

## Arcade

Generated by Doxygen 1.9.8



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AEventManager Class Reference	7
4.1.1 Detailed Description	9
4.1.2 Constructor & Destructor Documentation	9
4.1.2.1 AEventManager()	9
4.1.3 Member Function Documentation	9
4.1.3.1 getEventState()	9
4.1.3.2 getMouseX()	10
4.1.3.3 getMouseY()	10
4.1.3.4 pollEvents()	10
4.1.3.5 setEventState()	10
4.1.3.6 setMouseX()	11
4.1.3.7 setMouseY()	11
4.2 ComponentManager< T > Class Template Reference	11
4.2.1 Detailed Description	13
4.2.2 Member Function Documentation	13
4.2.2.1 addComponent()	13
4.2.2.2 getAll()	13
4.2.2.3 getComponent()	14
4.2.2.4 removeComponent()	14
4.3 Arcade::Core Class Reference	15
4.4 EntityManager Class Reference	16
4.4.1 Detailed Description	16
4.4.2 Member Function Documentation	16
4.4.2.1 createEntity()	16
4.4.2.2 destroyEntity()	17
4.4.2.3 getEntities()	17
4.5 Game1 Class Reference	17
4.6 Arcade::IDisplayModule Class Reference	18
4.7 Arcade::IEventManager Class Reference	19
4.7.1 Detailed Description	20
4.7.2 Member Function Documentation	20
4.7.2.1 getEventState()	20
4.7.2.2 getMouseX()	21

4.7.2.3 <a href="#">getMouseY()</a> . . . . .	21
4.7.2.4 <a href="#">pollEvents()</a> . . . . .	21
4.7.2.5 <a href="#">setEventState()</a> . . . . .	21
4.8 <a href="#">Arcade::IGameModule Class Reference</a> . . . . .	22
4.8.1 Detailed Description . . . . .	23
4.8.2 Member Function Documentation . . . . .	23
4.8.2.1 <a href="#">getElements()</a> . . . . .	23
4.8.2.2 <a href="#">getName()</a> . . . . .	23
4.8.2.3 <a href="#">getScore()</a> . . . . .	23
4.8.2.4 <a href="#">handleInput()</a> . . . . .	24
4.8.2.5 <a href="#">init()</a> . . . . .	24
4.8.2.6 <a href="#">render()</a> . . . . .	24
4.8.2.7 <a href="#">update()</a> . . . . .	24
4.9 <a href="#">Arcade::ISystem Class Reference</a> . . . . .	25
4.9.1 Detailed Description . . . . .	26
4.9.2 Constructor & Destructor Documentation . . . . .	26
4.9.2.1 <a href="#">~ISystem()</a> . . . . .	26
4.9.3 Member Function Documentation . . . . .	26
4.9.3.1 <a href="#">update()</a> . . . . .	26
4.10 <a href="#">MovementSystem Class Reference</a> . . . . .	27
4.10.1 Detailed Description . . . . .	27
4.10.2 Constructor & Destructor Documentation . . . . .	27
4.10.2.1 <a href="#">MovementSystem()</a> . . . . .	27
4.10.3 Member Function Documentation . . . . .	28
4.10.3.1 <a href="#">update()</a> . . . . .	28
4.11 <a href="#">PositionComponent Struct Reference</a> . . . . .	28
4.11.1 Detailed Description . . . . .	29
4.11.2 Constructor & Destructor Documentation . . . . .	29
4.11.2.1 <a href="#">PositionComponent()</a> . . . . .	29
4.12 <a href="#">SpriteComponent Struct Reference</a> . . . . .	29
4.12.1 Detailed Description . . . . .	30
4.12.2 Constructor & Destructor Documentation . . . . .	30
4.12.2.1 <a href="#">SpriteComponent()</a> . . . . .	30
4.13 <a href="#">SystemManager Class Reference</a> . . . . .	31
4.13.1 Detailed Description . . . . .	32
4.13.2 Member Function Documentation . . . . .	32
4.13.2.1 <a href="#">addSystem()</a> . . . . .	32
4.13.2.2 <a href="#">updateSystems()</a> . . . . .	32
4.14 <a href="#">Arcade::Vector2i Struct Reference</a> . . . . .	33
4.14.1 Detailed Description . . . . .	33
4.14.2 Constructor & Destructor Documentation . . . . .	33
4.14.2.1 <a href="#">Vector2i()</a> . . . . .	33

4.15 VelocityComponent Struct Reference	34
4.15.1 Detailed Description	34
4.15.2 Constructor & Destructor Documentation	35
4.15.2.1 VelocityComponent()	35
4.16 Arcade::WindowManager Class Reference	35
4.16.1 Detailed Description	36
4.16.2 Member Function Documentation	36
4.16.2.1 getFps()	36
4.16.2.2 getHeight()	37
4.16.2.3 getIconPath()	37
4.16.2.4 getTitle()	37
4.16.2.5 getWidth()	37
4.16.2.6 getWindowSize()	37
4.16.2.7 setFps()	37
4.16.2.8 setHeight()	38
4.16.2.9 setIconPath()	38
4.16.2.10 setTitle()	38
4.16.2.11 setWidth()	38
4.16.2.12 setWindowSize()	39
<b>5 File Documentation</b>	<b>41</b>
5.1 core.hpp	41
5.2 DLLoader.hpp	41
5.3 GameLoop.hpp	42
5.4 Game1.hpp	42
5.5 Game2.hpp	42
5.6 Game3.hpp	42
5.7 nCurses.hpp	42
5.8 SDL.hpp	43
5.9 SFML.hpp	43
5.10 Position.hpp	43
5.11 Sprite.hpp	43
5.12 Velocity.hpp	44
5.13 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.hpp File Reference	44
5.13.1 Detailed Description	45
5.14 AEventManager.hpp	45
5.15 IDisplayModule.hpp	45
5.16 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IEventManager.hpp File Reference	46
5.16.1 Detailed Description	48
5.17 IEventManager.hpp	48
5.18 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IGameModule.hpp File Reference	49
5.18.1 Detailed Description	50

5.19 IGameModule.hpp . . . . .	50
5.20 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/ISystem.hpp File Reference . . . .	51
5.20.1 Detailed Description . . . . .	51
5.21 ISystem.hpp . . . . .	52
5.22 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/Component↵ Manager.hpp File Reference . . . . .	52
5.22.1 Detailed Description . . . . .	53
5.23 ComponentManager.hpp . . . . .	53
5.24   /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/Entity↵ Manager.hpp File Reference . . . . .	54
5.24.1 Detailed Description . . . . .	55
5.25 EntityManager.hpp . . . . .	55
5.26 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.hpp File Reference . . . . .	55
5.26.1 Detailed Description . . . . .	56
5.27 SystemManager.hpp . . . . .	56
5.28 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.hpp File Reference . . . . .	56
5.28.1 Detailed Description . . . . .	57
5.29 WindowManager.hpp . . . . .	57
5.30 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp File Reference	58
5.30.1 Detailed Description . . . . .	58
5.31 MovementSystem.hpp . . . . .	58
<b>Index</b>	<b>61</b>

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

ComponentManager< T > . . . . .	11
ComponentManager< PositionComponent > . . . . .	11
ComponentManager< VelocityComponent > . . . . .	11
Arcade::Core . . . . .	15
EntityManager . . . . .	16
Game1 . . . . .	17
Arcade::IDisplayModule . . . . .	18
Arcade::IEventManager . . . . .	19
AEventManager . . . . .	7
Arcade::IGameModule . . . . .	22
Arcade::ISystem . . . . .	25
SystemManager . . . . .	31
MovementSystem . . . . .	27
PositionComponent . . . . .	28
SpriteComponent . . . . .	29
Arcade::Vector2i . . . . .	33
VelocityComponent . . . . .	34
Arcade::WindowManager . . . . .	35





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AEventManager</a>	
Abstract base class for event management . . . . .	7
<a href="#">ComponentManager&lt; T &gt;</a>	
Manages components of a specific type for entities . . . . .	11
<a href="#">Arcade::Core</a>	
<a href="#">EntityManager</a>	
Alias for entity identifiers . . . . .	16
<a href="#">Game1</a>	
. . . . .	17
<a href="#">Arcade::IDisplayModule</a>	
. . . . .	18
<a href="#">Arcade::IEventManager</a>	
Interface for event management . . . . .	19
<a href="#">Arcade::IGameModule</a>	
Interface for game logic modules . . . . .	22
<a href="#">Arcade::ISystem</a>	
Interface for game systems in the ECS . . . . .	25
<a href="#">MovementSystem</a>	
System responsible for updating entity movement . . . . .	27
<a href="#">PositionComponent</a>	
Represents the position of an entity in a 2D space . . . . .	28
<a href="#">SpriteComponent</a>	
Represents a graphical sprite associated with an entity . . . . .	29
<a href="#">SystemManager</a>	
Manages and updates game systems . . . . .	31
<a href="#">Arcade::Vector2i</a>	
Represents a 2D vector with integer coordinates . . . . .	33
<a href="#">VelocityComponent</a>	
Represents an entity's velocity in 2D space . . . . .	34
<a href="#">Arcade::WindowManager</a>	
Handles window properties such as size, title, frame rate, and icon . . . . .	35



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/core.hpp	41
/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/DLLoader/DLLoader.hpp	41
/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/GameLoop/GameLoop.hpp	42
/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game1/Game1.hpp	42
/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game2/Game2.hpp	42
/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game3/Game3.hpp	42
/home/stesson/Epitech/B-OOP-300/Arcade/src/Graphics/nCurses/nCurses.hpp	42
/home/stesson/Epitech/B-OOP-300/Arcade/src/Graphics/SDL/SDL.hpp	43
/home/stesson/Epitech/B-OOP-300/Arcade/src/Graphics/SFML/SFML.hpp	43
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Position.hpp	43
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Sprite.hpp	43
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/VelocityEngine.hpp	44
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.hpp	
Abstract class for event management in the Arcade project	44
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IDisplayModule.hpp	45
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IEventManager.hpp	
Interface for managing user input events in the Arcade project	46
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IGameModule.hpp	
Interface for game modules in the Arcade project	49
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/ISystem.hpp	
Interface for systems in the Entity Component System (ECS) architecture	51
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/ComponentManager.hpp	
Manages components for entities in an Entity Component System (ECS)	52
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp	
Manages entity creation and deletion in an Entity Component System (ECS)	54
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.hpp	
Manages multiple systems in an Entity Component System (ECS)	55
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.hpp	
Manages window properties for the Arcade project	56
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp	
System for handling entity movement in an ECS architecture	58



