

## 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 Entity Class Reference	8
4.4.1 Detailed Description	8
4.4.2 Constructor & Destructor Documentation	8
4.4.2.1 Entity() [1/2]	8
4.4.2.2 Entity() [2/2]	9
4.4.2.3 ~Entity()	9
4.4.3 Member Function Documentation	9
4.4.3.1 addComponent()	10
4.4.3.2 getComponent()	10
4.4.3.3 getName()	10
4.4.3.4 init()	11
4.4.3.5 setName()	11
4.5 EntityManager Class Reference	12
4.5.1 Detailed Description	12
4.5.2 Constructor & Destructor Documentation	12
4.5.2.1 EntityManager()	12
4.5.2.2 ~EntityManager()	13
4.5.3 Member Function Documentation	13
4.5.3.1 addEntity()	13
4.5.3.2 getEntities()	13
4.5.3.3 getEntity()	14
4.5.3.4 getEntityMap()	14
4.6 EntityManagerTest Class Reference	15
4.7 EntityTest Class Reference	15
4.8 Rendering Class Reference	15
4.9 Transform Class Reference	15
4.9.1 Detailed Description	16
4.9.2 Constructor & Destructor Documentation	16
4.9.2.1 Transform() [1/2]	16
4.9.2.2 Transform() [2/2]	17
4.9.2.3 ~Transform()	17

4.9.3 Member Function Documentation . . . . .	17
4.9.3.1 getBit() . . . . .	17
4.9.3.2 getPositionVector() . . . . .	18
4.9.3.3 getRotationVector() . . . . .	18
4.9.3.4 getScaleVector() . . . . .	18
4.9.3.5 setTransform() . . . . .	19
4.10 TransformTest Class Reference . . . . .	19
<b>Index</b>	<b>21</b>

## 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
Transform . . . . .	15
Rendering . . . . .	15
testing::Test	
EntityManagerTest . . . . .	15
EntityTest . . . . .	15
TransformTest . . . . .	19





## 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="#">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>	
<a href="#">EntityManager</a> class: <a href="#">EntityManager</a> is a class that represents an entity manager in the game	12
<a href="#">EntityManagerTest</a>	15
<a href="#">EntityTest</a>	15
<a href="#">Rendering</a>	15
<a href="#">Transform</a>	
<a href="#">Transform</a> class: <a href="#">Transform</a> is a class that represents the transform of a Component	15
<a href="#">TransformTest</a>	19



## Chapter 4

# Class Documentation

### 4.1 Archetypes Class Reference

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

- Archetype/Archetypes.h

### 4.2 Audio Class Reference

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

- Components/all\_components/Audio.h

### 4.3 Components Class Reference

Inheritance diagram for Components:

#### Public Member Functions

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

#### Protected Types

- using **ComponentTypeID** = std::size\_t
- using **ComponentBitset** = std::bitset< 3 >
- using **ComponentArray** = std::array< [Components](#) \*, 3 >

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

- Components/Components.h
- Components/Components.cpp

## 4.4 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*
- 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*

### Protected Types

- using [EntityMap](#) = std::map< std::string, [Entity](#) \* >

#### 4.4.1 Detailed Description

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

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

#### 4.4.2 Constructor & Destructor Documentation

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

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

Default [Entity](#) constructor.

## Parameters

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

## Returns

void

**4.4.2.2 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

**4.4.2.3 ~Entity()**

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

[Entity](#) destructor.

## Parameters

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

## Returns

void

**4.4.3 Member Function Documentation**

#### 4.4.3.1 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

#### 4.4.3.2 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

#### 4.4.3.3 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

**4.4.3.4 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](#).

**4.4.3.5 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:

- Entity/entity.h
- Entity/entity.cpp

## 4.5 EntityManager Class Reference

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

```
#include <entityManager.h>
```

Inheritance diagram for EntityManager:

Collaboration diagram for EntityManager:

### Public Member Functions

- [EntityManager](#) ()=default  
*Default [EntityManager](#) constructor.*
- [~EntityManager](#) ()=default  
*[EntityManager](#) destructor.*
- template<typename T , typename... TArgs>  
T & [addEntity](#) (TArgs &&...args)  
*[addEntity\(\)](#): Create and add a new entity to the entity manager.*
- template<typename T >  
T & [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.*
- Entity::EntityMap [getEntityMap](#) () const  
*[getEntityMap\(\)](#): Get the [EntityManager](#)'s entity map.*

### 4.5.1 Detailed Description

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

The [EntityManager](#) class manages entities.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 EntityManager()

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

Default [EntityManager](#) constructor.

