Arcade

Generated by Doxygen 1.9.8

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

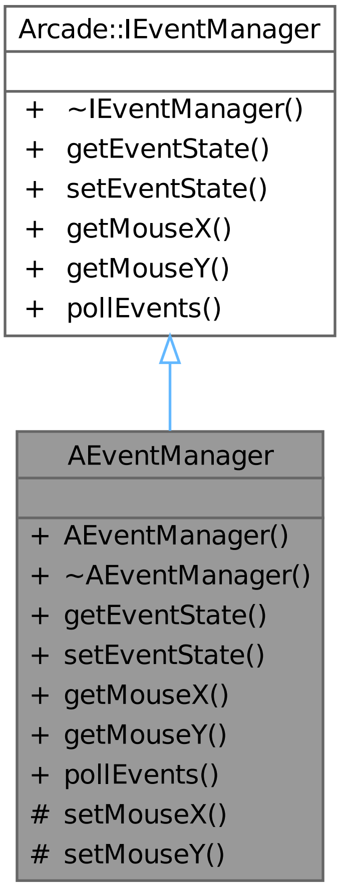
Class Documentation

4.1 AEventManager Class Reference

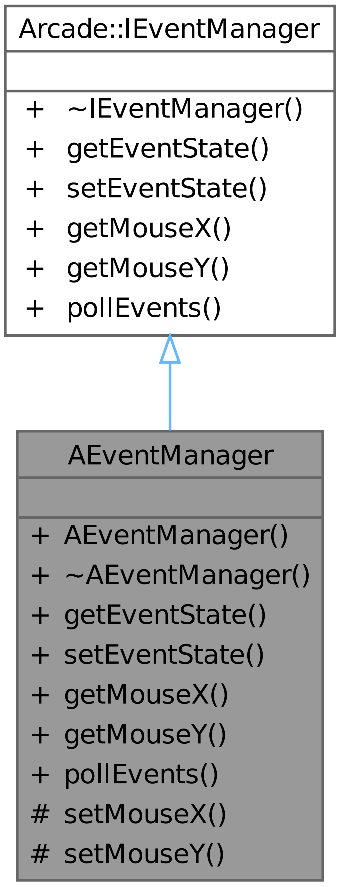
Abstract base class for event management.

`#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

key	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

key	The key/event identifier.
state	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:

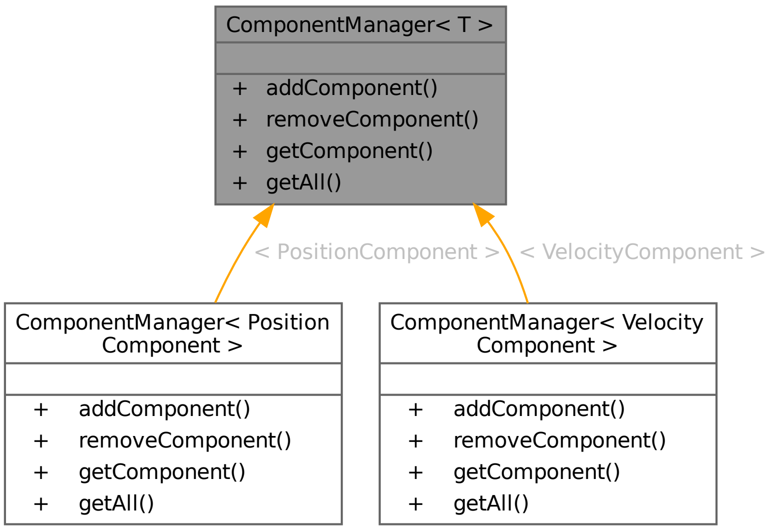
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.hpp
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.cpp

