Arcade

Generated by Doxygen 1.9.3

Chapter 1

Arcade

Arcade Interface

1.1

1.2 Building

1.2.1 Command Line (via CMake)

Required tools:

- CMake 3.17 (minimum)

on Linux:

```
# Create the build directory
mkdir build && cd build
# Configure the project
cmake .. -G 'Unix Makefiles' -DCMAKE_BUILD_TYPE=Release
# Build the executable and libraries
cmake --build .
# Return to previous directory
cd -
```

1.3 Documentation

The documatation is available online.

1.3.1 Local Docs

Required tools:

- Doxygen

on Linux:

```
# Run at the root of the project
doxygen
# Open the genrated pages
xdg-open doc/generated/html/index.html
```

Chapter 2

Implemententing a Game or a Graphics Backend

2.1 Game Implementation

2.1.1 Explanation

First, include the IGame interface, (and iostream too, for this example).

```
#include <arcade/IGame.hpp>
#include <arcade/types.hpp>
```

Then, create implement the IGame interface:

```
class ExampleGame : public IGame {
```

Store the game state as private fields:

```
  private:
    State _state;
```

Create the setup methods:

```
  public:
    ExampleGame()
    {
        // ...
    }
    void setup() override final { this->_state = State::Loaded; }
    void loadAssets(IAssetManager &manager, vec2u displaySize) override final
    {
        (void)manager;
        (void)displaySize;
    }
    void close() override final
    {
        // ...
    }
```

Then implement all the remaining methods:

```
    // [...]
    void setState(State state) override final { this->_state = state; }
    State getState() const override final { return this->_state; }
    unsigned int getScore() const override final { return -42; }
    void update(float delta) override final
    {
        (void)delta;
        // ...
    }
    void render(IRenderer &renderer) override final { (void)renderer; }
    void handleEvent(Event &event) override final
    {
        (void)event;
        // ...
    }
};
```

Declare the game's entry point

```
ARCADE_GAME_ENTRY_POINT
{
    std::cout « "[example game]: called entry point" « std::endl;
    return ARCADE_GAME_INSTANCE;
}
```

Finally, initialize the game instance on library load and destroy the instance on unload.

```
[[gnu::constructor]] void onConstruct()
{
    ARCADE_GAME_INSTANCE = new ExampleGame();
    std::cout « "[example game]: constructed" « std::endl;
}
[[gnu::destructor]] void onDestroy()
{
    delete ARCADE_GAME_INSTANCE;
    ARCADE_GAME_INSTANCE = nullptr;
    std::cout « "[example game]: destroyed" « std::endl;
}
```

2.1.2 Full Code

```
#include <arcade/IGame.hpp>
#include <arcade/types.hpp>
#include <iostream>
namespace arcade
{
    struct Event;
    class IRenderer;
    class IAssetManager;
} // namespace arcade
// Using 'using' imports to keep the example clean, do not use in final code!
using ::arcade::Event;
using ::arcade::IAssetManager;
using ::arcade::IGame;
using ::arcade::IRenderer;
using ::arcade::vec2u;
class ExampleGame : public IGame {
  private:
    State _state;
  public:
    ExampleGame()
    {
        // ...
    }
    void setup() override final { this->_state = State::Loaded; }
    void loadAssets(IAssetManager &manager, vec2u displaySize) override final
    {
        (void)manager;
        (void)displaySize;
    }
    void close() override final
    {
        // ...
    }
    // [...]
    void setState(State state) override final { this->_state = state; }
    State getState() const override final { return this->_state; }
    unsigned int getScore() const override final { return -42; }
    void update(float delta) override final
    {
        (void)delta;
        // ...
    }
    void render(IRenderer &renderer) override final { (void)renderer; }
    void handleEvent(Event &event) override final
    {
        (void)event;
        // ...
    }
};
static IGame *ARCADE_GAME_INSTANCE = nullptr;
ARCADE_GAME_ENTRY_POINT
{
    std::cout « "[example game]: called entry point" « std::endl;
    return ARCADE_GAME_INSTANCE;
}
[[gnu::constructor]] void onConstruct()
{
    ARCADE_GAME_INSTANCE = new ExampleGame();
    std::cout « "[example game]: constructed" « std::endl;
}
[[gnu::destructor]] void onDestroy()
{
```

```
    delete ARCADE_GAME_INSTANCE;
    ARCADE_GAME_INSTANCE = nullptr;
    std::cout « "[example game]: destroyed" « std::endl;
}
```

2.2 Display Implementation

2.2.1 Explanation

First, include the IDisplay interface, (and iostream too, for this example).

```
#include <functional>
#include <iostream>
#include <arcade/Color.hpp>
```

Then, create implement the IDisplay interface:

```
class ExampleDisplay : public IDisplay {
```

Then implement the needed methods:

```
  public:
    ExampleDisplay()
    {
        // ...
    }
    void setup() override final
    {
        // ...
    }
    void close() override final
    {
        // ...
    }
    Type getType() const override final { return Type::Graphical2D; }
    vec2u getSize() const override final { return {0, 0}; }
    virtual bool pollEvent(Event &event) override final
    {
        (void)event;
        return false;
    }
    void clear(Color color, DefaultColor backup) override final
    {
        (void)color;
        (void)backup;
    }
    void render(std::function<void(IRenderer &)> drawer) override final { (void)drawer; }
    void display() override final
    {
        // ...
    }
    void loadAssets(std::function<void(IAssetManager &)> loader) override final { (void)loader; }
};
```

Declare the display's entry point

```
ARCADE_DISPLAY_ENTRY_POINT
{
    std::cout « "[example display]: called entry point" « std::endl;
    return DISPLAY_INSTANCE;
}
```

Finally, initialize the display instance on library load and destroy the instance on unload.

```
[[gnu::constructor]] void onConstruct()
{
    DISPLAY_INSTANCE = new ExampleDisplay();
    std::cout « "[example display]: constructed" « std::endl;
}
[[gnu::destructor]] void onDestroy()
{
    delete DISPLAY_INSTANCE;
    DISPLAY_INSTANCE = nullptr;
    std::cout « "[example display]: destroyed" « std::endl;
}
```