## Chapter 4

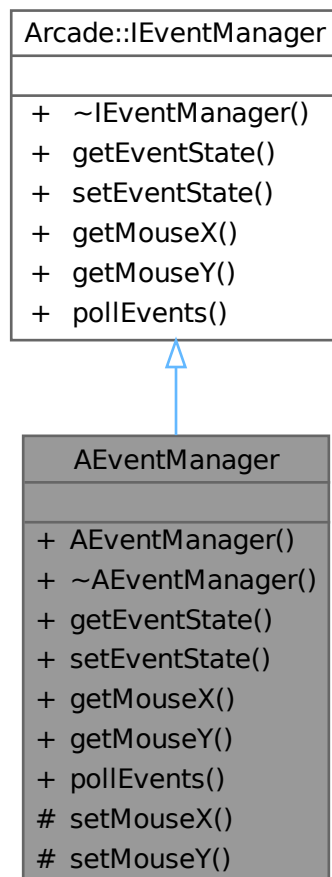
# Class Documentation

### 4.1 AEventManager Class Reference

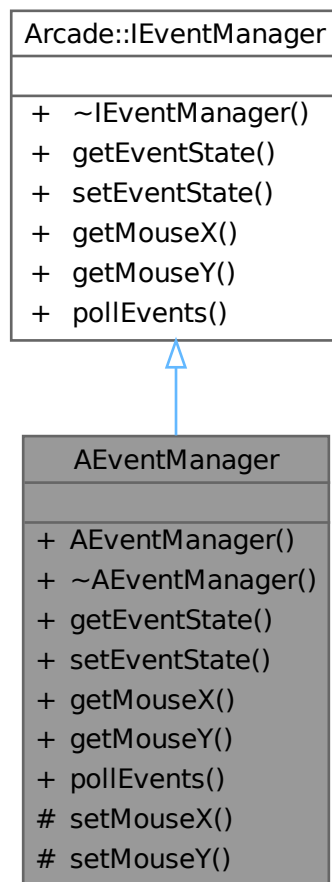
Abstract base class for event management.

```
#include <AEventManager.hpp>
```

Inheritance diagram for AEventManager:



Collaboration diagram for AEventManager:



## Public Member Functions

- [AEventManager](#) ()  
*Constructs an [AEventManager](#) object.*
- virtual `~AEventManager` ()=default  
*Virtual destructor.*
- bool [getEventState](#) (int key) override  
*Retrieves the state of a specific key event.*
- void [setEventState](#) (int key, bool state) override  
*Sets the state of a specific key event.*
- std::size\_t [getMouseX](#) () const override  
*Retrieves the current X-coordinate of the mouse.*
- std::size\_t [getMouseY](#) () const override  
*Retrieves the current Y-coordinate of the mouse.*
- virtual void [pollEvents](#) ()=0  
*Polls for new events.*

## Public Member Functions inherited from [Arcade::IEventManager](#)

- virtual `~IEventManager()`=default  
*Virtual destructor for [IEventManager](#).*

## Protected Member Functions

- void [setMouseX](#)(std::size\_t x)  
*Sets the X-coordinate of the mouse.*
- void [setMouseY](#)(std::size\_t y)  
*Sets the Y-coordinate of the mouse.*

### 4.1.1 Detailed Description

Abstract base class for event management.

The [AEventManager](#) class provides an implementation of the [IEventManager](#) interface with basic event state tracking, including key events and mouse position. It must be extended by concrete event managers that implement [pollEvents\(\)](#).

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 AEventManager()

```
AEventManager::AEventManager ( )
```

Constructs an [AEventManager](#) object.

Initializes key event states and mouse position.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 getEventState()

```
bool AEventManager::getEventState (
    int key ) [override], [virtual]
```

Retrieves the state of a specific key event.

#### Parameters

<i>key</i>	The key/event identifier.
------------	---------------------------

#### Returns

True if the key is currently active, false otherwise.

Implements [Arcade::IEventManager](#).

#### 4.1.3.2 getMouseX()

```
std::size_t AEventManager::getMouseX ( ) const [override], [virtual]
```

Retrieves the current X-coordinate of the mouse.

##### Returns

The current X position of the mouse cursor.

Implements [Arcade::IEventManager](#).

#### 4.1.3.3 getMouseY()

```
std::size_t AEventManager::getMouseY ( ) const [override], [virtual]
```

Retrieves the current Y-coordinate of the mouse.

##### Returns

The current Y position of the mouse cursor.

Implements [Arcade::IEventManager](#).

#### 4.1.3.4 pollEvents()

```
virtual void AEventManager::pollEvents ( ) [pure virtual]
```

Polls for new events.

This method is pure virtual and must be implemented by derived classes.

Implements [Arcade::IEventManager](#).

#### 4.1.3.5 setEventState()

```
void AEventManager::setEventState (
    int key,
    bool state ) [override], [virtual]
```

Sets the state of a specific key event.

##### Parameters

<i>key</i>	The key/event identifier.
<i>state</i>	The new state of the key (true = pressed, false = released).

Implements [Arcade::IEventManager](#).



#### 4.1.3.6 setMouseX()

```
void AEventManager::setMouseX (
    std::size_t x ) [protected]
```

Sets the X-coordinate of the mouse.

##### Parameters

x	The new X position of the mouse cursor.
---	---

#### 4.1.3.7 setMouseY()

```
void AEventManager::setMouseY (
    std::size_t y ) [protected]
```

Sets the Y-coordinate of the mouse.

##### Parameters

y	The new Y position of the mouse cursor.
---	---

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

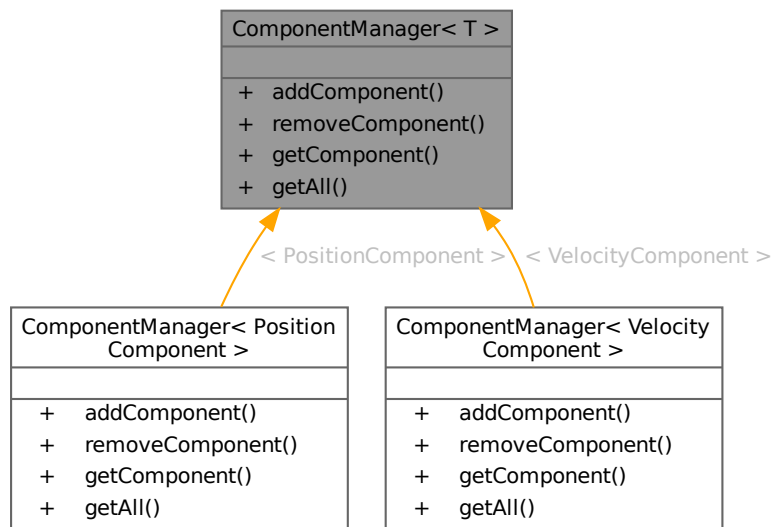
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/[AEventManager.hpp](#)
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.cpp

## 4.2 ComponentManager< T > Class Template Reference

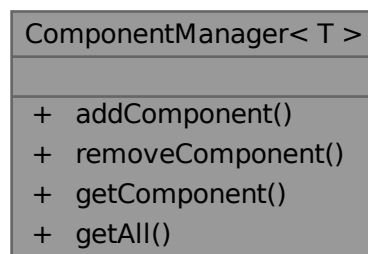
Manages components of a specific type for entities.

```
#include <ComponentManager.hpp>
```

Inheritance diagram for ComponentManager< T >:



Collaboration diagram for ComponentManager< T >:



## Public Member Functions

- void `addComponent` (Entity entity, T component)  
*Maps entities to their corresponding components.*
- void `removeComponent` (Entity entity)  
*Removes a component from an entity.*
- T \* `getComponent` (Entity entity)  
*Retrieves a pointer to an entity's component.*
- std::unordered\_map< Entity, T > & `getAll` ()  
*Retrieves all components.*

### 4.2.1 Detailed Description

**template**<typename T>  
**class** ComponentManager< T >

Manages components of a specific type for entities.

The [ComponentManager](#) class is responsible for storing, retrieving, and removing components associated with entities in an ECS. It uses an unordered map to efficiently store components.

#### Template Parameters

<i>T</i>	The type of component managed by this class.
----------	--

### 4.2.2 Member Function Documentation

#### 4.2.2.1 addComponent()

```
template<typename T >
void ComponentManager< T >::addComponent (
    Entity entity,
    T component )
```

Maps entities to their corresponding components.

Adds a component to an entity.

Associates the specified component with the given entity.

#### Parameters

<i>entity</i>	The entity to which the component is assigned.
<i>component</i>	The component to be added.

#### 4.2.2.2 getAll()

```
template<typename T >
std::unordered_map< Entity, T > & ComponentManager< T >::getAll ( )
```

Retrieves all components.

Provides access to the internal unordered map storing all components mapped to their respective entities.

#### Returns

A reference to the unordered map of entity-component pairs.

Here is the caller graph for this function:



#### 4.2.2.3 GetComponent()

```

template<typename T >
T * ComponentManager< T >::GetComponent (
    Entity entity )
  
```

Retrieves a pointer to an entity's component.

Returns a pointer to the component if it exists, or nullptr if the entity does not have the requested component.

##### Parameters

<i>entity</i>	The entity whose component is being retrieved.
---------------	--

##### Returns

A pointer to the component, or nullptr if not found.

Here is the caller graph for this function:



#### 4.2.2.4 removeComponent()

```

template<typename T >
void ComponentManager< T >::removeComponent (
    Entity entity )
  
```

Removes a component from an entity.

Deletes the component associated with the specified entity.

## Parameters

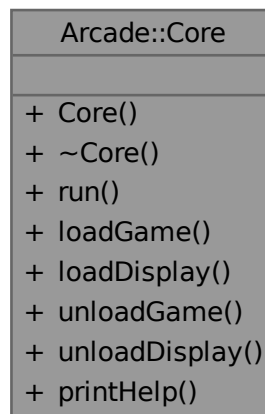
<i>entity</i>	The entity whose component should be removed.
---------------	---

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

- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/ComponentManager.hpp](#)
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/ComponentManager.cpp](#)

## 4.3 Arcade::Core Class Reference

Collaboration diagram for Arcade::Core:



### Public Member Functions

- void **run** ()
- void **loadGame** (const std::string &gameName)
- void **loadDisplay** (const std::string &displayName)
- void **unloadGame** ()
- void **unloadDisplay** ()
- void **printHelp** ()

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

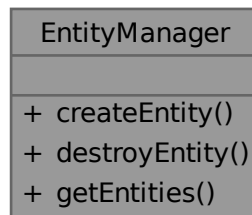
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/core.hpp](#)
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/core.cpp](#)

## 4.4 EntityManager Class Reference

Alias for entity identifiers.

```
#include <EntityManager.hpp>
```

Collaboration diagram for EntityManager:



### Public Member Functions

- Entity [createEntity](#) ()  
*Creates a new entity.*
- void [destroyEntity](#) (Entity entity)  
*Destroys an existing entity.*
- const std::vector< Entity > & [getEntities](#) () const  
*Retrieves the list of active entities.*

### 4.4.1 Detailed Description

Alias for entity identifiers.