4.2 ComponentManager< T > Class Template Reference

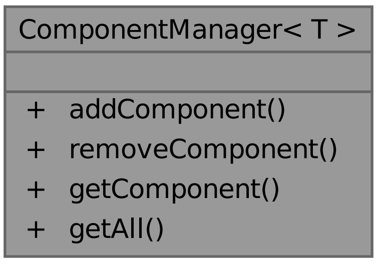
Manages components of a specific type for entities.

```
#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

T	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

entity	The entity to which the component is assigned.
component	The component to be added.

4.2.2.2 getAll()

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

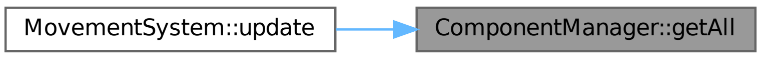
Retrieves all components.

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.

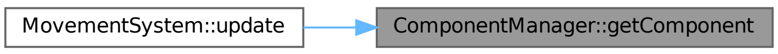
Parameters

entity	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

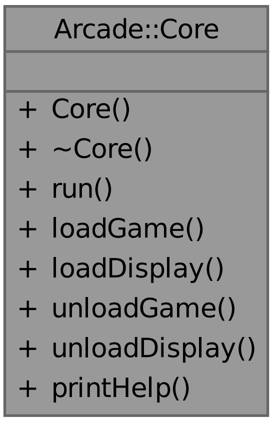
entity	The entity whose component should be removed.

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/Component↵
Manager.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:

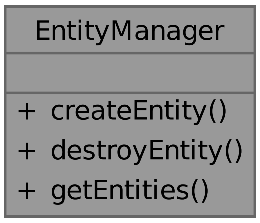
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Core/core.hpp
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Core/core.cpp

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

entity	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

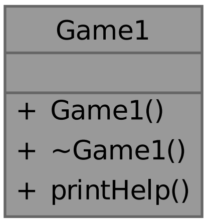
A constant reference to a vector containing all active entity identifiers.

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

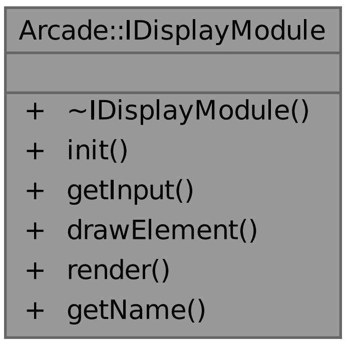
- void **printHelp** ()

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:



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:

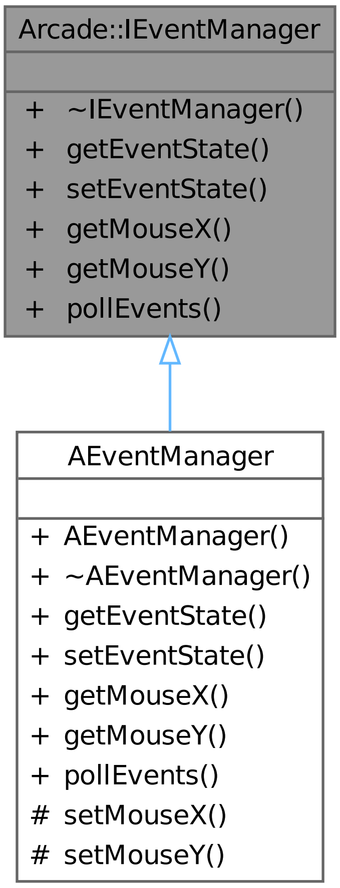
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IDisplayModule.hpp

4.7 Arcade::IEventManager Class Reference

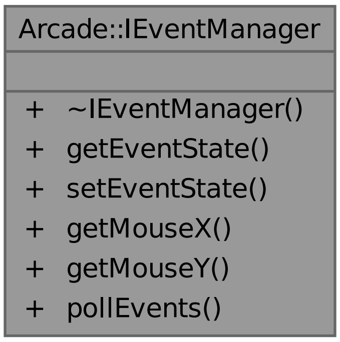
Interface for event management.

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

Inheritance diagram for Arcade::IEventManager:



Collaboration diagram for Arcade::IEventManager:



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

key	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

