Arcade

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 Game Namespace Reference

Namespace for game logic components of the Arcade project.

Classes

- class **AGame**

  *Abstract base class for games, implementing the **IGame** (p. 33) interface.*
- class **IGame**

  Interface for game logic in the Arcade project.
- class **Menu**
- class **Pacman**
- class **Snake**

5.1.1 Detailed Description

Namespace for game logic components of the Arcade project.

5.2 Graphic Namespace Reference

Namespace for graphical components of the Arcade project.

Classes

- class **ADisplay**

  *Abstract base class for display systems, implementing the **IDisplay** (p. 24) interface.*
- class **AEntity**

  *Abstract base class for game entities, implementing the **IEntity** (p. 29) interface.*
- class **IDisplay**

  Interface defining the display functionality for Arcade games.
- class **IEntity**

  Interface for game entities in the Arcade project.
- class **NCurses**
- class **SDL**
- class **RectangleShape**
- class **SFML**

Functions

- void **swhitchElem** (std::size_t i, std::size_t j, std::string line)
- void **swhitchBorder** (std::size_t i, std::size_t j, std::string line)
- void **treatMap** (std::size_t i, std::string line)

5.2.1 Detailed Description

Namespace for graphical components of the Arcade project.

Chapter 6

Class Documentation

6.1 Graphic::ADisplay Class Reference

Abstract base class for display systems, implementing the **IDisplay** (p. 24) interface.

`#include <ADisplay.hpp>`

Inherits **Graphic::IDisplay**.

Inherited by **Graphic::NCurses**, **Graphic::SDL**, and **Graphic::SFML**.

Public Member Functions

- void **setGameName** (std::string name) override

  Sets the name of the current game.
- int **generateRandomNumber** ()

  Generates a random number. Can be used for various game logic that requires randomization.
- void **parseMap** () override

  Parses the game map.
- std::vector< **IEntity** * > **getEntity** () override

  Gets the current entities in the game.
- bool **closeWindow** (int type) override

  Handles window close request.
- float **getElapsedTime** () override

  Gets the elapsed time since the last frame.
- void **resetClock** () override

  Resets the game clock.
- void **removeEntity** (**IEntity** *entity) override

  Removes an entity from the game.

Protected Attributes

- std::string **_nameWindow**

  Name of the game window.
- std::string **_nameLib**

  Name of the graphic library.
- std::string **_playerName**

  Name of the player.
- std::string **_gameName**

  Name of the current game.
- std::string **_mapPath**

  Path to the current game map.
- std::size_t **_score**

  Current score of the player.
- std::size_t **_life**

  Life count of the player.
- std::size_t **_direction**

  Current direction of something in the game.
- std::vector< **IEntity** * > **_entity**

  List of entities in the game.
- std::vector< std::string > **_map**

  Representation of the game map.

6.1.1 Detailed Description

Abstract base class for display systems, implementing the **IDisplay** (p. 24) interface.

Provides foundational functionalities for display systems, including handling game names, generating random numbers for game logic, managing game entities, and basic window operations.

6.1.2 Member Function Documentation

6.1.2.1 closeWindow()

```
bool Graphic::ADisplay::closeWindow (
            int type )  [override], [virtual]
```

Handles window close request.

Parameters

type	Type of close request.

Returns

true if the window should close, false otherwise.

Implements **Graphic::IDisplay** (p. 25).

Reimplemented in **Graphic::SFML** (p. 44), **Graphic::SDL** (p. 40), and **Graphic::NCurses** (p. 36).

6.1.2.2 generateRandomNumber()

```
int Graphic::ADisplay::generateRandomNumber ( )
```

Generates a random number. Can be used for various game logic that requires randomization.

Returns

A random integer.

6.1.2.3 getElapsedTime()

```
float Graphic::ADisplay::getElapsedTime ( )  [override], [virtual]
```

Gets the elapsed time since the last frame.

Returns

Elapsed time in float.

Implements **Graphic::IDisplay** (p. 26).

Reimplemented in **Graphic::SFML** (p. 46), **Graphic::SDL** (p. 42), and **Graphic::NCurses** (p. 37).

6.1.2.4 getEntity()

```
std::vector<  IEntity * > Graphic::ADisplay::getEntity ( )  [override], [virtual]
```

Gets the current entities in the game.

Returns

Vector of **IEntity** (p. 29) pointers.

Implements **Graphic::IDisplay** (p. 26).

6.1.2.5 removeEntity()

```
void Graphic::ADisplay::removeEntity (
              IEntity * entity )  [override], [virtual]
```

Removes an entity from the game.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be removed.

Implements **Graphic::IDisplay** (p. 27).

Reimplemented in **Graphic::SFML** (p. 47), **Graphic::SDL** (p. 43), and **Graphic::NCurses** (p. 38).

6.1.2.6 setGameName()

```
void Graphic::ADisplay::setGameName (
            std::string name )  [override], [virtual]
```

Sets the name of the current game.

Parameters

name	Name of the game.

Implements **Graphic::IDisplay** (p. 29).

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

- src/lib/ **ADisplay.hpp**
- src/lib/ADisplay.cpp

6.2 Graphic::AEntity Class Reference

Abstract base class for game entities, implementing the **IEntity** (p. 29) interface.

`#include <AEntity.hpp>`

Inherits **Graphic::IEntity**.

Inherited by **EntityNcurses**, **EntitySDL**, and **Sprite**.

Public Member Functions

- std::string **getType** () override
  Gets the entity type.
- void **setDirection** (std::size_t direction) override
  Sets the direction of the entity.
- std::size_t **getDirection** () override
  Gets the current direction of the entity.
- void **setRotation** (float angle) override
  Sets the rotation angle of the entity.
- char **getSymbol** () override
  Gets the symbol representing the entity.
- std::string **getTexturePath** () override
  Gets the path to the entity's texture.
- float **getRotation** () override
  Gets the current rotation angle of the entity.