Manages entities in an ECS.

The [EntityManager](#) class handles the creation and deletion of entities, ensuring each entity has a unique identifier and maintaining a list of active entities.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 createEntity()

```
Entity EntityManager::createEntity ( )
```

Creates a new entity.

Generates a new unique entity identifier and adds it to the list of active entities.

#### Returns

The newly created entity identifier.

#### 4.4.2.2 destroyEntity()

```
void EntityManager::destroyEntity (
    Entity entity )
```

Destroys an existing entity.

Removes the specified entity from the list of active entities.

##### Parameters

<i>entity</i>	The entity identifier to be destroyed.
---------------	--

#### 4.4.2.3 getEntities()

```
const std::vector< Entity > & EntityManager::getEntities ( ) const
```

Retrieves the list of active entities.

##### Returns

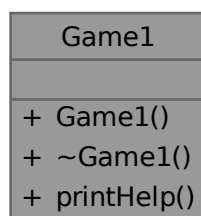
A constant reference to a vector containing all active entity identifiers.

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

- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/[EntityManager.hpp](#)
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.cpp

## 4.5 Game1 Class Reference

Collaboration diagram for Game1:



### Public Member Functions

- void **printHelp** ()

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

- /home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game1/Game1.hpp
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game1/Game1.cpp

## 4.6 Arcade::IDisplayModule Class Reference

Collaboration diagram for Arcade::IDisplayModule:

Arcade::IDisplayModule
<ul style="list-style-type: none"><li>+ ~IDisplayModule()</li><li>+ init()</li><li>+ getInput()</li><li>+ drawElement()</li><li>+ render()</li><li>+ getName()</li></ul>

### Public Member Functions

- virtual void **init** ()=0
- virtual [IEventManager](#) **getInput** (void)=0
- virtual void **drawElement** (const std::vector< Entity > &entities)=0
- virtual void **render** ()=0
- virtual std::string **getName** () const =0

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

- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IDisplayModule.hpp

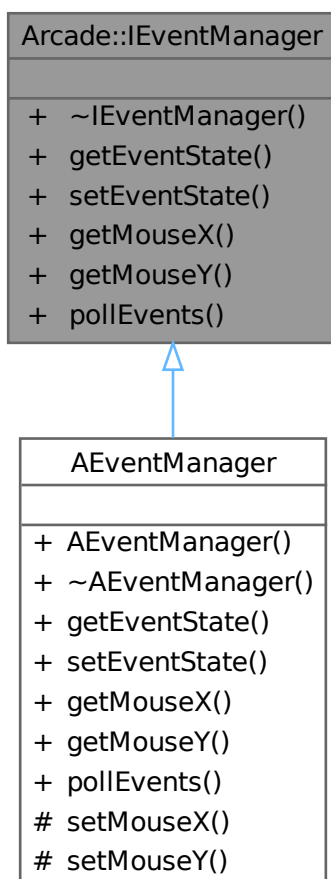


## 4.7 Arcade::IEventManager Class Reference

Interface for event management.

```
#include <IEventManager.hpp>
```

Inheritance diagram for Arcade::IEventManager:



Collaboration diagram for Arcade::IEventManager:

Arcade::IEventManager
<ul style="list-style-type: none"> <li>+ ~IEventManager()</li> <li>+ getEventState()</li> <li>+ setEventState()</li> <li>+ getMouseX()</li> <li>+ getMouseY()</li> <li>+ pollEvents()</li> </ul>

### Public Member Functions

- virtual **~IEventManager** ()=default  
*Virtual destructor for [IEventManager](#).*
- virtual bool [getEventState](#) (int key)=0  
*Checks if a specific event is active.*
- virtual void [setEventState](#) (int key, bool state)=0  
*Sets the state of a specific event.*
- virtual std::size\_t [getMouseX](#) () const =0  
*Retrieves the current X-coordinate of the mouse.*
- virtual std::size\_t [getMouseY](#) () const =0  
*Retrieves the current Y-coordinate of the mouse.*
- virtual void [pollEvents](#) ()=0  
*Polls for new events.*

## 4.7.1 Detailed Description

Interface for event management.

The [IEventManager](#) interface provides methods for handling user input events, including keyboard and mouse input, event polling, and retrieving the current mouse position.

## 4.7.2 Member Function Documentation

### 4.7.2.1 getEventState()

```
virtual bool Arcade::IEventManager::getEventState (
    int key ) [pure virtual]
```

Checks if a specific event is active.

This method allows querying the state of a particular key or event.

## Parameters

<i>key</i>	The key/event identifier.
------------	---------------------------

## Returns

True if the event is active, false otherwise.

Implemented in [AEventManager](#).

**4.7.2.2 getMouseX()**

```
virtual std::size_t Arcade::IEventManager::getMouseX ( ) const [pure virtual]
```

Retrieves the current X-coordinate of the mouse.

## Returns

The X-position of the mouse cursor.

Implemented in [AEventManager](#).

**4.7.2.3 getMouseY()**

```
virtual std::size_t Arcade::IEventManager::getMouseY ( ) const [pure virtual]
```

Retrieves the current Y-coordinate of the mouse.

## Returns

The Y-position of the mouse cursor.

Implemented in [AEventManager](#).

**4.7.2.4 pollEvents()**

```
virtual void Arcade::IEventManager::pollEvents ( ) [pure virtual]
```

Polls for new events.

This method updates the event states by checking for new input from the user.

Implemented in [AEventManager](#).

**4.7.2.5 setEventState()**

```
virtual void Arcade::IEventManager::setEventState (
    int key,
    bool state ) [pure virtual]
```

Sets the state of a specific event.

This method enables or disables a given key or event state.

## Parameters

<i>key</i>	The key/event identifier.
<i>state</i>	The new state of the event (true = active, false = inactive).

Implemented in [AEventManager](#).

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

- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IEventManager.hpp](#)

## 4.8 Arcade::IGameModule Class Reference

Interface for game logic modules.

```
#include <IGameModule.hpp>
```

Collaboration diagram for Arcade::IGameModule:

Arcade::IGameModule
<ul style="list-style-type: none"> <li>+ ~IGameModule()</li> <li>+ init()</li> <li>+ update()</li> <li>+ render()</li> <li>+ getElements()</li> <li>+ handleInput()</li> <li>+ getName()</li> <li>+ getScore()</li> </ul>

### Public Member Functions

- virtual **~IGameModule** ()=default  
*Virtual destructor for the interface.*
- virtual void **init** ()=0  
*Starts the game module.*
- virtual void **update** ()=0  
*Updates the game state.*
- virtual void **render** ()=0  
*Renders the game elements.*
- virtual std::vector< Entity > **getElements** () const =0

- Retrieves the game entities.*
  - virtual void `handleInput` (const `IEventManager` &event)=0
- Handles player input.*
  - virtual std::string `getName` () const =0
- Retrieves the game's name.*
  - virtual int `getScore` () const =0
- Retrieves the game's score.*

### 4.8.1 Detailed Description

Interface for game logic modules.

The `IGameModule` interface provides a set of pure virtual functions that must be implemented by any game module in the Arcade project. It includes methods for starting, updating, rendering, and handling input.

### 4.8.2 Member Function Documentation

#### 4.8.2.1 `getElements()`

```
virtual std::vector< Entity > Arcade::IGameModule::getElements ( ) const [pure virtual]
```

Retrieves the game entities.

This method should return the current entities involved in the game to be processed or displayed.

##### Returns

A map of entity names and their corresponding ECS components.

#### 4.8.2.2 `getName()`

```
virtual std::string Arcade::IGameModule::getName ( ) const [pure virtual]
```

Retrieves the game's name.

This method should return the name of the game module.

##### Returns

The name of the game module.

#### 4.8.2.3 `getScore()`

```
virtual int Arcade::IGameModule::getScore ( ) const [pure virtual]
```

Retrieves the game's score.

This method should return the current score of the game session.

##### Returns

The current score of the game session.

#### 4.8.2.4 handleInput()

```
virtual void Arcade::IGameModule::handleInput (
    const IEventManager & event ) [pure virtual]
```

Handles player input.

This method should process user input and apply the corresponding actions to the game logic.

#### 4.8.2.5 init()

```
virtual void Arcade::IGameModule::init ( ) [pure virtual]
```

Starts the game module.

This method is responsible for initializing the game state and preparing all necessary resources for the game session.

#### 4.8.2.6 render()

```
virtual void Arcade::IGameModule::render ( ) [pure virtual]
```

Renders the game elements.

This method should be responsible for preparing the game's visual elements before they are displayed.

#### 4.8.2.7 update()

```
virtual void Arcade::IGameModule::update ( ) [pure virtual]
```

Updates the game state.

This method should process game logic, update entities, and handle internal state transitions.