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

Implemented in AEventManager.

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

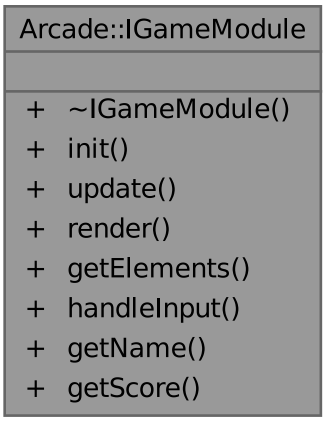
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IEventManager.hpp

4.8 Arcade::IGameModule Class Reference

Interface for game logic modules.

`#include <IGameModule.hpp>`

Collaboration diagram for Arcade::IGameModule:



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:

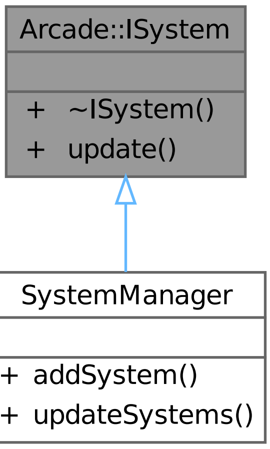
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/IGameModule.hpp

4.9 Arcade::ISystem Class Reference

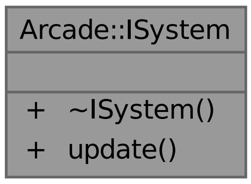
Interface for game systems in the ECS.

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

Inheritance diagram for Arcade::ISystem:



Collaboration diagram for Arcade::ISystem:



Public Member Functions

- virtual ~ISystem ()=default

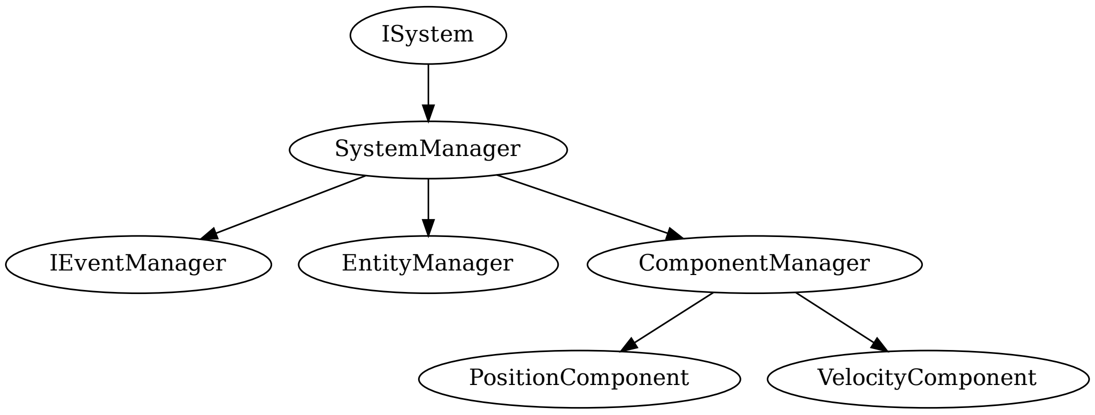
  Virtual destructor.
- virtual void update ()=0

  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:

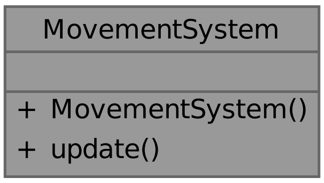
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Interface/ISystem.hpp

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

pm	Pointer to the PositionComponent manager.
vm	Pointer to the VelocityComponent manager.

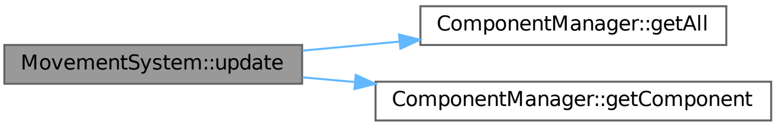
4.10.3 Member Function Documentation

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:

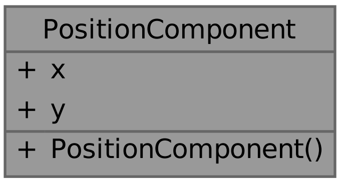
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.cpp

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:

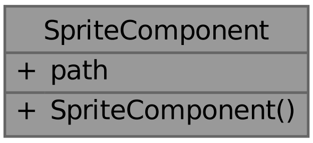
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Position.hpp

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

path	The file path to the sprite image.

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

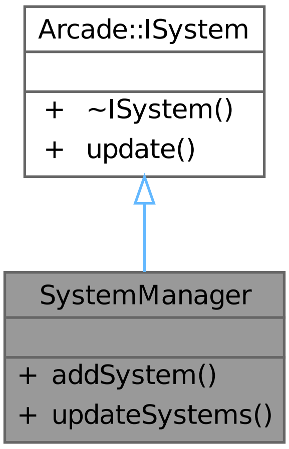
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Sprite.hpp

4.13 SystemManager Class Reference

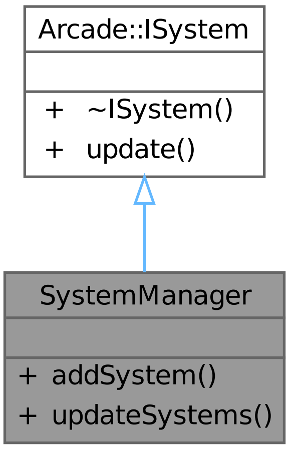
Manages and updates game systems.

