

## 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
4.2.1 Detailed Description	8
4.2.2 Constructor & Destructor Documentation	8
4.2.2.1 Audio() [1/2]	8
4.2.2.2 Audio() [2/2]	9
4.2.2.3 ~Audio()	9
4.2.3 Member Function Documentation	9
4.2.3.1 getSound()	9
4.2.3.2 getSoundBuffer()	10
4.2.3.3 getVolume()	10
4.2.3.4 isPlaying()	10
4.2.3.5 loadSoundBuffer()	11
4.2.3.6 pause()	11
4.2.3.7 play()	11
4.2.3.8 setLoop()	12
4.2.3.9 setSound()	12
4.2.3.10 setSoundBuffer()	12
4.2.3.11 setVolume()	13
4.2.3.12 stop()	13
4.3 AudioTest Class Reference	14
4.4 Components Class Reference	14
4.5 Entity Class Reference	15
4.5.1 Detailed Description	15
4.5.2 Constructor & Destructor Documentation	15
4.5.2.1 Entity() [1/2]	15
4.5.2.2 Entity() [2/2]	16
4.5.2.3 ~Entity()	16
4.5.3 Member Function Documentation	16
4.5.3.1 addComponent()	17
4.5.3.2 getComponent()	17
4.5.3.3 getName()	17
4.5.3.4 init()	18
4.5.3.5 setName()	18

---

4.6 EntityManager Class Reference . . . . .	19
4.6.1 Detailed Description . . . . .	19
4.6.2 Constructor & Destructor Documentation . . . . .	19
4.6.2.1 EntityManager() . . . . .	19
4.6.2.2 ~EntityManager() . . . . .	20
4.6.3 Member Function Documentation . . . . .	20
4.6.3.1 addEntity() . . . . .	20
4.6.3.2 getEntities() . . . . .	20
4.6.3.3 getEntity() . . . . .	21
4.6.3.4 getEntityMap() . . . . .	21
4.7 EntityManagerTest Class Reference . . . . .	22
4.8 EntityTest Class Reference . . . . .	22
4.9 Rendering Class Reference . . . . .	22
4.10 Transform Class Reference . . . . .	22
4.10.1 Detailed Description . . . . .	23
4.10.2 Constructor & Destructor Documentation . . . . .	23
4.10.2.1 Transform() [1/2] . . . . .	23
4.10.2.2 Transform() [2/2] . . . . .	24
4.10.2.3 ~Transform() . . . . .	24
4.10.3 Member Function Documentation . . . . .	24
4.10.3.1 getBit() . . . . .	24
4.10.3.2 getPositionVector() . . . . .	25
4.10.3.3 getRotationVector() . . . . .	25
4.10.3.4 getScaleVector() . . . . .	25
4.10.3.5 setTransform() . . . . .	26
4.11 TransformTest Class Reference . . . . .	26
<b>Index</b>	<b>27</b>

## Chapter 1

# Engine



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Archetypes . . . . .	7
Components . . . . .	14
Audio . . . . .	7
Entity . . . . .	15
EntityManager . . . . .	19
Transform . . . . .	22
Rendering . . . . .	22
testing::Test	
AudioTest . . . . .	14
EntityManagerTest . . . . .	22
EntityTest . . . . .	22
TransformTest . . . . .	26





## 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>	
<a href="#">Audio</a> class: <a href="#">Audio</a> is a class that represents the audio properties of a Component	7
<a href="#">AudioTest</a>	14
<a href="#">Components</a>	14
<a href="#">Entity</a>	
<a href="#">Entity</a> class: <a href="#">Entity</a> is a class that represents an entity in the game	15
<a href="#">EntityManager</a>	
<a href="#">EntityManager</a> class: <a href="#">EntityManager</a> is a class that represents an entity manager in the game	19
<a href="#">EntityManagerTest</a>	22
<a href="#">EntityTest</a>	22
<a href="#">Rendering</a>	22
<a href="#">Transform</a>	
<a href="#">Transform</a> class: <a href="#">Transform</a> is a class that represents the transform of a Component	22
<a href="#">TransformTest</a>	26



## 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

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

```
#include <Audio.h>
```

Inheritance diagram for [Audio](#):

Collaboration diagram for [Audio](#):

#### Public Member Functions

- [Audio](#) ()=default  
*Default [Audio](#) constructor.*
- [Audio](#) (const sf::SoundBuffer &buffer)  
*[Audio](#) constructor with an existing sound buffer. Automatically set the sound.*
- [~Audio](#) () override=default  
*[Audio](#) 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.*

- 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.*

## Additional Inherited Members

### 4.2.1 Detailed Description

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

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

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 [Audio\(\)](#) [1/2]

```
Audio::Audio ( ) [default]
```

Default [Audio](#) constructor.

Parameters

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

Returns

void

#### 4.2.2.2 Audio() [2/2]

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

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

##### Parameters

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

##### Returns

void

#### 4.2.2.3 ~Audio()

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

[Audio](#) destructor.

##### Parameters

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

##### Returns

void

### 4.2.3 Member Function Documentation

#### 4.2.3.1 getSound()

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

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

##### Parameters

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

**Returns**

sf::Sound: SFML Sound for audio.

**4.2.3.2 getSoundBuffer()**

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

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

**Parameters**

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

**Returns**

sf::SoundBuffer: SFML SoundBuffer for audio.

**4.2.3.3 getVolume()**

```
float Audio::getVolume ( ) const
```

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

**Parameters**

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

**Returns**

float: Volume level (0 to 100).

