

## R-Type - Engine

Generated by Doxygen 1.9.1



<b>1 Engine</b>	<b>1</b>
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 Archetypes Class Reference	7
4.2 Audio Class Reference	7
4.3 Components Class Reference	7
4.4 DrawableComponent Class Reference	8
4.5 Entity Class Reference	8
4.5.1 Detailed Description	9
4.5.2 Constructor & Destructor Documentation	9
4.5.2.1 Entity() [1/2]	9
4.5.2.2 Entity() [2/2]	9
4.5.2.3 ~Entity()	10
4.5.3 Member Function Documentation	10
4.5.3.1 addComponent()	10
4.5.3.2 getComponent()	11
4.5.3.3 getName()	11
4.5.3.4 init()	11
4.5.3.5 setName()	12
4.6 EntityManager Class Reference	12
4.6.1 Constructor & Destructor Documentation	13
4.6.1.1 EntityManager()	13
4.6.1.2 ~EntityManager()	13
4.6.2 Member Function Documentation	14
4.6.2.1 addComponent()	14
4.6.2.2 addEntity()	14
4.6.2.3 getComponent()	15
4.6.2.4 getEntities()	15
4.6.2.5 getEntity()	15
4.6.2.6 getEntityMap()	16
4.6.2.7 getName()	16
4.6.2.8 init()	17
4.6.2.9 setName()	17
4.7 EntityManagerTest Class Reference	17
4.8 EntityTest Class Reference	18
4.9 EventEngine Class Reference	18
4.10 GameEngine Class Reference	19

4.10.1 Member Function Documentation	20
4.10.1.1 addComponent()	20
4.10.1.2 addEntity()	21
4.10.1.3 getComponent()	21
4.10.1.4 getEntities()	22
4.10.1.5 getEntity()	22
4.10.1.6 getEntityMap()	22
4.10.1.7 getName()	23
4.10.1.8 init()	23
4.10.1.9 setName()	23
4.11 Sprite Class Reference	24
4.12 Transform Class Reference	24
4.12.1 Detailed Description	25
4.12.2 Constructor & Destructor Documentation	25
4.12.2.1 Transform() [1/2]	25
4.12.2.2 Transform() [2/2]	25
4.12.2.3 ~Transform()	26
4.12.3 Member Function Documentation	26
4.12.3.1 getBit()	26
4.12.3.2 getPositionVector()	26
4.12.3.3 getRotationVector()	27
4.12.3.4 getScaleVector()	27
4.12.3.5 setTransform()	27
4.13 TransformTest Class Reference	28
4.14 World Class Reference	28
4.14.1 Member Function Documentation	29
4.14.1.1 addComponent()	29
4.14.1.2 addEntity()	30
4.14.1.3 getComponent()	30
4.14.1.4 getEntities()	31
4.14.1.5 getEntity()	31
4.14.1.6 getEntityMap()	31
4.14.1.7 getName()	32
4.14.1.8 init()	32
4.14.1.9 setName()	32

## Chapter 1

# Engine



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Archetypes . . . . .	7
Audio . . . . .	7
Components . . . . .	7
Entity . . . . .	8
EntityManager . . . . .	12
World . . . . .	28
GameEngine . . . . .	19
Sprite . . . . .	24
Transform . . . . .	24
DrawableComponent . . . . .	8
Sprite . . . . .	24
EventEngine . . . . .	18
GameEngine . . . . .	19
testing::Test	
EntityManagerTest . . . . .	17
EntityTest . . . . .	18
TransformTest . . . . .	28





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">Archetypes</a>	7
<a href="#">Audio</a>	7
<a href="#">Components</a>	7
<a href="#">DrawableComponent</a>	8
<a href="#">Entity</a>	
<a href="#">Entity</a> class: <a href="#">Entity</a> is a class that represents an entity in the game	8
<a href="#">EntityManager</a>	12
<a href="#">EntityManagerTest</a>	17
<a href="#">EntityTest</a>	18
<a href="#">EventEngine</a>	18
<a href="#">GameEngine</a>	19
<a href="#">Sprite</a>	
<a href="#">Sprite</a> class: <a href="#">Sprite</a> is a class that represents the rendering properties of a Component	24
<a href="#">Transform</a>	
<a href="#">Transform</a> class: <a href="#">Transform</a> is a class that represents the transform of a Component	24
<a href="#">TransformTest</a>	28
<a href="#">World</a>	28