```
#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

system	A pointer to an ISystem 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:

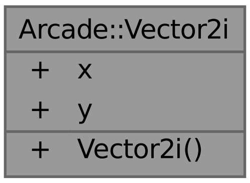
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.hpp
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.cpp

4.14 Arcade::Vector2i Struct Reference

Represents a 2D vector with integer coordinates.

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

Collaboration diagram for Arcade::Vector2i:



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

x	The X-coordinate (default is 0).
y	The Y-coordinate (default is 0).

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

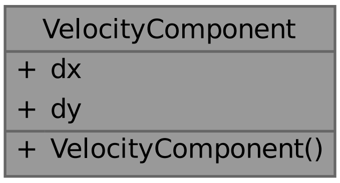
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/WindowManager/WindowManager.hpp

4.15 VelocityComponent Struct Reference

Represents an entity's velocity in 2D space.

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

Collaboration diagram for VelocityComponent:



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

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

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

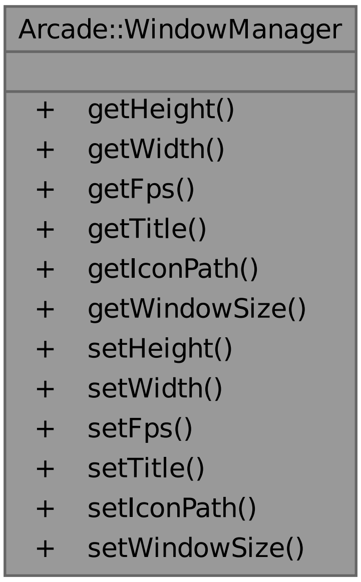
- /home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Components/Velocity.hpp

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:



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

fps	The new FPS value.

4.16.2.8 setHeight()

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

Sets the height of the window.

Parameters

height	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

iconPath	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

title	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

width	The new width value in pixels.

4.16.2.12 setWindowSize()

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

Sets the window size.

Parameters

vec	A Vector2i 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

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** Arcade
** File description:
** core
*/

#ifndef SRC_CORE_CORE_HPP_
    #define SRC_CORE_CORE_HPP_
    #include <memory>
    #include <string>
    #include "../Shared/Interface/IGameModule.hpp"
    #include "../Shared/Interface/IDisplayModule.hpp"

namespace Arcade {
class Core {
 public:
    Core() : _currentGame(nullptr), _currentDisplay(nullptr) {}
    ~Core() {}
    void run();
    void loadGame(const std::string &gameName);
    void loadDisplay(const std::string &displayName);
    void unloadGame();
    void unloadDisplay();
    void printHelp();

 private:
    std::unique_ptr<IGameModule> _currentGame;
    std::unique_ptr<IDisplayModule> _currentDisplay;
};
}  // namespace Arcade
#endif  // SRC_CORE_CORE_HPP_
```

5.2 DLLoader.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** DLoader class
*/

#ifndef SRC_CORE_DLLOADER_DLLOADER_HPP_
    #define SRC_CORE_DLLOADER_DLLOADER_HPP_

#endif  // SRC_CORE_DLLOADER_DLLOADER_HPP_
```

5.3 GameLoop.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** GameLoop class
*/

#ifndef SRC_CORE_GAMELOOP_GAMELOOP_HPP_
    #define SRC_CORE_GAMELOOP_GAMELOOP_HPP_

#endif  // SRC_CORE_GAMELOOP_GAMELOOP_HPP_
```

5.4 Game1.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** Game1 class
*/

#ifndef SRC_GAMES_GAME1_GAME1_HPP_
    #define SRC_GAMES_GAME1_GAME1_HPP_

class Game1 {
 public:
    Game1() = default;
    ~Game1() = default;
    void printHelp();
};

#endif  // SRC_GAMES_GAME1_GAME1_HPP_
```

5.5 Game2.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** Game2
*/

#ifndef SRC_GAMES_GAME2_GAME2_HPP_
    #define SRC_GAMES_GAME2_GAME2_HPP_

#endif  // SRC_GAMES_GAME2_GAME2_HPP_
```

5.6 Game3.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** Game3 class
*/
#ifndef SRC_GAMES_GAME3_GAME3_HPP_
    #define SRC_GAMES_GAME3_GAME3_HPP_

#endif  // SRC_GAMES_GAME3_GAME3_HPP_
```

5.7 nCurses.hpp

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

```
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** nCurses class
*/

#ifndef SRC_GRAPHICS_NCURSES_NCURSES_HPP_
    #define SRC_GRAPHICS_NCURSES_NCURSES_HPP_

#endif  // SRC_GRAPHICS_NCURSES_NCURSES_HPP_
```

5.8 SDL.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** SDL class
*/

#ifndef SRC_GRAPHICS_SDL_SDL_HPP_
    #define SRC_GRAPHICS_SDL_SDL_HPP_

#endif  // SRC_GRAPHICS_SDL_SDL_HPP_
```

5.9 SFML.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** SFML class
*/

#ifndef SRC_GRAPHICS_SFML_SFML_HPP_
    #define SRC_GRAPHICS_SFML_SFML_HPP_

#endif  // SRC_GRAPHICS_SFML_SFML_HPP_
```

5.10 Position.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400
** File description:
** Position Component
*/
#ifndef SRC_SHARED_COMPONENTS_POSITION_HPP_
    #define SRC_SHARED_COMPONENTS_POSITION_HPP_

struct PositionComponent {
    int x;
    int y;

    explicit PositionComponent(int x = 0, int y = 0) : x(x), y(y) {}
};