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

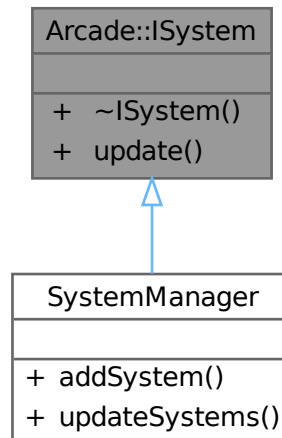
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IGameModule.hpp](#)

## 4.9 Arcade::ISystem Class Reference

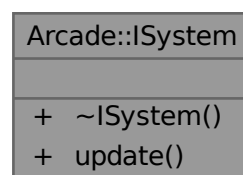
Interface for game systems in the ECS.

```
#include <ISystem.hpp>
```

Inheritance diagram for Arcade::ISystem:



Collaboration diagram for Arcade::ISystem:



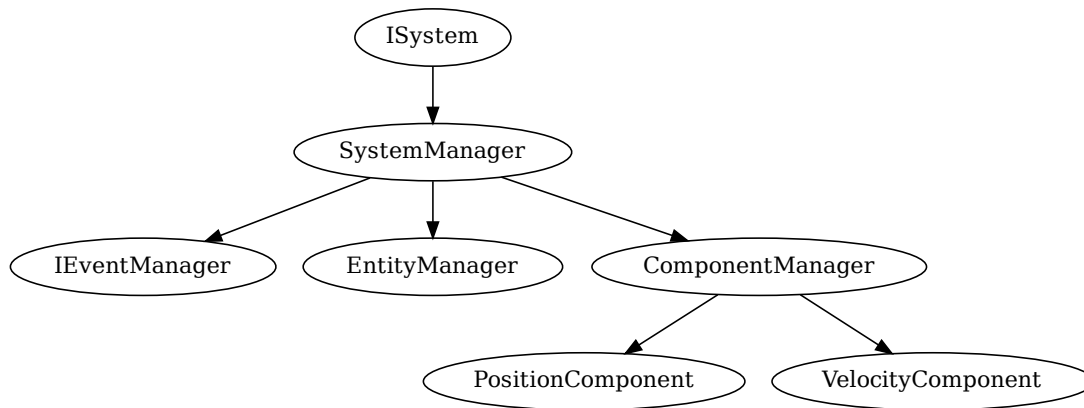
### Public Member Functions

- virtual `~ISystem()`=default  
*Virtual destructor.*
- virtual void `update()`=0  
*Updates the system.*

### 4.9.1 Detailed Description

Interface for game systems in the ECS.

The `ISystem` interface provides a pure virtual method `update()`, which must be implemented by any system managing game logic or entity behavior.



### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 ~ISystem()

```
virtual Arcade::ISystem::~~ISystem ( ) [virtual], [default]
```

Virtual destructor.

Ensures proper cleanup of derived system instances.

### 4.9.3 Member Function Documentation

#### 4.9.3.1 update()

```
virtual void Arcade::ISystem::update ( ) [pure virtual]
```

Updates the system.

This method is meant to be implemented by derived classes to handle system-specific logic that needs to be executed every update cycle.

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

- `/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/ISystem.hpp`

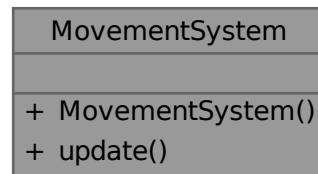


## 4.10 MovementSystem Class Reference

System responsible for updating entity movement.

```
#include <MovementSystem.hpp>
```

Collaboration diagram for MovementSystem:



### Public Member Functions

- [MovementSystem](#) ([ComponentManager](#)< [PositionComponent](#) > \*pm, [ComponentManager](#)< [VelocityComponent](#) > \*vm)  
*Constructs a [MovementSystem](#) with component managers.*
- void [update](#) ()  
*Updates entity positions based on velocity.*

### 4.10.1 Detailed Description

System responsible for updating entity movement.

The [MovementSystem](#) class processes all entities that have both Position and Velocity components, updating their positions based on their velocity values.

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 MovementSystem()

```

MovementSystem::MovementSystem (
    ComponentManager< PositionComponent > * pm,
    ComponentManager< VelocityComponent > * vm ) [inline]
  
```

Constructs a [MovementSystem](#) with component managers.

The system requires pointers to the component managers for Position and Velocity.

## Parameters

<i>pm</i>	Pointer to the <a href="#">PositionComponent</a> manager.
<i>vm</i>	Pointer to the <a href="#">VelocityComponent</a> manager.

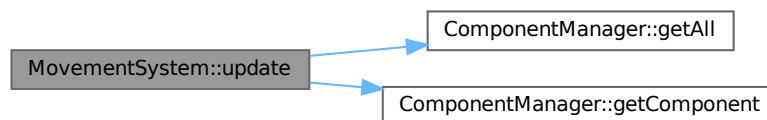
### 4.10.3 Member Function Documentation

#### 4.10.3.1 update()

```
void MovementSystem::update ( )
```

Updates entity positions based on velocity.

This function iterates through all entities that have both Position and Velocity components and updates their positions accordingly. Here is the call graph for this function:



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

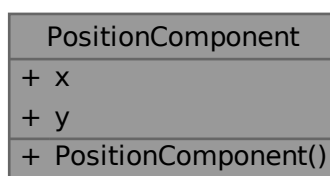
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp](#)
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.cpp](#)

## 4.11 PositionComponent Struct Reference

Represents the position of an entity in a 2D space.

```
#include <Position.hpp>
```

Collaboration diagram for PositionComponent:



### Public Member Functions

- [PositionComponent](#) (int x=0, int y=0)  
*The Y-coordinate of the entity.*

### Public Attributes

- int **x**
- int **y**  
*The X-coordinate of the entity.*

#### 4.11.1 Detailed Description

Represents the position of an entity in a 2D space.

The [PositionComponent](#) structure stores the X and Y coordinates of an entity within the game world.

#### 4.11.2 Constructor & Destructor Documentation

##### 4.11.2.1 PositionComponent()

```
PositionComponent::PositionComponent (
    int x = 0,
    int y = 0 ) [inline], [explicit]
```

The Y-coordinate of the entity.

Constructs a [PositionComponent](#) with optional initial values.

##### Parameters

x	The initial X position (default is 0).
y	The initial Y position (default is 0).

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

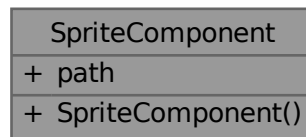
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Position.hpp

## 4.12 SpriteComponent Struct Reference

Represents a graphical sprite associated with an entity.

```
#include <Sprite.hpp>
```

Collaboration diagram for SpriteComponent:



### Public Member Functions

- [SpriteComponent](#) (std::string path)  
*File path to the sprite image.*

### Public Attributes

- std::string **path**

## 4.12.1 Detailed Description

Represents a graphical sprite associated with an entity.

The [SpriteComponent](#) stores the file path to an image that is used as the graphical representation of an entity.

## 4.12.2 Constructor & Destructor Documentation

### 4.12.2.1 SpriteComponent()

```
SpriteComponent::SpriteComponent (  
    std::string path ) [inline], [explicit]
```

File path to the sprite image.

Constructs a [SpriteComponent](#) with a specified image path.

#### Parameters

<i>path</i>	The file path to the sprite image.
-------------	------------------------------------

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

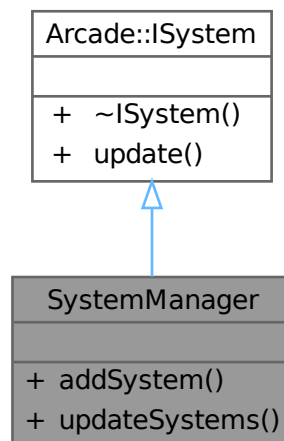
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Sprite.hpp

## 4.13 SystemManager Class Reference

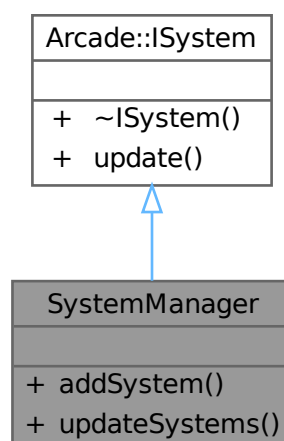
Manages and updates game systems.

```
#include <SystemManager.hpp>
```

Inheritance diagram for SystemManager:



Collaboration diagram for SystemManager:



## Public Member Functions

- void `addSystem` (`Arcade::ISystem *system`)  
*Adds a system to the manager.*
- void `updateSystems` ()  
*Updates all registered systems.*

## Public Member Functions inherited from `Arcade::ISystem`

- virtual `~ISystem` ()=default  
*Virtual destructor.*
- virtual void `update` ()=0  
*Updates the system.*

### 4.13.1 Detailed Description

Manages and updates game systems.

The `SystemManager` class is responsible for managing multiple game systems that implement the `Arcade::ISystem` interface. It provides methods to add systems and update them collectively.

### 4.13.2 Member Function Documentation

#### 4.13.2.1 `addSystem()`

```
void SystemManager::addSystem (
    Arcade::ISystem * system )
```

Adds a system to the manager.

This method stores a pointer to a system that implements the `ISystem` interface.

#### Parameters

<code>system</code>	A pointer to an <code>ISystem</code> instance to be managed.
---------------------	--

#### 4.13.2.2 `updateSystems()`

```
void SystemManager::updateSystems ( )
```

Updates all registered systems.

This method iterates through the stored systems and calls their update function.

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

- `/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.hpp`
- `/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.cpp`

## 4.14 Arcade::Vector2i Struct Reference

Represents a 2D vector with integer coordinates.

```
#include <WindowManager.hpp>
```

Collaboration diagram for Arcade::Vector2i:

Arcade::Vector2i
+ x
+ y
+ Vector2i()

### Public Member Functions

- [Vector2i](#) (int *x*=0, int *y*=0)  
*Constructs a [Vector2i](#) object with default or specified values.*

### Public Attributes

- int *x*  
*X-coordinate or width.*
- int *y*  
*Y-coordinate or height.*

### 4.14.1 Detailed Description

Represents a 2D vector with integer coordinates.

This structure is used for handling 2D positions or sizes.

### 4.14.2 Constructor & Destructor Documentation

#### 4.14.2.1 Vector2i()

```
Arcade::Vector2i::Vector2i (
    int x = 0,
    int y = 0 ) [inline], [explicit]
```

Constructs a [Vector2i](#) object with default or specified values.

**Parameters**

<i>x</i>	The X-coordinate (default is 0).
<i>y</i>	The Y-coordinate (default is 0).

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

- </home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.hpp>

## 4.15 VelocityComponent Struct Reference

Represents an entity's velocity in 2D space.

```
#include <Velocity.hpp>
```

Collaboration diagram for VelocityComponent:

VelocityComponent
+ dx
+ dy
+ VelocityComponent()

**Public Member Functions**

- [VelocityComponent](#) (int dx=0, int dy=0)  
*Change in position along the Y-axis.*

**Public Attributes**

- int **dx**
- int **dy**  
*Change in position along the X-axis.*

### 4.15.1 Detailed Description

Represents an entity's velocity in 2D space.

The [VelocityComponent](#) structure defines movement speed in the X and Y directions. It is used to update an entity's position within the game world.



## 4.15.2 Constructor & Destructor Documentation

### 4.15.2.1 VelocityComponent()

```
VelocityComponent::VelocityComponent (
    int dx = 0,
    int dy = 0 ) [inline], [explicit]
```

Change in position along the Y-axis.

Constructs a [VelocityComponent](#) with optional initial values.

#### Parameters

<i>dx</i>	The initial velocity along the X-axis (default is 0).
<i>dy</i>	The initial velocity along the Y-axis (default is 0).