Protected Attributes

- std::string **_type**
  Type of the entity.
- std::string **_texturePath**
  Path to the texture of the entity.
- std::pair< float, float > **_position**
  Position of the entity (x, y).
- std::size_t **_direction**
  Direction of the entity.
- float **_angle**
  Rotation angle of the entity.
- char **_symbol**
  Symbol representing the entity.

6.2.1 Detailed Description

Abstract base class for game entities, implementing the **IEntity** (p. 29) interface.

This class provides basic implementations for common entity functionalities, such as handling type, texture, direction, and rotation.

6.2.2 Member Function Documentation

6.2.2.1 getDirection()

```
std::size_t Graphic::AEntity::getDirection ( )  [override], [virtual]
```

Gets the current direction of the entity.

Returns

The current direction of the entity as size_t.

Implements **Graphic::IEntity** (p. 30).

6.2.2.2 getRotation()

```
float Graphic::AEntity::getRotation ( )  [override], [virtual]
```

Gets the current rotation angle of the entity.

Returns

The rotation angle in degrees.

Implements **Graphic::IEntity** (p. 31).

6.2.2.3 getSymbol()

```
char Graphic::AEntity::getSymbol ( )  [override], [virtual]
```

Gets the symbol representing the entity.

Returns

A char representing the entity symbol.

Implements **Graphic::IEntity** (p. 31).

Reimplemented in **EntityNcurses** (p. 20).

6.2.2.4 getTexturePath()

```
std::string Graphic::AEntity::getTexturePath ( )  [override], [virtual]
```

Gets the path to the entity's texture.

Returns

A string representing the path to the texture.

Implements **Graphic::IEntity** (p. 31).

6.2.2.5 getType()

```
std::string Graphic::AEntity::getType ( )  [override], [virtual]
```

Gets the entity type.

Returns

A string representing the type of the entity.

Implements **Graphic::IEntity** (p. 31).

6.2.2.6 setDirection()

```
void Graphic::AEntity::setDirection (
            std::size_t direction )  [override], [virtual]
```

Sets the direction of the entity.

Parameters

direction	The new direction of the entity.

Implements **Graphic::IEntity** (p. 32).

6.2.2.7 setRotation()

```
void Graphic::AEntity::setRotation (
            float angle )  [override], [virtual]
```

Sets the rotation angle of the entity.

Parameters

angle	The new rotation angle in degrees.

Implements **Graphic::IEntity** (p. 32).

Reimplemented in **Sprite** (p. 49), and **EntitySDL** (p. 23).

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

- src/lib/ **AEntity.hpp**
- src/lib/AEntity.cpp

6.3 Game::AGame Class Reference

Abstract base class for games, implementing the **IGame** (p. 33) interface.

`#include <AGame.hpp>`

Inherits **Game::IGame**.

Inherited by **Game::Menu**, **Game::Pacman**, and **Game::Snake**.

Public Member Functions

- **AGame** (**Graphic::IDisplay** *display)
- void **finishGame** () override
- void **setScore** (std::size_t score) override
- std::size_t **getScore** () override
- void **setLife** (std::size_t life) override
- std::size_t **getLife** () override
- void **setMap** (std::vector< std::string > map) override
- std::vector< std::string > **getMap** () override

- void **setMapPath** (std::string mapPath) override
- std::string **getMapPath** () override
- void **setPlayerName** (std::string playerName) override
- std::string **getPlayerName** () override
- void **setGameName** (std::string gameName) override
- std::string **getGameName** () override
- std::vector< **Graphic::IEntity** * > **getEntity** () override
- void **setEntity** (std::vector< **InfoEntity** > entity) override
- bool **createEntityWithPrev** ()

  Creates game entities based on the previous state.

Protected Attributes

- std::string **_gameName**

  Name of the game.
- std::string **_playerName**

  Name of the player.
- std::string **_mapPath**

  Path to the game map.
- std::size_t **_score**

  Player's score.
- std::size_t **_input**

  Player's input.
- std::vector< **InfoEntity** > **_prevEntity**

  Previous state of game entities.
- std::size_t **_life**

  Player's remaining lives.
- std::vector< std::string > **_map**

  Game *(p. 9) map.*
- **Graphic::IDisplay** * **_display**

  Display interface for graphical operations.

6.3.1 Detailed Description

Abstract base class for games, implementing the **IGame** (p. 33) interface.

Provides foundational functionalities for game operations, including game state management, player interactions, and game entity management.

6.3.2 Member Function Documentation

6.3.2.1 createEntityWithPrev()

```
bool Game::AGame::createEntityWithPrev ( )
```

Creates game entities based on the previous state.

Returns

True if entities are successfully created, false otherwise.

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

- src/game/ **AGame.hpp**
- src/game/AGame.cpp

6.4 Core::Arcade Class Reference

Public Member Functions

- **Arcade** (std::string libName)
- bool **start** ()
- bool **startMenu** ()
- bool **startGame** ()
- void **reloadToMenu** ()
- void **openMap** (std::string path)

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

- src/core/Arcade.hpp
- src/core/Arcade.cpp

6.5 Core::DLopener Class Reference

Public Member Functions

- void **loadLib** (std::size_t type, **Graphic::IDisplay** ∗display)
- void **clearLib** ()
- void **setLib** (std::string libName)
- **Game::IGame** ∗ **getGame** ()
- **Graphic::IDisplay** ∗ **getDisplay** ()

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

- src/core/DLopener.hpp
- src/core/DLopener.cpp

6.6 EntityNcurses Class Reference

Inherits **Graphic::AEntity**.

Public Member Functions

- **EntityNcurses** (std::string type, float x, float y, std::string texturePath, char symbol)
- virtual void **setTexture** (std::string &path) override

  Sets the texture of the entity.
- virtual void **setPosition** (float x, float y) override

  Sets the position of the entity.
- virtual std::pair< float, float > **getPosition** () override

  Gets the current position of the entity.
- virtual void **setSize** (float x, float y) override

  Sets the size of the entity.
- char **getSymbol** ()

  Gets the symbol representing the entity.

