CurveFeverClone

Generated by Doxygen 1.9.4

Chapter 1

CurveFeverAlike

This is our final project for second semester.

1.1 Details

In this project we focused on mimicking the mechanics used in original web game "CurveFever" in order to create clone of it. It uses SFML and ImGui libraries.

- ImGui is used to create simple menu in which you can switch between different game modes
- SFML is used to provide graphics to represent mechanics and game itself Game is really simple to handle, you only need few keyboard keys. Original game `CurveFeverPro`

1.2 How it works?

- Players spawning point: Each player has its own random angle (float type) drawn by lot. It is passed to the constructor of *PositionManager* class and then used in matrix equation to optain random spawn point on the circle perimeter (which radius can be changed inside the code).
- Placing lines: Each player has its own *VertexArray* of pointers. In this place function adds every following point to it, then using funtion *draw*, draws lines using *TriangleStrip* for better performance.
- Players movement: When line navigation by keyboard keys occurs, there is an angle in which the line curves. The line itself is a vector and we change its sense. Value of the angle is incremented every moment we press key or hold it.
- Collision between player and edge: Functions *checkForCllision* and *updateCollisionQueue* iterate while the game occurs, when the distance is too short to the edges of the window first of them return information which is then captured by other functions in order to restart the game. They count the distance using pointers.
- Collision between players: There is a vector array of pointers with atributes from *Player* class and for loop with another for loop inside it, checking through whole gameplay if any collision occured between *p* and *q* points in above-mentioned array.

1.3 How to setup?

To setup the game on your device you need to clone this repository and add both ImGui and SFML in project properties. You can find brief description on how to do this in these videos:

- ImGui
- SFML

1.4 How to play?

1.4.1 Local multiplayer

To open the game you need to compile the project. When game menu is visible, click "Local multiplayer" button, after that the gameplay begins. Both players can change the direction of their lines by changing angle using keyboard keys:

- First player uses **A** to turn left and **D** to turn right
- Second player uses **J** to turn left and **L** to turn right You can change these keys inside the code.

1.4.2 Server multiplayer

To open the game you need to compile the project. When game menu is visible, click "Server multiplayer" button, after that the gameplay begins. Both players can change the direction of their lines by changing angle using keyboard keys:

- First player uses **A** to turn left and **D** to turn right
- Second player uses **J** to turn left and **L** to turn right You can change these keys inside the code.

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 files with brief descriptions:

Chapter 5

Class Documentation

5.1 BackgroundImage Class Reference

Class for managing background of the main screen.

`#include <Window.h>`

Public Member Functions

- BackgroundImage (std::string pngname="Texture/gg2.png")
- ~BackgroundImage ()

Public Attributes

- std::string pngname
- sf::Vector2u scrres
- sf::Texture imag
- sf::Sprite sprit

5.1.1 Detailed Description

Class for managing background of the main screen.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BackgroundImage()

```
BackgroundImage::BackgroundImage (
            std::string pngname = "Texture/gg2.png" )  [inline]
```

5.1.2.2 ~BackgroundImage()

```
BackgroundImage::~BackgroundImage ( )  [inline]
```

5.1.3 Member Data Documentation

5.1.3.1 imag

```
sf::Texture BackgroundImage::imag
```

5.1.3.2 pngname

```
std::string BackgroundImage::pngname
```

5.1.3.3 scrres

```
sf::Vector2u BackgroundImage::scrres
```

5.1.3.4 sprit

```
sf::Sprite BackgroundImage::sprit
```

5.2 ControlSignals Class Reference

```
#include <Network.h>
```

Public Member Functions

- ControlSignals (bool l=false, bool r=false, bool s=false)

Public Attributes

- bool left
- bool right
- bool space

5.2.1 Constructor & Destructor Documentation

5.2.1.1 ControlSignals()

```
ControlSignals::ControlSignals (
            bool l = false,
            bool r = false,
            bool s = false )  [inline]
```

5.2.2 Member Data Documentation

5.2.2.1 left

```
bool ControlSignals::left
```

5.2.2.2 right

```
bool ControlSignals::right
```

5.2.2.3 space

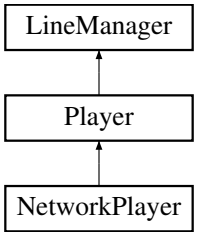
```
bool ControlSignals::space
```

5.3 LineManager Class Reference

Class for managing line indexes, line placement.

```
#include <Player.h>
```

Inheritance diagram for LineManager:



