity::GetComponentArrays ( ) const [inline]
```

[GetComponentArrays\(\)](#): Get the array of components

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::array<Components*, 6>: array of components

4.6.3.6 GetComponentBitset()

```
std::bitset<6> Entity::GetComponentBitset ( ) const [inline]
```

[GetComponentBitset\(\)](#): Get the bitset of the components

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::bitset<6>: bitset of the components

4.6.3.7 GetComponentTypeID()

```
template<typename T >
template std::size_t Entity::GetComponentTypeID< Text > ( ) [noexcept]
```

[GetComponentTypeID\(\)](#): Get the ID of a component

Template Parameters

| | |
|----------|-----------------------|
| <i>T</i> | Type of the component |
|----------|-----------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::size_t: ID of the component

4.6.3.8 getDrawableComponents()

```
std::vector<DrawableComponent*> Entity::getDrawableComponents ( ) const [inline]
```

[getDrawableComponents\(\)](#): Get the drawable components of the entity

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::vector<DrawableComponent*>: drawable components of the entity

4.6.3.9 getName()

```
std::string Entity::getName ( ) const
```

[getName\(\)](#): Get the name of the entity

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::string: name of the entity

4.6.3.10 initEntity()

```
bool Entity::initEntity ( )
```

[init\(\)](#): Initialize the entity

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: true if the entity is initialized, false otherwise

4.6.3.11 setName()

```
void Entity::setName (
    std::string newName )
```

[setName\(\)](#): Set the name of the entity

Parameters

| | |
|----------------|------------------------|
| <i>newName</i> | new name of the entity |
|----------------|------------------------|

Returns

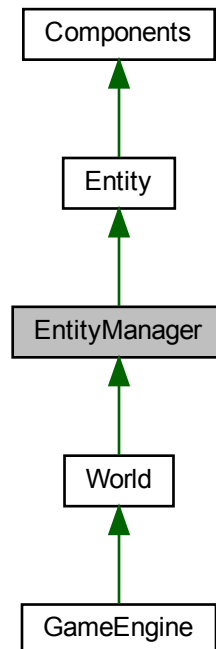
void

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

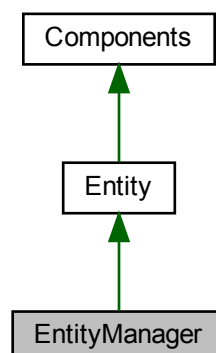
- src/Entity/include/entity.h
- src/Entity/entity.cpp

4.7 EntityManager Class Reference

Inheritance diagram for EntityManager:



Collaboration diagram for EntityManager:



Public Member Functions

- [EntityManager](#) ()=default
Default [EntityManager](#) constructor.
- [~EntityManager](#) ()=default
[EntityManager](#) destructor.
- [Entity](#) & [addEntity](#) (std::string nameEntity, [Archetypes](#) newArchetype=[Archetypes](#)())
[addEntity\(\)](#): Create and add a new entity to the entity manager.
- [Entity](#) & [getEntity](#) (std::string nameEntity)
[getEntity\(\)](#): Get an entity from the entity manager by its name.
- std::map< std::string, [Entity](#) * > [getEntities](#) () const
[getEntities\(\)](#): Get the [EntityManager](#)'s entities.
- std::map< std::string, [Entity](#) * > [getEntityMap](#) () const
[getEntityMap\(\)](#): Get the [EntityManager](#)'s entity map.
- bool [initEntityManager](#) ()
[initEntityManager\(\)](#): Initialize the [EntityManager](#).

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 EntityManager()

```
EntityManager::EntityManager ( ) [default]
```

Default [EntityManager](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.7.1.2 ~EntityManager()

```
EntityManager::~~EntityManager ( ) [default]
```

[EntityManager](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.7.2 Member Function Documentation

4.7.2.1 addEntity()

```
Entity & EntityManager::addEntity (
    std::string nameEntity,
    Archetypes newArchetype = Archetypes() )
```

addEntity(): Create and add a new entity to the entity manager.

Template Parameters

| | |
|--------------|------------------------|
| <i>T</i> | Type of the entity. |
| <i>TArgs</i> | Type of the arguments. |

Parameters

| | |
|-------------|--------------------------|
| <i>args</i> | Arguments of the entity. |
|-------------|--------------------------|

4.7.2.2 getEntities()

```
std::map< std::string, Entity * > EntityManager::getEntities ( ) const
```

getEntities(): Get the **EntityManager**'s entities.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::map<std::string, Entity *>: Entities.

4.7.2.3 getEntity()

```
Entity & EntityManager::getEntity (
    std::string nameEntity )
```

getEntity(): Get an entity from the entity manager by its name.

Template Parameters

| | |
|----------|---------------------|
| <i>T</i> | Type of the entity. |
|----------|---------------------|

Parameters

| | |
|-------------------|---------------------|
| <i>nameEntity</i> | Name of the entity. |
|-------------------|---------------------|

Returns

T&: Reference of the entity.

4.7.2.4 `getEntityMap()`

```
std::map<std::string, Entity*> EntityManager::getEntityMap ( ) const [inline]
```

`getEntityMap()`: Get the `EntityManager`'s entity map.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

Entity::EntityMap: `Entity` map.

4.7.2.5 `initEntityManager()`

```
bool EntityManager::initEntityManager ( ) [inline]
```

`initEntityManager()`: Initialize the `EntityManager`.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

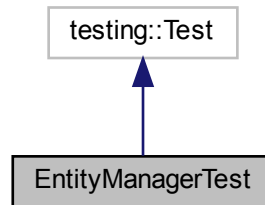
bool: true if the `EntityManager` is initialized, false otherwise.

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

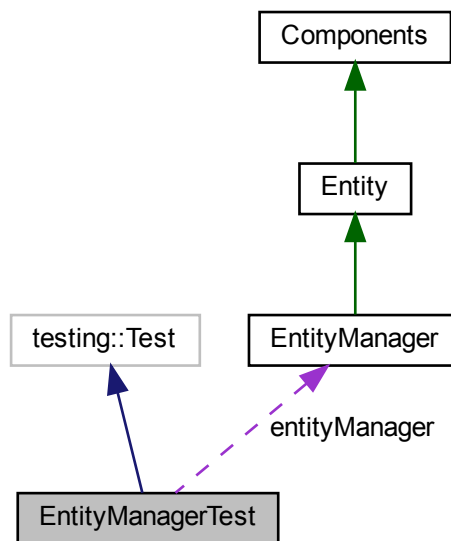
- `src/Entity/include/entityManager.h`
- `src/Entity/entityManager.cpp`

4.8 EntityManagerTest Class Reference

Inheritance diagram for EntityManagerTest:



Collaboration diagram for EntityManagerTest:



Protected Member Functions

- void **SetUp** () override
- void **TearDown** () override

Protected Attributes

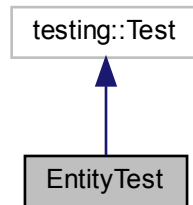
- [EntityManager](#) entityManager {}

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

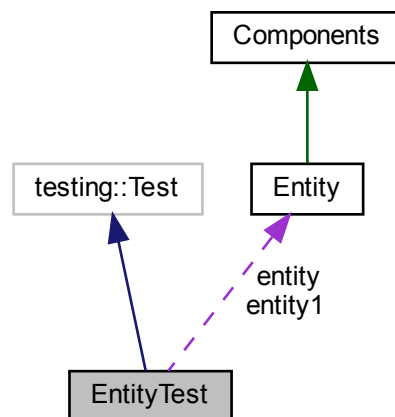
- tests/Entity/TestEntityManager.cpp

4.9 EntityTest Class Reference

Inheritance diagram for EntityTest:



Collaboration diagram for EntityTest:



Protected Attributes

- [Entity](#) entity
- [Entity](#) entity1

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

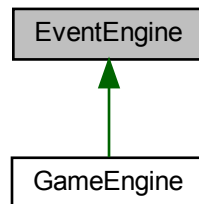
- tests/Entity/TestEntity.cpp

4.10 EventEngine Class Reference

`EventEngine` class: `EventEngine` is a class that represents the event engine of the game.

```
#include <eventEngine.h>
```

Inheritance diagram for `EventEngine`:



Public Member Functions

- `EventEngine ()=default`
Default `EventEngine` constructor.
- `virtual ~EventEngine ()=default`
`EventEngine` destructor.
- `bool init () const`
`init()`: Initialize the `EventEngine`.
- `sf::Event & getEvent ()`
`getEvent()`: Get the SFML Event.
- `void addKeyPressed (sf::Keyboard::Key keyboard, std::function< void()> function)`
`addKeyPressed()`: Add a key pressed to the map.
- `void addMouseButtonPressed (sf::Mouse::Button mouse, std::function< void()> function)`
`addMouseButtonPressed()`: Add a mouse button pressed to the map.
- `void addMouseMoved (std::string nameEntity, std::function< void()> function)`
`addMouseMoved()`: Add a mouse moved to the map.
- `std::map< sf::Keyboard::Key, std::function< void()> > & getKeyPressedMap ()`
`getKeyPressedMap()`: Get the map of the key pressed.
- `std::map< sf::Mouse::Button, std::function< void()> > & getMouseButtonPressedMap ()`
`getMouseButtonPressedMap()`: Get the map of the mouse button pressed.
- `std::map< std::string, std::function< void()> > & getMouseMovedMap ()`
`getMouseMovedPressedMap()`: Get the map of the key pressed.

4.10.1 Detailed Description

`EventEngine` class: `EventEngine` is a class that represents the event engine of the game.

The `EventEngine` class manages the events of the game.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 EventEngine()

```
EventEngine::EventEngine ( ) [default]
```

Default [EventEngine](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.10.2.2 ~EventEngine()

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

[EventEngine](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.10.3 Member Function Documentation

4.10.3.1 addKeyPressed()

```
void EventEngine::addKeyPressed (
    sf::Keyboard::Key keyboard,
    std::function< void()> function )
```

[addKeyPressed\(\)](#): Add a key pressed to the map.

Parameters

| | |
|-----------------|--|
| <i>keyboard</i> | SFML Keyboard::Key of the key pressed. |
| <i>function</i> | Function to execute when the key is pressed. |

Returns

void

4.10.3.2 addMouseButtonPressed()