## Chapter 4

# Class Documentation

### 4.1 Archetypes Class Reference

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

- `src/Archetype/Archetypes.h`

### 4.2 Audio Class Reference

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

- `src/Components/all_components/Audio.h`

### 4.3 Components Class Reference

Inheritance diagram for Components:

#### Public Member Functions

- virtual bool **init** ()
- virtual void **update** ()
- template<typename T >  
ComponentTypeID **GetComponentTypeID** () noexcept

#### Protected Types

- using **ComponentTypeID** = std::size\_t

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

- `src/Components/Components.h`
- `src/Components/Components.cpp`

## 4.4 DrawableComponent Class Reference

Inheritance diagram for DrawableComponent:

### Public Member Functions

- virtual void **draw** (sf::RenderWindow &window) const =0

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

- src/Components/DrawableComponent.h

## 4.5 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 [init](#) () override  
*[init\(\)](#): Initialize the entity*
- std::string [getName](#) () const  
*[genName\(\)](#): Get the name of the entity*
- void [setName](#) (std::string newName)  
*[setName\(\)](#): Set the name of the entity*
- void **addDrawable** ([Components](#) \*component)
- void **draw** (sf::RenderWindow &window)
- 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*
- std::bitset< 3 > **getComponentBitset** () const
- std::vector< [DrawableComponent](#) \* > **getDrawableComponents** () const
- std::array< [Components](#) \*, 3 > **getComponentArrays** () const

## Protected Types

- using **ComponentTypeID** = std::size\_t

## Protected Member Functions

- virtual void **update** ()
- template<typename T >  
ComponentTypeID **GetComponentTypeID** () noexcept

### 4.5.1 Detailed Description

**Entity** class: **Entity** is a class that represents an entity in the game.

The **Entity** class manages components associated with the entity.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 Entity() [1/2]

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

Default **Entity** constructor.

##### Parameters

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

##### Returns

void

#### 4.5.2.2 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

**4.5.2.3 ~Entity()**

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

[Entity](#) destructor.

**Parameters**

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

**Returns**

void

**4.5.3 Member Function Documentation****4.5.3.1 addComponent()**

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

#### 4.5.3.2 GetComponent()

```
template<typename T >  
T & Entity::GetComponent
```

[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

#### 4.5.3.3 getName()

```
std::string Entity::getName ( ) const [inline]
```

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

##### Parameters

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

##### Returns

std::string: name of the entity

#### 4.5.3.4 init()

```
bool Entity::init ( ) [inline], [override], [virtual]
```

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

##### Parameters

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

**Returns**

bool: true if the entity is initialized, false otherwise

Reimplemented from [Components](#).

Reimplemented in [World](#), and [EntityManager](#).

**4.5.3.5 setName()**

```
void Entity::setName (
    std::string newName ) [inline]
```

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

**4.6 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 [init](#) () override  
*[init\(\)](#): Initialize the entity*



## Protected Types

- using **ComponentTypeID** = std::size\_t

## Protected Member Functions

- std::string **getName** () const  
*getName(): Get the name of the entity*
- void **setName** (std::string newName)  
*setName(): Set the name of the entity*
- void **addDrawable** (**Components** \*component)
- void **draw** (sf::RenderWindow &window)
- 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*
- std::bitset< 3 > **getComponentBitset** () const
- std::vector< **DrawableComponent** \* > **getDrawableComponents** () const
- std::array< **Components** \*, 3 > **getComponentArrays** () const
- virtual void **update** ()
- template<typename T >  
ComponentTypeID **getComponentTypeID** () noexcept

## 4.6.1 Constructor & Destructor Documentation

### 4.6.1.1 EntityManager()

EntityManager::EntityManager ( ) [default]

Default **EntityManager** constructor.