#endif  // SRC_SHARED_COMPONENTS_POSITION_HPP_
```

5.11 Sprite.hpp

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400
** File description:
** Position Component
*/
```

```
#ifndef SRC_SHARED_COMPONENTS_SPRITE_HPP_
#define SRC_SHARED_COMPONENTS_SPRITE_HPP_
#include <utility>
#include <string>

struct SpriteComponent {
    std::string path;
    explicit SpriteComponent(std::string path) : path(std::move(path)) {}
};

#endif  // SRC_SHARED_COMPONENTS_SPRITE_HPP_
```

5.12 Velocity.hpp

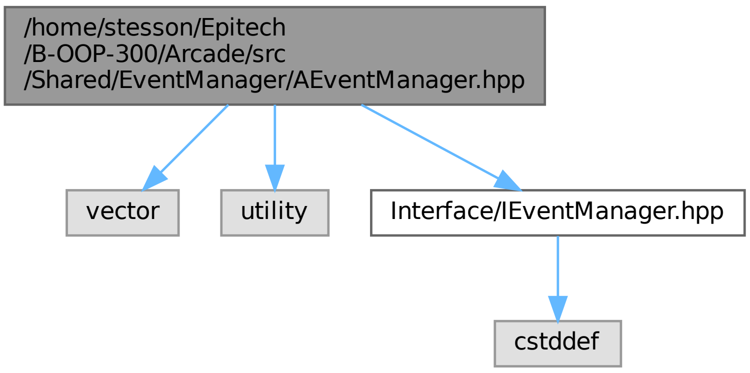
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400
** File description:
** Position Component
*/
#ifndef SRC_SHARED_COMPONENTS_VELOCITY_HPP_
    #define SRC_SHARED_COMPONENTS_VELOCITY_HPP_

struct VelocityComponent {
    int dx;
    int dy;

    explicit VelocityComponent(int dx = 0, int dy = 0) : dx(dx), dy(dy) {}
};

#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:


/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/EventManager/AEventManager.hpp
vector
utility
Interface/IEventManager.hpp
cstddef

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.

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** Arcade
** File description:
** AEventManager abstract class
*/

// Copyright 2025 <Epitech>
#pragma once

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

class AEventManager : public Arcade::IEventManager {
 public:
    AEventManager();

    virtual ~AEventManager() = default;

    bool getEventState(int key) override;

    void setEventState(int key, bool state) override;

    std::size_t getMouseX() const override;

    std::size_t getMouseY() const override;

    virtual void pollEvents() = 0;

 protected:
    void setMouseX(std::size_t x);

    void setMouseY(std::size_t y);

 private:
    std::vector<std::pair<int, bool» _keyEvents;
    std::size_t _mouseX;
    std::size_t _mouseY;
};
```

5.15 IDisplayModule.hpp

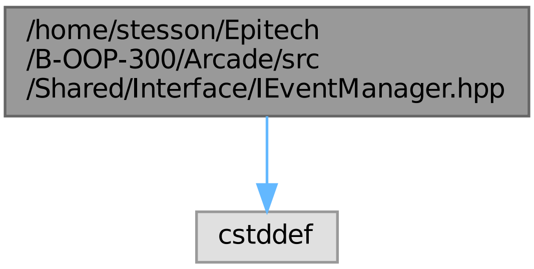
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** Paradigm pool
** File description:
** Seminar
*/

#ifndef SRC_SHARED_INTERFACE_IDISPLAYMODULE_HPP_
    #define SRC_SHARED_INTERFACE_IDISPLAYMODULE_HPP_
    #include <string>
    #include <vector>
    #include "Shared/Models/EntityComponentSystem/EntityManager.hpp"
    #include "Interface/IEventManager.hpp"

namespace Arcade {
class IDisplayModule {
 public:
    virtual ~IDisplayModule() = default;
    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;
};
}  // namespace Arcade
#endif  // SRC_SHARED_INTERFACE_IDISPLAYMODULE_HPP_
```

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

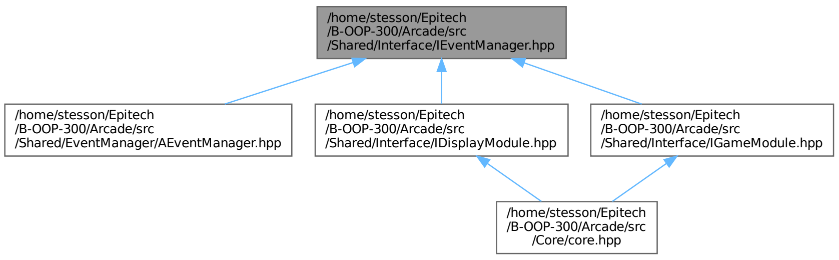
Interface for managing user input events in the Arcade project.