```
void EventEngine::addMouseButtonPressed (
    sf::Mouse::Button mouse,
    std::function< void()> function )
```

[addMouseButtonPressed\(\)](#): Add a mouse button pressed to the map.

Parameters

| | |
|-----------------|---|
| <i>mouse</i> | SFML Mouse::Button of the mouse button pressed. |
| <i>function</i> | Function to execute when the mouse button is pressed. |

Returns

void

4.10.3.3 addMouseMoved()

```
void EventEngine::addMouseMoved (
    std::string nameEntity,
    std::function< void()> function )
```

[addMouseMoved\(\)](#): Add a mouse moved to the map.

Parameters

| | |
|-------------------|---|
| <i>nameEntity</i> | : Name of the Entity you want. |
| <i>function</i> | Function to execute when the mouse moved on entity. |

Returns

void

4.10.3.4 `getEvent()`

```
sf::Event& EventEngine::getEvent ( ) [inline]
```

[`getEvent\(\)`](#): Get the SFML Event.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

sf::Event: The SFML Event.

4.10.3.5 `getKeyPressedMap()`

```
std::map<sf::Keyboard::Key, std::function<void()> >& EventEngine::getKeyPressedMap ( ) [inline]
```

[`getKeyPressedMap\(\)`](#): Get the map of the key pressed.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::map<sf::Keyboard::Key, std::function<void()>>: The map of the key pressed.

4.10.3.6 `getMouseButtonPressedMap()`

```
std::map<sf::Mouse::Button, std::function<void()> >& EventEngine::getMouseButtonPressedMap ( ) [inline]
```

[`getMouseButtonPressedMap\(\)`](#): Get the map of the mouse button pressed.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::map<sf::Mouse::Button, std::function<void()>>: The map of the mouse button pressed.

4.10.3.7 getMouseMovedMap()

```
std::map<std::string, std::function<void()> >& EventEngine::getMouseMovedMap ( ) [inline]
```

getMouseMovedPressedMap(): Get the map of the key pressed.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::map<std::string, std::function<void()>>: The map of the mouse moved.

4.10.3.8 init()

```
bool EventEngine::init ( ) const [inline]
```

init(): Initialize the [EventEngine](#).

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

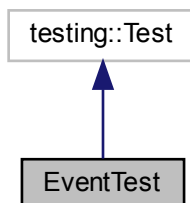
bool: True if the [EventEngine](#) is initialized, false otherwise.

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

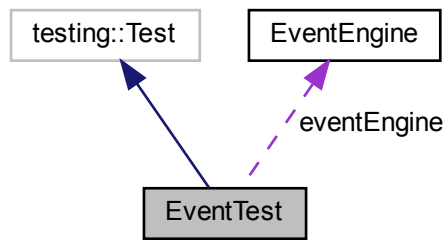
- src/Event/include/eventEngine.h
- src/Event/eventEngine.cpp

4.11 EventTest Class Reference

Inheritance diagram for EventTest:



Collaboration diagram for EventTest:



Protected Attributes

- [EventEngine](#) `eventEngine`

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

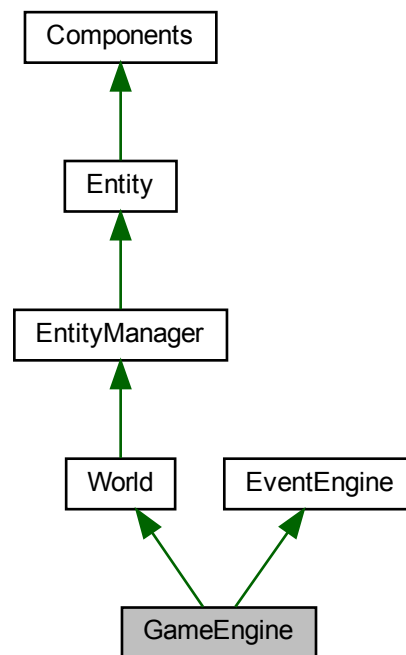
- `tests/Event/TestEvent.cpp`

4.12 GameEngine Class Reference

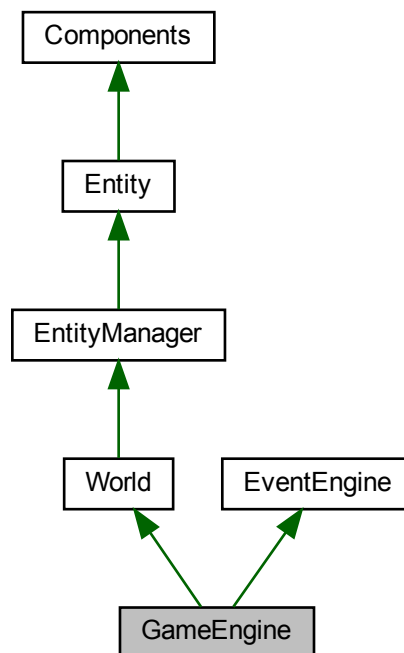
[GameEngine](#) class: [GameEngine](#) is a class that represents the game engine.

```
#include <gameEngine.h>
```


Inheritance diagram for GameEngine:



Collaboration diagram for GameEngine:



Public Member Functions

- [GameEngine](#) ()=default
< Time of the game. Using with the Clock.
- [GameEngine](#) (sf::VideoMode mode, std::string type, sf::String title, sf::Uint32 style=sf::Style::Default, const sf::ContextSettings &settings=sf::ContextSettings())
GameEngine constructor with parameters.
- [~GameEngine](#) ()=default
GameEngine destructor.
- void [run](#) (std::map< std::string, std::unique_ptr< [World](#) >> mapWorld, std::map< std::string, std::string > pathRessources, std::string firstScene)
run(): Run the game engine (with parameters).
- void [run](#) ()
run(): Run the game engine (without parameters).
- void [renderGameEngine](#) ()
renderGameEngine(): Render the game engine.
- void [eventGameEngine](#) ()
eventGameEngine(): Manage the events of the game engine.
- bool [isWindowOpen](#) ()
isWindowOpen(): Check if the window is open.
- void [updateGameEngine](#) ()
updateGameEngine(): Update the game engine.
- std::vector< std::string > [getFilesRessources](#) (std::string pathDirectory)

- *getFilesRessources()*: Get all the ressources type files in the given directory.
- void **initialize** (std::map< std::string, std::unique_ptr< **World** >> mapWorld, std::map< std::string, std::string > pathRessources, std::string firstScene)
 - initialize()*: Initialize the game engine.
- void **initializeSpriteFunction** ()
 - initializeSpriteFunction()*: Initialize the sprites function.
- void **initializeSoundFunction** ()
 - initializeSoundFunction()*: Initialize the sound function.
- void **initializeMusicFunction** ()
 - initializeMusicFunction()*: Initialize the music function.
- void **initializeTextFunction** ()
- void **initializeAllFiles** (std::map< std::string, std::string > pathRessources)
 - initializeAllFiles()*: Initialize all the ressources files the engine need.
- void **initializeTexture** (std::string path)
 - initializeTexture()*: Initialize the textures with their path.
- void **initializeSound** (std::string path)
 - initializeSound()*: Initialize the sound with their path.
- void **initializeMusic** (std::string path)
 - initializeMusic()*: Initialize the music with their path.
- void **initializeFont** (std::string path)
- void **initializeWorldMap** (std::map< std::string, std::unique_ptr< **World** >> mapWorld)
 - initializeWorldMap()*: Initialize the world map.
- const auto & **getWindow** ()
 - getWindow()*: Get the window.
- void **setWindow** ()
 - setWindow()*: Set the window.
- **EventEngine** & **getEventEngine** ()
 - getEventEngine()*: Get the event engine.
- void **setCurrentWorld** (**World** *world)
 - setCurrentWorld()*: Set **GameEngine**'s current world.
- **World** * **getCurrentWorld** ()
 - getCurrentWorld()*: Get **GameEngine**'s current world.
- **World** & **addWorld** (std::string nameWorld, std::unique_ptr< **World** > world)
 - addWorld()*: Add a world to the world map.
- **World** & **getWorld** (std::string nameWorld)
 - getWorld()*: Get a world from the world map with its name.
- std::map< std::string, std::shared_ptr< sf::Texture > > **getMapTexture** () const
 - getMapTexture()*: Get **GameEngine**'s map of the textures.
- std::map< std::string, **World** * > **getWorldMap** () const
 - getWorldMap()*: Get **GameEngine**'s map of the worlds.
- std::map< std::string, std::shared_ptr< sf::Music > > **getMapMusic** () const
 - getMapMusic()*: Get **GameEngine**'s map of the music.
- std::map< std::string, std::shared_ptr< sf::SoundBuffer > > **getMapSound** () const
- std::map< std::string, std::shared_ptr< sf::Font > > **getMapFont** () const

Additional Inherited Members

4.12.1 Detailed Description

GameEngine class: **GameEngine** is a class that represents the game engine.

The **GameEngine** class manages the game engine.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 `GameEngine()` [1/2]

```
GameEngine::GameEngine ( ) [default]
```

< Time of the game. Using with the Clock.

Default [GameEngine](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.2.2 `GameEngine()` [2/2]

```
GameEngine::GameEngine (
    sf::VideoMode mode,
    std::string type,
    sf::String title,
    sf::Uint32 style = sf::Style::Default,
    const sf::ContextSettings & settings = sf::ContextSettings() ) [explicit]
```

[GameEngine](#) constructor with parameters.

Parameters

| | |
|-----------------|--|
| <i>mode</i> | Video mode. |
| <i>type</i> | Type of the graphics ("2D" or "3D"). |
| <i>title</i> | Title of the window. |
| <i>style</i> | Style of the window (sf::Style::Default by default). |
| <i>settings</i> | Settings of the window. |

Returns

void

4.12.2.3 ~GameEngine()

```
GameEngine::~~GameEngine ( ) [default]
```

[GameEngine](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3 Member Function Documentation

4.12.3.1 addWorld()

```
World & GameEngine::addWorld (
    std::string nameWorld,
    std::unique_ptr< World > world )
```

[addWorld\(\)](#): Add a world to the world map.

Parameters

| | |
|------------------|-------------------------------|
| <i>nameWorld</i> | Name of the world. |
| <i>world</i> | World to add. |

Returns

[World&](#): The world.

4.12.3.2 eventGameEngine()

```
void GameEngine::eventGameEngine ( )
```

[eventGameEngine\(\)](#): Manage the events of the game engine.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.3 getCurrentWorld()

```
World* GameEngine::getCurrentWorld ( ) [inline]
```

[getCurrentWorld\(\)](#): Get [GameEngine](#)'s current world.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

World*: [GameEngine](#)'s current world.

4.12.3.4 getEventEngine()