2.2.2 Full Code

```
#include <functional>
#include <iostream>
#include <arcade/Color.hpp>
#include <arcade/IDisplay.hpp>
#include <arcade/types.hpp>
namespace arcade
{
    struct Event;
    class IAsset;
    class IAssetManager;
    class IGameObject;
    class IRenderer;
} // namespace arcade
// Using 'using' imports to keep the example clean, do not use in final code!
using ::arcade::Color;
using ::arcade::DefaultColor;
using ::arcade::Event;
using ::arcade::IAsset;
using ::arcade::IAssetManager;
using ::arcade::IDisplay;
using ::arcade::IGameObject;
using ::arcade::IRenderer;
using ::arcade::vec2u;
class ExampleDisplay : public IDisplay {
  public:
    ExampleDisplay()
    {
        // ...
    }
    void setup() override final
    {
        // ...
    }
    void close() override final
    {
        // ...
    }
    Type getType() const override final { return Type::Graphical2D; }
    vec2u getSize() const override final { return {0, 0}; }
    virtual bool pollEvent(Event &event) override final
    {
        (void)event;
        return false;
    }
    void clear(Color color, DefaultColor backup) override final
    {
        (void)color;
        (void)backup;
    }
    void render(std::function<void(IRenderer &)> drawer) override final { (void)drawer; }
    void display() override final
    {
        // ...
    }
    void loadAssets(std::function<void(IAssetManager &)> loader) override final { (void)loader; }
};
static IDisplay *DISPLAY_INSTANCE = nullptr;
ARCADE_DISPLAY_ENTRY_POINT
{
    std::cout << "[example display]: called entry point" << std::endl;
    return DISPLAY_INSTANCE;
}
[[gnu::constructor]] void onConstruct()
{
    DISPLAY_INSTANCE = new ExampleDisplay();
    std::cout << "[example display]: constructed" << std::endl;
}
[[gnu::destructor]] void onDestroy()
{
    delete DISPLAY_INSTANCE;
    DISPLAY_INSTANCE = nullptr;
    std::cout << "[example display]: destroyed" << std::endl;
}
```

Chapter 3

How to contribute

3.0.0.1 Did you find a bug? / Do you want to suggest somthing?

- Create an issue at `this issue page`.

3.0.0.2 Do you want to fix an issue?

- Create a branch formatted as `fix/<ISSUENUMBER>-<TITLE>` for bug fixes or `feature/<ISSUENUMBER>-<TITL` for features, example: `fix/4221-infinite-loop`.
- Submit a `pull request`.
- Once validated, merge to PR to `master` and remove the source branch (with `git branch -D <branch_name>`.

3.0.1 How to title commits?

- Use present tense (avoid past tense).
- The title of the commit must be a summuary of the content and not be too long (less than 80 characters).
- Prefer putting detailed informations inside the commit's description.
- Example:

```
$> git commit -m 'Fix infinite loop when pressing Alt-F4
This was caused by a missing check in the event loop
The program now checks when the window is set to close'
```

3.0.1.1 DOs and DONTs

- :x: **DONT**: Push to the `master` (or `main`) branch for any reason, please submit a PR instead.
- :x: **DONT**: Create a branch with your username as the title
- :heavy_check_mark: **DO**: Commit often! allows everyone to see your progress and eventually make suggestions on it.
- :heavy_check_mark: **DO**: Format your code, using either `clang-format` directly or your IDE's capabilities (and yes, VSCode can format your code for you!)

Thanks! :heart: :heart: :heart:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

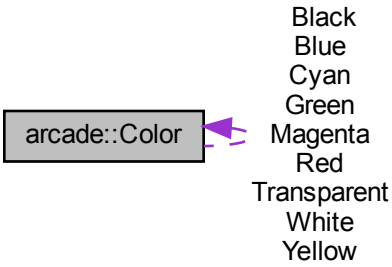
Class Documentation

6.1 arcade::Color Class Reference

32-bit ARGB Color.

`#include <Color.hpp>`

Collaboration diagram for arcade::Color:



Public Member Functions

- constexpr **Color** ()

  Default constructor. Create a black color (all components set to 0).
- constexpr Color (uint32_t color)
- constexpr Color (std::byte red, std::byte green, std::byte blue, std::byte alpha=std::byte(0))
- constexpr uint32_t toInteger () const

Public Attributes

- std::byte **a**

  Alpha component. 255 means transparent.
- std::byte **r**

  Red component.
- std::byte **g**

  Green component.
- std::byte **b**

  Blue component.

Static Public Attributes

- static const Color **Black** = Color(0x00000000)

  Black predefined color.
- static const Color **White** = Color(0x00ffffff)

  White predefined color.
- static const Color **Transparent** = Color(0xffffffff)

  White Transparent predefined color.
- static const Color **Red** = Color(0x00ff0000)

  Red predefined color.
- static const Color **Green** = Color(0x0000ff00)

  Green predefined color.
- static const Color **Blue** = Color(0x000000ff)

  Blue predefined color.
- static const Color **Yellow** = Color(0x00ffff00)

  Yellow predefined color.
- static const Color **Magenta** = Color(0x00ff00ff)

  Magenta predefined color.
- static const Color **Cyan** = Color(0x0000ffff)

  Cyan predefined color.

6.1.1 Detailed Description

32-bit ARGB Color.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Color() [1/2]

```
constexpr arcade::Color::Color (
            uint32_t color )  [inline], [constexpr]
```

Construct a color from an 32-bit ARGB value.

Parameters

color	32-bit ARGB color.

6.1.2.2 Color() [2/2]

```
constexpr arcade::Color::Color (
            std::byte red,
            std::byte green,
            std::byte blue,
            std::byte alpha = std::byte(0) )  [inline], [constexpr]
```

Construct a color from its components values.

Parameters

red	red component.
green	green component.
blue	blue component.
alpha	alpha component.

6.1.3 Member Function Documentation

6.1.3.1 toInteger()

`constexpr uint32_t arcade::Color::toInteger ( ) const [inline], [constexpr]`
Convert a color to a 32-bit ARGB integer.

Returns

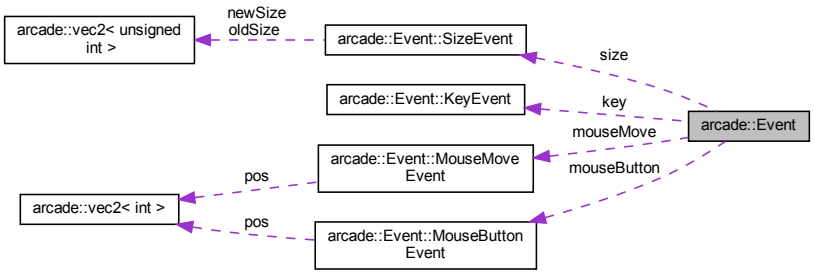
uint32_t 32-bit ARGB color.

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

- src/interface/include/arcade/Color.hpp

6.2 arcade::Event Struct Reference

`#include <Event.hpp>`
Collaboration diagram for arcade::Event:



Classes

- struct KeyEvent

  Keyboard event parameters (Event::Type::KeyPressed, Event::Type::KeyReleased).
- struct MouseButtonEvent

  Mouse buttons events parameters (Event::Type::MouseButtonPressed, Event::Type::MouseButtonReleased).
- struct MouseMoveEvent

  Mouse move event parameters (Event::Type::MouseMoved)
- struct SizeEvent

  Size events parameters (Event::Type::Resized).