#### Parameters

void	
------	--

#### Returns

void

### 4.6.1.2 ~EntityManager()

EntityManager::~EntityManager ( ) [default]

**EntityManager** destructor.

## Parameters

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

## Returns

void

## 4.6.2 Member Function Documentation

### 4.6.2.1 addComponent()

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

[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

### 4.6.2.2 addEntity()

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

[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.
-------------	--------------------------

## 4.6.2.3 GetComponent()

```
template<typename T >
T & Entity::GetComponent [inherited]
```

[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

## 4.6.2.4 getEntities()

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

[getEntities\(\)](#): Get the [EntityManager](#)'s entities.

## Parameters

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

## Returns

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

## 4.6.2.5 getEntity()

```
Entity & EntityManager::getEntity (
    std::string nameEntity ) [inline]
```

[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.

#### 4.6.2.6 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.

#### 4.6.2.7 getName()

```
std::string Entity::getName ( ) const [inline], [inherited]
```

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

### Parameters

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

### Returns

std::string: name of the entity

#### 4.6.2.8 init()

```
bool EntityManager::init ( ) [inline], [override], [virtual]
```

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

##### Parameters

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

##### Returns

bool: true if the entity is initialized, false otherwise

Reimplemented from [Entity](#).

Reimplemented in [World](#).

#### 4.6.2.9 setName()

```
void Entity::setName (
    std::string newName ) [inline], [inherited]
```

[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/entityManager.h
- src/Entity/entityManager.cpp

## 4.7 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:

- src/tests/Entity/TestEntityManager.cpp

## 4.8 EntityTest Class Reference

Inheritance diagram for EntityTest:

Collaboration diagram for EntityTest:

## Protected Attributes

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

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

- src/tests/Entity/TestEntity.cpp

## 4.9 EventEngine Class Reference

Inheritance diagram for EventEngine:

## Public Member Functions

- bool **init** () const
- sf::Event & **getEvent** ()
- void **addKeyPressed** (sf::Keyboard::Key keyboard, std::function< void()> function)
- std::map< sf::Keyboard::Key, std::function< void()> > & **getKeyPressedMap** ()

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

- src/Event/event.h
- src/Event/event.cpp

## 4.10 GameEngine Class Reference

Inheritance diagram for GameEngine:

Collaboration diagram for GameEngine:

### Public Member Functions

- **GameEngine** (sf::VideoMode mode, std::string type, sf::String title, sf::Uint32 style=sf::Style::Default, const sf::ContextSettings &settings=sf::ContextSettings())
- void **run** (std::map< std::string, std::unique\_ptr< [World](#) >> mapWorld, std::map< std::string, std::string > pathResources, std::string firstScene)
- void **run** ()
- void **renderGameEngine** ()
- void **eventGameEngine** ()
- bool **isWindowOpen** ()
- void **updateGameEngine** ()
- void **initialize** (std::map< std::string, std::unique\_ptr< [World](#) >> mapWorld, std::map< std::string, std::string > pathResources, std::string firstScene)
- void **initializeSprite** ()
- void **initializeTexture** (std::string path)
- void **initializeWorldMap** (std::map< std::string, std::unique\_ptr< [World](#) >> mapWorld)
- const auto & **getWindow** ()
- void **setWindow** ()
- [EventEngine](#) & **getEventEngine** ()
- void **setCurrentWorld** ([World](#) \*world)
- [World](#) \* **getCurrentWorld** ()
- [World](#) & **addWorld** (std::string nameWorld, std::unique\_ptr< [World](#) > world)
- [World](#) & **getWorld** (std::string nameWorld)
- std::map< std::string, sf::Texture > **getMapTexture** () const
- std::map< std::string, [World](#) \* > **getWorldMap** () const

