# R-Type - Engine

Generated by Doxygen 1.9.1

# Chapter 1

# Engine

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# 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**

| | |
|---|---|
| *void* | |

**Returns**

void

#### 4.4.2.2 Entity() [2/2]

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

Entity constructor.

**Parameters**

| | |
|---|---|
| *nameEntity* | name of the entity |
| *newArchetype* | archetype of the entity (optional, default = new archetype) |

**Returns**

void

#### 4.4.2.3 ~Entity()

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

Entity destructor.

**Parameters**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *T* | Type of the component |
| *TArgs* | Variadic template for component constructor arguments. |

**Parameters**

| | |
|---|---|
| *args* | 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**

| | |
|---|---|
| *T* | Type of the component |

**Parameters**

| | |
|---|---|
| *void* | |

**Returns**

T&: reference of the component

#### 4.4.3.3 getName()

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

genName(): Get the name of the entity

**Parameters**

| | |
|---|---|
| *void* | |

**Returns**

std::string: name of the entity

#### 4.4.3.4 init()

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

init(): Initialize the entity

**Parameters**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *newName* | 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**

| | |
|---|---|
| *void* | |

**Returns**

void

#### 4.5.2.2 ~EntityManager()

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

EntityManager destructor.

**Parameters**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *T* | Type of the entity. |
| *TArgs* | Type of the arguments. |

**Parameters**

| | |
|---|---|
| *args* | 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**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *T* | Type of the entity. |

**Parameters**

| | |
|---|---|
| *nameEntity* | 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**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *void* | |

**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**

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

**Returns**

void

#### 4.9.2.3 ~Transform()

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

Transform destructor.

**Parameters**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *void* | |

**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**

| | |
|---|---|
| *void* | |

**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**

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

**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