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

- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Velocity.hpp

## 4.16 Arcade::WindowManager Class Reference

Handles window properties such as size, title, frame rate, and icon.

```
#include <WindowManager.hpp>
```

Collaboration diagram for Arcade::WindowManager:

Arcade::WindowManager
<ul style="list-style-type: none"> <li>+ getHeight()</li> <li>+ getWidth()</li> <li>+ getFps()</li> <li>+ getTitle()</li> <li>+ getIconPath()</li> <li>+ getWindowSize()</li> <li>+ setHeight()</li> <li>+ setWidth()</li> <li>+ setFps()</li> <li>+ setTitle()</li> <li>+ setIconPath()</li> <li>+ setWindowSize()</li> </ul>

## Public Member Functions

- `int getHeight () const`  
*Retrieves the height of the window.*
- `int getWidth () const`  
*Retrieves the width of the window.*
- `float getFps () const`  
*Retrieves the frames per second (FPS) setting.*
- `std::string getTitle () const`  
*Retrieves the window title.*
- `std::string getIconPath () const`  
*Retrieves the file path of the window icon.*
- `Vector2i getWindowSize () const`  
*Retrieves the size of the window.*
- `void setHeight (int height)`  
*Sets the height of the window.*
- `void setWidth (int width)`  
*Sets the width of the window.*
- `void setFps (float fps)`  
*Sets the frames per second (FPS) value.*
- `void setTitle (const std::string &title)`  
*Sets the title of the window.*
- `void setIconPath (const std::string &iconPath)`  
*Sets the file path of the window icon.*
- `void setWindowSize (Vector2i vec)`  
*Sets the window size.*

### 4.16.1 Detailed Description

Handles window properties such as size, title, frame rate, and icon.

The [WindowManager](#) class provides getter and setter methods for managing window dimensions, FPS settings, window title, and associated icon.

### 4.16.2 Member Function Documentation

#### 4.16.2.1 getFps()

```
float Arcade::WindowManager::getFps ( ) const
```

Retrieves the frames per second (FPS) setting.

#### Returns

The current FPS value.

#### 4.16.2.2 getHeight()

```
int Arcade::WindowManager::getHeight ( ) const
```

Retrieves the height of the window.

##### Returns

The window height in pixels.

#### 4.16.2.3 getIconPath()

```
std::string Arcade::WindowManager::getIconPath ( ) const
```

Retrieves the file path of the window icon.

##### Returns

A string representing the path to the icon file.

#### 4.16.2.4 getTitle()

```
std::string Arcade::WindowManager::getTitle ( ) const
```

Retrieves the window title.

##### Returns

A string containing the window title.

#### 4.16.2.5 getWidth()

```
int Arcade::WindowManager::getWidth ( ) const
```

Retrieves the width of the window.

##### Returns

The window width in pixels.

#### 4.16.2.6 getWindowSize()

```
Arcade::Vector2i Arcade::WindowManager::getWindowSize ( ) const
```

Retrieves the size of the window.

##### Returns

A [Vector2i](#) representing the window dimensions (width, height).

#### 4.16.2.7 setFps()

```
void Arcade::WindowManager::setFps (
    float fps )
```

Sets the frames per second (FPS) value.

**Parameters**

<i>fps</i>	The new FPS value.
------------	--------------------

**4.16.2.8 setHeight()**

```
void Arcade::WindowManager::setHeight (
    int height )
```

Sets the height of the window.

**Parameters**

<i>height</i>	The new height value in pixels.
---------------	---------------------------------

**4.16.2.9 setIconPath()**

```
void Arcade::WindowManager::setIconPath (
    const std::string & iconPath )
```

Sets the file path of the window icon.

**Parameters**

<i>iconPath</i>	A string representing the path to the new icon file.
-----------------	--

**4.16.2.10 setTitle()**

```
void Arcade::WindowManager::setTitle (
    const std::string & title )
```

Sets the title of the window.

**Parameters**

<i>title</i>	A string containing the new window title.
--------------	---

**4.16.2.11 setWidth()**

```
void Arcade::WindowManager::setWidth (
    int width )
```

Sets the width of the window.

## Parameters

<i>width</i>	The new width value in pixels.
--------------	--------------------------------

**4.16.2.12 setWindowSize()**

```
void Arcade::WindowManager::setWindowSize (
    Vector2i vec )
```

Sets the window size.

## Parameters

<i>vec</i>	A <a href="#">Vector2i</a> representing the new window dimensions (width, height).
------------	--

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

- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.hpp](#)
- [/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.cpp](#)



# Chapter 5

## File Documentation

### 5.1 core.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003  ** EPITECH PROJECT, 2025
00004  ** Arcade
00005  ** File description:
00006  ** core
00007  */
00008
00009 #ifndef SRC_CORE_CORE_HPP_
00010     #define SRC_CORE_CORE_HPP_
00011     #include <memory>
00012     #include <string>
00013     #include "../Shared/Interface/IGameModule.hpp"
00014     #include "../Shared/Interface/IDisplayModule.hpp"
00015
00016 namespace Arcade {
00017     class Core {
00018     public:
00019         Core() : _currentGame(nullptr), _currentDisplay(nullptr) {}
00020         ~Core() {}
00021         void run();
00022         void loadGame(const std::string &gameName);
00023         void loadDisplay(const std::string &displayName);
00024         void unloadGame();
00025         void unloadDisplay();
00026         void printHelp();
00027
00028     private:
00029         std::unique_ptr<IGameModule> _currentGame;
00030         std::unique_ptr<IDisplayModule> _currentDisplay;
00031     };
00032 } // namespace Arcade
00033 #endif // SRC_CORE_CORE_HPP_
```

### 5.2 DLloader.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400 Arcade
00005  ** File description:
00006  ** DLoader class
00007  */
00008
00009 #ifndef SRC_CORE_DLLOADER_DLLOADER_HPP_
00010     #define SRC_CORE_DLLOADER_DLLOADER_HPP_
00011
00012 #endif // SRC_CORE_DLLOADER_DLLOADER_HPP_
```

## 5.3 GameLoop.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400 Arcade
00005 ** File description:
00006 ** GameLoop class
00007 */
00008
00009 #ifndef SRC_CORE_GAMELOOP_GAMELOOP_HPP_
00010     #define SRC_CORE_GAMELOOP_GAMELOOP_HPP_
00011
00012 #endif // SRC_CORE_GAMELOOP_GAMELOOP_HPP_
```

## 5.4 Game1.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400 Arcade
00005 ** File description:
00006 ** Game1 class
00007 */
00008
00009 #ifndef SRC_GAMES_GAME1_GAME1_HPP_
00010     #define SRC_GAMES_GAME1_GAME1_HPP_
00011
00012     class Game1 {
00013     public:
00014         Game1() = default;
00015         ~Game1() = default;
00016         void printHelp();
00017     };
00018
00019 #endif // SRC_GAMES_GAME1_GAME1_HPP_
```

## 5.5 Game2.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400 Arcade
00005 ** File description:
00006 ** Game2
00007 */
00008
00009 #ifndef SRC_GAMES_GAME2_GAME2_HPP_
00010     #define SRC_GAMES_GAME2_GAME2_HPP_
00011
00012 #endif // SRC_GAMES_GAME2_GAME2_HPP_
```

## 5.6 Game3.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400 Arcade
00005 ** File description:
00006 ** Game3 class
00007 */
00008 #ifndef SRC_GAMES_GAME3_GAME3_HPP_
00009     #define SRC_GAMES_GAME3_GAME3_HPP_
00010
00011 #endif // SRC_GAMES_GAME3_GAME3_HPP_
```

## 5.7 nCurses.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
```



```
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400 Arcade
00005  ** File description:
00006  ** nCurses class
00007  */
00008
00009  #ifndef SRC_GRAPHICS_NCURSES_NCURSES_HPP_
00010      #define SRC_GRAPHICS_NCURSES_NCURSES_HPP_
00011
00012  #endif  // SRC_GRAPHICS_NCURSES_NCURSES_HPP_
```

## 5.8 SDL.hpp

```
00001  // Copyright 2025 <Epitech>
00002  /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400 Arcade
00005  ** File description:
00006  ** SDL class
00007  */
00008
00009  #ifndef SRC_GRAPHICS_SDL_SDL_HPP_
00010      #define SRC_GRAPHICS_SDL_SDL_HPP_
00011
00012  #endif  // SRC_GRAPHICS_SDL_SDL_HPP_
```

## 5.9 SFML.hpp

```
00001  // Copyright 2025 <Epitech>
00002  /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400 Arcade
00005  ** File description:
00006  ** SFML class
00007  */
00008
00009  #ifndef SRC_GRAPHICS_SFML_SFML_HPP_
00010      #define SRC_GRAPHICS_SFML_SFML_HPP_
00011
00012  #endif  // SRC_GRAPHICS_SFML_SFML_HPP_
```

## 5.10 Position.hpp

```
00001  // Copyright 2025 <Epitech>
00002  /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400
00005  ** File description:
00006  ** Position Component
00007  */
00016  #ifndef SRC_SHARED_COMPONENTS_POSITION_HPP_
00017      #define SRC_SHARED_COMPONENTS_POSITION_HPP_
00018
00026  struct PositionComponent {
00027      int x;
00028      int y;
00029
00036      explicit PositionComponent(int x = 0, int y = 0) : x(x), y(y) {}
00037  };
00038
00039  #endif  // SRC_SHARED_COMPONENTS_POSITION_HPP_
```

## 5.11 Sprite.hpp

```
00001  // Copyright 2025 <Epitech>
00002  /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400
00005  ** File description:
00006  ** Position Component
00007  */
```

```

00016 #ifndef SRC_SHARED_COMPONENTS_SPRITE_HPP_
00017 #define SRC_SHARED_COMPONENTS_SPRITE_HPP_
00018 #include <utility>
00019 #include <string>
00020
00028 struct SpriteComponent {
00029     std::string path;
00035     explicit SpriteComponent(std::string path) : path(std::move(path)) {}
00036 };
00037
00038 #endif // SRC_SHARED_COMPONENTS_SPRITE_HPP_

```

## 5.12 Velocity.hpp

```

00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400
00005 ** File description:
00006 ** Position Component
00007 */
00016 #ifndef SRC_SHARED_COMPONENTS_VELOCITY_HPP_
00017 #define SRC_SHARED_COMPONENTS_VELOCITY_HPP_
00018
00026 struct VelocityComponent {
00027     int dx;
00028     int dy;
00029
00036     explicit VelocityComponent(int dx = 0, int dy = 0) : dx(dx), dy(dy) {}
00037 };
00038
00039 #endif // SRC_SHARED_COMPONENTS_VELOCITY_HPP_

```

## 5.13 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.hpp File Reference

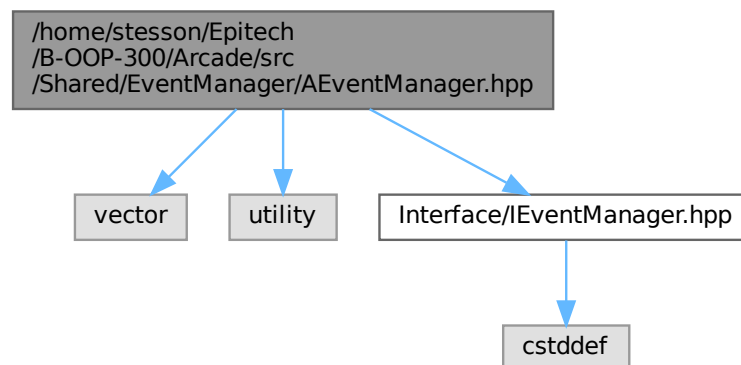
Abstract class for event management in the Arcade project.