### Protected Types

- using **ComponentTypeID** = std::size\_t

### Protected Member Functions

- void **createEntities** (std::map< std::string, std::pair< std::unique\_ptr< [EntityManager](#) >, std::vector< std::string >>> &mapEntityManager, std::string keyEntityManager)
- [EntityManager](#) & **addEntityManager** (std::string NameEntityManager)
- [EntityManager](#) & **getEntityManager** (std::string NameEntityManager)
- void **setNameWorld** (std::string newName)
- std::string **getNameWorld** () const
- std::map< std::string, [EntityManager](#) \* > **getEntityManagerMap** () const
- bool **init** () override
  - init(): Initialize the entity*
- [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.*
- `std::string getName ()` const  
*getName(): Get the name of the entity*
- `void setName (std::string newName)`  
*setName(): Set the name of the entity*
- `void addDrawable (Components *component)`
- `void draw (sf::RenderWindow &window)`
- `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*
- `std::bitset< 3 > getComponentBitset ()` const
- `std::vector< DrawableComponent * > getDrawableComponents ()` const
- `std::array< Components *, 3 > getComponentArrays ()` const
- `virtual void update ()`
- `template<typename T >`  
`ComponentTypeID getComponentTypeID ()` noexcept
- `bool init ()` const
- `sf::Event & getEvent ()`
- `void addKeyPressed (sf::Keyboard::Key keyboard, std::function< void()> function)`
- `std::map< sf::Keyboard::Key, std::function< void()> > & getKeyPressedMap ()`

## 4.10.1 Member Function Documentation

### 4.10.1.1 addComponent()

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

*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

**4.10.1.2 addEntity()**

```
Entity & EntityManager::addEntity (
    std::string nameEntity,
    Archetypes newArchetype = Archetypes() ) [inline], [inherited]
```

[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.
-------------	--------------------------

**4.10.1.3 getComponent()**

```
template<typename T >
T & Entity::getComponent [inherited]
```

[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

#### 4.10.1.4 getEntities()

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

[getEntities\(\)](#): Get the [EntityManager](#)'s entities.

##### Parameters

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

##### Returns

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

#### 4.10.1.5 getEntity()

```
Entity & EntityManager::getEntity (
    std::string nameEntity ) [inline], [inherited]
```

[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.

#### 4.10.1.6 getEntityMap()

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

[getEntityMap\(\)](#): Get the [EntityManager](#)'s entity map.

##### Parameters

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

**Returns**

Entity::EntityMap: [Entity](#) map.

**4.10.1.7 getName()**

```
std::string Entity::getName ( ) const [inline], [inherited]
```

getName(): Get the name of the entity

**Parameters**

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

**Returns**

std::string: name of the entity

**4.10.1.8 init()**

```
bool World::init ( ) [inline], [override], [virtual], [inherited]
```

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

**Parameters**

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

**Returns**

bool: true if the entity is initialized, false otherwise

Reimplemented from [EntityManager](#).

**4.10.1.9 setName()**

```
void Entity::setName (
    std::string newName ) [inline], [inherited]
```

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

## 4.11 Sprite Class Reference

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

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

Inheritance diagram for [Sprite](#):

## 4.12 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  
*Default [Transform](#) constructor.*
- bool **init** () const
- [Transform](#) (const std::map< std::string, std::vector< float >> &mapTransform)  
*[Transform](#) constructor.*
- [~Transform](#) () override=default  
*[Transform](#) destructor.*
- int **getBit** () const  
*[getBit\(\)](#): Get the bitmask of the component*
- std::vector< float > **getPositionVector** () const  
*[getPositionVector\(\)](#): Get the position vector of the component;*
- std::vector< float > **getRotationVector** () const  
*[getRotationVector\(\)](#): Get the rotation vector of the component;*
- std::vector< float > **getScaleVector** () const  
*[getScaleVector\(\)](#): Get the scale vector of the component;*
- void **setTransform** (const std::map< std::string, std::vector< float >> &mapTransform)  
*[setTransform\(\)](#): Set the transformation properties of the component*
- virtual bool **init** ()
- virtual void **update** ()
- template<typename T >  
ComponentTypeID **getComponentTypeID** () noexcept

## Protected Types

- using **ComponentTypeID** = std::size\_t

### 4.12.1 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.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 Transform() [1/2]

```
Transform::Transform ( ) [default]
```

Default **Transform** constructor.