Public Types

- enum class MouseButton { Left , Right , Middle , Count }

  Represents a mouse button.
- enum class Type {
  Closed , Resized , KeyPressed , KeyReleased ,
  MouseButtonPressed , MouseButtonReleased , MouseMoved , Count }

  Enumeration of the different types of events.
- using Key = char

Public Attributes

- Type **type**

  Type of the event.

-

  union {
  SizeEvent **size**
  Size event parameters (Event::Resized).
  KeyEvent **key**
  Key event parameters (Event::KeyPressed, Event::KeyReleased).
  MouseMoveEvent **mouseMove**
  Mouse move event parameters (Event::MouseMoved).
  MouseButtonEvent **mouseButton**
  Mouse button event parameters (Event::MouseButtonPressed, Event::MouseButtonReleased).
  };

6.2.1 Detailed Description

Contains an event's data.
Event data is fetched by first checking its type and then accessing the correspoding member.

6.2.2 Member Typedef Documentation

6.2.2.1 Key

```
using arcade::Event::Key = char
```

Key code for keyboard events.
Keys are represented by their corresponding character, meaning that some keys are not currently representable using this format (such as the Enter key).

6.2.3 Member Enumeration Documentation

6.2.3.1 MouseButton

```
enum class arcade::Event::MouseButton  [strong]
```

Represents a mouse button.

Enumerator

Left	The left mouse button.
Right	The right mouse button.
Middle	The middle (wheel) mouse button.
Count	Keep last – the total number of mouse buttons.

6.2.3.2 Type

```
enum class arcade::Event::Type  [strong]
```

Enumeration of the different types of events.

Enumerator

Closed	The window requested to be closed (no data).
Resized	The window was resized (data in Event.size).
KeyPressed	A key was pressed (data in Event.key).
KeyReleased	A key was released (data in Event.key).
MouseButtonPressed	A mouse button was pressed (data in Event.mouseButton).
MouseButtonReleased	A mouse button was released (data in Event.mouseButton).
MouseMoved	The mouse cursor moved (data in Event.mouseMove).
Count	Keep last – the total number of event types.

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

- src/interface/include/arcade/Event.hpp

6.3 arcade::IAsset Class Reference

`#include <IAsset.hpp>`

Public Types

- enum class Type { Font , Texture , CharSet }

  The type of an asset.

Public Member Functions

- virtual Type getType () const =0

6.3.1 Detailed Description

Generic asset.
An asset can be any a font, an image, a texture, and more.

6.3.2 Member Enumeration Documentation

6.3.2.1 Type

`enum class arcade::IAsset::Type [strong]`
The type of an asset.

Enumerator

Font	A font.
Texture	A graphical sprite made of pixels.
CharSet	A set of characters that acts as a sprite (aka: ASCII-art).

6.3.3 Member Function Documentation

6.3.3.1 getType()

`virtual Type arcade::IAsset::getType ( ) const [pure virtual]`

Returns

The type of this asset.

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

- src/interface/include/arcade/IAsset.hpp

6.4 arcade::IAssetManager Class Reference

Loads assets and creates game objects.
`#include <IAssetManager.hpp>`

Public Member Functions

- virtual IDisplay::Type getType () const =0
- virtual std::unique_ptr< IAsset > loadAsset (std::string_view name, IAsset::Type type)=0
- virtual std::unique_ptr< IGameObject > createTextObject (std::string_view text, IAsset const *font=nullptr) const =0
- virtual std::unique_ptr< IGameObject > createRectObject (vec2u size, IAsset const *texture=nullptr) const =0

6.4.1 Detailed Description

Loads assets and creates game objects.

6.4.2 Member Function Documentation

6.4.2.1 createRectObject()

```
virtual std::unique_ptr< IGameObject > arcade::IAssetManager::createRectObject (
            vec2u size,
            IAsset const * texture = nullptr ) const  [pure virtual]
```

Creates a textured rectangle object.

Parameters

size	The dimensions (in units) of the rectangle.
texture	The texture to use. If `nullptr`, the texture isn't used.

Exceptions

std::logic_error	When `texture` is not a texture asset.
std::logic_error	When `texture` is `nullptr` and a RectObject can't be created without a texture.

Returns

A boxed IGameObject instance.

6.4.2.2 createTextObject()

```
virtual std::unique_ptr< IGameObject > arcade::IAssetManager::createTextObject (
            std::string_view text,
            IAsset const * font = nullptr ) const  [pure virtual]
```

Creates a text (as in string) object instance.

Parameters

text	The string to display.
font	The font to use. If `nullptr`, the font isn't used.

Exceptions

std::logic_error	When `font` is not a font asset.
std::logic_error	When `font` is `nulltptr` and a TextObject can't be created without a font.

Returns

A boxed IGameObject instance.

6.4.2.3 getType()

```
virtual IDisplay::Type arcade::IAssetManager::getType ( ) const  [pure virtual]
```

Returns

The type of the linked display output.

6.4.2.4 loadAsset()

```
virtual std::unique_ptr< IAsset > arcade::IAssetManager::loadAsset (
            std::string_view name,
            IAsset::Type type )  [pure virtual]
```

Fetches an asset by name, loading it if necessary.
The returned IAsset instance is a reference to the real underlying asset, when switching displays, all existing instances of IAsset will attempt to convert to equivalent assets for the new display mode.

Parameters

name	The name of the requested asset.
type	Which type of asset to fetch?

Returns

A reference to the loaded asset, or a null reference if the requested asset failed to load.

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

- src/interface/include/arcade/IAssetManager.hpp

6.5 arcade::IDisplay Class Reference

```
#include <IDisplay.hpp>
```

Public Types

- enum class Type { Terminal , Graphical2D }

The graphics mode.

- using EntryPoint = IDisplay *(*)()

Public Member Functions

- virtual void setup ()=0
- virtual void close ()=0
- virtual Type getType () const =0
- virtual vec2u getSize () const =0
- virtual bool pollEvent (Event &event)=0
- virtual void clear (Color color, DefaultColor backup)=0
- virtual void render (std::function< void(IRenderer &)> drawer)=0
- virtual void display ()=0
- virtual void loadAssets (std::function< void(IAssetManager &)> loader)=0

Static Public Attributes

- static constexpr std::string_view **ENTRY_POINT** = "arcade_DisplayEntryPoint"

  Expected name of the Display entry point.

6.5.1 Detailed Description

Graphics backend.
Instances of IDisplay are reponsible for managing the display, events and assets of IGame instances.

6.5.2 Member Typedef Documentation

6.5.2.1 EntryPoint

```
using arcade::IDisplay::EntryPoint = IDisplay *(*)()
```

Type of the entry point of the library to get an instance of IGame. The function used as EntryPoint must be named as the DisplayEntryPointName below.