Additional Inherited Members

6.6.1 Member Function Documentation

6.6.1.1 getPosition()

```
std::pair< float, float > EntityNcurses::getPosition ( )  [override], [virtual]
```

Gets the current position of the entity.

Returns

A pair of floats representing the X and Y coordinates of the entity.

Implements **Graphic::IEntity** (p. 30).

6.6.1.2 getSymbol()

```
char EntityNcurses::getSymbol ( )  [virtual]
```

Gets the symbol representing the entity.

Returns

A char representing the entity symbol.

Reimplemented from **Graphic::AEntity** (p. 15).

6.6.1.3 setPosition()

```
void EntityNcurses::setPosition (
            float x,
            float y )  [override], [virtual]
```

Sets the position of the entity.

Parameters

x	X coordinate of the new position.
y	Y coordinate of the new position.

Implements **Graphic::IEntity** (p. 32).

6.6.1.4 setSize()

```
void EntityNcurses::setSize (
            float x,
            float y )  [override], [virtual]
```

Sets the size of the entity.

Parameters

x	Width of the entity.
y	Height of the entity.

Implements **Graphic::IEntity** (p. 33).

6.6.1.5 setTexture()

```
void EntityNcurses::setTexture (
            std::string & path )  [override], [virtual]
```

Sets the texture of the entity.

Parameters

path	Reference to a string representing the path to the texture.

Implements **Graphic::IEntity** (p. 33).

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

- src/lib/ncurses/Entity.hpp
- src/lib/ncurses/Entity.cpp

6.7 EntitySDL Class Reference

Inherits **Graphic::AEntity**.

Public Member Functions

- **EntitySDL** (std::string type, float x, float y, std::string texturePath, char symbol, SDL_Renderer ∗renderer)
- void **setTexture** (std::string &path) override

  Sets the texture of the entity.
- void **setPosition** (float x, float y) override

  Sets the position of the entity.
- std::pair< float, float > **getPosition** () override

  Gets the current position of the entity.
- void **setSize** (float x, float y) override

  Sets the size of the entity.
- void **setRotation** (float angle) override

  Sets the rotation angle of the entity.
- SDL_Texture ∗ **getTexture** ()
- SDL_Rect ∗ **getRect** ()
- float **getAngle** ()

Additional Inherited Members

6.7.1 Member Function Documentation

6.7.1.1 getPosition()

```
std::pair< float, float > EntitySDL::getPosition ( )  [override], [virtual]
```

Gets the current position of the entity.

Returns

A pair of floats representing the X and Y coordinates of the entity.

Implements **Graphic::IEntity** (p. 30).

6.7.1.2 setPosition()

```
void EntitySDL::setPosition (
            float x,
            float y )  [override], [virtual]
```

Sets the position of the entity.

Parameters

x	X coordinate of the new position.
y	Y coordinate of the new position.

Implements **Graphic::IEntity** (p. 32).

6.7.1.3 setRotation()

```
void EntitySDL::setRotation (
            float angle )  [override], [virtual]
```

Sets the rotation angle of the entity.

Parameters

angle	The new rotation angle in degrees.

Reimplemented from **Graphic::AEntity** (p. 17).

6.7.1.4 setSize()

```
void EntitySDL::setSize (
            float x,
            float y )  [override], [virtual]
```

Sets the size of the entity.

Parameters

x	Width of the entity.
y	Height of the entity.

Implements **Graphic::IEntity** (p. 33).

6.7.1.5 setTexture()

```
void EntitySDL::setTexture (
            std::string & path )  [override], [virtual]
```

Sets the texture of the entity.

Parameters

path	Reference to a string representing the path to the texture.

Implements **Graphic::IEntity** (p. 33).

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

- src/lib/sdl2/EntitySDL.hpp
- src/lib/sdl2/EntitySDL.cpp

6.8 Graphic::IDisplay Class Reference

Interface defining the display functionality for Arcade games.

`#include <IDisplay.hpp>`

Inherited by **Graphic::ADisplay**.

Public Member Functions

- **IDisplay** ()=default
  Default constructor.
- virtual ∼**IDisplay** ()=default
  Default destructor.
- virtual void **setGameName** (std::string name)=0
  Sets the name of the current game.
- virtual void **initWindow** (float x, float y)=0
  Initializes the game window.
- virtual void **createEntity** (std::string type, float x, float y, std::string path, char symbol)=0
  Creates a game entity.
- virtual void **setMap** (std::vector< std::string > map)=0
  Sets the game map.
- virtual bool **closeWindow** (int type)=0
  Handles window close request.
- virtual void **clearWindow** ()=0
  Clears the game window.
- virtual void **displayWindow** ()=0
  Displays the content of the game window.
- virtual std::vector< **IEntity** ∗ > **getEntity** ()=0
  Gets the current entities in the game.
- virtual std::size_t **getInput** ()=0
  Gets the user input.
- virtual void **parseMap** ()=0
  Parses the game map.
- virtual void **displayMap** ()=0
  Displays the game map.
- virtual void **displayEntity** (**IEntity** ∗entity)=0
  Displays a game entity.
- virtual bool **getCollision** (**IEntity** &entity1, **IEntity** &entity2)=0
  Checks for collision between two entities.
- virtual float **getElapsedTime** ()=0
  Gets the elapsed time since the last frame.
- virtual void **removeEntity** (**IEntity** ∗entity)=0
  Removes an entity from the game.
- virtual void **resetClock** ()=0
  Resets the game clock.

6.8.1 Detailed Description

Interface defining the display functionality for Arcade games.

This interface outlines the necessary methods for implementing game displays, including window management, entity rendering, and event handling.