##### Parameters

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

##### Returns

void

#### 4.12.2.2 Transform() [2/2]

```
Transform::Transform (
    const std::map< std::string, std::vector< float >> & mapTransform ) [inline],
[explicit]
```

**Transform** constructor.

##### Parameters

<i>mapTransform</i>	Map containing transformation properties (std::string, std::vector<float>).
---------------------	---

##### Returns

void

#### 4.12.2.3 ~Transform()

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

[Transform](#) destructor.

##### Parameters

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

##### Returns

void

### 4.12.3 Member Function Documentation

#### 4.12.3.1 getBit()

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

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

##### Parameters

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

##### Returns

int: bitmask of the component

#### 4.12.3.2 getPositionVector()

```
std::vector< float > Transform::getPositionVector ( ) const
```

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

##### Parameters

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

##### Returns

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

#### 4.12.3.3 getRotationVector()

```
std::vector< float > Transform::getRotationVector ( ) const
```

[getRotationVector\(\)](#): Get the rotation vector of the component;

##### Parameters

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

##### Returns

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

#### 4.12.3.4 getScaleVector()

```
std::vector< float > Transform::getScaleVector ( ) const
```

[getScaleVector\(\)](#): Get the scale vector of the component;

##### Parameters

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

##### Returns

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

#### 4.12.3.5 setTransform()

```
void Transform::setTransform (
    const std::map< std::string, std::vector< float >> & mapTransform )
```

[setTransform\(\)](#): Set the transformation properties of the component

##### Parameters

<i>mapTransform</i>	Map containing transformation properties (std::string, std::vector<float>).
---------------------	---

**Returns**

void

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

- src/Components/all\_components/Transform.h
- src/Components/all\_components/Transform.cpp

## 4.13 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:

- src/tests/Components/all\_components/TestTransform.cpp

## 4.14 World Class Reference

Inheritance diagram for World:

Collaboration diagram for World:

**Public Member Functions**

- void **createEntities** (std::map< std::string, std::pair< std::unique\_ptr< [EntityManager](#) >, std::vector< std::string >>> &mapEntityManager, std::string keyEntityManager)
- [EntityManager](#) & **addEntityManager** (std::string NameEntityManager)
- [EntityManager](#) & **getEntityManager** (std::string NameEntityManager)
- void **setNameWorld** (std::string newName)
- std::string **getNameWorld** () const
- std::map< std::string, [EntityManager](#) \* > **getEntityManagerMap** () const
- bool **init** () override  
*init(): Initialize the entity*

**Protected Types**

- using **ComponentTypeID** = std::size\_t



## Protected Member Functions

- **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.*
- std::string **getName** () const  
*genName(): Get the name of the entity*
- void **setName** (std::string newName)  
*setName(): Set the name of the entity*
- void **addDrawable** (**Components** \*component)
- void **draw** (sf::RenderWindow &window)
- 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*
- std::bitset< 3 > **getComponentBitset** () const
- std::vector< **DrawableComponent** \* > **getDrawableComponents** () const
- std::array< **Components** \*, 3 > **getComponentArrays** () const
- virtual void **update** ()
- template<typename T >  
ComponentTypeID **getComponentTypeID** () noexcept

### 4.14.1 Member Function Documentation

#### 4.14.1.1 addComponent()

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

**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

**4.14.1.2 addEntity()**

```
Entity & EntityManager::addEntity (
    std::string nameEntity,
    Archetypes newArchetype = Archetypes() ) [inline], [inherited]
```

[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.
-------------	--------------------------

**4.14.1.3 getComponent()**

```
template<typename T >
T & Entity::getComponent [inherited]
```

[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

#### 4.14.1.4 getEntities()

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

[getEntities\(\)](#): Get the [EntityManager](#)'s entities.

##### Parameters

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

##### Returns

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

#### 4.14.1.5 getEntity()

```
Entity & EntityManager::getEntity (
    std::string nameEntity ) [inline], [inherited]
```

[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.