```
EventEngine& GameEngine::getEventEngine ( ) [inline]
```

[getEventEngine\(\)](#): Get the event engine.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

[EventEngine&](#): [GameEngine](#)'s [EventEngine](#).

4.12.3.5 getFilesRessources()

```
std::vector< std::string > GameEngine::getFilesRessources (
    std::string pathDirectory )
```

[getFilesRessources\(\)](#): Get all the ressources type files in the given directory.

Parameters

| | |
|----------------------|------------------------|
| <i>pathDirectory</i> | Path of the directory. |
|----------------------|------------------------|

Returns

`std::vector<std::string>`: Vector of the ressources type files' names.

4.12.3.6 getMapMusic()

```
std::map<std::string, std::shared_ptr<sf::Music> > GameEngine::getMapMusic ( ) const [inline]
```

[getMapMusic\(\)](#): Get [GameEngine](#)'s map of the music.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

`std::map<std::string, std::shared_ptr<sf::Music>>`: [GameEngine](#)'s map of the musics.

4.12.3.7 getMapTexture()

```
std::map<std::string, std::shared_ptr<sf::Texture> > GameEngine::getMapTexture ( ) const [inline]
```

[getMapTexture\(\)](#): Get [GameEngine](#)'s map of the textures.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

`std::map<std::string, std::shared_ptr<sf::Texture>>`: [GameEngine](#)'s map of the textures.

4.12.3.8 getWindow()

```
const auto& GameEngine::getWindow ( ) [inline]
```

[getWindow\(\)](#): Get the window.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

`std::variant<std::unique_ptr<sf::Window>, std::unique_ptr<sf::RenderWindow>>`: The [GameEngine](#)'s window

4.12.3.9 `getWorld()`

```
World & GameEngine::getWorld (
    std::string nameWorld )
```

[getWorld\(\)](#): Get a world from the world map with its name.

Parameters

| | |
|------------------|--------------------|
| <i>nameWorld</i> | Name of the world. |
|------------------|--------------------|

Returns

[World&](#): [GameEngine](#)'s world.

4.12.3.10 `getWorldMap()`

```
std::map<std::string, World *> GameEngine::getWorldMap ( ) const [inline]
```

[getWorldMap\(\)](#): Get [GameEngine](#)'s map of the worlds.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

`std::map<std::string, World*>`: [GameEngine](#)'s map of the worlds.

4.12.3.11 `initialize()`

```
void GameEngine::initialize (
    std::map< std::string, std::unique_ptr< World >> mapWorld,
    std::map< std::string, std::string > pathResources,
    std::string firstScene )
```

[initialize\(\)](#): Initialize the game engine.

Parameters

| | |
|-----------------------|--|
| <i>mapWorld</i> | Map of World classes' unique pointers. |
| <i>pathRessources</i> | Map of the path of the ressources (assets). |
| <i>firstScene</i> | Name of the first scene. |

Returns

void

4.12.3.12 initializeAllFiles()

```
void GameEngine::initializeAllFiles (
    std::map< std::string, std::string > pathRessources )
```

[initializeAllFiles\(\)](#): Initialize all the ressources files the engine need.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.13 initializeMusic()

```
void GameEngine::initializeMusic (
    std::string path )
```

[initializeMusic\(\)](#): Initialize the music with their path.

Parameters

| | |
|-------------|----------------------|
| <i>path</i> | Path of the texture. |
|-------------|----------------------|

Returns

void

4.12.3.14 initializeMusicFunction()

```
void GameEngine::initializeMusicFunction ( )
```

[initializeMusicFunction\(\)](#): Initialize the music function.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.15 initializeSound()

```
void GameEngine::initializeSound (
    std::string path )
```

[initializeSound\(\)](#): Initialize the sound with their path.

Parameters

| | |
|-------------|----------------------|
| <i>path</i> | Path of the texture. |
|-------------|----------------------|

Returns

void

4.12.3.16 initializeSoundFunction()

```
void GameEngine::initializeSoundFunction ( )
```

[initializeSoundFunction\(\)](#): Initialize the sound function.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.17 initializeSpriteFunction()

```
void GameEngine::initializeSpriteFunction ( )
```

[initializeSpriteFunction\(\)](#): Initialize the sprites function.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.18 initializeTexture()

```
void GameEngine::initializeTexture (
    std::string path )
```

[initializeTexture\(\)](#): Initialize the textures with their path.

Parameters

| | |
|-------------|----------------------|
| <i>path</i> | Path of the texture. |
|-------------|----------------------|

Returns

void

4.12.3.19 initializeWorldMap()

```
void GameEngine::initializeWorldMap (
    std::map< std::string, std::unique_ptr< World >> mapWorld )
```

[initializeWorldMap\(\)](#): Initialize the world map.

Parameters

| | |
|-----------------|--|
| <i>mapWorld</i> | Map of World classes' unique pointers. |
|-----------------|--|

Returns

void

4.12.3.20 isWindowOpen()

```
bool GameEngine::isWindowOpen ( )
```

[isWindowOpen\(\)](#): Check if the window is open.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: True if the window is open, false otherwise.

4.12.3.21 renderGameEngine()

```
void GameEngine::renderGameEngine ( )
```

[renderGameEngine\(\)](#): Render the game engine.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.22 run() [1/2]

```
void GameEngine::run ( )
```

[run\(\)](#): Run the game engine (without parameters).

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.23 run() [2/2]

```
void GameEngine::run (
    std::map< std::string, std::unique_ptr< World >> mapWorld,
    std::map< std::string, std::string > pathResources,
    std::string firstScene )
```

[run\(\)](#): Run the game engine (with parameters).

Parameters

| | |
|----------------------|--|
| <i>mapWorld</i> | Map of World classes' unique pointers. |
| <i>pathResources</i> | Map of the path of the ressources (assets). |
| <i>firstScene</i> | Name of the first scene. |

Returns

void

4.12.3.24 setCurrentWorld()

```
void GameEngine::setCurrentWorld (
    World * world )
```

[setCurrentWorld\(\)](#): Set [GameEngine](#)'s current world.

Parameters

| | |
|--------------|-------------------------------|
| <i>world</i> | World to set. |
|--------------|-------------------------------|

Returns

void

4.12.3.25 setWindow()

```
void GameEngine::setWindow ( )
```

[setWindow\(\)](#): Set the window.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.12.3.26 updateGameEngine()

```
void GameEngine::updateGameEngine ( )
```

[updateGameEngine\(\)](#): Update the game engine.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

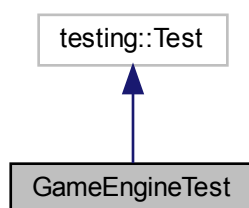
void

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

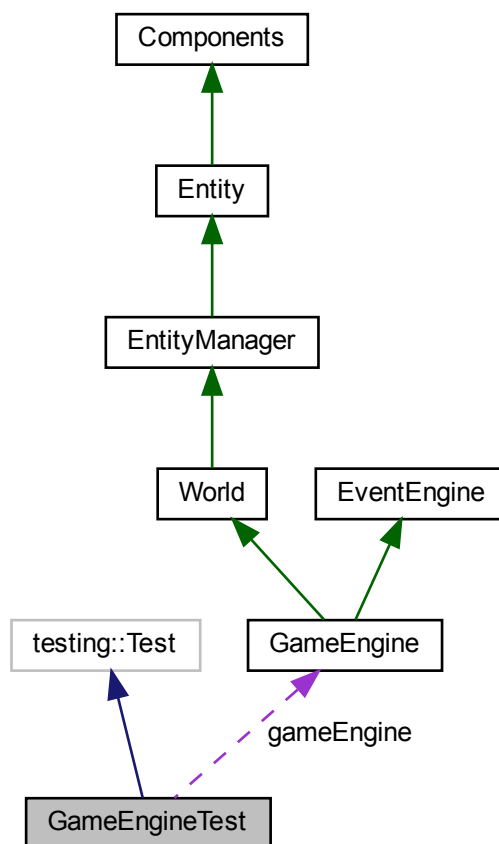
- src/GameEngine/include/gameEngine.h
- src/GameEngine/gameEngine.cpp

4.13 GameEngineTest Class Reference

Inheritance diagram for GameEngineTest:



Collaboration diagram for GameEngineTest:



Protected Member Functions

- void **TearDown** () override

Protected Attributes

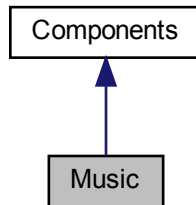
- [GameEngine](#) * **gameEngine**

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

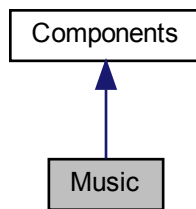
- tests/GameEngine/TestGameEngine.cpp

4.14 Music Class Reference

Inheritance diagram for Music:



Collaboration diagram for Music:



Public Member Functions

- void **setMusic** (std::map< std::string, std::shared_ptr< sf::Music >> mapMusic, std::string nameMusic)
- void **setDeferredMusic** (std::function< void()> setter)
- void **applyDeferredMusic** ()
- std::shared_ptr< sf::Music > **getMusic** () const
- void **play** ()
- void **play** (int seconds)
- void **stop** ()
- int **getBit** () const

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

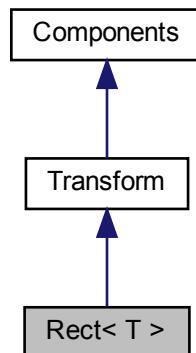
- src/Components/all_components/include/Music.h
- src/Components/all_components/Music.cpp

4.15 Rect< T > Class Template Reference

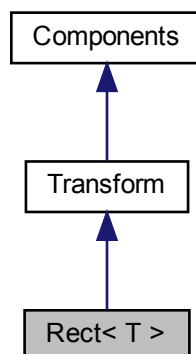
Rect class: Rect is a class that represents a rectangle.