6.8.2 Member Function Documentation

6.8.2.1 closeWindow()

```
virtual bool Graphic::IDisplay::closeWindow (
            int type )  [pure virtual]
```

Handles window close request.

Parameters

type	Type of close request.

Returns

true if the window should close, false otherwise.

Implemented in **Graphic::SFML** (p. 44), **Graphic::SDL** (p. 40), **Graphic::NCurses** (p. 36), and **Graphic::↩ ADisplay** (p. 12).

6.8.2.2 createEntity()

```
virtual void Graphic::IDisplay::createEntity (
            std::string type,
            float x,
            float y,
            std::string path,
            char symbol )  [pure virtual]
```

Creates a game entity.

Parameters

type	Type of the entity.
x	X position of the entity.
y	Y position of the entity.
path	Path to the entity's texture or representation.
symbol	Character symbol representing the entity.

Implemented in **Graphic::SFML** (p. 45), **Graphic::SDL** (p. 41), and **Graphic::NCurses** (p. 36).

6.8.2.3 displayEntity()

```
virtual void Graphic::IDisplay::displayEntity (
              IEntity * entity )  [pure virtual]
```

Displays a game entity.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be displayed.

Implemented in **Graphic::SFML** (p. 45), **Graphic::SDL** (p. 41), and **Graphic::NCurses** (p. 36).

6.8.2.4 getCollision()

```
virtual bool Graphic::IDisplay::getCollision (
              IEntity & entity1,
              IEntity & entity2 )  [pure virtual]
```

Checks for collision between two entities.

Parameters

entity1	First entity.
entity2	Second entity.

Returns

true if a collision occurs, false otherwise.

Implemented in **Graphic::SFML** (p. 45), **Graphic::SDL** (p. 41), and **Graphic::NCurses** (p. 37).

6.8.2.5 getElapsedTime()

```
virtual float Graphic::IDisplay::getElapsedTime ( )  [pure virtual]
```

Gets the elapsed time since the last frame.

Returns

Elapsed time in float.

Implemented in **Graphic::SFML** (p. 46), **Graphic::SDL** (p. 42), **Graphic::NCurses** (p. 37), and **Graphic::ADisplay** (p. 13).

6.8.2.6 getEntity()

```
virtual std::vector< IEntity *> Graphic::IDisplay::getEntity ( )  [pure virtual]
```

Gets the current entities in the game.

Returns

Vector of **IEntity** (p. 29) pointers.

Implemented in **Graphic::ADisplay** (p. 13).

6.8.2.7 getInput()

```
virtual std::size_t Graphic::IDisplay::getInput ( )  [pure virtual]
```

Gets the user input.

Returns

Size_t value representing the user input.

Implemented in **Graphic::SFML** (p. 46), **Graphic::SDL** (p. 42), and **Graphic::NCurses** (p. 37).

6.8.2.8 initWindow()

```
virtual void Graphic::IDisplay::initWindow (
            float x,
            float y )  [pure virtual]
```

Initializes the game window.

Parameters

x	Width of the window.
y	Height of the window.

Implemented in **Graphic::SFML** (p. 46), **Graphic::SDL** (p. 42), and **Graphic::NCurses** (p. 38).

6.8.2.9 removeEntity()

```
virtual void Graphic::IDisplay::removeEntity (
             IEntity * entity )  [pure virtual]
```

Removes an entity from the game.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be removed.

Implemented in **Graphic::SFML** (p. 47), **Graphic::SDL** (p. 43), **Graphic::NCurses** (p. 38), and **Graphic::ADisplay** (p. 13).

6.8.2.10 setGameName()

```
virtual void Graphic::IDisplay::setGameName (
            std::string name )  [pure virtual]
```

Sets the name of the current game.

Parameters

name	Name of the game.

Implemented in **Graphic::ADisplay** (p. 14).

6.8.2.11 setMap()

```
virtual void Graphic::IDisplay::setMap (
            std::vector< std::string > map )  [pure virtual]
```

Sets the game map.

Parameters

map	Vector of strings representing the game map.

Implemented in **Graphic::SFML** (p. 47), **Graphic::SDL** (p. 43), and **Graphic::NCurses** (p. 38).

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

- src/lib/ **IDisplay.hpp**

6.9 Graphic::IEntity Class Reference

Interface for game entities in the Arcade project.

`#include <IEntity.hpp>`

Inherited by **Graphic::AEntity**.

Public Member Functions

- **IEntity** ()=default
  Default constructor.
- virtual ~**IEntity** ()=default
  Default destructor.
- virtual std::string **getType** ()=0
  Gets the entity type.
- virtual void **setDirection** (std::size_t direction)=0
  Sets the direction of the entity.
- virtual std::size_t **getDirection** ()=0
  Gets the current direction of the entity.
- virtual void **setTexture** (std::string &path)=0
  Sets the texture of the entity.
- virtual void **setPosition** (float x, float y)=0
  Sets the position of the entity.
- virtual std::pair< float, float > **getPosition** ()=0
  Gets the current position of the entity.
- virtual void **setSize** (float x, float y)=0
  Sets the size of the entity.
- virtual void **setRotation** (float angle)=0
  Sets the rotation angle of the entity.
- virtual char **getSymbol** ()=0
  Gets the symbol representing the entity.
- virtual std::string **getTexturePath** ()=0
  Gets the path to the entity's texture.
- virtual float **getRotation** ()=0
  Gets the current rotation angle of the entity.

6.9.1 Detailed Description

Interface for game entities in the Arcade project.

This interface defines the basic functionalities required for any game entity, including type identification, texture handling, and spatial properties.

6.9.2 Member Function Documentation

6.9.2.1 getDirection()

```
virtual std::size_t Graphic::IEntity::getDirection ( )  [pure virtual]
```

Gets the current direction of the entity.

Returns

The current direction of the entity as size_t.

Implemented in **Graphic::AEntity** (p. 15).