```

#include <vector>
#include <utility>
#include "Interface/IEventManager.hpp"

```

Include dependency graph for AEventManager.hpp:



## Classes

- class [AEventManager](#)

*Abstract base class for event management.*

### 5.13.1 Detailed Description

Abstract class for event management in the Arcade project.

The [AEventManager](#) class provides a base implementation for event handling, including keyboard event states and mouse position tracking.

## 5.14 AEventManager.hpp

[Go to the documentation of this file.](#)

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** Arcade
00005 ** File description:
00006 ** AEventManager abstract class
00007 */
00008
00009 // Copyright 2025 <Epitech>
00010 #pragma once
00011
00012 #include <vector>
00013 #include <utility>
00014 #include "Interface/IEventManager.hpp"
00015
00016 class AEventManager : public Arcade::IEventManager {
00017 public:
00018     AEventManager();
00019
00020     virtual ~AEventManager() = default;
00021
00022     bool getEventState(int key) override;
00023
00024     void setEventState(int key, bool state) override;
00025
00026     std::size_t getMouseX() const override;
00027
00028     std::size_t getMouseY() const override;
00029
00030     virtual void pollEvents() = 0;
00031
00032 protected:
00033     void setMouseX(std::size_t x);
00034
00035     void setMouseY(std::size_t y);
00036
00037 private:
00038     std::vector<std::pair<int, bool>> _keyEvents;
00039     std::size_t _mouseX;
00040     std::size_t _mouseY;
00041 };
```

## 5.15 IDisplayModule.hpp

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** Paradigm pool
00005 ** File description:
00006 ** Seminar
00007 */
00008
00009 #ifndef SRC_SHARED_INTERFACE_IDISPLAYMODULE_HPP_
00010 #define SRC_SHARED_INTERFACE_IDISPLAYMODULE_HPP_
```

```

00011     #include <string>
00012     #include <vector>
00013     #include "Shared/Models/EntityComponentSystem/EntityManager.hpp"
00014     #include "Interface/IEventManager.hpp"
00015
00016 namespace Arcade {
00017 class IDisplayModule {
00018 public:
00019     virtual ~IDisplayModule() = default;
00020     virtual void init() = 0;
00021     virtual IEventManager getInput(void) = 0;
00022     virtual void drawElement(const std::vector<Entity> &entities) = 0;
00023     virtual void render() = 0;
00024     virtual std::string getName() const = 0;
00025 };
00026 } // namespace Arcade
00027 #endif // SRC_SHARED_INTERFACE_IDISPLAYMODULE_HPP_

```

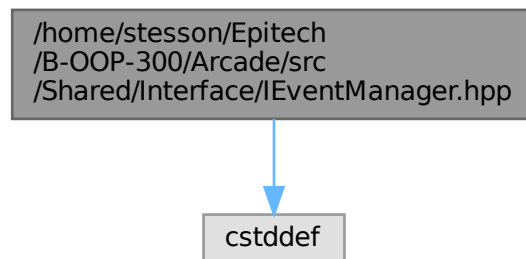
## 5.16 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IEventManager.hpp ↩

### IEventManager.hpp File Reference

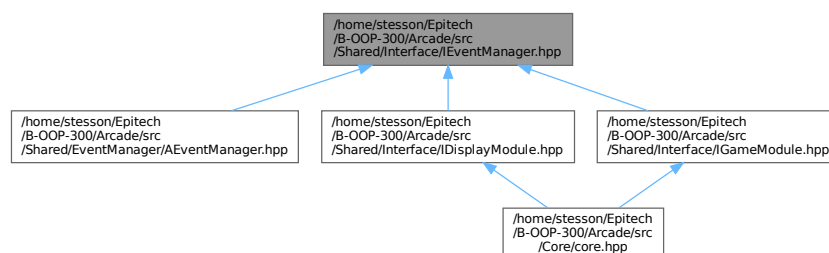
Interface for managing user input events in the Arcade project.

```
#include <cstddef>
```

Include dependency graph for IEventManager.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Arcade::IEventManager](#)  
*Interface for event management.*

## Macros

- **#define KEY\_UP 0**  
*Key definitions for user input handling.*
- **#define KEY\_DOWN 1**
- **#define KEY\_LEFT 2**
- **#define KEY\_RIGHT 3**
- **#define KEY\_A 4**
- **#define KEY\_B 5**
- **#define KEY\_C 6**
- **#define KEY\_D 7**
- **#define KEY\_E 8**
- **#define KEY\_F 9**
- **#define KEY\_G 10**
- **#define KEY\_H 11**
- **#define KEY\_I 12**
- **#define KEY\_J 13**
- **#define KEY\_K 14**
- **#define KEY\_L 15**
- **#define KEY\_M 16**
- **#define KEY\_N 17**
- **#define KEY\_O 18**
- **#define KEY\_P 19**
- **#define KEY\_Q 20**
- **#define KEY\_R 21**
- **#define KEY\_S 22**
- **#define KEY\_T 23**
- **#define KEY\_U 24**
- **#define KEY\_V 25**
- **#define KEY\_W 26**
- **#define KEY\_X 27**
- **#define KEY\_Y 28**
- **#define KEY\_Z 29**
- **#define KEY\_0 30**
- **#define KEY\_1 31**
- **#define KEY\_2 32**
- **#define KEY\_3 33**
- **#define KEY\_4 34**
- **#define KEY\_5 35**
- **#define KEY\_6 36**
- **#define KEY\_7 37**
- **#define KEY\_8 38**
- **#define KEY\_9 39**
- **#define KEY\_SPACE 40**
- **#define KEY\_ENTER 41**
- **#define KEY\_ESCAPE 42**
- **#define KEY\_BACKSPACE 43**
- **#define KEY\_TAB 44**
- **#define KEY\_SHIFT 45**

- `#define KEY_CTRL` 46
- `#define KEY_ALT` 47
- `#define KEY_SUPER` 48
- `#define LEFT_CLICK` 49
- `#define RIGHT_CLICK` 50
- `#define MIDDLE_CLICK` 51
- `#define WINDOW_CLOSE` 52

### 5.16.1 Detailed Description

Interface for managing user input events in the Arcade project.

The `IEventManager` interface defines a set of methods for handling keyboard and mouse events, including retrieving event states and polling for new events.

## 5.17 IEventManager.hpp

[Go to the documentation of this file.](#)

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003  ** EPITECH PROJECT, 2025
00004  ** Arcade
00005  ** File description:
00006  ** IEventManager interface
00007  */
00008 // Copyright 2025 <Epitech>
00017 #pragma once
00018 #include <cstdint>
00019
00021 #define KEY_UP 0
00022 #define KEY_DOWN 1
00023 #define KEY_LEFT 2
00024 #define KEY_RIGHT 3
00025 #define KEY_A 4
00026 #define KEY_B 5
00027 #define KEY_C 6
00028 #define KEY_D 7
00029 #define KEY_E 8
00030 #define KEY_F 9
00031 #define KEY_G 10
00032 #define KEY_H 11
00033 #define KEY_I 12
00034 #define KEY_J 13
00035 #define KEY_K 14
00036 #define KEY_L 15
00037 #define KEY_M 16
00038 #define KEY_N 17
00039 #define KEY_O 18
00040 #define KEY_P 19
00041 #define KEY_Q 20
00042 #define KEY_R 21
00043 #define KEY_S 22
00044 #define KEY_T 23
00045 #define KEY_U 24
00046 #define KEY_V 25
00047 #define KEY_W 26
00048 #define KEY_X 27
00049 #define KEY_Y 28
00050 #define KEY_Z 29
00051 #define KEY_0 30
00052 #define KEY_1 31
00053 #define KEY_2 32
00054 #define KEY_3 33
00055 #define KEY_4 34
00056 #define KEY_5 35
00057 #define KEY_6 36
00058 #define KEY_7 37
00059 #define KEY_8 38
00060 #define KEY_9 39
00061 #define KEY_SPACE 40
00062 #define KEY_ENTER 41
```

```

00063 #define KEY_ESCAPE 42
00064 #define KEY_BACKSPACE 43
00065 #define KEY_TAB 44
00066 #define KEY_SHIFT 45
00067 #define KEY_CTRL 46
00068 #define KEY_ALT 47
00069 #define KEY_SUPER 48
00070 #define LEFT_CLICK 49
00071 #define RIGHT_CLICK 50
00072 #define MIDDLE_CLICK 51
00073 #define WINDOW_CLOSE 52
00074
00075 namespace Arcade {
00076
00085 class IEventManager {
00086 public:
00090     virtual ~IEventManager() = default;
00091
00100     virtual bool getEventState(int key) = 0;
00101
00110     virtual void setEventState(int key, bool state) = 0;
00111
00117     virtual std::size_t getMouseX() const = 0;
00118
00124     virtual std::size_t getMouseY() const = 0;
00125
00131     virtual void pollEvents() = 0;
00132 };
00133
00134 } // namespace Arcade

```

## 5.18 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IGameModule.hpp File Reference

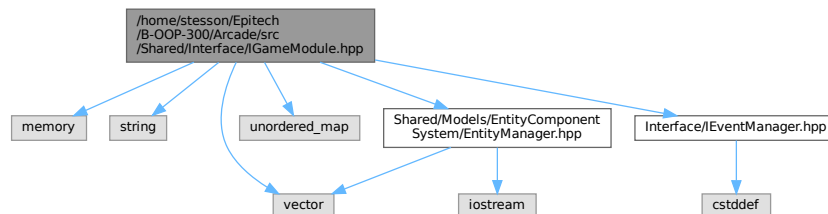
Interface for game modules in the Arcade project.

```

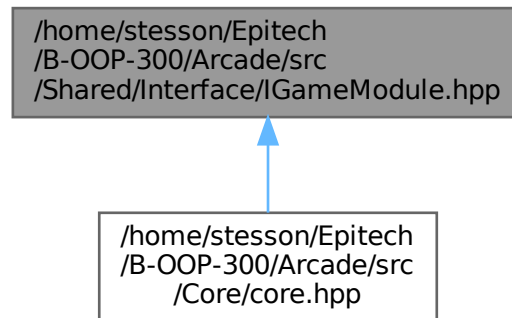
#include <memory>
#include <string>
#include <vector>
#include <unordered_map>
#include "Shared/Models/EntityComponentSystem/EntityManager.hpp"
#include "Interface/IEventManager.hpp"

```

Include dependency graph for IGameModule.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Arcade::IGameModule](#)  
*Interface for game logic modules.*

### 5.18.1 Detailed Description

Interface for game modules in the Arcade project.