```
#include <Rect.h>
```

Inheritance diagram for Rect< T >:



Collaboration diagram for Rect< T >:



Public Member Functions

- Rect (T left, T top, T width, T height)
 < Rect is the variable you can use for change the data in RectStruct.
- ~Rect ()=default

- *Rect* destructor.
- RectStruct [getRect](#) () const
getRect(): Get the using RectStruct.
- T [getLeft](#) () const
getLeft(): Get the using RectStruct left.
- T [getTop](#) () const
getTop(): Get the using RectStruct top.
- T [getWidth](#) () const
getWidth(): Get the using RectStruct width.
- T [getHeight](#) () const
getHeight(): Get the using RectStruct height.
- bool [contains](#) (T x, T y) const
contains(): Check if a point is in the rectangle.

4.15.1 Detailed Description

```
template<typename T>
class Rect< T >
```

[Rect](#) class: [Rect](#) is a class that represents a rectangle.

This create a rectangle and using for what you want.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Rect()

```
template<typename T >
Rect< T >::Rect (
    T left,
    T top,
    T width,
    T height ) [inline]
```

< [Rect](#) is the variable you can use for change the data in RectStruct.

[Rect](#) constructor with parameters.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|-------------|
| <i>left</i> | Position x. |
| <i>top</i> | Position y. |

Parameters

| | |
|---------------|---------------------------|
| <i>width</i> | Width of your rectangle. |
| <i>height</i> | Height of your rectangle. |

Returns

void

4.15.2.2 ~Rect()

```
template<typename T >
Rect< T >::~~Rect ( ) [default]
```

Rect destructor.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.15.3 Member Function Documentation**4.15.3.1 contains()**

```
template<typename T >
template bool Rect< T >::contains (
    T x,
    T y ) const
```

[contains\(\)](#): Check if a point is in the rectangle.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|----------|----------------------------|
| <i>x</i> | : Position x of the point. |
| <i>y</i> | : Position y of the point. |

Returns

T : T is the type you want (float, int,...).

4.15.3.2 getHeight()

```
template<typename T >
T Rect< T >::getHeight ( ) const [inline]
```

[getHeight\(\)](#): Get the using RectStruct height.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

T : T is the type you want (float, int,...).

4.15.3.3 getLeft()

```
template<typename T >
T Rect< T >::getLeft ( ) const [inline]
```

[getLeft\(\)](#): Get the using RectStruct left.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

T : T is the type you want (float, int,...).

4.15.3.4 getRect()

```
template<typename T >
RectStruct Rect< T >::getRect ( ) const [inline]
```

[getRect\(\)](#): Get the using RectStruct.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

[Rect](#)

4.15.3.5 getTop()

```
template<typename T >
T Rect< T >::getTop ( ) const [inline]
```

[getTop\(\)](#): Get the using RectStruct top.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

T : T is the type you want (float, int,...).

4.15.3.6 getWidth()

```
template<typename T >
T Rect< T >::getWidth ( ) const [inline]
```

[getWidth\(\)](#): Get the using RectStruct width.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

T : T is the type you want (float, int,...).

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

- src/Other/include/Rect.h
- src/Other/Rect.cpp

4.16 Script Class Reference

Public Member Functions

- virtual void **execute** ()=0

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

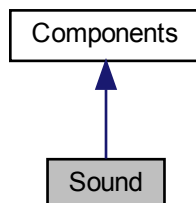
- src/Script/include/Script.h

4.17 Sound Class Reference

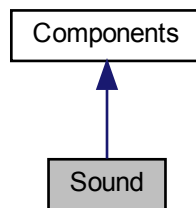
[Sound](#) class: [Sound](#) is a class that represents the audio properties of a Component.

```
#include <Sound.h>
```

Inheritance diagram for Sound:



Collaboration diagram for Sound:



Public Member Functions

- **Sound** ()=default
*Default **Sound** constructor.*
- **Sound** (const sf::SoundBuffer &buffer)
***Sound** constructor with an existing sound buffer. Automatically set the sound.*
- **~Sound** () override=default
***Sound** destructor.*
- bool **loadSoundBuffer** (const std::string &filePath)
***loadSoundBuffer()**: Load the sound buffer from a file. Automatically set the component sound. //! Only supports .wav, .ogg and FLAC files.*
- bool **setSoundBuffer** (const sf::SoundBuffer &buffer)
***setSoundBuffer()**: Set the sound buffer with an existing one. Automatically set the component sound.*
- const sf::SoundBuffer & **getSoundBuffer** () const
***getSoundBuffer()**: Get the current sound buffer.*
- bool **setSound** (const sf::Sound &sound)
***setSound()**: Set the sound with an existing one. Automatically set the component sound buffer.*
- void **setSound** (std::map< std::string, std::shared_ptr< sf::SoundBuffer >> mapSound, std::string name← Sound)
- void **setDeferredSound** (std::function< void()> setter)
- void **applyDeferredSound** ()
- const sf::Sound & **getSound** () const
***getSound()**: Get the current sound.*
- void **play** ()
***play()**: Play the audio.*
- void **pause** ()
***pause()**: Pause the audio.*
- void **stop** ()
***stop()**: Stop the audio.*
- void **setLoop** (bool loop)
***setLoop()**: Set whether the audio should loop or not.*
- void **setVolume** (float volume)
***setVolume()**: Set the volume of the audio.*
- float **getVolume** () const
***getVolume()**: Get the current volume level.*
- bool **isPlaying** () const
***isPlaying()**: Check if the audio is currently playing.*
- int **getBit** () const

4.17.1 Detailed Description

[Sound](#) class: [Sound](#) is a class that represents the audio properties of a Component.

The [Sound](#) class manages the audio representation of a Component using SFML.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 [Sound\(\)](#) [1/2]

```
Sound::Sound ( ) [default]
```

Default [Sound](#) constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.17.2.2 [Sound\(\)](#) [2/2]

```
Sound::Sound (
    const sf::SoundBuffer & buffer ) [explicit]
```

[Sound](#) constructor with an existing sound buffer. Automatically set the sound.

Parameters

| | |
|---------------|-----------------------------|
| <i>buffer</i> | SFML SoundBuffer for audio. |
|---------------|-----------------------------|

Returns

void

4.17.2.3 [~Sound\(\)](#)

```
Sound::~Sound ( ) [override], [default]
```

[Sound](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.17.3 Member Function Documentation

4.17.3.1 `getSound()`

```
const sf::Sound & Sound::getSound ( ) const
```

[`getSound\(\)`](#): Get the current sound.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

sf::Sound: SFML [Sound](#) for audio.

4.17.3.2 `getSoundBuffer()`

```
const sf::SoundBuffer & Sound::getSoundBuffer ( ) const
```

[`getSoundBuffer\(\)`](#): Get the current sound buffer.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

sf::SoundBuffer: SFML SoundBuffer for audio.

4.17.3.3 `getVolume()`

```
float Sound::getVolume ( ) const
```

[getVolume\(\)](#): Get the current volume level.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

float: Volume level (0 to 100).

4.17.3.4 isPlaying()

```
bool Sound::isPlaying ( ) const
```

[isPlaying\(\)](#): Check if the audio is currently playing.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: True if the audio is playing, false otherwise.

4.17.3.5 loadSoundBuffer()

```
bool Sound::loadSoundBuffer (
    const std::string & filePath )
```

[loadSoundBuffer\(\)](#): Load the sound buffer from a file. Automatically set the component sound. /\ Only supports .wav, .ogg and FLAC files.

Parameters

| | |
|-----------------|-------------------------|
| <i>filePath</i> | Path to the audio file. |
|-----------------|-------------------------|

Returns

bool: True if the sound buffer has been loaded, false otherwise.

4.17.3.6 pause()

```
void Sound::pause ( )
```

[pause\(\)](#): Pause the audio.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.17.3.7 play()

```
void Sound::play ( )
```

[play\(\)](#): Play the audio.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.17.3.8 setLoop()

```
void Sound::setLoop (
    bool loop )
```

[setLoop\(\)](#): Set whether the audio should loop or not.

Parameters

| | |
|-------------|---|
| <i>loop</i> | True to enable looping, false to disable. |
|-------------|---|

Returns

void

4.17.3.9 setSound()

```
bool Sound::setSound (
    const sf::Sound & sound )
```

[setSound\(\)](#): Set the sound with an existing one. Automatically set the component sound buffer.

Parameters

| | |
|--------------|---------------------------------------|
| <i>sound</i> | SFML Sound for audio. |
|--------------|---------------------------------------|

Returns

bool: True if the sound has been set, false otherwise.

4.17.3.10 setSoundBuffer()

```
bool Sound::setSoundBuffer (
    const sf::SoundBuffer & buffer )
```

[setSoundBuffer\(\)](#): Set the sound buffer with an existing one. Automatically set the component sound.

Parameters

| | |
|---------------|-----------------------------|
| <i>buffer</i> | SFML SoundBuffer for audio. |
|---------------|-----------------------------|

Returns

bool: True if the sound buffer has been set, false otherwise.

4.17.3.11 setVolume()

```
void Sound::setVolume (
    float volume )
```

[setVolume\(\)](#): Set the volume of the audio.

Parameters

| | |
|---------------|--------------------------|
| <i>volume</i> | Volume level (0 to 100). |
|---------------|--------------------------|

Returns

void

4.17.3.12 stop()

```
void Sound::stop ( )
```

[stop\(\)](#): Stop the audio.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

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

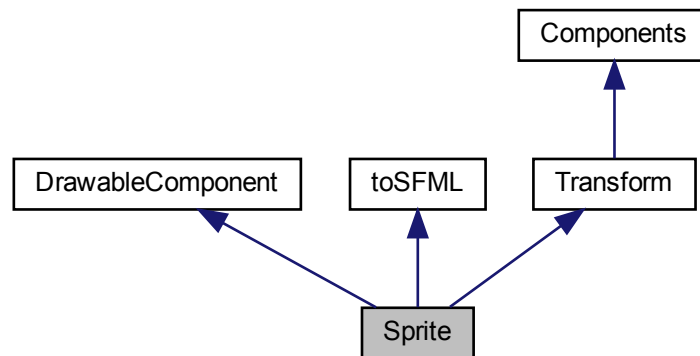
- src/Components/all_components/include/Sound.h
- src/Components/all_components/Sound.cpp

4.18 Sprite Class Reference

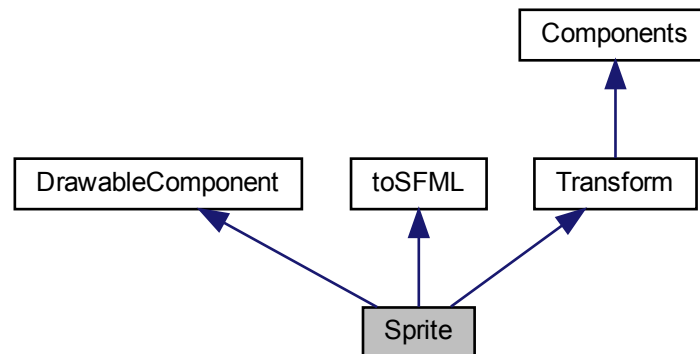
[Sprite](#) class: [Sprite](#) is a class that represents the rendering properties of a Component.