`#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.

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** Arcade
** File description:
** IEventManager interface
*/
// Copyright 2025 <Epitech>
#pragma once
#include <cstddef>

#define KEY_UP 0
#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
```

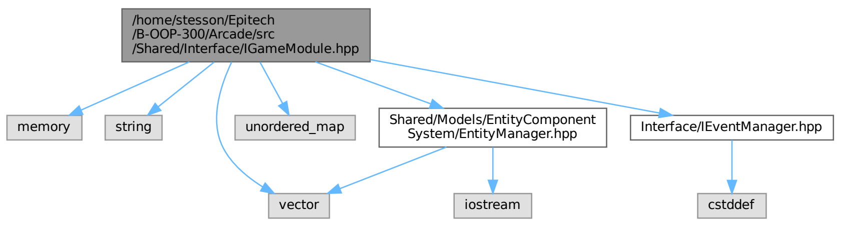
```
#define KEY_ESCAPE 42
#define KEY_BACKSPACE 43
#define KEY_TAB 44
#define KEY_SHIFT 45
#define KEY_CTRL 46
#define KEY_ALT 47
#define KEY_SUPER 48
#define LEFT_CLICK 49
#define RIGHT_CLICK 50
#define MIDDLE_CLICK 51
#define WINDOW_CLOSE 52

namespace Arcade {

class IEventManager {
 public:
    virtual ~IEventManager() = default;

    virtual bool getEventState(int key) = 0;

    virtual void setEventState(int key, bool state) = 0;

    virtual std::size_t getMouseX() const = 0;

    virtual std::size_t getMouseY() const = 0;

    virtual void pollEvents() = 0;
};

}  // namespace Arcade
```

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

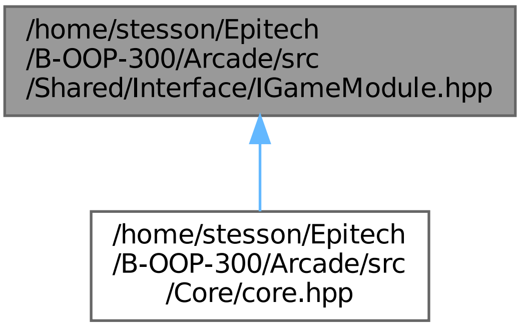
Interface for game modules in the Arcade project.

```
#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

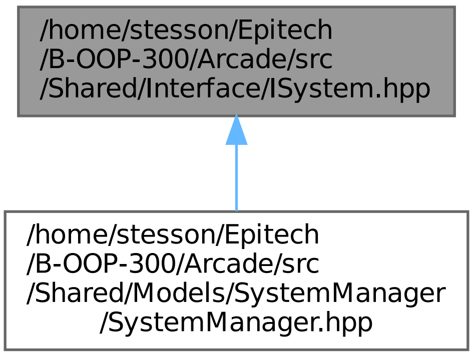
Go to the documentation of this file.

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** IGameModule interface
*/
#ifndef SRC_SHARED_INTERFACE_IGAMEMODULE_HPP_
    #define SRC_SHARED_INTERFACE_IGAMEMODULE_HPP_
    #include <memory>
    #include <string>
    #include <vector>
    #include <unordered_map>
    #include "Shared/Models/EntityComponentSystem/EntityManager.hpp"
    #include "Interface/IEventManager.hpp"

namespace Arcade {
class IGameModule {
 public:
    virtual ~IGameModule() = default;
    virtual void init() = 0;

    virtual void update() = 0;

    virtual void render() = 0;

    virtual std::vector<Entity> getElements() const = 0;
    virtual void handleInput(const IEventManager &event) = 0;

    virtual std::string getName() const = 0;

    virtual int getScore() const = 0;
};
}  // namespace Arcade
#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.

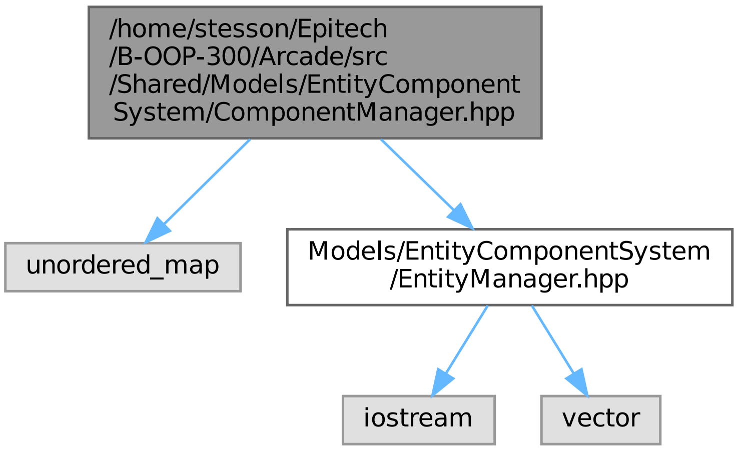
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400
** File description:
** System Interface
*/