6.9.2.2 getPosition()

```
virtual std::pair<float, float> Graphic::IEntity::getPosition ( )  [pure virtual]
```

Gets the current position of the entity.

Returns

A pair of floats representing the X and Y coordinates of the entity.

Implemented in **Sprite** (p. 48), **EntitySDL** (p. 22), and **EntityNcurses** (p. 20).

6.9.2.3 getRotation()

```
virtual float Graphic::IEntity::getRotation ( )  [pure virtual]
```

Gets the current rotation angle of the entity.

Returns

The rotation angle in degrees.

Implemented in **Graphic::AEntity** (p. 15).

6.9.2.4 getSymbol()

```
virtual char Graphic::IEntity::getSymbol ( )  [pure virtual]
```

Gets the symbol representing the entity.

Returns

A char representing the entity symbol.

Implemented in **Graphic::AEntity** (p. 15), and **EntityNcurses** (p. 20).

6.9.2.5 getTexturePath()

```
virtual std::string Graphic::IEntity::getTexturePath ( )  [pure virtual]
```

Gets the path to the entity's texture.

Returns

A string representing the path to the texture.

Implemented in **Graphic::AEntity** (p. 16).

6.9.2.6 getType()

```
virtual std::string Graphic::IEntity::getType ( )  [pure virtual]
```

Gets the entity type.

Returns

A string representing the type of the entity.

Implemented in **Graphic::AEntity** (p. 16).

6.9.2.7 setDirection()

```
virtual void Graphic::IEntity::setDirection (
            std::size_t direction )  [pure virtual]
```

Sets the direction of the entity.

Parameters

direction	The new direction of the entity.

Implemented in **Graphic::AEntity** (p. 16).

6.9.2.8 setPosition()

```
virtual void Graphic::IEntity::setPosition (
            float x,
            float y )  [pure virtual]
```

Sets the position of the entity.

Parameters

x	X coordinate of the new position.
y	Y coordinate of the new position.

Implemented in **Sprite** (p. 48), **EntitySDL** (p. 22), and **EntityNcurses** (p. 20).

6.9.2.9 setRotation()

```
virtual void Graphic::IEntity::setRotation (
            float angle )  [pure virtual]
```

Sets the rotation angle of the entity.

Parameters

angle	The new rotation angle in degrees.

Implemented in **Sprite** (p. 49), **EntitySDL** (p. 23), and **Graphic::AEntity** (p. 17).

6.9.2.10 setSize()

```
virtual void Graphic::IEntity::setSize (
            float x,
            float y )  [pure virtual]
```

Sets the size of the entity.

Parameters

x	Width of the entity.
y	Height of the entity.

Implemented in **Sprite** (p. 49), **EntitySDL** (p. 23), and **EntityNcurses** (p. 21).

6.9.2.11 setTexture()

```
virtual void Graphic::IEntity::setTexture (
            std::string & path )  [pure virtual]
```

Sets the texture of the entity.

Parameters

path	Reference to a string representing the path to the texture.

Implemented in **Sprite** (p. 49), **EntitySDL** (p. 23), and **EntityNcurses** (p. 21).

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

- src/lib/ **IEntity.hpp**

6.10 Game::IGame Class Reference

Interface for game logic in the Arcade project.

`#include <IGame.hpp>`

Inherited by **Game::AGame**.

Public Member Functions

- virtual std::size_t **startGame** ()=0
- virtual void **finishGame** ()=0
- virtual void **setScore** (std::size_t score)=0
- virtual std::size_t **getScore** ()=0
- virtual void **setLife** (std::size_t life)=0
- virtual std::size_t **getLife** ()=0
- virtual void **setMap** (std::vector< std::string > map)=0
- virtual std::vector< std::string > **getMap** ()=0
- virtual void **setMapPath** (std::string mapPath)=0
- virtual std::string **getMapPath** ()=0
- virtual void **setPlayerName** (std::string playerName)=0
- virtual std::string **getPlayerName** ()=0
- virtual void **setGameName** (std::string gameName)=0
- virtual std::string **getGameName** ()=0
- virtual std::vector< **Graphic::IEntity** * > **getEntity** ()=0
- virtual void **setEntity** (std::vector< **InfoEntity** > entity)=0

6.10.1 Detailed Description

Interface for game logic in the Arcade project.

Defines the essential functions for game operations, including starting and finishing a game, managing scores, lives, maps, and entities.

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

- src/game/ **IGame.hpp**

6.11 InfoEntity Struct Reference

Public Attributes

- std::string **type**
- std::string **texturePath**
- std::pair< float, float > **position**
- std::size_t **direction**
- float **angle**
- char **symbol**

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

- src/core/InfoEntity.hpp

6.12 Game::Menu Class Reference

Inherits **Game::AGame**.

Public Member Functions

- **Menu** (**Graphic::IDisplay** ∗display)
- std::size_t **startGame** () override
- void **displayEntity** ()
- void **MoveEntity** ()
- void **endGame** ()
- void **createEntities** ()

Additional Inherited Members

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

- src/game/menu/Menu.hpp
- src/game/menu/Menu.cpp

6.13 Graphic::NCurses Class Reference

Inherits **Graphic::ADisplay**.

Public Member Functions

- void **initWindow** (float x, float y) override
  Initializes the game window.
- void **createEntity** (std::string type, float x, float y, std::string path, char symbol) override
  Creates a game entity.
- void **setMap** (std::vector< std::string > map) override
  Sets the game map.
- void **displayMap** () override
  Displays the game map.
- void **displayEntity** (**IEntity** ∗entity) override
  Displays a game entity.
- std::size_t **getInput** () override
  Gets the user input.
- void **clearWindow** () override
  Clears the game window.
- void **displayWindow** () override
  Displays the content of the game window.
- bool **getCollision** (**IEntity** &entity1, **IEntity** &entity2) override
  Checks for collision between two entities.
- bool **closeWindow** (int type) override
  Handles window close request.
- void **removeEntity** (**IEntity** ∗entity) override
  Removes an entity from the game.
- float **getElapsedTime** () override
  Gets the elapsed time since the last frame.
- void **resetClock** () override
  Resets the game clock.

