R-Type

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Rtype Class Reference

`#include <r_type_client.hpp>`

Public Member Functions

- Rtype ()

  Construct a new Rtype object This will init the player.
- void run ()

  If _mainMenu variable is true, call mainMenu.
- void mainMenu ()

  Open window.
- void gameLoop ()

  Open window.
- void handleEvents ()

  This is where I will handle the events for the window & player (key input, etc.).
- void processServerMessages ()

  This is where I will process the info from the server.
- void updateGame ()

  This is where I will update the time, position of sprites, etc.
- void renderGame ()

  This is where I will render the game.

Private Attributes

- Scenes * _scenes

  Set the Game Mode object.
- sf::RenderWindow _window

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Rtype()

```
Rtype::Rtype ( )
```

Construct a new Rtype object This will init the player.

Construct a new Rtype:: Rtype object.

Default easy mode and normal daltonism mode. Ex: renderSystem.addEntity(player), inputSystem.add↩ Entity(player), collisionSystem.addEntity(player), etc.

3.1.2 Member Function Documentation

3.1.2.1 gameLoop()

```
void Rtype::gameLoop ( )
```

Open window.

This is where I will call the handleEvents, updateGame, processCommands, and render functions.

3.1.2.2 handleEvents()

```
void Rtype::handleEvents ( )
```

This is where I will handle the events for the window & player (key input, etc.).

When key is pressed, move player, and send new player info to server.

3.1.2.3 mainMenu()

```
void Rtype::mainMenu ( )
```

Open window.

(handleEvents). Display the main menu with start, help, daltonic mode, and speed selection buttons. On start, set _mainMenu to false, close window, and return. When active, daltonic_mode will be set to true, and draw a color filter over the screen until deactivated. Can set keybindings as well, either default or customized

3.1.2.4 processServerMessages()

```
void Rtype::processServerMessages ( )
```

This is where I will process the info from the server.

3.1.2.5 renderGame()

```
void Rtype::renderGame ( )
```

This is where I will render the game.

Ex: window.clear(), window.draw(background), renderSystem.render(window), window.display, etc.

3.1.2.6 run()

```
void Rtype::run ( )
```

If _mainMenu variable is true, call mainMenu.

While _mainMenu is false, call gameLoop.

3.1.2.7 updateGame()

```
void Rtype::updateGame ( )
```

This is where I will update the time, position of sprites, etc.

Ex: inputSystem.update(deltaTime.asSeconds()), renderSystem.update(deltaTime.asSeconds()), etc.

3.1.3 Member Data Documentation

3.1.3.1 _scenes

```
Scenes* Rtype::_scenes  [private]
```

Set the Game Mode object.

Parameters

mode	

3.1.3.2 _window

```
sf::RenderWindow Rtype::_window  [private]
```

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

- /home/runner/work/R-Type/R-Type/Client/Interface/Include/r_type_client.hpp
- /home/runner/work/R-Type/R-Type/Client/Src/r-type_client.cpp

Chapter 4

File Documentation

4.1 /home/runner/work/R-Type/R-Type/Client/Interface/↩ Include/mainmenu.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <r_type_client.hpp>
```

Functions

- int MainMenu (sf::RenderWindow *window, Rtype *rtype)

4.1.1 Function Documentation

4.1.1.1 MainMenu()

```
int MainMenu (
            sf::RenderWindow * window,
            Rtype * rtype )
```

4.2 /home/runner/work/R-Type/R-Type/Client/Interface/Include/r_type_↩ client.hpp File Reference

```
#include "error_handling.hpp"
#include "scenes.hpp"
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
```

Classes

- class Rtype

4.3 /home/runner/work/R-Type/R-Type/Client/Src/main.cpp File Reference

```
#include <r_type_client.hpp>
```

Functions

- int main ()

  The entry point of the program.

4.3.1 Function Documentation

4.3.1.1 main()

```
int main ( )
```

The entry point of the program.

This function initializes the Rtype object and runs the game.

Returns

0 indicating successful program execution.

int

4.4 /home/runner/work/R-Type/R-Type/Server/Src/main.cpp File Reference

```
#include <Net/server.hpp>
#include <iostream>
```

Functions

- int main ()

4.4.1 Function Documentation

4.4.1.1 main()

```
int main ( )
```

4.5 /home/runner/work/R-Type/R-Type/Client/Src/r-type_client.cpp File Reference

```
#include <Components/component_manager.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Systems/systems.hpp>
#include <iostream>
#include <r_type_client.hpp>
#include <texture_manager.hpp>
```

4.6 /home/runner/work/R-Type/R-Type/Client/Src/scenes.cpp File Reference

```
#include <Components/component_manager.hpp>
#include <Components/components.hpp>
#include <Entities/entity_factory.hpp>
#include <Entities/entity_manager.hpp>
#include <Net/client.hpp>
#include <Systems/systems.hpp>
#include <creatable_client_object.hpp>
#include <functional>
#include <iostream>
#include <r_type_client.hpp>
#include <scenes.hpp>
#include <texture_manager.hpp>
```

Functions

- void handleEvents (sf::Event event, ComponentManager &componentManager, sf::RenderWindow *_↩ window, std::vector< Entity * > buttons, Scenes *scenes)
- void createDaltonismChoiceButtons (std::vector< Entity * > *buttons, ComponentManager &component↩ Manager, EntityManager &entityManager, TextureManager &textureManager, EntityFactory &entityFactory)
- void createGameModeChoiceButtons (std::vector< Entity * > *buttons, ComponentManager &component↩ Manager, EntityManager &entityManager, TextureManager &textureManager, EntityFactory &entityFactory)
- sf::Keyboard::Key waitForKey (sf::RenderWindow *_window)
- void createKeyBindingButtons (std::vector< Entity * > *buttons, ComponentManager &componentManager, EntityManager &entityManager, TextureManager &textureManager, EntityFactory &entityFactory)

4.6.1 Function Documentation

4.6.1.1 createDaltonismChoiceButtons()

```
void createDaltonismChoiceButtons (
            std::vector< Entity * > * buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            EntityFactory & entityFactory )
```

4.6.1.2 createGameModeChoiceButtons()

```
void createGameModeChoiceButtons (
            std::vector< Entity * > * buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            EntityFactory & entityFactory )
```

4.6.1.3 createKeyBindingButtons()

```
void createKeyBindingButtons (
            std::vector< Entity * > * buttons,
            ComponentManager & componentManager,
            EntityManager & entityManager,
            TextureManager & textureManager,
            EntityFactory & entityFactory )
```

4.6.1.4 handleEvents()

```
void handleEvents (
            sf::Event event,
            ComponentManager & componentManager,
            sf::RenderWindow * _window,
            std::vector< Entity * > buttons,
            Scenes * scenes )
```

4.6.1.5 waitForKey()

```
sf::Keyboard::Key waitForKey (
            sf::RenderWindow * _window )
```

4.7 /home/runner/work/R-Type/R-Type/Server/Src/server.cpp File Reference

```
#include <Net/server.hpp>
#include <creatable_client_object.hpp>
```

Index