```
#include <Sprite.h>
```

Inheritance diagram for Sprite:



Collaboration diagram for Sprite:



Public Member Functions

- `Sprite ()`
< Doing the animation.
- `Sprite (const std::string &texturePath)`
Sprite constructor with an existing texture path.
- `~Sprite ()` override=default
Sprite destructor.
- `Transform * getTransform ()` const
- void `setTransform (Transform &newTransform)`
- bool `initSprite ()` const
init(): Initialize the Sprite.
- int `getBit ()` const
getBit(): Get the bit of the Sprite.
- void `draw (sf::RenderWindow &window)` const override
draw(): Draw the Sprite.
- void `update (sf::Time deltaTime)` override
- void `createSprite (const std::string &texturePath)`
createSprite(): Create the SFML Sprite with a texture path for rendering.
- void `createSprite (const sf::Texture &existingTexture)`
createSprite(): Create the SFML Sprite with an existing texture for rendering.
- void `createSprite ()`
createSprite(): Create the SFML Sprite with the component's texture for rendering.
- `sf::Sprite getSprite ()` const
getSprite(): Get the SFML Sprite for rendering.
- `sf::Texture getTexture ()` const
getTexture(): Get the SFML Texture for the sprite.
- bool `isTextureLoaded ()` const
isTextureLoaded(): Check if the texture is loaded.
- void `setSprite (const sf::Sprite &sprite)`
setSprite(): Set the SFML Sprite with an existing one for rendering.

- void `setSprite` (std::map< std::string, std::shared_ptr< sf::Texture >> mapTexture, std::string nameTexture, bool animate=false, std::vector< `Rect`< int >> newFrames=std::vector< `Rect`< int >>(), int durationOfFrame=100)
setSprite(): Set the SFML `Sprite` with a map of string and textures, a texture name and a map of string and vector of floats.
- void `setTransformSprite` (`Vector2`< float > newPosition, float newRotation, `Vector2`< float > newScale)
setTransformSprite(): Set the sprite transform with new value and set the value on the `Transform` component.
- void `setTransformSprite` ()
setTransformSprite(): Set the transform of the sprite based on the `Transform` component value.
- void `setPosition` (`Vector2`< float > newPosition)
setPosition(): Set the position of the sprite with new value.
- void `setPosition` ()
setPosition(): Set the position of the sprite based on the `Transform` component value.
- void `setRotation` (float newRotation)
setRotation(): Set the rotation of the sprite with new value.
- void `setRotation` ()
setRotation(): Set the rotation of the sprite based on the `Transform` component value.
- void `setScale` (`Vector2`< float > newScale)
setScale(): Set the the scale of the sprite with new value.
- void `setScale` ()
setScale(): Set the scale of the sprite based on the `Transform` component value.
- void `setDeferredSprite` (std::function< void()> setter)
setDeferredSprite(): Set the deferred sprite.
- void `applyDeferredSprite` ()
applyDeferredSprite(): Apply the deferred sprite.
- void `setTexture` (const sf::Texture &existingTexture)
setTexture(): Set the texture with an existing one for the sprite.
- `Rect`< float > `getBounds` () const

4.18.1 Detailed Description

`Sprite` class: `Sprite` is a class that represents the rendering properties of a Component.

The `Sprite` class manages the graphical representation of a Component using SFML.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 `Sprite()` [1/2]

```
Sprite::Sprite ( ) [inline]
```

< Doing the animation.

Default `Sprite` constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.2.2 Sprite() [2/2]

```
Sprite::Sprite (  
    const std::string & texturePath ) [inline]
```

[Sprite](#) constructor with an existing texture path.

Parameters

| | |
|--------------------|--|
| <i>texturePath</i> | Path to the texture file for the sprite. |
|--------------------|--|

Returns

void

4.18.2.3 ~Sprite()

```
Sprite::~Sprite ( ) [override], [default]
```

[Sprite](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3 Member Function Documentation

4.18.3.1 applyDeferredSprite()

```
void Sprite::applyDeferredSprite ( )
```

[applyDeferredSprite\(\)](#): Apply the deferred sprite.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3.2 createSprite() [1/3]

```
void Sprite::createSprite ( )
```

[createSprite\(\)](#): Create the SFML [Sprite](#) with the component's texture for rendering.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3.3 createSprite() [2/3]

```
void Sprite::createSprite (
    const sf::Texture & existingTexture )
```

[createSprite\(\)](#): Create the SFML [Sprite](#) with an existing texture for rendering.

Parameters

| | |
|------------------------|-----------------------------|
| <i>existingTexture</i> | SFML Texture for the sprite |
|------------------------|-----------------------------|

Returns

void

4.18.3.4 createSprite() [3/3]

```
void Sprite::createSprite (
    const std::string & texturePath )
```

[createSprite\(\)](#): Create the SFML [Sprite](#) with a texture path for rendering.

Parameters

| | |
|--------------------|--|
| <i>texturePath</i> | Path to the texture file for the sprite. |
|--------------------|--|

Returns

void

4.18.3.5 draw()

```
void Sprite::draw (
    sf::RenderWindow & window ) const [override], [virtual]
```

[draw\(\)](#): Draw the [Sprite](#).

Parameters

| | |
|---------------|---|
| <i>window</i> | SFML RenderWindow where the Sprite will be drawn. |
|---------------|---|

Returns

void

Implements [DrawableComponent](#).

4.18.3.6 getBit()

```
int Sprite::getBit ( ) const [inline]
```

[getBit\(\)](#): Get the bit of the [Sprite](#).

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

int: The bit of the [Sprite](#).

4.18.3.7 getSprite()

```
sf::Sprite Sprite::getSprite ( ) const
```

[getSprite\(\)](#): Get the SFML [Sprite](#) for rendering.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

sf::Sprite: SFML [Sprite](#) for rendering

4.18.3.8 getTexture()

```
sf::Texture Sprite::getTexture ( ) const
```

[getTexture\(\)](#): Get the SFML Texture for the sprite.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

sf::Texture: SFML Texture for the sprite

4.18.3.9 initSprite()

```
bool Sprite::initSprite ( ) const [inline]
```

[init\(\)](#): Initialize the [Sprite](#).

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: True if the [Sprite](#) is initialized, false otherwise.

4.18.3.10 isTextureLoaded()

```
bool Sprite::isTextureLoaded ( ) const [inline]
```

[isTextureLoaded\(\)](#): Check if the texture is loaded.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: True if the texture is loaded, false otherwise.

4.18.3.11 setDeferredSprite()

```
void Sprite::setDeferredSprite (
    std::function< void()> setter )
```

[setDeferredSprite\(\)](#): Set the deferred sprite.

Parameters

| | |
|---------------|------------------------------------|
| <i>setter</i> | Function that will set the sprite. |
|---------------|------------------------------------|

Returns

void

4.18.3.12 setPosition() [1/2]

```
void Sprite::setPosition ( )
```

[setPosition\(\)](#): Set the position of the sprite based on the [Transform](#) component value.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3.13 setPosition() [2/2]

```
void Sprite::setPosition (
    Vector2< float > newPosition )
```

[setPosition\(\)](#): Set the position of the sprite with new value.

Parameters

| | |
|--------------------|--|
| <i>newPosition</i> | The new Vector2<float> position. |
|--------------------|--|

Returns

void

4.18.3.14 setRotation() [1/2]

```
void Sprite::setRotation ( )
```

[setRotation\(\)](#): Set the rotation of the sprite based on the [Transform](#) component value.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3.15 setRotation() [2/2]

```
void Sprite::setRotation (
    float newRotation )
```

[setRotation\(\)](#): Set the rotation of the sprite with new value.

Parameters

| | |
|--------------------|-------------------------|
| <i>newRotation</i> | The new float rotation. |
|--------------------|-------------------------|

Returns

void

4.18.3.16 setScale() [1/2]

```
void Sprite::setScale ( )
```

[setScale\(\)](#): Set the scale of the sprite based on the [Transform](#) component value.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3.17 setScale() [2/2]

```
void Sprite::setScale (
    Vector2< float > newScale )
```

[setScale\(\)](#): Set the the scale of the sprite with new value.

Parameters

| | |
|-----------------|--|
| <i>newScale</i> | The new Vector2 <float> scale. |
|-----------------|--|

Returns

void

4.18.3.18 setSprite() [1/2]

```
void Sprite::setSprite (
    const sf::Sprite & sprite )
```

[setSprite\(\)](#): Set the SFML [Sprite](#) with an existing one for rendering.

Parameters

| | |
|---------------|---|
| <i>sprite</i> | SFML Sprite for rendering |
|---------------|---|

Returns

void

4.18.3.19 setSprite() [2/2]

```
void Sprite::setSprite (
    std::map< std::string, std::shared_ptr< sf::Texture >> mapTexture,
    std::string nameTexture,
    bool animate = false,
    std::vector< Rect< int >> newFrames = std::vector<Rect<int>>(),
    int durationOfFrame = 100 )
```

[setSprite\(\)](#): Set the SFML [Sprite](#) with a map of string and textures, a texture name and a map of string and vector of floats.

Parameters

| | |
|---------------------|-------------------------------------|
| <i>mapTexture</i> | Map of string and textures. |
| <i>nameTexture</i> | Name of the texture. |
| <i>mapTransform</i> | Map of string and vector of floats. |

Returns

void

4.18.3.20 setTexture()

```
void Sprite::setTexture (
    const sf::Texture & existingTexture )
```

[setTexture\(\)](#): Set the texture with an existing one for the sprite.

Parameters

| | |
|------------------------|-----------------------------|
| <i>existingTexture</i> | SFML Texture for the sprite |
|------------------------|-----------------------------|

Returns

void

4.18.3.21 setTransformSprite() [1/2]

```
void Sprite::setTransformSprite ( )
```

[setTransformSprite\(\)](#): Set the transform of the sprite based on the [Transform](#) component value.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.18.3.22 setTransformSprite() [2/2]