6.5.3 Member Enumeration Documentation

6.5.3.1 Type

```
enum class arcade::IDisplay::Type  [strong]
```

The graphics mode.

Enumerator

Terminal	Text-only output.
Graphical2D	Flat graphical output.

6.5.4 Member Function Documentation

6.5.4.1 clear()

```
virtual void arcade::IDisplay::clear (
            Color color,
            DefaultColor backup )  [pure virtual]
```

Clears the render target by filling with the given color.

Note

This function must be called before IDisplay::render().

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

Parameters

color	32-bit ARGB color to set.
backup	color to set if the display doesn't support 32-bit ARGB colors.

6.5.4.2 close()

```
virtual void arcade::IDisplay::close ( )  [pure virtual]
```

Releases the ressources allocated by this graphics backend.
Each call to IDisplay::close() **must** be preceded by a call to IDisplay::setup(). Calling this function again without calling IDisplay::setup() leads to **undefined behavior**.

6.5.4.3 display()

```
virtual void arcade::IDisplay::display ( )  [pure virtual]
```

Displays the rendered frame to the screen

Note

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

The rendered frame is immediately shown to the user at the end of the call.

6.5.4.4 getSize()

```
virtual vec2u arcade::IDisplay::getSize ( ) const  [pure virtual]
```

Note

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

Returns

The size of the display, in units.

6.5.4.5 getType()

```
virtual Type arcade::IDisplay::getType ( ) const  [pure virtual]
```

Note

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

Returns

The type of display output.

6.5.4.6 loadAssets()

```
virtual void arcade::IDisplay::loadAssets (
            std::function< void(IAssetManager &)> loader )  [pure virtual]
```

Calls a game's asset loading function.

Note

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

Parameters

loader	A function that loads assets and game objects using the supplied asset manager.

6.5.4.7 pollEvent()

```
virtual bool arcade::IDisplay::pollEvent (
            Event & event )  [pure virtual]
```

Fetches the next event in the event queue, this operation is non-blocking.

Note

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

Parameters

out	*event*	Where the event will be stored, may be uninitialized.

Returns

Whether an event was loaded into `event`, false means that the event queue is currently empty.

6.5.4.8 render()

```
virtual void arcade::IDisplay::render (
            std::function< void(IRenderer &)> drawer )  [pure virtual]
```

Renders objects.

Note

Calling this method without calling IDisplay::setup() leads to **undefined behavior**.

Nothing is shown to the user until IDisplay::display() is called.

Parameters

drawer	A function that renders game objects using the supplied renderer.

6.5.4.9 setup()

```
virtual void arcade::IDisplay::setup ( )  [pure virtual]
```

Intializes this graphics backend.
Each call to IDisplay::setup() **must** be followed by a call to IDisplay::close(). Calling this function again without calling IDisplay::close() leads to **undefined behavior**.

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

- src/interface/include/arcade/IDisplay.hpp

6.6 arcade::IGame Class Reference

A game instance.
`#include <IGame.hpp>`

Public Types

- enum class State { Loaded , Running , Paused , Ended }

  A game's state.
- using EntryPoint = IGame ∗(∗)()

Public Member Functions

- virtual void setup ()=0
- virtual void loadAssets (IAssetManager &manager, vec2u displaySize)=0
- virtual void close ()=0
- virtual void setState (State state)=0
- virtual State getState () const =0
- virtual unsigned int getScore () const =0
- virtual void update (float delta)=0
- virtual void render (IRenderer &renderer)=0
- virtual void handleEvent (Event &event)=0

Static Public Attributes

- static constexpr std::string_view **ENTRY_POINT** = "arcade_GameEntryPoint"

  Expected name of the Game entry point.

6.6.1 Detailed Description

A game instance.

6.6.2 Member Typedef Documentation

6.6.2.1 EntryPoint

`using arcade::IGame::EntryPoint = IGame ∗(∗)()`
Type of the entry point of the library to get an instance of IGame. The function used as EntryPoint must be named as the GameEntryPointName below.

6.6.3 Member Enumeration Documentation

6.6.3.1 State

`enum class arcade::IGame::State [strong]`
A game's state.

Enumerator

Loaded	The initial state of the game.

Enumerator

Running	After the first update.
Paused	When a previously running game is paused.
Ended	When the game is ended.

6.6.4 Member Function Documentation

6.6.4.1 close()

```
virtual void arcade::IGame::close ( )  [pure virtual]
```

Releases the ressources allocated by this game.
Each call to IGame::close() **must** be preceded by a call to IGame::setup(). Calling this function again without calling IGame::setup() leads to **undefined behavior**.

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

6.6.4.2 getScore()

```
virtual unsigned int arcade::IGame::getScore ( ) const  [pure virtual]
```

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Returns

The current score of the player.

6.6.4.3 getState()

```
virtual State arcade::IGame::getState ( ) const  [pure virtual]
```

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Returns

The current state.

6.6.4.4 handleEvent()

```
virtual void arcade::IGame::handleEvent (
            Event & event )  [pure virtual]
```

Handles the given event.

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Parameters

event	The event to process.

6.6.4.5 loadAssets()

```
virtual void arcade::IGame::loadAssets (
            IAssetManager & manager,
            vec2u displaySize )  [pure virtual]
```

(Re)-loads assets and game objects.
This method is called each time the underlying graphics backend is switched.

Parameters

manager	The assets manager.
displaySize	The size of the display.

6.6.4.6 render()

```
virtual void arcade::IGame::render (
            IRenderer & renderer )  [pure virtual]
```

Draw the game GameObjects.

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Parameters

renderer	The renderer.

6.6.4.7 setState()

```
virtual void arcade::IGame::setState (
            State state )  [pure virtual]
```

Alters the state of the game.

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Parameters

state	The new state.

6.6.4.8 setup()

```
virtual void arcade::IGame::setup ( )  [pure virtual]
```

Intializes this game.

Each call to IGame::setup() **must** be followed by a call to IGame::close(). Calling this function again without calling IGame::close() leads to **undefined behavior**.

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Parameters

display	The starting display manager.

6.6.4.9 update()

```
virtual void arcade::IGame::update (
            float delta )  [pure virtual]
```

Updates the game logic.

Note

Calling this method without calling IGame::setup() leads to **undefined behavior**.

Parameters

delta	The time (in seconds) elapsed since the last update.

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

- src/interface/include/arcade/IGame.hpp

6.7 arcade::IGameObject Class Reference

A movable, drawable game object such as text or sprites.
`#include <IGameObject.hpp>`

Public Types

- enum class Type { Text , Rect }

  The type of a game object.