// Copyright 2025 <Epitech>
#ifndef SRC_SHARED_INTERFACE_ISYSTEM_HPP_
    #define SRC_SHARED_INTERFACE_ISYSTEM_HPP_

namespace Arcade {
class ISystem {
 public:
    virtual ~ISystem() = default;

    virtual void update() = 0;
};
}  // namespace Arcade

#endif  // SRC_SHARED_INTERFACE_ISYSTEM_HPP_
```

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

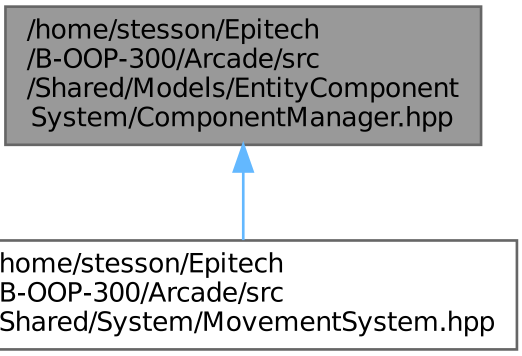
Manages components for entities in an Entity Component System (ECS).

```
#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:


/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/EntityComponentSystem/ComponentManager.hpp
/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp

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.

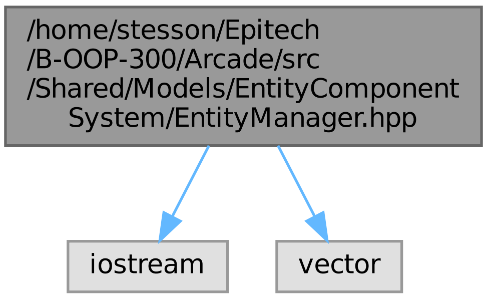
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** Component Manager
*/

// Copyright 2025 <Epitech>
#ifndef SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_COMPONENTMANAGER_HPP_
    #define SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_COMPONENTMANAGER_HPP_
    #include <unordered_map>
    #include "Models/EntityComponentSystem/EntityManager.hpp"

template <typename T>
class ComponentManager {
 private:
    std::unordered_map<Entity, T> components;

 public:
    void addComponent(Entity entity, T component);

    void removeComponent(Entity entity);

    T* getComponent(Entity entity);

    std::unordered_map<Entity, T>& getAll();
};

#endif  // SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_COMPONENTMANAGER_HPP_

```

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

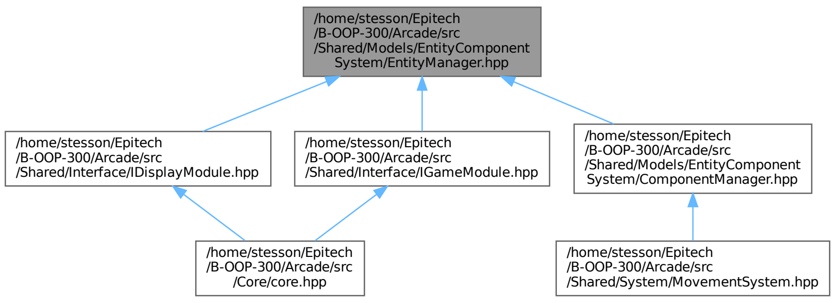
Manages entity creation and deletion in an Entity Component System (ECS).

```
#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.

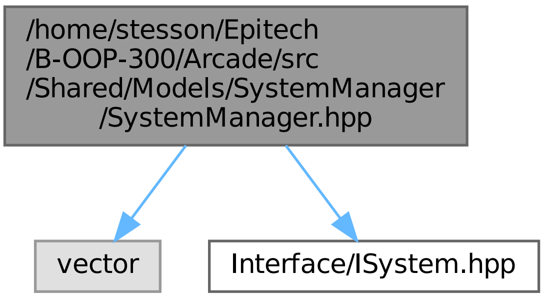
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400
** File description:
** Entity manager
*/
// Copyright 2025 <Epitech>
#ifndef SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_ENTITYMANAGER_HPP_
    #define SRC_SHARED_MODELS_ENTITYCOMPONENTSYSTEM_ENTITYMANAGER_HPP_
    #include <iostream>
    #include <vector>

using Entity = std::size_t;
class EntityManager {
 public:
    Entity createEntity();

    void destroyEntity(Entity entity);

    const std::vector<Entity>& getEntities() const;

 private:
    Entity _nextEntityId = 0;
    std::vector<Entity> _activeEntities;
};

#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:


/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/Models/SystemManager/SystemManager.hpp
vector
Interface/ISystem.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.