```
void Sprite::setTransformSprite (
    Vector2< float > newPosition,
    float newRotation,
    Vector2< float > newScale )
```

[setTransformSprite\(\)](#): Set the sprite transform with new value and set the value on the [Transform](#) component.

Parameters

| | |
|--------------------|---|
| <i>newPosition</i> | The new Vector2 <float> position. |
| <i>newRotation</i> | The new float rotation. |
| <i>newScale</i> | The new Vector2 <float> scale. |

Returns

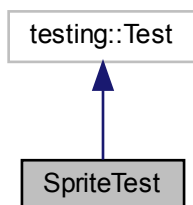
void

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

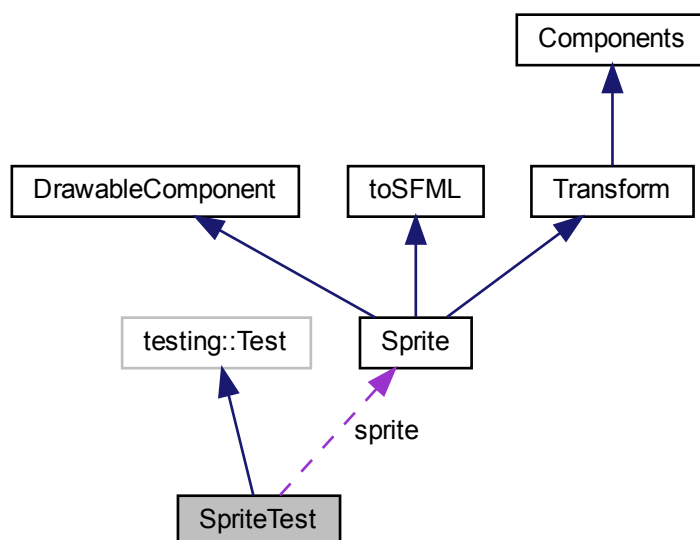
- src/Components/all_components/include/Sprite.h
- src/Components/all_components/Sprite.cpp

4.19 SpriteTest Class Reference

Inheritance diagram for SpriteTest:



Collaboration diagram for SpriteTest:



Protected Attributes

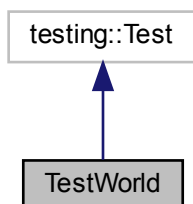
- [Sprite](#) `sprite`

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

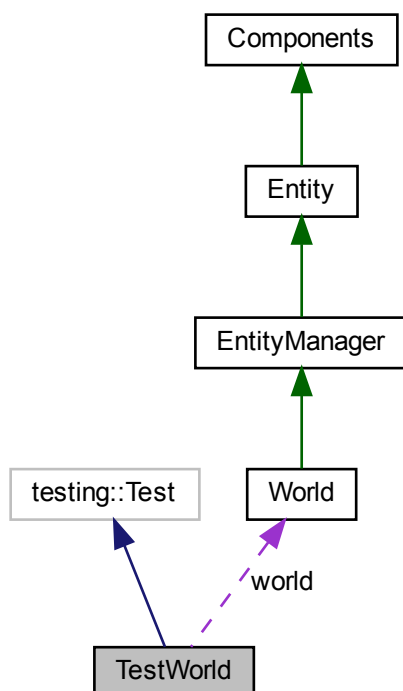
- `tests/Components/all_components/TestSprite.cpp`

4.20 TestWorld Class Reference

Inheritance diagram for TestWorld:



Collaboration diagram for TestWorld:



Protected Attributes

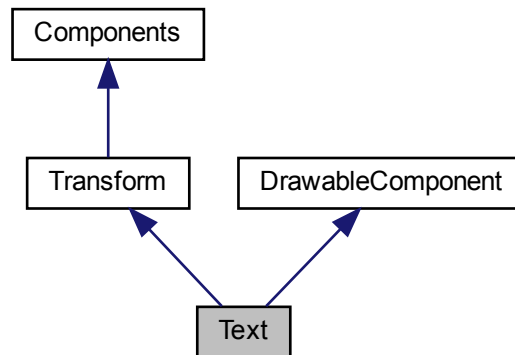
- [World](#) world

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

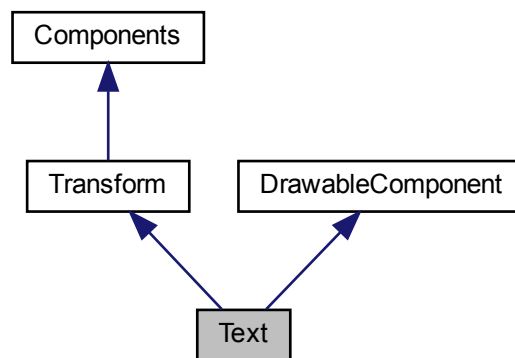
- tests/World/TestWorld.cpp

4.21 Text Class Reference

Inheritance diagram for Text:



Collaboration diagram for Text:



Public Member Functions

- int **getBit** () const
- void **draw** (sf::RenderWindow &window) const override
draw(): Draw the component
- void **update** (sf::Time deltaTime) override
- void **setText** (std::map< std::string, std::shared_ptr< sf::Font >> mapFont, std::string nameFont, std::string newStringText, int sizeText, Color color)

- void **setText** (std::map< std::string, std::shared_ptr< sf::Font >> mapFont, std::string nameFont, std::string newStringText, int sizeText, [Color](#) newColorFill, [Color](#) newColorOutline)
- void **setFont** (std::map< std::string, std::shared_ptr< sf::Font >> mapFont, std::string nameFont)
- void **setString** (std::string nameText)
- void **setSize** (int sizeText)
- void **setOutlineColor** ([Color](#) color)
- void **setFillColor** ([Color](#) color)
- void **setPosition** ([Vector2](#)< float > position)
- void **setRotation** (float rotation)
- void **setScale** ([Vector2](#)< float > scale)
- sf::Text **getText** () const
- sf::Font **getFont** () const
- std::string **getStringText** () const
- int **getSize** () const
- [Color](#) **getColorFill** () const
- [Color](#) **getColorOutline** () const
- [Transform](#) * **getTransform** () const
- void **setTransform** ([Transform](#) &newTransform)
- void **setDeferredText** (std::function< void()> setter)
- void **applyDeferredText** ()

4.21.1 Member Function Documentation

4.21.1.1 draw()

```
void Text::draw (
    sf::RenderWindow & window ) const [override], [virtual]
```

[draw\(\)](#): Draw the component

Parameters

| | |
|---------------|---------------------------------|
| <i>window</i> | Window to draw the component on |
|---------------|---------------------------------|

Returns

void

Implements [DrawableComponent](#).

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

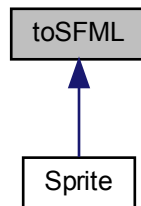
- src/Components/all_components/include/Text.h
- src/Components/all_components/Text.cpp

4.22 toSFML Class Reference

`toSFML` class: `toSFML` is a class that convert some class into SFML class.

```
#include <toSFML.h>
```

Inheritance diagram for `toSFML`:



Public Member Functions

- `toSFML()`=default
Default `toSFML` constructor.
- `~toSFML()`=default
`toSFML` destructor.
- `template<typename T>`
`sf::Rect< T > toSFMLRect (Rect< T > rect)`
`toSFMLRect()`: Convert your `Rect<T>` into `sf::Rect<T>`.

4.22.1 Detailed Description

`toSFML` class: `toSFML` is a class that convert some class into SFML class.

Convert some class in SFML class.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 toSFML()

```
toSFML::toSFML ( ) [default]
```

Default `toSFML` constructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.22.2.2 ~toSFML()

```
toSFML::~~toSFML ( ) [default]
```

[toSFML](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.22.3 Member Function Documentation**4.22.3.1 toSFMLRect()**

```
template<typename T >
template sf::Rect< float > toSFML::toSFMLRect (
    Rect< T > rect )
```

[toSFMLRect\(\)](#): Convert your Rect<T> into sf::Rect<T>.

Template Parameters

| | |
|----------|-------------------|
| <i>T</i> | Type of the rect. |
|----------|-------------------|

Parameters

| | |
|-------------|-------------------------------|
| <i>rect</i> | The rect you want to convert. |
|-------------|-------------------------------|

Returns

sf:Rect<T>: SFML rect.

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

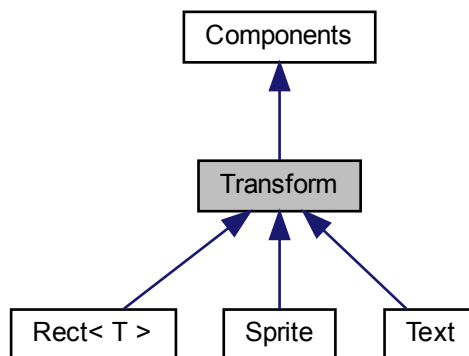
- src/toSFML/include/toSFML.h
- src/toSFML/toSFML.cpp

4.23 Transform Class Reference

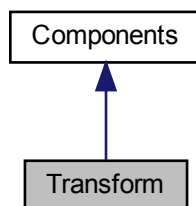
[Transform](#) class: [Transform](#) is a class that represents the transform of a Component.

```
#include <Transform.h>
```

Inheritance diagram for Transform:



Collaboration diagram for Transform:



Public Member Functions

- [Transform](#) ()
Default [Transform](#) constructor.
- bool [init](#) () const
[init\(\)](#): Initialize the component
- [~Transform](#) () override=default
[Transform](#) destructor.
- void **update** (sf::Time deltaTime) override
- int [getBit](#) () const
[getBit\(\)](#): Get the bitmask of the component
- [Vector2](#)< float > [getPosition](#) () const
[getPositionVector\(\)](#): Get the position vector of the component;
- float [getRotation](#) () const
[getRotationVector\(\)](#): Get the rotation vector of the component;
- [Vector2](#)< float > [getScale](#) () const
[getScaleVector\(\)](#): Get the scale vector of the component;
- TransformStruct [getTransformStruct](#) () const
[getTransformStruct\(\)](#): Get the the transform of the component;
- void [setTransform](#) ([Vector2](#)< float > newPosition, float newRotation, [Vector2](#)< float > newScale)
[setTransformStruct\(\)](#): Set the transform of the component;
- void [setTransformPosition](#) ([Vector2](#)< float > newPosition)
[setTransformPosition\(\)](#): Set the transform position of the component;
- void [setTransformRotation](#) (float newRotation)
[setTransformRotation\(\)](#): Set the transform rotation of the component;
- void [setTransformScale](#) ([Vector2](#)< float > newScale)
[setTransformScale\(\)](#): Set the transform scale of the component;

4.23.1 Detailed Description

[Transform](#) class: [Transform](#) is a class that represents the transform of a Component.

The [Transform](#) class manages the position, rotation and scale of a Component.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Transform()