Public Member Functions

- virtual Type getType () const =0
- virtual vec2u getSize () const =0
- virtual vec2i getPosition () const =0
- virtual void setPosition (vec2i pos)=0
- virtual void setForeground (Color color, DefaultColor backup=DefaultColor::White)=0
- virtual void setBackground (Color color, DefaultColor backup=DefaultColor::Transparent)=0

6.7.1 Detailed Description

A movable, drawable game object such as text or sprites.

6.7.2 Member Enumeration Documentation

6.7.2.1 Type

```
enum class arcade::IGameObject::Type  [strong]
```

The type of a game object.

Enumerator

Text	Textual object.
Rect	Textured rectangle object.

6.7.3 Member Function Documentation

6.7.3.1 getPosition()

```
virtual vec2i arcade::IGameObject::getPosition ( ) const  [pure virtual]
```

TODO: we might want to change to type to an vec2f instead.

Returns

The position (in units) of thiss object.

6.7.3.2 getSize()

```
virtual vec2u arcade::IGameObject::getSize ( ) const  [pure virtual]
```

Returns

The dimensions (in units) of this object.

6.7.3.3 getType()

```
virtual Type arcade::IGameObject::getType ( ) const  [pure virtual]
```

Returns

The type of this object.

6.7.3.4 setBackground()

```
virtual void arcade::IGameObject::setBackground (
            Color color,
            DefaultColor backup = DefaultColor::Transparent )  [pure virtual]
```

Change the background color of the game object.

Parameters

color	32-bit ARGB color to set.
backup	color to set if the display doesn't support 32-bit ARGB colors.

6.7.3.5 setForeground()

```
virtual void arcade::IGameObject::setForeground (
            Color color,
            DefaultColor backup = DefaultColor::White )  [pure virtual]
```

Change the foreground color of the game object.

Parameters

color	32-bit ARGB Color to set.
backup	color to set if the display doesn't support 32-bit ARGB colors.

6.7.3.6 setPosition()

```
virtual void arcade::IGameObject::setPosition (
            vec2i pos )  [pure virtual]
```

Moves the game object.

Parameters

pos	The new position.

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

- src/interface/include/arcade/IGameObject.hpp

6.8 arcade::IRenderer Class Reference

Rendering interface.

`#include <IRenderer.hpp>`

Public Member Functions

- virtual void draw (IGameObject const &object)=0

6.8.1 Detailed Description

Rendering interface.

6.8.2 Member Function Documentation

6.8.2.1 draw()

```
virtual void arcade::IRenderer::draw (
            IGameObject const & object )  [pure virtual]
```

Draws a game object to the display's internal buffer(s).
Drawn IGameObject instances will not display until the next call to IDisplay::render().

Parameters

object	The object to draw.

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

- src/interface/include/arcade/IRenderer.hpp

6.9 arcade::Event::KeyEvent Struct Reference

Keyboard event parameters (Event::Type::KeyPressed, Event::Type::KeyReleased).
`#include <Event.hpp>`

Public Attributes

- bool **alt**

  Is the Alt key pressed?
- bool **control**

  Is the Control key pressed?
- bool **shift**

  Is the Shift key pressed?
- bool **system**

  Is the System key pressed?
- Key **code**

  Code of the key that has been pressed.

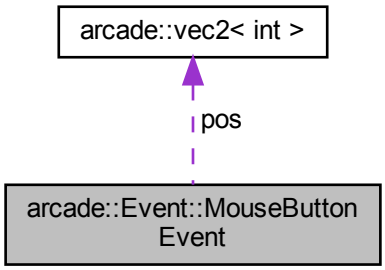
6.9.1 Detailed Description

Keyboard event parameters (Event::Type::KeyPressed, Event::Type::KeyReleased).
The documentation for this struct was generated from the following file:

- src/interface/include/arcade/Event.hpp

6.10 arcade::Event::MouseButtonEvent Struct Reference

Mouse buttons events parameters (Event::Type::MouseButtonPressed, Event::Type::MouseButtonReleased).
`#include <Event.hpp>`
Collaboration diagram for arcade::Event::MouseButtonEvent:



Public Attributes

- MouseButton **button**

  Code of the button that has been pressed.
- vec2i **pos**

  Position of the mouse pointer, relative to the left of the owner window.

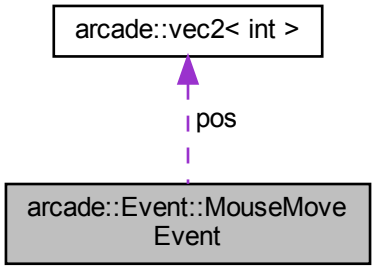
6.10.1 Detailed Description

Mouse buttons events parameters (Event::Type::MouseButtonPressed, Event::Type::MouseButtonReleased).
The documentation for this struct was generated from the following file:

- src/interface/include/arcade/Event.hpp

6.11 arcade::Event::MouseMoveEvent Struct Reference

Mouse move event parameters (Event::Type::MouseMoved)
`#include <Event.hpp>`
Collaboration diagram for arcade::Event::MouseMoveEvent:



Public Attributes

- vec2i **pos**

  Position of the mouse pointer, relative to the left of the owner window.

6.11.1 Detailed Description

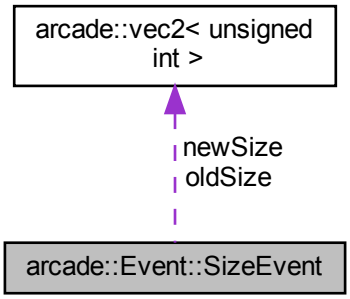
Mouse move event parameters (Event::Type::MouseMoved)
The documentation for this struct was generated from the following file:

- src/interface/include/arcade/Event.hpp

6.12 arcade::Event::SizeEvent Struct Reference

Size events parameters (Event::Type::Resized).
`#include <Event.hpp>`

Collaboration diagram for arcade::Event::SizeEvent:



Public Attributes

- vec2u **oldSize**

  Old size, in units.
- vec2u **newSize**

  New size, in units.

6.12.1 Detailed Description

Size events parameters (Event::Type::Resized).
The documentation for this struct was generated from the following file:

- src/interface/include/arcade/Event.hpp

6.13 arcade::vec2< T > Struct Template Reference

A 2D vector.
`#include <types.hpp>`

Public Member Functions

- template<typename U >
  constexpr **operator vec2**< **U** > () const

  Direct vector casting support.
- constexpr vec2< T > **operator+** () const
- constexpr vec2< T > **operator-** () const
- constexpr vec2< T > **operator+** (T const &v) const
- constexpr vec2< T > **operator+** (vec2< T > const &v) const
- constexpr vec2< T > **operator-** (T const &v) const
- constexpr vec2< T > **operator-** (vec2< T > const &v) const
- constexpr vec2< T > **operator*** (T const &v) const
- constexpr T **operator*** (vec2< T > const &v) const

  Dot product.
- constexpr vec2< T > **operator/** (T const &v) const
- constexpr vec2< T > & **operator+=** (T const &v)
- constexpr vec2< T > & **operator+=** (vec2< T > const &v)

- constexpr vec2< T > & **operator-=** (T const &v)
- constexpr vec2< T > & **operator-=** (vec2< T > const &v)
- constexpr vec2< T > & **operator*=** (T const &v)
- constexpr vec2< T > & **operator/=** (T const &v)
- constexpr bool **operator==** (vec2< T > const &v) const
- constexpr bool **operator!=** (vec2< T > const &v) const

Public Attributes

- T **x**

  x coordinate.
- T **y**

  y coordinate.