#### 4.14.1.6 getEntityMap()

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

[getEntityMap\(\)](#): Get the [EntityManager](#)'s entity map.

##### Parameters

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

**Returns**

Entity::EntityMap: [Entity](#) map.

**4.14.1.7 getName()**

```
std::string Entity::getName ( ) const [inline], [inherited]
```

getName(): Get the name of the entity

**Parameters**

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

**Returns**

std::string: name of the entity

**4.14.1.8 init()**

```
bool World::init ( ) [inline], [override], [virtual]
```

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

**Parameters**

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

**Returns**

bool: true if the entity is initialized, false otherwise

Reimplemented from [EntityManager](#).

**4.14.1.9 setName()**

```
void Entity::setName (
    std::string newName ) [inline], [inherited]
```

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



# Index

- ~Entity
  - Entity, [10](#)
- ~EntityManager
  - EntityManager, [13](#)
- ~Transform
  - Transform, [25](#)
- addComponent
  - Entity, [10](#)
  - EntityManager, [14](#)
  - GameEngine, [20](#)
  - World, [29](#)
- addEntity
  - EntityManager, [14](#)
  - GameEngine, [21](#)
  - World, [30](#)
- Archetypes, [7](#)
- Audio, [7](#)
- Components, [7](#)
- DrawableComponent, [8](#)
- Entity, [8](#)
  - ~Entity, [10](#)
  - addComponent, [10](#)
  - Entity, [9](#)
  - getComponent, [10](#)
  - getName, [11](#)
  - init, [11](#)
  - setName, [12](#)
- EntityManager, [12](#)
  - ~EntityManager, [13](#)
  - addComponent, [14](#)
  - addEntity, [14](#)
  - EntityManager, [13](#)
  - getComponent, [15](#)
  - getEntities, [15](#)
  - getEntity, [15](#)
  - getEntityMap, [16](#)
  - getName, [16](#)
  - init, [16](#)
  - setName, [17](#)
- EntityManagerTest, [17](#)
- EntityTest, [18](#)
- EventEngine, [18](#)
- GameEngine, [19](#)
  - addComponent, [20](#)
  - addEntity, [21](#)
  - getComponent, [21](#)
  - getEntities, [21](#)
  - getEntity, [22](#)
  - getEntityMap, [22](#)
  - getName, [23](#)
  - init, [23](#)
  - setName, [23](#)
- getBit
  - Transform, [26](#)
- getComponent
  - Entity, [10](#)
  - EntityManager, [15](#)
  - GameEngine, [21](#)
  - World, [30](#)
- getEntities
  - EntityManager, [15](#)
  - GameEngine, [21](#)
  - World, [30](#)
- getEntity
  - EntityManager, [15](#)
  - GameEngine, [22](#)
  - World, [31](#)
- getEntityMap
  - EntityManager, [16](#)
  - GameEngine, [22](#)
  - World, [31](#)
- getName
  - Entity, [11](#)
  - EntityManager, [16](#)
  - GameEngine, [23](#)
  - World, [32](#)
- getPositionVector
  - Transform, [26](#)
- getRotationVector
  - Transform, [27](#)
- getScaleVector
  - Transform, [27](#)
- init
  - Entity, [11](#)
  - EntityManager, [16](#)
  - GameEngine, [23](#)
  - World, [32](#)
- setName
  - Entity, [12](#)
  - EntityManager, [17](#)
  - GameEngine, [23](#)
  - World, [32](#)
- setTransform
  - Transform, [27](#)

Sprite, [24](#)

Transform, [24](#)

    ~Transform, [25](#)

    getBit, [26](#)

    getPositionVector, [26](#)

    getRotationVector, [27](#)

    getScaleVector, [27](#)

    setTransform, [27](#)

    Transform, [25](#)

TransformTest, [28](#)

World, [28](#)

    addComponent, [29](#)

    addEntity, [30](#)

    getComponent, [30](#)

    getEntities, [30](#)

    getEntity, [31](#)

    getEntityMap, [31](#)

    getName, [32](#)

    init, [32](#)

    setName, [32](#)