```
Transform::Transform ( ) [inline]
```

Default [Transform](#) constructor.

Parameters

| | |
|-------------------|--|
| <code>void</code> | |
|-------------------|--|

Returns

void

4.23.2.2 ~Transform()

```
Transform::~Transform ( ) [override], [default]
```

[Transform](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.23.3 Member Function Documentation**4.23.3.1 getBit()**

```
int Transform::getBit ( ) const
```

[getBit\(\)](#): Get the bitmask of the component

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

int: bitmask of the component

4.23.3.2 getPosition()

```
Vector2<float> Transform::getPosition ( ) const [inline]
```

[getPositionVector\(\)](#): Get the position vector of the component;

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::vector<float>: position vector of the component

4.23.3.3 getRotation()

```
float Transform::getRotation ( ) const [inline]
```

getRotationVector(): Get the rotation vector of the component;

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::vector<float>: rotation vector of the component

4.23.3.4 getScale()

```
Vector2<float> Transform::getScale ( ) const [inline]
```

getScaleVector(): Get the scale vector of the component;

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::vector<float>: scale vector of the component

4.23.3.5 getTransformStruct()

```
TransformStruct Transform::getTransformStruct ( ) const [inline]
```

[getTransformStruct\(\)](#): Get the the transform of the component;

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

TransformStruct: struct of the [Transform](#).

4.23.3.6 init()

```
bool Transform::init ( ) const [inline]
```

[init\(\)](#): Initialize the component

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: true if the component is initialized, false otherwise

4.23.3.7 setTransform()

```
void Transform::setTransform (
    Vector2< float > newPosition,
    float newRotation,
    Vector2< float > newScale )
```

setTransformStruct(): Set the transform of the component;

Parameters

| | |
|--------------------|---|
| <i>newPosition</i> | : the new Vector2 <float> position. |
| <i>newRotation</i> | : the new float rotation. |
| <i>newScale</i> | : the new Vector2 <float> scale. |

Returns

void

4.23.3.8 setTransformPosition()

```
void Transform::setTransformPosition (
    Vector2< float > newPosition )
```

[setTransformPosition\(\)](#): Set the transform position of the component;

Parameters

| | |
|--------------------|--|
| <i>newPosition</i> | : the new Vector2<float> position. |
|--------------------|--|

Returns

void

4.23.3.9 setTransformRotation()

```
void Transform::setTransformRotation (
    float newRotation )
```

[setTransformRotation\(\)](#): Set the transform rotation of the component;

Parameters

| | |
|--------------------|---------------------------|
| <i>newRotation</i> | : the new float rotation. |
|--------------------|---------------------------|

Returns

void

4.23.3.10 setTransformScale()

```
void Transform::setTransformScale (
    Vector2< float > newScale )
```

[setTransformScale\(\)](#): Set the transform scale of the component;

Parameters

| | |
|-----------------|---|
| <i>newScale</i> | : the new Vector2<float> scale. |
|-----------------|---|

Returns

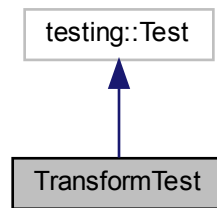
void

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

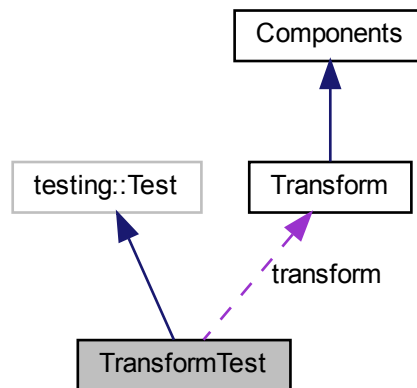
- `src/Components/all_components/include/Transform.h`
- `src/Components/all_components/Transform.cpp`

4.24 TransformTest Class Reference

Inheritance diagram for TransformTest:



Collaboration diagram for TransformTest:



Protected Attributes

- `Transform transform`

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

- `tests/Components/all_components/TestTransform.cpp`

4.25 Vector2< T > Class Template Reference

Vector class: Vector is a class that represents a vector in 2 dimensions.

```
#include <Vector2.h>
```

Public Member Functions

- [Vector2](#) (T x, T y)
< Variable for using the value of the Vector2Struct.
- [~Vector2](#) ()=default
Vector2 destructor.
- Vector2Struct [getVector2Struct](#) () const
getVector2Struct(): Get the using Vector2Struct.
- T [getX](#) () const
getX(): Get x of Vector2Struct.
- T [getY](#) () const
getY(): Get y of Vector2Struct.

4.25.1 Detailed Description

```
template<typename T>
class Vector2< T >
```

Vector class: Vector is a class that represents a vector in 2 dimensions.

This create a vector with 2 value.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Vector2()

```
template<typename T >
Vector2< T >::Vector2 (
    T x,
    T y ) [inline]
```

< Variable for using the value of the Vector2Struct.

[Vector2](#) constructor with parameters.

Template Parameters

| | |
|----------|---------------------|
| <i>T</i> | Type of the vector. |
|----------|---------------------|

Parameters

| | |
|----------|-------------|
| <i>x</i> | Position x. |
| <i>y</i> | Position y. |

Returns

void

4.25.2.2 ~Vector2()

```
template<typename T >  
Vector2< T >::~~Vector2 ( ) [default]
```

[Vector2](#) destructor.

Template Parameters

| | |
|----------|---------------------|
| <i>T</i> | Type of the vector. |
|----------|---------------------|

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.25.3 Member Function Documentation**4.25.3.1 getVector2Struct()**

```
template<typename T >  
Vector2Struct Vector2< T >::getVector2Struct ( ) const [inline]
```

[getVector2Struct\(\)](#): Get the using Vector2Struct.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

Vector2Struct

4.25.3.2 getX()

```
template<typename T >
T Vector2< T >::getX ( ) const [inline]
```

[getX\(\)](#): Get x of Vector2Struct.

Template Parameters

| | |
|--|--|
| | |
|--|--|

4.25.3.3 getY()

```
template<typename T >
T Vector2< T >::getY ( ) const [inline]
```

[getY\(\)](#): Get y of Vector2Struct.

Template Parameters

| | |
|--|--|
| | |
|--|--|

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

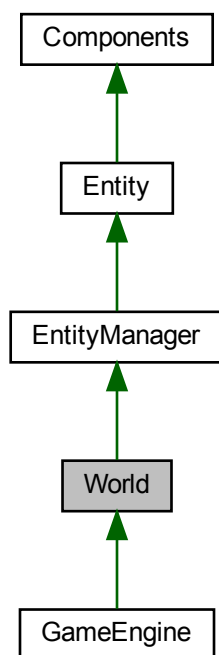
- src/Other/include/Vector2.h

4.26 World Class Reference

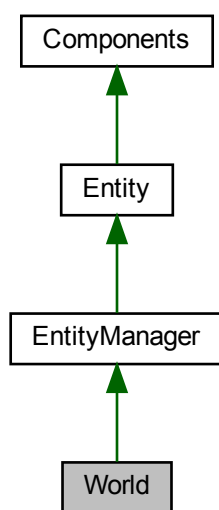
[World](#) class: [World](#) is a class that represents the world of the game.

```
#include <world.h>
```

Inheritance diagram for World:



Collaboration diagram for World:



Public Member Functions

- `World ()`=default
Default `World` constructor.
- `~World ()` override=default
`World` destructor.
- void `createEntities` (std::map< std::string, std::pair< std::unique_ptr< `EntityManager` >, std::vector< std::string >>> &mapEntityManager)
`createEntities()`: Create the entities.
- `EntityManager` & `addEntityManager` (std::string NameEntityManager)
`addEntityManager()`: Add an entity manager to the map.
- `EntityManager` & `getEntityManager` (std::string NameEntityManager)
`getEntityManager()`: Get the entity manager.
- void `setNameWorld` (std::string newName)
`setNameWorld()`: Set the name of the world.
- std::string `getNameWorld` () const
`getNameWorld()`: Get the name of the world.
- std::map< std::string, `EntityManager` * > `getEntityManagerMap` () const
`getEntityManagerMap()`: Get the map of the entity manager.
- bool `initWorld` ()
`init()`: Initialize the `World`.

Additional Inherited Members

4.26.1 Detailed Description

`World` class: `World` is a class that represents the world of the game.

The `World` class manages the world of the game.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 `World()`

```
World::World ( ) [default]
```

Default `World` constructor.

Parameters

| | |
|-------------------|--|
| <code>void</code> | |
|-------------------|--|

Returns

`void`

4.26.2.2 ~World()

```
World::~~World ( ) [override], [default]
```

[World](#) destructor.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

void

4.26.3 Member Function Documentation

4.26.3.1 addEntityManager()

```
EntityManager & World::addEntityManager (
    std::string NameEntityManager )
```

[addEntityManager\(\)](#): Add an entity manager to the map.

Parameters

| | |
|--------------------------|-----------------------------|
| <i>NameEntityManager</i> | Name of the entity manager. |
|--------------------------|-----------------------------|

Returns

[EntityManager&](#): The entity manager.

4.26.3.2 createEntities()

```
void World::createEntities (
    std::map< std::string, std::pair< std::unique_ptr< EntityManager >, std::vector<
std::string >>> & mapEntityManager )
```

[createEntities\(\)](#): Create the entities.

Parameters

| | |
|-------------------------|--|
| <i>mapEntityManager</i> | Map of the entities manager's unique pointers. |
| <i>keyEntityManager</i> | Key of the entities manager. |

Returns

void

4.26.3.3 getEntityManager()

```
EntityManager & World::getEntityManager (
    std::string NameEntityManager )
```

[getEntityManager\(\)](#): Get the entity manager.

Parameters

| | |
|--------------------------|-----------------------------|
| <i>NameEntityManager</i> | Name of the entity manager. |
|--------------------------|-----------------------------|

Returns

[EntityManager&](#): The entity manager.

4.26.3.4 getEntityManagerMap()

```
std::map<std::string, EntityManager*> World::getEntityManagerMap ( ) const [inline]
```

[getEntityManagerMap\(\)](#): Get the map of the entity manager.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

`std::map<std::string, EntityManager*>`: The map of the entity manager.

4.26.3.5 getNameWorld()

```
std::string World::getNameWorld ( ) const [inline]
```

[getNameWorld\(\)](#): Get the name of the world.

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

std::string: The name of the world.

4.26.3.6 initWorld()