Parameters

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



## Returns

void

**4.5.2.2 ~EntityManager()**`EntityManager::~EntityManager ( ) [default]`[EntityManager](#) destructor.

## Parameters

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

## Returns

void

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

```
template<typename T , typename... TArgs>
T & EntityManager::addEntity (
    TArgs &&... args )
```

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

**4.5.3.2 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.

**4.5.3.3 getEntity()**

```
template<typename T >
T & 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.

**4.5.3.4 getEntityMap()**

```
Entity::EntityMap EntityManager::getEntityMap ( ) const [inline]
```

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

**Parameters**

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

**Returns**

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

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

- Entity/entityManager.h
- Entity/entityManager.cpp

## 4.6 EntityManagerTest Class Reference

Inheritance diagram for EntityManagerTest:

## 4.7 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:

- tests/Entity/TestEntity.cpp

## 4.8 Rendering Class Reference

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

- Components/all\_components/Rendering.h

## 4.9 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.*
- [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*

## Additional Inherited Members

### 4.9.1 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.

### 4.9.2 Constructor & Destructor Documentation

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

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

Default [Transform](#) constructor.

#### Parameters

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

#### Returns

void

#### 4.9.2.2 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

#### 4.9.2.3 ~Transform()

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

[Transform](#) destructor.

##### Parameters

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

##### Returns

void

### 4.9.3 Member Function Documentation

#### 4.9.3.1 getBit()

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

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

##### Parameters

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

**Returns**

int: bitmask of the component

**4.9.3.2 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

**4.9.3.3 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

**4.9.3.4 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

**4.9.3.5 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 ( <code>std::string</code> , <code>std::vector&lt;float&gt;</code> ).
---------------------	--

**Returns**

`void`

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

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

**4.10 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`





# Index

- ~Entity
  - Entity, [9](#)
- ~EntityManager
  - EntityManager, [13](#)
- ~Transform
  - Transform, [17](#)
- addComponent
  - Entity, [9](#)
- addEntity
  - EntityManager, [13](#)
- Archetypes, [7](#)
- Audio, [7](#)
- Components, [7](#)
- Entity, [8](#)
  - ~Entity, [9](#)
  - addComponent, [9](#)
  - Entity, [8](#), [9](#)
  - getComponent, [10](#)
  - getName, [10](#)
  - init, [11](#)
  - setName, [11](#)
- EntityManager, [12](#)
  - ~EntityManager, [13](#)
  - addEntity, [13](#)
  - EntityManager, [12](#)
  - getEntities, [13](#)
  - getEntity, [14](#)
  - getEntityMap, [14](#)
- EntityManagerTest, [15](#)
- EntityTest, [15](#)
- getBit
  - Transform, [17](#)
- getComponent
  - Entity, [10](#)
- getEntities
  - EntityManager, [13](#)
- getEntity
  - EntityManager, [14](#)
- getEntityMap
  - EntityManager, [14](#)
- getName
  - Entity, [10](#)
- getPositionVector
  - Transform, [18](#)
- getRotationVector
  - Transform, [18](#)
- getScaleVector
  - Transform, [18](#)
- init
  - Entity, [11](#)
- Rendering, [15](#)
- setName
  - Entity, [11](#)
- setTransform
  - Transform, [19](#)
- Transform, [15](#)
  - ~Transform, [17](#)
  - getBit, [17](#)
  - getPositionVector, [18](#)
  - getRotationVector, [18](#)
  - getScaleVector, [18](#)
  - setTransform, [19](#)
  - Transform, [16](#)
- TransformTest, [19](#)