This interface defines the necessary methods for implementing a game module in the Arcade project.

## 5.19 IGameModule.hpp

[Go to the documentation of this file.](#)

```

00001 // Copyright 2025 <Epitech>
00002 /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400 Arcade
00005  ** File description:
00006  ** IGameModule interface
00007  */
00016 #ifndef SRC_SHARED_INTERFACE_IGAMEMODULE_HPP_
00017     #define SRC_SHARED_INTERFACE_IGAMEMODULE_HPP_
00018     #include <memory>
00019     #include <string>
00020     #include <vector>
00021     #include <unordered_map>
00022     #include "Shared/Models/EntityComponentSystem/EntityManager.hpp"
00023     #include "Interface/IEventManager.hpp"
00024
00025 namespace Arcade {
00034 class IGameModule {
00035 public:
00039     virtual ~IGameModule() = default;
00046     virtual void init() = 0;
00047
00054     virtual void update() = 0;
00055
00062     virtual void render() = 0;
00063
00071     virtual std::vector<Entity> getElements() const = 0;
  
```



```

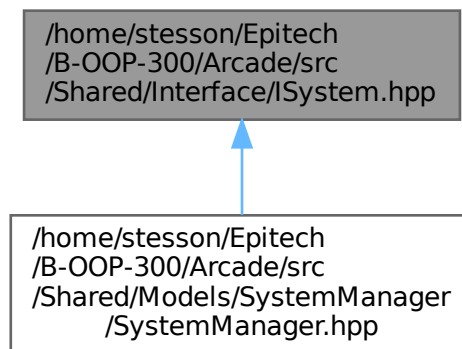
00078     virtual void handleInput(const IEventManager &event) = 0;
00079
00086     virtual std::string getName() const = 0;
00087
00094     virtual int getScore() const = 0;
00095 };
00096 } // namespace Arcade
00097 #endif // SRC_SHARED_INTERFACE_IGAMEMODULE_HPP_

```

## 5.20 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/ISystem.hpp File Reference ↩

Interface for systems in the Entity Component System (ECS) architecture.

This graph shows which files directly or indirectly include this file:



### Classes

- class [Arcade::ISystem](#)  
*Interface for game systems in the ECS.*

### 5.20.1 Detailed Description

Interface for systems in the Entity Component System (ECS) architecture.

The ISystem interface defines a standard structure for all systems that operate within the ECS of the Arcade project.

## 5.21 ISystem.hpp

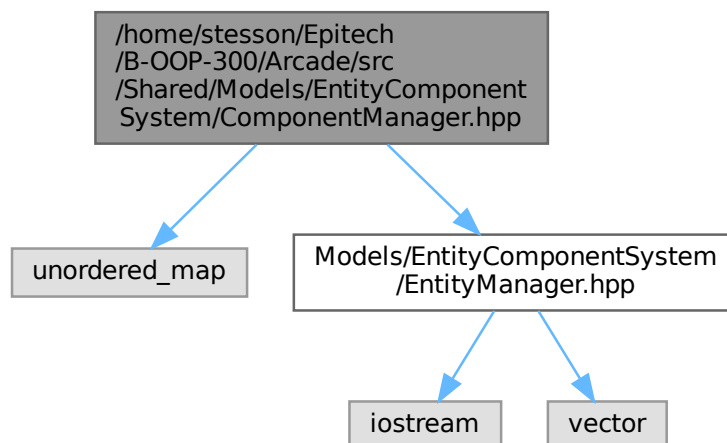
[Go to the documentation of this file.](#)

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400
00005 ** File description:
00006 ** System Interface
00007 */
00008
00009 // Copyright 2025 <Epitech>
00018 #ifndef SRC_SHARED_INTERFACE_ISYSTEM_HPP_
00019     #define SRC_SHARED_INTERFACE_ISYSTEM_HPP_
00020
00021 namespace Arcade {
00041 class ISystem {
00042 public:
00048     virtual ~ISystem() = default;
00049
00057     virtual void update() = 0;
00058 };
00059 } // namespace Arcade
00060
00061 #endif // SRC_SHARED_INTERFACE_ISYSTEM_HPP_
```

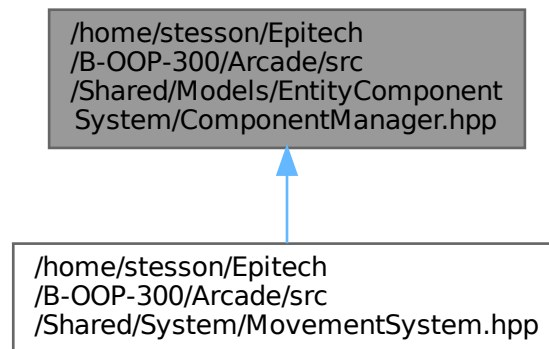
## 5.22 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/ComponentManager.hpp File Reference

Manages components for entities in an Entity Component System (ECS).

```
#include <unordered_map>
#include "Models/EntityComponentSystem/EntityManager.hpp"
Include dependency graph for ComponentManager.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `ComponentManager< T >`  
*Manages components of a specific type for entities.*

### 5.22.1 Detailed Description

Manages components for entities in an Entity Component System (ECS).

The `ComponentManager` class is a template-based system that allows entities to have associated components, enabling efficient storage and retrieval within an ECS architecture.

## 5.23 ComponentManager.hpp

[Go to the documentation of this file.](#)

```

00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400 Arcade
00005 ** File description:
00006 ** Component Manager
00007 */
00008
00009 // Copyright 2025 <Epitech>
00010 #ifndef SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_COMPONENTMANAGER_HPP_
00011 #define SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_COMPONENTMANAGER_HPP_
00012 #include <unordered_map>
00013 #include "Models/EntityComponentSystem/EntityManager.hpp"
00014
00015 template <typename T>
00016 class ComponentManager {
00017 private:
00018     std::unordered_map<Entity, T> components;
00019
00020 public:
00021     void addComponent(Entity entity, T component);
00022     void removeComponent(Entity entity);
00023     T* getComponent(Entity entity);
00024     std::unordered_map<Entity, T>& getAll();
00025 };
00026 #endif // SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_COMPONENTMANAGER_HPP_

```

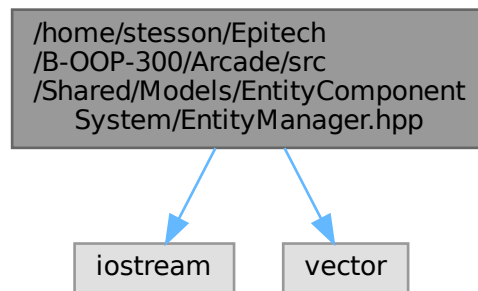
## 5.24 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp File Reference

Manages entity creation and deletion in an Entity Component System (ECS).

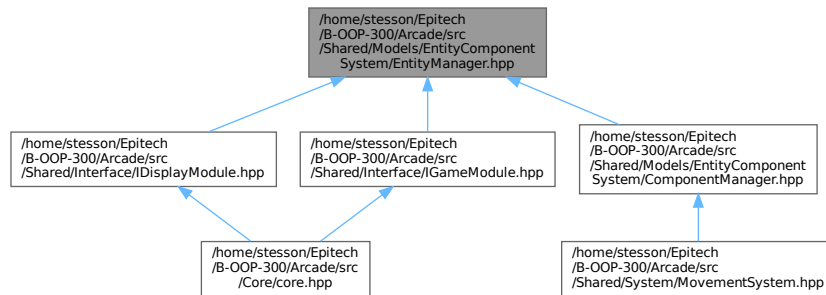
```
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for EntityManager.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

- class [EntityManager](#)  
*Alias for entity identifiers.*

### Typedefs

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

### 5.24.1 Detailed Description

Manages entity creation and deletion in an Entity Component System (ECS).

The [EntityManager](#) class is responsible for generating unique entity identifiers and tracking active entities.

## 5.25 EntityManager.hpp

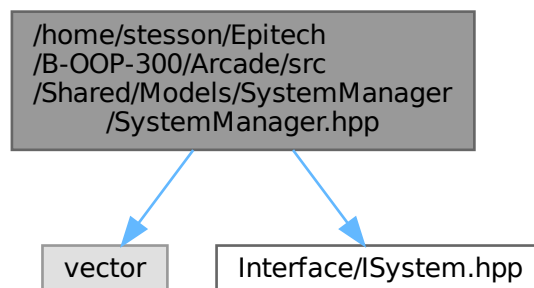
[Go to the documentation of this file.](#)

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400
00005  ** File description:
00006  ** Entity manager
00007  */
00008 // Copyright 2025 <Epitech>
00017 #ifndef SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_ENTITYMANAGER_HPP_
00018     #define SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_ENTITYMANAGER_HPP_
00019     #include <iostream>
00020     #include <vector>
00021
00022     using Entity = std::size_t;
00030     class EntityManager {
00031     public:
00039         Entity createEntity();
00040
00048         void destroyEntity(Entity entity);
00049
00055         const std::vector<Entity>& getEntities() const;
00056
00057     private:
00058         Entity _nextEntityId = 0;
00059         std::vector<Entity> _activeEntities;
00060     };
00061
00062 #endif // SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_ENTITYMANAGER_HPP_
```

## 5.26 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.hpp File Reference

Manages multiple systems in an Entity Component System (ECS).

```
#include <vector>
#include "Interface/ISystem.hpp"
Include dependency graph for SystemManager.hpp:
```



## Classes

- class [SystemManager](#)  
*Manages and updates game systems.*

### 5.26.1 Detailed Description

Manages multiple systems in an Entity Component System (ECS).