```
bool World::initWorld ( ) [inline]
```

[init\(\)](#): Initialize the [World](#).

Parameters

| | |
|-------------|--|
| <i>void</i> | |
|-------------|--|

Returns

bool: True if the world is initialized, false otherwise.

4.26.3.7 setNameWorld()

```
void World::setNameWorld (
    std::string newName )
```

[setNameWorld\(\)](#): Set the name of the world.

Parameters

| | |
|----------------|------------------------|
| <i>newName</i> | New name of the world. |
|----------------|------------------------|

Returns

void

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

- src/World/include/world.h
- src/World/world.cpp

Index

- ~Components
 - Components, [10](#)
- ~DrawableComponent
 - DrawableComponent, [12](#)
- ~Entity
 - Entity, [15](#)
- ~EntityManager
 - EntityManager, [22](#)
- ~EventEngine
 - EventEngine, [28](#)
- ~GameEngine
 - GameEngine, [36](#)
- ~Rect
 - Rect< T >, [51](#)
- ~Sound
 - Sound, [56](#)
- ~Sprite
 - Sprite, [65](#)
- ~Transform
 - Transform, [82](#)
- ~Vector2
 - Vector2< T >, [88](#)
- ~World
 - World, [92](#)
- ~toSFML
 - toSFML, [79](#)
- addComponent
 - Entity, [16](#)
- addDrawable
 - Entity, [16](#)
- addEntity
 - EntityManager, [23](#)
- addEntityManager
 - World, [92](#)
- addKeyPressed
 - EventEngine, [28](#)
- addMouseButtonPressed
 - EventEngine, [29](#)
- addMouseMoved
 - EventEngine, [29](#)
- addWorld
 - GameEngine, [37](#)
- applyDeferredSprite
 - Sprite, [65](#)
- Archetypes, [7](#)
- AudioTest, [7](#)
- Color, [8](#)
- Components, [9](#)
- ~Components, [10](#)
- Components, [10](#)
- init, [11](#)
- contains
 - Rect< T >, [51](#)
- createEntities
 - World, [92](#)
- createSprite
 - Sprite, [66](#)
- draw
 - DrawableComponent, [12](#)
 - Sprite, [67](#)
 - Text, [77](#)
- DrawableComponent, [11](#)
 - ~DrawableComponent, [12](#)
 - draw, [12](#)
- drawEntity
 - Entity, [17](#)
- Entity, [13](#)
 - ~Entity, [15](#)
 - addComponent, [16](#)
 - addDrawable, [16](#)
 - drawEntity, [17](#)
 - Entity, [15](#)
 - getComponent, [17](#)
 - getComponentArrays, [17](#)
 - getComponentBitset, [18](#)
 - getComponentTypeID, [18](#)
 - getDrawableComponents, [19](#)
 - getName, [19](#)
 - initEntity, [19](#)
 - setName, [20](#)
- EntityManager, [21](#)
 - ~EntityManager, [22](#)
 - addEntity, [23](#)
 - EntityManager, [22](#)
 - getEntities, [23](#)
 - getEntity, [23](#)
 - getEntityMap, [24](#)
 - initEntityManager, [24](#)
- EntityManagerTest, [25](#)
- EntityTest, [26](#)
- EventEngine, [27](#)
 - ~EventEngine, [28](#)
 - addKeyPressed, [28](#)
 - addMouseButtonPressed, [29](#)
 - addMouseMoved, [29](#)
 - EventEngine, [28](#)

- getEvent, 29
 - getKeyPressedMap, 30
 - getMouseButtonPressedMap, 30
 - getMouseMovedMap, 30
 - init, 31
- eventGameEngine
 - GameEngine, 37
- EventTest, 31
- GameEngine, 32
 - ~GameEngine, 36
 - addWorld, 37
 - eventGameEngine, 37
 - GameEngine, 36
 - getCurrentWorld, 38
 - getEventEngine, 38
 - getFilesRessources, 38
 - getMapMusic, 39
 - getMapTexture, 39
 - getWindow, 39
 - getWorld, 40
 - getWorldMap, 40
 - initialize, 40
 - initializeAllFiles, 41
 - initializeMusic, 41
 - initializeMusicFunction, 41
 - initializeSound, 42
 - initializeSoundFunction, 42
 - initializeSpriteFunction, 42
 - initializeTexture, 43
 - initializeWorldMap, 43
 - isWindowOpen, 43
 - renderGameEngine, 44
 - run, 44
 - setCurrentWorld, 45
 - setWindow, 45
 - updateGameEngine, 46
- GameEngineTest, 46
- getBit
 - Sprite, 67
 - Transform, 82
- getComponent
 - Entity, 17
- getComponentArrays
 - Entity, 17
- getComponentBitset
 - Entity, 18
- getComponentTypeID
 - Entity, 18
- getCurrentWorld
 - GameEngine, 38
- getDrawableComponents
 - Entity, 19
- getEntities
 - EntityManager, 23
- getEntity
 - EntityManager, 23
- getEntityManager
 - World, 93
- getEntityManagerMap
 - World, 93
- getEntityMap
 - EntityManager, 24
- getEvent
 - EventEngine, 29
- getEventEngine
 - GameEngine, 38
- getFilesRessources
 - GameEngine, 38
- getHeight
 - Rect< T >, 52
- getKeyPressedMap
 - EventEngine, 30
- getLeft
 - Rect< T >, 52
- getMapMusic
 - GameEngine, 39
- getMapTexture
 - GameEngine, 39
- getMouseButtonPressedMap
 - EventEngine, 30
- getMouseMovedMap
 - EventEngine, 30
- getName
 - Entity, 19
- getNameWorld
 - World, 93
- getPosition
 - Transform, 82
- getRect
 - Rect< T >, 53
- getRotation
 - Transform, 83
- getScale
 - Transform, 83
- getSound
 - Sound, 57
- getSoundBuffer
 - Sound, 57
- getSprite
 - Sprite, 68
- getTexture
 - Sprite, 68
- getTop
 - Rect< T >, 53
- getTransformStruct
 - Transform, 83
- getVector2Struct
 - Vector2< T >, 88
- getVolume
 - Sound, 57
- getWidth
 - Rect< T >, 53
- getWindow
 - GameEngine, 39
- getWorld
 - GameEngine, 40

- getWorldMap
 - GameEngine, 40
- getX
 - Vector2< T >, 89
- getY
 - Vector2< T >, 89
- init
 - Components, 11
 - EventEngine, 31
 - Transform, 84
- initEntity
 - Entity, 19
- initEntityManager
 - EntityManager, 24
- initialize
 - GameEngine, 40
- initializeAllFiles
 - GameEngine, 41
- initializeMusic
 - GameEngine, 41
- initializeMusicFunction
 - GameEngine, 41
- initializeSound
 - GameEngine, 42
- initializeSoundFunction
 - GameEngine, 42
- initializeSpriteFunction
 - GameEngine, 42
- initializeTexture
 - GameEngine, 43
- initializeWorldMap
 - GameEngine, 43
- initSprite
 - Sprite, 68
- initWorld
 - World, 94
- isPlaying
 - Sound, 59
- isTextureLoaded
 - Sprite, 69
- isWindowOpen
 - GameEngine, 43
- loadSoundBuffer
 - Sound, 59
- Music, 48
- pause
 - Sound, 59
- play
 - Sound, 60
- Rect
 - Rect< T >, 50
- Rect< T >, 49
 - ~Rect, 51
 - contains, 51
- getHeight, 52
- getLeft, 52
- getRect, 53
- getTop, 53
- getWidth, 53
- Rect, 50
- renderGameEngine
 - GameEngine, 44
- run
 - GameEngine, 44
- Script, 54
- setCurrentWorld
 - GameEngine, 45
- setDeferredSprite
 - Sprite, 69
- setLoop
 - Sound, 60
- setName
 - Entity, 20
- setNameWorld
 - World, 94
- setPosition
 - Sprite, 69, 70
- setRotation
 - Sprite, 70
- setScale
 - Sprite, 71
- setSound
 - Sound, 60
- setSoundBuffer
 - Sound, 61
- setSprite
 - Sprite, 71, 72
- setTexture
 - Sprite, 72
- setTransform
 - Transform, 84
- setTransformPosition
 - Transform, 84
- setTransformRotation
 - Transform, 85
- setTransformScale
 - Transform, 85
- setTransformSprite
 - Sprite, 73
- setVolume
 - Sound, 61
- setWindow
 - GameEngine, 45
- Sound, 54
 - ~Sound, 56
 - getSound, 57
 - getSoundBuffer, 57
 - getVolume, 57
 - isPlaying, 59
 - loadSoundBuffer, 59
 - pause, 59
 - play, 60

- setLoop, 60
 - setSound, 60
 - setSoundBuffer, 61
 - setVolume, 61
 - Sound, 56
 - stop, 61
- Sprite, 62
 - ~Sprite, 65
 - applyDeferredSprite, 65
 - createSprite, 66
 - draw, 67
 - getBit, 67
 - getSprite, 68
 - getTexture, 68
 - initSprite, 68
 - isTextureLoaded, 69
 - setDeferredSprite, 69
 - setPosition, 69, 70
 - setRotation, 70
 - setScale, 71
 - setSprite, 71, 72
 - setTexture, 72
 - setTransformSprite, 73
 - Sprite, 64, 65
- SpriteTest, 74
- stop
 - Sound, 61
- TestWorld, 75
- Text, 76
 - draw, 77
- toSFML, 78
 - ~toSFML, 79
 - toSFML, 78
 - toSFMLRect, 79
- toSFMLRect
 - toSFML, 79
- Transform, 80
 - ~Transform, 82
 - getBit, 82
 - getPosition, 82
 - getRotation, 83
 - getScale, 83
 - getTransformStruct, 83
 - init, 84
 - setTransform, 84
 - setTransformPosition, 84
 - setTransformRotation, 85
 - setTransformScale, 85
 - Transform, 81
- TransformTest, 86
- updateGameEngine
 - GameEngine, 46
- Vector2
 - Vector2< T >, 87
- Vector2< T >, 87
 - ~Vector2, 88
- getVector2Struct, 88
 - getX, 89
 - getY, 89
 - Vector2, 87
- World, 89
 - ~World, 92
 - addEntityManager, 92
 - createEntities, 92
 - getEntityManager, 93
 - getEntityManagerMap, 93
 - getNameWorld, 93
 - initWorld, 94
 - setNameWorld, 94
 - World, 91