6.13.1 Detailed Description

template<typename T>
struct arcade::vec2< T >

A 2D vector.
The documentation for this struct was generated from the following file:

- src/interface/include/arcade/types.hpp

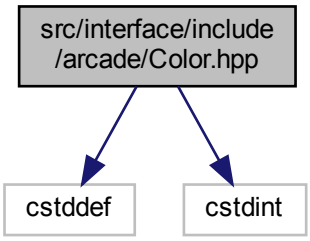
Chapter 7

File Documentation

7.1 src/interface/include/arcade/Color.hpp File Reference

```
#include <cstddef>
#include <cstdint>
```

Include dependency graph for Color.hpp:



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

7.2 Color.hpp

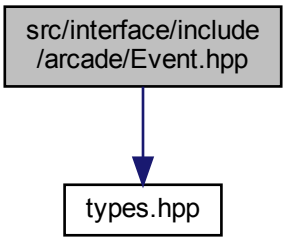
Go to the documentation of this file.

```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade
4 ** File description:
5 ** Color
6 */
7
11
12 #ifndef ARCADE_COLOR_HPP_
13 #define ARCADE_COLOR_HPP_
14
15 #include <cstddef>
16 #include <cstdint>
17
18 namespace arcade
19 {
21     enum class DefaultColor {
22         Black,
23         White,
24         Transparent,
25         Red,
26         Green,
27         Blue,
28         Yellow,
```

```
29          Magenta,
30          Cyan,
31      };
32
34      class Color {
35        public:
37          constexpr inline Color() : a(std::byte(0)), r(std::byte(0)), g(std::byte(0)), b(std::byte(0))
38          {
39          }
40
44          constexpr inline Color(uint32_t color)
45              : a(std::byte((color & (0xff « 24)) » 24)), r(std::byte((color & (0xff « 16)) » 16)),
46                g(std::byte((color & (0xff « 8)) » 8)), b(std::byte(color & 0xff))
47          {
48          }
49
56          constexpr inline Color(std::byte red, std::byte green, std::byte blue, std::byte alpha =
    std::byte(0))
57              : a(alpha), r(red), g(green), b(blue)
58          {
59          }
60
64          constexpr inline uint32_t toInteger() const
65          {
66              return (std::to_integer<uint32_t>(a) « 24) | (std::to_integer<uint32_t>(r) « 16)
67                  | (std::to_integer<uint32_t>(g) « 8) | std::to_integer<uint32_t>(b);
68          }
69
71          static const Color Black;
73          static const Color White;
75          static const Color Transparent;
77          static const Color Red;
79          static const Color Green;
81          static const Color Blue;
83          static const Color Yellow;
85          static const Color Magenta;
87          static const Color Cyan;
88
90          std::byte a;
92          std::byte r;
94          std::byte g;
96          std::byte b;
97      };
98
99      constexpr Color Color::Black = Color(0x00000000);
100      constexpr Color Color::White = Color(0x00ffffff);
101      constexpr Color Color::Transparent = Color(0xffffffff);
102      constexpr Color Color::Red = Color(0x00ff0000);
103      constexpr Color Color::Green = Color(0x0000ff00);
104      constexpr Color Color::Blue = Color(0x000000ff);
105      constexpr Color Color::Yellow = Color(0x00ffff00);
106      constexpr Color Color::Magenta = Color(0x00ff00ff);
107      constexpr Color Color::Cyan = Color(0x0000ffff);
108
109 } // namespace arcade
110
111 #endif /* !ARCADE_COLOR_HPP_ */
```

7.3 src/interface/include/arcade/Event.hpp File Reference

`#include "types.hpp"`
Include dependency graph for Event.hpp:



Classes

- struct arcade::Event
- struct arcade::Event::SizeEvent

  Size events parameters (Event::Type::Resized).
- struct arcade::Event::KeyEvent

  Keyboard event parameters (Event::Type::KeyPressed, Event::Type::KeyReleased).
- struct arcade::Event::MouseMoveEvent

  Mouse move event parameters (Event::Type::MouseMoved)
- struct arcade::Event::MouseButtonEvent

  Mouse buttons events parameters (Event::Type::MouseButtonPressed, Event::Type::MouseButtonReleased).

7.3.1 Detailed Description

The Event data structure.

7.4 Event.hpp

Go to the documentation of this file.

```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade: Interface
4 ** File description:
5 ** Event data structure
6 */
7
11
12 #ifndef ARCADE_EVENT_HPP_
13 #define ARCADE_EVENT_HPP_
14
15 #include "types.hpp"
16
17 namespace arcade
18 {
22     struct Event {
27         using Key = char;
28
30         enum class MouseButton {
32             Left,
34             Right,
36             Middle,
37
39             Count
40         };
```

```
41
43          struct SizeEvent {
45              vec2u oldSize;
47              vec2u newSize;
48          };
49
51          struct KeyEvent {
53              bool alt;
55              bool control;
57              bool shift;
59              bool system;
61              Key code;
62          };
63
65          struct MouseMoveEvent {
67              vec2i pos;
68          };
69
71          struct MouseButtonEvent {
73              MouseButton button;
75              vec2i pos;
76          };
77
79          enum class Type {
81              Closed,
83              Resized,
85              KeyPressed,
87              KeyReleased,
89              MouseButtonPressed,
91              MouseButtonReleased,
93              MouseMoved,
94
96              Count
97          };
98
99          // Member data
100
102          Type type;
103
104          union {
106              SizeEvent size;
108              KeyEvent key;
110              MouseMoveEvent mouseMove;
112              MouseButtonEvent mouseButton;
113          };
114      };
115 } // namespace arcade
116
117 #endif // !defined(ARCADE_EVENT_HPP_)
```

7.5 src/interface/include/arcade/IAsset.hpp File Reference

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

Classes

- class arcade::IAsset

7.5.1 Detailed Description

The asset interface.