**4.2.3.4 isPlaying()**

```
bool Audio::isPlaying ( ) const
```

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

**Parameters**

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

**Returns**

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

**4.2.3.5 loadSoundBuffer()**

```
bool Audio::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.

**4.2.3.6 pause()**

```
void Audio::pause ( )
```

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

**Parameters**

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

**Returns**

void

**4.2.3.7 play()**

```
void Audio::play ( )
```

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

**Parameters**

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

**Returns**

void

**4.2.3.8 setLoop()**

```
void Audio::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

**4.2.3.9 setSound()**

```
bool Audio::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.

**4.2.3.10 setSoundBuffer()**

```
bool Audio::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.

**4.2.3.11 setVolume()**

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

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

## Parameters

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

## Returns

void

**4.2.3.12 stop()**

```
void Audio::stop ( )
```

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

## Parameters

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

## Returns

void

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

- Components/all\_components/Audio.h
- Components/all\_components/Audio.cpp

## 4.3 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

## 4.4 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

## 4.5 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  
*`getName()`: 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` \* >

#### 4.5.1 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](#).

**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:

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

## 4.6 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.*

### 4.6.1 Detailed Description

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

The [EntityManager](#) class manages entities.

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 EntityManager()

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

Default [EntityManager](#) constructor.

Parameters

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

**Returns**

void

**4.6.2.2 ~EntityManager()**

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

[EntityManager](#) destructor.

**Parameters**

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

**Returns**

void

**4.6.3 Member Function Documentation****4.6.3.1 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.
-------------	--------------------------

**4.6.3.2 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.

### 4.6.3.3 `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.

### 4.6.3.4 `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

## 4.7 EntityManagerTest Class Reference

Inheritance diagram for EntityManagerTest:

## 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:

- tests/Entity/TestEntity.cpp

## 4.9 Rendering Class Reference

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

- Components/all\_components/Rendering.h

## 4.10 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

### 4.10.1 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.

### 4.10.2 Constructor & Destructor Documentation

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

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

Default [Transform](#) constructor.

Parameters

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

Returns

void

#### 4.10.2.2 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

#### 4.10.2.3 ~Transform()

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

[Transform](#) destructor.

##### Parameters

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

##### Returns

void

### 4.10.3 Member Function Documentation

#### 4.10.3.1 getBit()

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

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

##### Parameters

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

**Returns**

int: bitmask of the component

**4.10.3.2 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

**4.10.3.3 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

**4.10.3.4 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

**4.10.3.5 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`

**4.11 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

- ~Audio
  - Audio, [9](#)
- ~Entity
  - Entity, [16](#)
- ~EntityManager
  - EntityManager, [20](#)
- ~Transform
  - Transform, [24](#)
- addComponent
  - Entity, [16](#)
- addEntity
  - EntityManager, [20](#)
- Archetypes, [7](#)
- Audio, [7](#)
  - ~Audio, [9](#)
  - Audio, [8](#)
  - getSound, [9](#)
  - getSoundBuffer, [10](#)
  - getVolume, [10](#)
  - isPlaying, [10](#)
  - loadSoundBuffer, [11](#)
  - pause, [11](#)
  - play, [11](#)
  - setLoop, [12](#)
  - setSound, [12](#)
  - setSoundBuffer, [12](#)
  - setVolume, [13](#)
  - stop, [13](#)
- AudioTest, [14](#)
- Components, [14](#)
- Entity, [15](#)
  - ~Entity, [16](#)
  - addComponent, [16](#)
  - Entity, [15](#), [16](#)
  - getComponent, [17](#)
  - getName, [17](#)
  - init, [18](#)
  - setName, [18](#)
- EntityManager, [19](#)
  - ~EntityManager, [20](#)
  - addEntity, [20](#)
  - EntityManager, [19](#)
  - getEntities, [20](#)
  - getEntity, [21](#)
  - getEntityMap, [21](#)
- EntityManagerTest, [22](#)
- EntityTest, [22](#)
  - getBit
    - Transform, [24](#)
  - getComponent
    - Entity, [17](#)
  - getEntities
    - EntityManager, [20](#)
  - getEntity
    - EntityManager, [21](#)
  - getEntityMap
    - EntityManager, [21](#)
  - getName
    - Entity, [17](#)
  - getPositionVector
    - Transform, [25](#)
  - getRotationVector
    - Transform, [25](#)
  - getScaleVector
    - Transform, [25](#)
  - getSound
    - Audio, [9](#)
  - getSoundBuffer
    - Audio, [10](#)
  - getVolume
    - Audio, [10](#)
  - init
    - Entity, [18](#)
  - isPlaying
    - Audio, [10](#)
  - loadSoundBuffer
    - Audio, [11](#)
  - pause
    - Audio, [11](#)
  - play
    - Audio, [11](#)
- Rendering, [22](#)
- setLoop
  - Audio, [12](#)
- setName
  - Entity, [18](#)
- setSound
  - Audio, [12](#)
- setSoundBuffer
  - Audio, [12](#)
- setTransform
  - Transform, [26](#)
- setVolume

- Audio, [13](#)
- stop
  - Audio, [13](#)
- Transform, [22](#)
  - ~Transform, [24](#)
  - getBit, [24](#)
  - getPositionVector, [25](#)
  - getRotationVector, [25](#)
  - getScaleVector, [25](#)
  - setTransform, [26](#)
  - Transform, [23](#)
- TransformTest, [26](#)