The [SystemManager](#) class handles the addition and updating of systems that implement the `ISystem` interface.

## 5.27 SystemManager.hpp

[Go to the documentation of this file.](#)

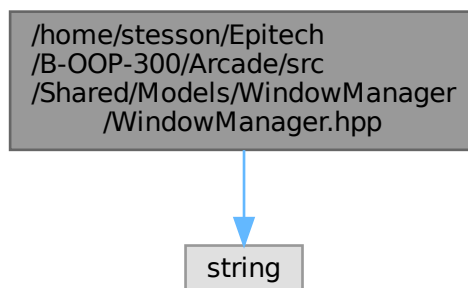
```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400 Arcade
00005 ** File description:
00006 ** System Manager
00007 */
00016 #ifndef SRC_SHARED_MODELS_SYSTEMMANAGER_SYSTEMMANAGER_HPP_
00017 #define SRC_SHARED_MODELS_SYSTEMMANAGER_SYSTEMMANAGER_HPP_
00018     #include <vector>
00019     #include "Interface/ISystem.hpp"
00020
00029 class SystemManager : public Arcade::ISystem {
00030 public:
00038     void addSystem(Arcade::ISystem* system);
00044     void updateSystems();
00045
00046 private:
00047     std::vector<Arcade::ISystem*> _systems;
00048 };
00049
00050 #endif // SRC_SHARED_MODELS_SYSTEMMANAGER_SYSTEMMANAGER_HPP_
```

## 5.28 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.hpp File Reference

Manages window properties for the Arcade project.

```
#include <string>
```

Include dependency graph for `WindowManager.hpp`:



**Classes**

- struct [Arcade::Vector2i](#)  
*Represents a 2D vector with integer coordinates.*
- class [Arcade::WindowManager](#)  
*Handles window properties such as size, title, frame rate, and icon.*

**5.28.1 Detailed Description**

Manages window properties for the Arcade project.

The WindowManager class handles various attributes of a graphical window, such as its size, title, frame rate, and icon.

**5.29 WindowManager.hpp**

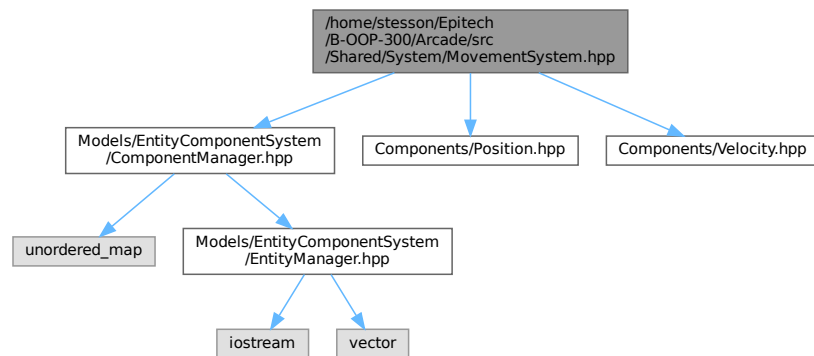
[Go to the documentation of this file.](#)

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003 ** EPITECH PROJECT, 2025
00004 ** B-OOP-400-NAN-4-1-arcade-marin.lamy
00005 ** File description:
00006 ** windowManager
00007 */
00016 #ifndef SRC_SHARED_MODELS_WINDOWMANAGER_WINDOWMANAGER_HPP_
00017     #define SRC_SHARED_MODELS_WINDOWMANAGER_WINDOWMANAGER_HPP_
00018     #include <string>
00019
00020 namespace Arcade {
00027 struct Vector2i {
00028     int x;
00029     int y;
00036     explicit Vector2i(int x = 0, int y = 0) : x(x), y(y) {}
00037 };
00038
00046 class WindowManager {
00047 public:
00052     int getHeight() const;
00053
00058     int getWidth() const;
00059
00064     float getFps() const;
00065
00070     std::string getTitle() const;
00071
00076     std::string getIconPath() const;
00077
00082     Vector2i getWindowSize() const;
00083
00088     void setHeight(int height);
00089
00094     void setWidth(int width);
00095
00100     void setFps(float fps);
00101
00106     void setTitle(const std::string &title);
00107
00112     void setIconPath(const std::string &iconPath);
00113
00118     void setWindowSize(Vector2i vec);
00119
00120 private:
00121     int _width;
00122     int _height;
00123     float _fps;
00124     std::string _title;
00125     std::string _iconPath;
00126 };
00127
00128 } // namespace Arcade
00129
00130 #endif // SRC_SHARED_MODELS_WINDOWMANAGER_WINDOWMANAGER_HPP_
```

## 5.30 /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp File Reference

System for handling entity movement in an ECS architecture.

```
#include "Models/EntityComponentSystem/ComponentManager.hpp"
#include "Components/Position.hpp"
#include "Components/VelocityEngine"
Include dependency graph for MovementSystem.hpp:
```



### Classes

- class [MovementSystem](#)  
*System responsible for updating entity movement.*

### 5.30.1 Detailed Description

System for handling entity movement in an ECS architecture.

The [MovementSystem](#) class is responsible for updating entity positions based on their velocity components within an Entity Component System (ECS).

## 5.31 MovementSystem.hpp

[Go to the documentation of this file.](#)

```
00001 // Copyright 2025 <Epitech>
00002 /*
00003  ** EPITECH PROJECT, 2025
00004  ** B-OOP-400
00005  ** File description:
00006  ** MovementSystem
00007  */
00016 #ifndef SRC_SHARED_SYSTEM_MOVEMENTSYSTEM_HPP_
00017     #define SRC_SHARED_SYSTEM_MOVEMENTSYSTEM_HPP_
00018     #include "Models/EntityComponentSystem/ComponentManager.hpp"
00019     #include "Components/Position.hpp"
00020     #include "Components/VelocityEngine"
00021
00030 class MovementSystem {
```



```
00031 public:
00040     MovementSystem(ComponentManager<PositionComponent>* pm,
00041                     ComponentManager<VelocityComponent>* vm)
00042         : positionManager(pm), velocityManager(vm) {}
00043
00050     void update();
00051
00052 private:
00053     ComponentManager<PositionComponent>* positionManager;
00055     ComponentManager<VelocityComponent>* velocityManager;
00057 };
00058
00059 #endif // SRC_SHARED_SYSTEM_MOVEMENTSYSTEM_HPP_
```



# Index

[/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/DLLoader/DLLoaderManager, 32](#)  
[41](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/GameLoop/GameLoopManager, 9](#)  
[42](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Core/core.hpp, 9](#)  
[41](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game1/Game1.hpp, 10](#)  
[42](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game2/Game2.hpp, 10](#)  
[42](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game2/Game2.hpp, 11](#)  
[42](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Games/Game2/Game2.hpp, 15](#)  
[42](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Graphics/SFML/SFML.hpp, 19](#)  
[43](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Graphics/SFML/SFML.hpp, 21](#)  
[43](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Graphics/nCurses/nCurses.hpp, 21](#)  
[42](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/ComponentManager/ComponentManager, 22](#)  
[43](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/ComponentManager/ComponentManager, 23](#)  
[43](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/ComponentManager/ComponentManager, 23](#)  
[44](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/ComponentManager/ComponentManager, 24](#)  
[44, 45](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/Interface, 25](#)  
[45](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/Interface, 26](#)  
[46, 48](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/Interface, 33](#)  
[49, 50](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/Interface, 35](#)  
[51, 52](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp, 37](#)  
[52, 53](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp, 37](#)  
[54, 55](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp, 37](#)  
[55, 56](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp, 38](#)  
[56, 57](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/EntityManager.hpp, 38](#)  
[58](#)  
[/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/EntityManager, 39](#)  
[58](#)  
[~ISystem](#)  
[Arcade::ISystem, 26](#)  
[addComponent](#)  
[ComponentManager< T >, 13](#)  
[addSystem](#)  
[ComponentManager< T >, 11](#)  
[addComponent, 13](#)  
[getAll, 13](#)  
[getComponent, 14](#)  
[removeComponent, 14](#)

- createEntity
  - EntityManager, 16
- destroyEntity
  - EntityManager, 16
- EntityManager, 16
  - createEntity, 16
  - destroyEntity, 16
  - getEntities, 17
- Game1, 17
- getAll
  - ComponentManager< T >, 13
- getComponent
  - ComponentManager< T >, 14
- getElements
  - Arcade::IGameModule, 23
- getEntities
  - EntityManager, 17
- getEventState
  - AEventManager, 9
  - Arcade::IEventManager, 20
- getFps
  - Arcade::WindowManager, 36
- getHeight
  - Arcade::WindowManager, 36
- getIconPath
  - Arcade::WindowManager, 37
- getMouseX
  - AEventManager, 9
  - Arcade::IEventManager, 21
- getMouseY
  - AEventManager, 10
  - Arcade::IEventManager, 21
- getName
  - Arcade::IGameModule, 23
- getScore
  - Arcade::IGameModule, 23
- getTitle
  - Arcade::WindowManager, 37
- getWidth
  - Arcade::WindowManager, 37
- getWindowSize
  - Arcade::WindowManager, 37
- handleInput
  - Arcade::IGameModule, 23
- init
  - Arcade::IGameModule, 24
- MovementSystem, 27
  - MovementSystem, 27
  - update, 28
- pollEvents
  - AEventManager, 10
  - Arcade::IEventManager, 21
- PositionComponent, 28
  - PositionComponent, 29
- removeComponent
  - ComponentManager< T >, 14
- render
  - Arcade::IGameModule, 24
- setEventState
  - AEventManager, 10
  - Arcade::IEventManager, 21
- setFps
  - Arcade::WindowManager, 37
- setHeight
  - Arcade::WindowManager, 38
- setIconPath
  - Arcade::WindowManager, 38
- setMouseX
  - AEventManager, 10
- setMouseY
  - AEventManager, 11
- setTitle
  - Arcade::WindowManager, 38
- setWidth
  - Arcade::WindowManager, 38
- setWindowSize
  - Arcade::WindowManager, 39
- SpriteComponent, 29
  - SpriteComponent, 30
- SystemManager, 31
  - addSystem, 32
  - updateSystems, 32
- update
  - Arcade::IGameModule, 24
  - Arcade::ISystem, 26
  - MovementSystem, 28
- updateSystems
  - SystemManager, 32
- Vector2i
  - Arcade::Vector2i, 33
- VelocityComponent, 34
  - VelocityComponent, 35