7.6 IAsset.hpp

Go to the documentation of this file.

```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade: Interface
4 ** File description:
5 ** The Asset interface
6 */
7
11
12 #ifndef ARCADE_IASSET_HPP_
13 #define ARCADE_IASSET_HPP_
14
15 namespace arcade
```

```
16 {
20     class IAsset {
21       public:
23         enum class Type {
25             Font,
27             Texture,
29             CharSet,
30         };
31
32         virtual ~IAsset() = default;
33
35         virtual Type getType() const = 0;
36     };
37 } // namespace arcade
38
39 #endif // !defined(ARCADE_IASSET_HPP_)
```

7.7 src/interface/include/arcade/IAssetManager.hpp File Reference

```
#include <memory>
#include <string_view>
#include "IAsset.hpp"
#include "IDisplay.hpp"
#include "types.hpp"
```

Include dependency graph for IAssetManager.hpp:

Classes

- class arcade::IAssetManager

  Loads assets and creates game objects.

7.7.1 Detailed Description

Creation of Assets and Game Objects.

7.8 IAssetManager.hpp

Go to the documentation of this file.

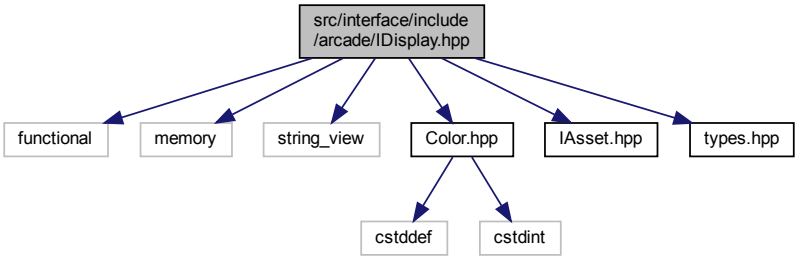
```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade: Interface
4 ** File description:
5 ** Creation of Assets and Game objects
6 */
7
11
12 #ifndef ARCADE_IASSET_MANAGER_HPP_
13 #define ARCADE_IASSET_MANAGER_HPP_
14
15 #include <memory>
16 #include <string_view>
17
18 #include "IAsset.hpp"
19 #include "IDisplay.hpp"
20 #include "types.hpp"
21
22 namespace arcade
23 {
24     class IGameObject;
25
27     class IAssetManager {
28       public:
29         virtual ~IAssetManager() = default;
30
32         virtual IDisplay::Type getType() const = 0;
33
43         virtual std::unique_ptr<IAsset> loadAsset(std::string_view name, IAsset::Type type) = 0;
44
54         virtual std::unique_ptr<IGameObject> createTextObject(
55             std::string_view text, IAsset const *font = nullptr) const = 0;
56
66         virtual std::unique_ptr<IGameObject> createRectObject(vec2u size, IAsset const *texture =
       nullptr) const = 0;
```

```
67     };
68 } // namespace arcade
69
70 #endif /* !ARCADE_IASSET_MANAGER_HPP_ */
```

7.9 src/interface/include/arcade/IDisplay.hpp File Reference

```
#include <functional>
#include <memory>
#include <string_view>
#include "Color.hpp"
#include "IAsset.hpp"
#include "types.hpp"
```

Include dependency graph for IDisplay.hpp:



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

Classes

- class arcade::IDisplay

Macros

- #define **ARCADE_DISPLAY_ENTRY_POINT** extern "C" ::arcade::IDisplay *arcade_DisplayEntryPoint()

  Entry point to get an instance of IDisplay.

7.9.1 Detailed Description

The display interface.

7.10 IDisplay.hpp

Go to the documentation of this file.

```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade: Interface
4 ** File description:
5 ** The display interface
6 */
7
11
12 #ifndef ARCADE_IDISPLAY_HPP_
13 #define ARCADE_IDISPLAY_HPP_
14
16 #define ARCADE_DISPLAY_ENTRY_POINT extern "C" ::arcade::IDisplay *arcade_DisplayEntryPoint()
17
18 #include <functional>
19 #include <memory>
```

```
20 #include <string_view>
21
22 #include "Color.hpp"
23 #include "IAsset.hpp"
24 #include "types.hpp"
25
26 namespace arcade
27 {
28     class IGameObject;
29     struct Event;
30     class IRenderer;
31     class IAssetManager;
32
36     class IDisplay {
37       public:
39         enum class Type {
41             Terminal,
43             Graphical2D,
44         };
45
48         using EntryPoint = IDisplay *(*)();
50         static constexpr std::string_view ENTRY_POINT = "arcade_DisplayEntryPoint";
51
52         virtual ~IDisplay() = default;
53
58         virtual void setup() = 0;
59
64         virtual void close() = 0;
65
69         virtual Type getType() const = 0;
70
74         virtual vec2u getSize() const = 0;
75
83         virtual bool pollEvent(Event &event) = 0;
84
93         virtual void clear(Color color, DefaultColor backup) = 0;
94
102          virtual void render(std::function<void(IRenderer &)> drawer) = 0;
103
109          virtual void display() = 0;
110
116          virtual void loadAssets(std::function<void(IAssetManager &)> loader) = 0;
117     };
118 } // namespace arcade
119
120 #endif // !defined(ARCADE_DISPLAY_HPP_)
```

7.11 src/interface/include/arcade/IGame.hpp File Reference

```
#include "types.hpp"
#include <string_view>
```

Include dependency graph for IGame.hpp:

Classes

- class arcade::IGame

  A game instance.

Macros

- #define **ARCADE_GAME_ENTRY_POINT** extern "C" ::arcade::IGame *arcade_GameEntryPoint()

  Entry point to get an instance of IGame.

7.11.1 Detailed Description

The game interface.

7.12 IGame.hpp

Go to the documentation of this file.

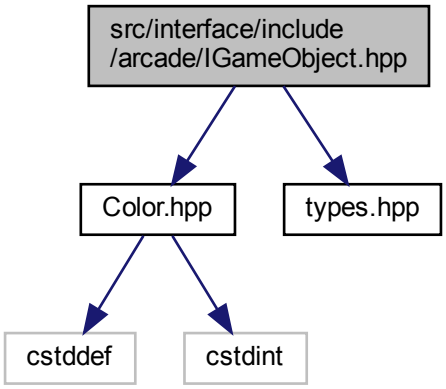
```
1 /*
2 ** EPITECH PROJECT, 2022
```

```
3 ** Arcade: Interface
4 ** File description:
5 ** The game interface
6 */
7
11
12 #ifndef ARCADE_IGAME_HPP_
13 #define ARCADE_IGAME_HPP_
14
16 #define ARCADE_GAME_ENTRY_POINT extern "C" ::arcade::IGame *arcade_GameEntryPoint()
17
18 #include "types.hpp"
19 #include <string_view>
20
21 namespace arcade
22 {
23     struct Event;
24     class IAssetManager;
25     class IRenderer;
26
28     class IGame {
29       public:
31         enum class State {
33             Loaded,
35             Running,
37             Paused,
39             Ended,
40         };
41
44         using EntryPoint = IGame *(*)();
46         static constexpr std::string_view ENTRY_POINT = "arcade_GameEntryPoint";
47
48         virtual ~IGame() = default;
49
58         virtual void setup() = 0;
59
66         virtual void loadAssets(IAssetManager &manager, vec2u displaySize) = 0;
67
74         virtual void close() = 0;
75
81         virtual void setState(State state) = 0;
82
86         virtual State getState() const = 0;
87
91         virtual unsigned int getScore() const = 0;
92
98         virtual void update(float delta) = 0;
99
105         virtual void render(IRenderer &renderer) = 0;
106
112         virtual void handleEvent(Event &event) = 0;
113     };
114 } // namespace arcade
115
116 #endif // !defined(ARCADE_IGAME_HPP_)
```