Additional Inherited Members

6.13.1 Member Function Documentation

6.13.1.1 closeWindow()

```
bool Graphic::NCurses::closeWindow (
            int type )  [override], [virtual]
```

Handles window close request.

Parameters

type	Type of close request.

Returns

true if the window should close, false otherwise.

Reimplemented from **Graphic::ADisplay** (p. 12).

6.13.1.2 createEntity()

```
void Graphic::NCurses::createEntity (
            std::string type,
            float x,
            float y,
            std::string path,
            char symbol )  [override], [virtual]
```

Creates a game entity.

Parameters

type	Type of the entity.
x	X position of the entity.
y	Y position of the entity.
path	Path to the entity's texture or representation.
symbol	Character symbol representing the entity.

Implements **Graphic::IDisplay** (p. 25).

6.13.1.3 displayEntity()

```
void Graphic::NCurses::displayEntity (
              IEntity * entity )  [override], [virtual]
```

Displays a game entity.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be displayed.

Implements **Graphic::IDisplay** (p. 26).

6.13.1.4 getCollision()

```
bool Graphic::NCurses::getCollision (
              IEntity & entity1,
              IEntity & entity2 )  [override], [virtual]
```

Checks for collision between two entities.

Parameters

entity1	First entity.
entity2	Second entity.

Returns

true if a collision occurs, false otherwise.

Implements **Graphic::IDisplay** (p. 26).

6.13.1.5 getElapsedTime()

```
float Graphic::NCurses::getElapsedTime ( )  [override], [virtual]
```

Gets the elapsed time since the last frame.

Returns

Elapsed time in float.

Reimplemented from **Graphic::ADisplay** (p. 13).

6.13.1.6 getInput()

```
std::size_t Graphic::NCurses::getInput ( )  [override], [virtual]
```

Gets the user input.

Returns

Size_t value representing the user input.

Implements **Graphic::IDisplay** (p. 27).

6.13.1.7 initWindow()

```
void Graphic::NCurses::initWindow (
            float x,
            float y )  [override], [virtual]
```

Initializes the game window.

Parameters

x	Width of the window.
y	Height of the window.

Implements **Graphic::IDisplay** (p. 27).

6.13.1.8 removeEntity()

```
void Graphic::NCurses::removeEntity (
             IEntity * entity )  [override], [virtual]
```

Removes an entity from the game.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be removed.

Reimplemented from **Graphic::ADisplay** (p. 13).

6.13.1.9 setMap()

```
void Graphic::NCurses::setMap (
            std::vector< std::string > map )  [override], [virtual]
```

Sets the game map.

Parameters

map	Vector of strings representing the game map.

Implements **Graphic::IDisplay** (p. 29).

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

- src/lib/ncurses/NCurses.hpp
- src/lib/ncurses/NCurses.cpp

6.14 Game::Pacman Class Reference

Inherits **Game::AGame**.

Public Member Functions

- **Pacman** (**Graphic::IDisplay** ∗display)
- std::size_t **startGame** () override
- void **createSmallPacgums** ()
- bool **isWall** (std::pair< float, float > newpos)
- bool **isIntersec** (std::pair< float, float > newpos, std::size_t direction)
- void **moveGhost** ()
- void **movePlayer** ()
- void **checklose** ()

Additional Inherited Members

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

- src/game/pacman/Pacman.hpp
- src/game/pacman/Pacman.cpp

6.15 Graphic::RectangleShape Class Reference

Public Member Functions

- void **setColor** (const sf::Color &color)
- void **setPosition** (const sf::Vector2f &position)
- void **draw** (sf::RenderWindow &window)

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

- src/lib/sfml/RectangleShape.hpp
- src/lib/sfml/RectangleShape.cpp

6.16 Graphic::SDL Class Reference

Inherits **Graphic::ADisplay**.

Public Member Functions

- void **initWindow** (float x, float y) override
  Initializes the game window.
- void **createEntity** (std::string type, float x, float y, std::string path, char symbol) override
  Creates a game entity.
- void **setMap** (std::vector< std::string > map) override
  Sets the game map.
- void **clearWindow** () override
  Clears the game window.
- void **displayWindow** () override
  Displays the content of the game window.
- std::size_t **getInput** () override
  Gets the user input.
- void **parseMap** () override
  Parses the game map.
- void **displayMap** () override
  Displays the game map.
- void **displayEntity** (**IEntity** *entity) override
  Displays a game entity.
- bool **getCollision** (**IEntity** &entity1, **IEntity** &entity2) override
  Checks for collision between two entities.
- bool **closeWindow** (int type) override
  Handles window close request.
- float **getElapsedTime** () override
  Gets the elapsed time since the last frame.
- void **resetClock** () override
  Resets the game clock.
- void **removeEntity** (**IEntity** *entity) override
  Removes an entity from the game.

Additional Inherited Members

6.16.1 Member Function Documentation

6.16.1.1 closeWindow()

```
bool Graphic::SDL::closeWindow (
            int type )  [override], [virtual]
```

Handles window close request.

Parameters

type	Type of close request.

Returns

true if the window should close, false otherwise.

Reimplemented from **Graphic::ADisplay** (p. 12).

6.16.1.2 createEntity()

```
void Graphic::SDL::createEntity (
            std::string type,
            float x,
            float y,
            std::string path,
            char symbol )  [override], [virtual]
```

Creates a game entity.

Parameters

type	Type of the entity.
x	X position of the entity.
y	Y position of the entity.
path	Path to the entity's texture or representation.
symbol	Character symbol representing the entity.

Implements **Graphic::IDisplay** (p. 25).