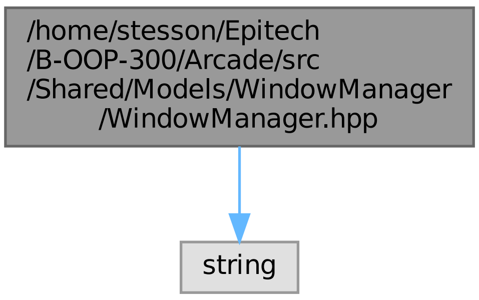
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400 Arcade
** File description:
** System Manager
*/
#ifndef SRC_SHARED_MODELS_SYSTEMMANAGER_SYSTEMMANAGER_HPP_
    #define SRC_SHARED_MODELS_SYSTEMMANAGER_SYSTEMMANAGER_HPP_
    #include <vector>
    #include "Interface/ISystem.hpp"

class SystemManager : public Arcade::ISystem {
 public:
    void addSystem(Arcade::ISystem* system);
    void updateSystems();

 private:
    std::vector<Arcade::ISystem*> _systems;
};

#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.

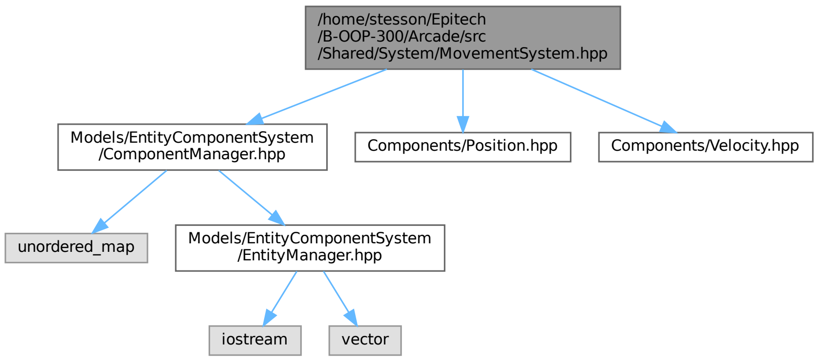
```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400-NAN-4-1-arcade-marin.lamy
** File description:
** windowManager
*/
#ifndef SRC_SHARED_MODELS_WINDOWMANAGER_WINDOWMANAGER_HPP_
    #define SRC_SHARED_MODELS_WINDOWMANAGER_WINDOWMANAGER_HPP_
    #include <string>

namespace Arcade {
struct Vector2i {
    int x;
    int y;
    explicit Vector2i(int x = 0, int y = 0) : x(x), y(y) {}
};

class WindowManager {
 public:
    int getHeight() const;

    int getWidth() const;

    float getFps() const;

    std::string getTitle() const;

    std::string getIconPath() const;

    Vector2i getWindowSize() const;

    void setHeight(int height);

    void setWidth(int width);

    void setFps(float fps);

    void setTitle(const std::string &title);

    void setIconPath(const std::string &iconPath);

    void setWindowSize(Vector2i vec);

 private:
    int _width;
    int _height;
    float _fps;
    std::string _title;
    std::string _iconPath;
};

}  // namespace Arcade

#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/Velocity.hpp"
```

Include dependency graph for MovementSystem.hpp:


/home/stesson/Epitech/B-OOP-300/Arcade/src/Shared/System/MovementSystem.hpp
Models/EntityComponentSystem/ComponentManager.hpp
Components/Position.hpp
Components/Velocity.hpp
unordered_map
Models/EntityComponentSystem/EntityManager.hpp
iostream
vector

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.

```
// Copyright 2025 <Epitech>
/*
** EPITECH PROJECT, 2025
** B-OOP-400
** File description:
** MovementSystem
*/
#ifndef SRC_SHARED_SYSTEM_MOVEMENTSYSTEM_HPP_
    #define SRC_SHARED_SYSTEM_MOVEMENTSYSTEM_HPP_
    #include "Models/EntityComponentSystem/ComponentManager.hpp"
    #include "Components/Position.hpp"
    #include "Components/Velocity.hpp"

class MovementSystem {
 public:
    MovementSystem(ComponentManager<PositionComponent>* pm,
                   ComponentManager<VelocityComponent>* vm)
        : positionManager(pm), velocityManager(vm) {}

    void update();

 private:
    ComponentManager<PositionComponent>* positionManager;
    ComponentManager<VelocityComponent>* velocityManager;
};

#endif  // SRC_SHARED_SYSTEM_MOVEMENTSYSTEM_HPP_
```

Index