7.13 src/interface/include/arcade/IGameObject.hpp File Reference

```
#include "Color.hpp"
#include "types.hpp"
```

Include dependency graph for IGameObject.hpp:



Classes

- class arcade::IGameObject

  A movable, drawable game object such as text or sprites.

7.13.1 Detailed Description

Game object interface.

7.14 IGameObject.hpp

Go to the documentation of this file.

```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade: Interface
4 ** File description:
5 ** Game object interface
6 */
7
11
12 #ifndef ARCADE_GAME_OBJECT_HPP_
13 #define ARCADE_GAME_OBJECT_HPP_
14
15 #include "Color.hpp"
16 #include "types.hpp"
17
18 namespace arcade
19 {
21     class IGameObject {
22       public:
24         enum class Type {
26             Text,
28             Rect,
29         };
30
31         virtual ~IGameObject() = default;
32
34         virtual Type getType() const = 0;
35
37         virtual vec2u getSize() const = 0;
38
41         virtual vec2i getPosition() const = 0;
42
46         virtual void setPosition(vec2i pos) = 0;
47
53         virtual void setForeground(Color color, DefaultColor backup = DefaultColor::White) = 0;
```

```
54
60          virtual void setBackground(Color color, DefaultColor backup = DefaultColor::Transparent) = 0;
61      };
62  } // namespace arcade
63
64  #endif // !defined(ARCADE_GAME_OBJECT_HPP_)
```

7.15 src/interface/include/arcade/IRenderer.hpp File Reference

Classes

- class arcade::IRenderer

  Rendering interface.

7.15.1 Detailed Description

Game rendering.

7.16 IRenderer.hpp

Go to the documentation of this file.

```
1  /*
2  ** EPITECH PROJECT, 2022
3  ** Arcade: Interface
4  ** File description:
5  ** Game rendering
6  */
7
11
12 #ifndef ARCADE_IRENDERER_HPP_
13 #define ARCADE_IRENDERER_HPP_
14
15 namespace arcade
16 {
17      class IGameObject;
18
20      class IRenderer {
21        public:
22          virtual ~IRenderer() = default;
23
29          virtual void draw(IGameObject const &object) = 0;
30      };
31 } // namespace arcade
32
33 #endif // !defined(ARCADE_IRENDERER_HPP_)
```

7.17 src/interface/include/arcade/types.hpp File Reference

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

Classes

- struct arcade::vec2< T >

  A 2D vector.

Typedefs

- using **arcade::vec2u** = vec2< unsigned int >

  A 2D vector of unsigned ints.
- using **arcade::vec2i** = vec2< int >

  A 2D vector of ints.
- using **arcade::vec2f** = vec2< float >

  A 2D vector of floats.
- using **arcade::vec2d** = vec2< double >

  A 2D vector of doubles.

7.17.1 Detailed Description

Misc types.

7.18 types.hpp

Go to the documentation of this file.

```
1 /*
2 ** EPITECH PROJECT, 2022
3 ** Arcade: Interface
4 ** File description:
5 ** Misc types
6 */
7
11
12 #ifndef ARCADE_TYPES_HPP_
13 #define ARCADE_TYPES_HPP_
14
15 namespace arcade
16 {
18     template <typename T> struct vec2 {
20         T x;
22         T y;
23
25         template <typename U> explicit constexpr operator vec2<U>() const
26         {
27             return vec2<U>{static_cast<U>(x), static_cast<U>(y)};
28         }
29
30         constexpr vec2<T> operator+() const
31         {
32             return *this;
33         }
34
35         constexpr vec2<T> operator-() const
36         {
37             return vec2<T>{-x, -y};
38         }
39
40         constexpr vec2<T> operator+(T const &v) const
41         {
42             return vec2<T>{x + v, y + v};
43         }
44
45         constexpr vec2<T> operator+(vec2<T> const &v) const
46         {
47             return vec2<T>{x + v.x, y + v.y};
48         }
49
50         constexpr vec2<T> operator-(T const &v) const
51         {
52             return vec2<T>{x - v, y - v};
53         }
54
55         constexpr vec2<T> operator-(vec2<T> const &v) const
56         {
57             return vec2<T>{x - v.x, y - v.y};
58         }
59
60         constexpr vec2<T> operator*(T const &v) const
61         {
62             return vec2<T>{x * v, y * v};
63         }
64
66         constexpr T operator*(vec2<T> const &v) const
67         {
68             return x * v.x + y * v.y;
69         }
70
71         constexpr vec2<T> operator/(T const &v) const
72         {
73             return vec2<T>{x / v, y / v};
74         }
75
76         constexpr vec2<T> &operator+=(T const &v)
77         {
78             x += v;
79             y += v;
80             return *this;
81         }
82
83         constexpr vec2<T> &operator+=(vec2<T> const &v)
84         {
85             x += v.x;
```

```
86              y += v.y;
87              return *this;
88          }
89
90          constexpr vec2<T> &operator-=(T const &v)
91          {
92              x -= v;
93              y -= v;
94              return *this;
95          }
96
97          constexpr vec2<T> &operator-=(vec2<T> const &v)
98          {
99              x -= v.x;
100              y -= v.y;
101              return *this;
102          }
103
104          constexpr vec2<T> &operator*=(T const &v)
105          {
106              x *= v;
107              y *= v;
108              return *this;
109          }
110
111          constexpr vec2<T> &operator/=(T const &v)
112          {
113              x /= v;
114              y /= v;
115              return *this;
116          }
117
118          constexpr bool operator==(vec2<T> const &v) const
119          {
120              return x == v.x && y == v.y;
121          }
122
123          constexpr bool operator!=(vec2<T> const &v) const
124          {
125              return x != v.x || y != v.y;
126          }
127      };
128
130      using vec2u = vec2<unsigned int>;
131
133      using vec2i = vec2<int>;
134
136      using vec2f = vec2<float>;
137
139      using vec2d = vec2<double>;
140  } // namespace arcade
141
142  #endif // !defined(ARCADE_TYPES_HPP_)
```

Index