6.16.1.3 displayEntity()

```
void Graphic::SDL::displayEntity (
             IEntity * entity )  [override], [virtual]
```

Displays a game entity.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be displayed.

Implements **Graphic::IDisplay** (p. 26).

6.16.1.4 getCollision()

```
bool Graphic::SDL::getCollision (
              IEntity & entity1,
              IEntity & entity2 )  [override], [virtual]
```

Checks for collision between two entities.

Parameters

entity1	First entity.
entity2	Second entity.

Returns

true if a collision occurs, false otherwise.

Implements **Graphic::IDisplay** (p. 26).

6.16.1.5 getElapsedTime()

```
float Graphic::SDL::getElapsedTime ( )  [override], [virtual]
```

Gets the elapsed time since the last frame.

Returns

Elapsed time in float.

Reimplemented from **Graphic::ADisplay** (p. 13).

6.16.1.6 getInput()

```
std::size_t Graphic::SDL::getInput ( )  [override], [virtual]
```

Gets the user input.

Returns

Size_t value representing the user input.

Implements **Graphic::IDisplay** (p. 27).

6.16.1.7 initWindow()

```
void Graphic::SDL::initWindow (
             float x,
             float y )  [override], [virtual]
```

Initializes the game window.

Parameters

x	Width of the window.
y	Height of the window.

Implements **Graphic::IDisplay** (p. 27).

6.16.1.8 removeEntity()

```
void Graphic::SDL::removeEntity (
              IEntity * entity )  [override], [virtual]
```

Removes an entity from the game.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be removed.

Reimplemented from **Graphic::ADisplay** (p. 13).

6.16.1.9 setMap()

```
void Graphic::SDL::setMap (
             std::vector< std::string > map )  [override], [virtual]
```

Sets the game map.

Parameters

map	Vector of strings representing the game map.

Implements **Graphic::IDisplay** (p. 29).

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

- src/lib/sdl2/SDL.hpp
- src/lib/sdl2/SDL.cpp

6.17 Graphic::SFML Class Reference

Inherits **Graphic::ADisplay**.

Public Member Functions

- void **initWindow** (float x, float y) override

  Initializes the game window.
- void **createEntity** (std::string type, float x, float y, std::string path, char symbol) override

  Creates a game entity.
- void **setMap** (std::vector< std::string > map) override

  Sets the game map.
- void **clearWindow** () override

  Clears the game window.
- void **displayWindow** () override

  Displays the content of the game window.
- std::size_t **getInput** () override

  Gets the user input.
- void **parseMap** () override

  Parses the game map.
- void **displayMap** () override

  Displays the game map.
- void **displayEntity** (**IEntity** *entity) override

  Displays a game entity.
- bool **getCollision** (**IEntity** &entity1, **IEntity** &entity2) override

  Checks for collision between two entities.
- bool **closeWindow** (int type) override

  Handles window close request.
- float **getElapsedTime** () override

  Gets the elapsed time since the last frame.
- void **resetClock** () override

  Resets the game clock.
- void **removeEntity** (**IEntity** *entity) override

  Removes an entity from the game.

Additional Inherited Members

6.17.1 Member Function Documentation

6.17.1.1 closeWindow()

```
bool Graphic::SFML::closeWindow (
            int type )  [override], [virtual]
```

Handles window close request.

Parameters

type	Type of close request.

Returns

true if the window should close, false otherwise.

Reimplemented from **Graphic::ADisplay** (p. 12).

6.17.1.2 createEntity()

```
void Graphic::SFML::createEntity (
            std::string type,
            float x,
            float y,
            std::string path,
            char symbol )  [override], [virtual]
```

Creates a game entity.

Parameters

type	Type of the entity.
x	X position of the entity.
y	Y position of the entity.
path	Path to the entity's texture or representation.
symbol	Character symbol representing the entity.

Implements **Graphic::IDisplay** (p. 25).

6.17.1.3 displayEntity()

```
void Graphic::SFML::displayEntity (
            IEntity * entity )  [override], [virtual]
```

Displays a game entity.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be displayed.

Implements **Graphic::IDisplay** (p. 26).

6.17.1.4 getCollision()

```
bool Graphic::SFML::getCollision (
            IEntity & entity1,
            IEntity & entity2 )  [override], [virtual]
```

Checks for collision between two entities.

Parameters

entity1	First entity.
entity2	Second entity.

Returns

true if a collision occurs, false otherwise.

Implements **Graphic::IDisplay** (p. 26).

6.17.1.5 getElapsedTime()

```
float Graphic::SFML::getElapsedTime ( )  [override], [virtual]
```

Gets the elapsed time since the last frame.

Returns

Elapsed time in float.

Reimplemented from **Graphic::ADisplay** (p. 13).

6.17.1.6 getInput()

```
std::size_t Graphic::SFML::getInput ( )  [override], [virtual]
```

Gets the user input.

Returns

Size_t value representing the user input.

Implements **Graphic::IDisplay** (p. 27).

6.17.1.7 initWindow()

```
void Graphic::SFML::initWindow (
            float x,
            float y )  [override], [virtual]
```

Initializes the game window.

Parameters

x	Width of the window.
y	Height of the window.

Implements **Graphic::IDisplay** (p. 27).

6.17.1.8 removeEntity()

```
void Graphic::SFML::removeEntity (
             IEntity * entity )  [override], [virtual]
```

Removes an entity from the game.

Parameters

entity	Pointer to the **IEntity** (p. 29) to be removed.

Reimplemented from **Graphic::ADisplay** (p. 13).

6.17.1.9 setMap()

```
void Graphic::SFML::setMap (
            std::vector< std::string > map )  [override], [virtual]
```

Sets the game map.

Parameters

map	Vector of strings representing the game map.

Implements **Graphic::IDisplay** (p. 29).

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

- src/lib/sfml/SFML.hpp
- src/lib/sfml/SFML.cpp

6.18 Game::Snake Class Reference

Inherits **Game::AGame**.

Public Member Functions

- **Snake** (**Graphic::IDisplay** ∗display)
- std::size_t **startGame** () override
- int **generateRandomNumber** ()
- void **checkLoose** ()

Additional Inherited Members

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

- src/game/snake/Snake.hpp
- src/game/snake/Snake.cpp

6.19 Sprite Class Reference

Inherits **Graphic::AEntity**.

Public Member Functions

- **Sprite** (std::string type, float x, float y, std::string texturePath, char symbol)
- void **setTexture** (std::string &path) override

  Sets the texture of the entity.
- void **setPosition** (float x, float y) override

  Sets the position of the entity.
- std::pair< float, float > **getPosition** () override

  Gets the current position of the entity.
- void **setSize** (float x, float y) override

  Sets the size of the entity.
- sf::Sprite **getSprite** ()
- void **setRotation** (float angle) override

  Sets the rotation angle of the entity.

Additional Inherited Members

6.19.1 Member Function Documentation

6.19.1.1 getPosition()

```
std::pair< float, float > Sprite::getPosition ( )  [override], [virtual]
```

Gets the current position of the entity.

Returns

A pair of floats representing the X and Y coordinates of the entity.

Implements **Graphic::IEntity** (p. 30).

6.19.1.2 setPosition()

```
void Sprite::setPosition (
            float x,
            float y )  [override], [virtual]
```

Sets the position of the entity.

Parameters

x	X coordinate of the new position.
y	Y coordinate of the new position.

Implements **Graphic::IEntity** (p. 32).

6.19.1.3 setRotation()

```
void Sprite::setRotation (
            float angle )  [override], [virtual]
```

Sets the rotation angle of the entity.

Parameters

angle	The new rotation angle in degrees.

Reimplemented from **Graphic::AEntity** (p. 17).

6.19.1.4 setSize()

```
void Sprite::setSize (
            float x,
            float y )  [override], [virtual]
```

Sets the size of the entity.

Parameters

x	Width of the entity.
y	Height of the entity.

Implements **Graphic::IEntity** (p. 33).

6.19.1.5 setTexture()

```
void Sprite::setTexture (
            std::string & path )  [override], [virtual]
```

Sets the texture of the entity.

Parameters

path	Reference to a string representing the path to the texture.

Implements **Graphic::IEntity** (p. 33).

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

- src/lib/sfml/Sprite.hpp
- src/lib/sfml/Sprite.cpp

Chapter 7

File Documentation

7.1 src/game/AGame.hpp File Reference

Header for the AGame abstract class in the Arcade project.

```
#include "IGame.hpp"
#include "../lib/IDisplay.hpp"
```

Classes

- class **Game::AGame**

  *Abstract base class for games, implementing the **IGame** (p. 33) interface.*

Namespaces

- **Game**

  Namespace for game logic components of the Arcade project.

7.1.1 Detailed Description

Header for the AGame abstract class in the Arcade project.

Version

1.0

Date

2024 Epitech Project, 2024 Arcade **Game** (p. 9) Framework

7.2 src/game/IGame.hpp File Reference

Header for the IGame interface in the Arcade project.

```
#include <iostream>
#include <vector>
#include <memory>
#include "../lib/IEntity.hpp"
#include "../core/InfoEntity.hpp"
```

Classes

- class **Game::IGame**

  Interface for game logic in the Arcade project.

Namespaces

- **Game**

  Namespace for game logic components of the Arcade project.

7.2.1 Detailed Description

Header for the IGame interface in the Arcade project.

Version

1.0

Date

2024 Epitech Project, 2024 Arcade **Game** (p. 9) Framework

7.3 src/lib/ADisplay.hpp File Reference

Header for the ADisplay abstract class in the Arcade project.

```
#include <random>
#include <algorithm>
#include "IDisplay.hpp"
```

Classes

- class **Graphic::ADisplay**

  *Abstract base class for display systems, implementing the **IDisplay** (p. 24) interface.*

Namespaces

- **Graphic**
  Namespace for graphical components of the Arcade project.

7.3.1 Detailed Description

Header for the ADisplay abstract class in the Arcade project.

Version

1.0

Date

2024 Epitech Project, 2024 Arcade **Game** (p. 9) Framework

7.4 src/lib/AEntity.hpp File Reference

Header for the AEntity abstract class in the Arcade project.

```
#include "IEntity.hpp"
```

Classes

- class **Graphic::AEntity**
  *Abstract base class for game entities, implementing the **IEntity** (p. 29) interface.*

Namespaces

- **Graphic**
  Namespace for graphical components of the Arcade project.

7.4.1 Detailed Description

Header for the AEntity abstract class in the Arcade project.

Version

1.0

Date

2024 Epitech Project, 2024 Arcade **Game** (p. 9) Framework

7.5 src/lib/IDisplay.hpp File Reference

Header for the IDisplay interface in the Arcade project.

```
#include <iostream>
#include <vector>
#include <memory>
#include "IEntity.hpp"
```

Classes

- class **Graphic::IDisplay**

  Interface defining the display functionality for Arcade games.

Namespaces

- **Graphic**

  Namespace for graphical components of the Arcade project.

7.5.1 Detailed Description

Header for the IDisplay interface in the Arcade project.

Version

1.0

Date

2024 Epitech Project, 2024 Arcade **Game** (p. 9) Framework

7.6 src/lib/IEntity.hpp File Reference

Header for the IEntity interface in the Arcade project.

```
#include <iostream>
#include <memory>
```

Classes

- class **Graphic::IEntity**

  Interface for game entities in the Arcade project.

Namespaces

- **Graphic**

  Namespace for graphical components of the Arcade project.

7.6.1 Detailed Description

Header for the IEntity interface in the Arcade project.

Version

1.0

Date

2024 Epitech Project, 2024 Arcade **Game** (p. 9) Framework