Public Types

- enum class LineModes { both =1 , visual =2 , collision =3 }

Public Member Functions

- LineManager ()
- void initiateLine ()

  Function that initiate line, incrementing array of vertices.
- void restart ()

  Function that clears arrays, restarting them.
- int getLineIndex ()
- void setLineindex (int i)

  Incrementing line index.
- virtual ~LineManager ()

Public Attributes

- std::vector< sf::VertexArray * > linesArray
- std::vector< std::pair< float, float > > collisionPointMap
- std::vector< std::pair< float, float > > collisionPointQueue

5.3.1 Detailed Description

Class for managing line indexes, line placement.

5.3.2 Member Enumeration Documentation

5.3.2.1 LineModes

```
enum class LineManager::LineModes  [strong]
```

Enumerator

both	
visual	
collision	

5.3.3 Constructor & Destructor Documentation

5.3.3.1 LineManager()

```
LineManager::LineManager ( )  [inline]
```

5.3.3.2 ∼LineManager()

```
virtual LineManager::∼LineManager ( )  [inline], [virtual]
```

5.3.4 Member Function Documentation

5.3.4.1 getLineIndex()

```
int LineManager::getLineIndex ( )
```

5.3.4.2 initiateLine()

```
void LineManager::initiateLine ( )
```

Function that initiate line, incrementing array of vertices.

5.3.4.3 restart()

```
void LineManager::restart ( )
```

Function that clears arrays, restarting them.

5.3.4.4 setLineindex()

```
void LineManager::setLineindex (
            int i )
```

Incrementing line index.

5.3.5 Member Data Documentation

5.3.5.1 collisionPointMap

```
std::vector<std::pair<float, float> > LineManager::collisionPointMap
```

5.3.5.2 collisionPointQueue

```
std::vector<std::pair<float, float> > LineManager::collisionPointQueue
```

5.3.5.3 linesArray

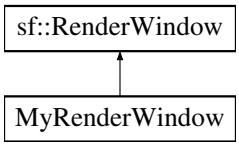
```
std::vector<sf::VertexArray*> LineManager::linesArray
```

5.4 MyRenderWindow Class Reference

Class for initializing window.

```
#include <Window.h>
```

Inheritance diagram for MyRenderWindow:



Public Member Functions

- MyRenderWindow (sf::VideoMode v, std::string title, sf::ContextSettings c)
- void draw (Player &player)

  Function for drawing player, his position and size.
- void draw (sf::Text &text)
- void draw (BackgroundImage &bcg)

  Function for drawing background.
- void draw (sf::Shape &s)

  Function for drawing shape of player.
- void draw (sf::VertexArray &v)

  Fuction for drawing vertex array, points that form lines.

Public Attributes

- sf::Clock guiClock

5.4.1 Detailed Description

Class for initializing window.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 MyRenderWindow()

```
MyRenderWindow::MyRenderWindow (
            sf::VideoMode v,
            std::string title,
            sf::ContextSettings c )  [inline]
```

5.4.3 Member Function Documentation

5.4.3.1 draw() [1/5]

```
void MyRenderWindow::draw (
            BackgroundImage & bcg )
```

Function for drawing background.

5.4.3.2 draw() [2/5]

```
void MyRenderWindow::draw (
            Player & player )
```

Function for drawing player, his position and size.

5.4.3.3 draw() [3/5]

```
void MyRenderWindow::draw (
            sf::Shape & s )
```

Function for drawing shape of player.

5.4.3.4 draw() [4/5]

```
void MyRenderWindow::draw (
            sf::Text & text )  [inline]
```

5.4.3.5 draw() [5/5]

```
void MyRenderWindow::draw (
            sf::VertexArray & v )
```

Fuction for drawing vertex array, points that form lines.

5.4.4 Member Data Documentation

5.4.4.1 guiClock

```
sf::Clock MyRenderWindow::guiClock
```

5.5 networkClient Class Reference

```
#include <Network.h>
```

Public Member Functions

- networkClient ()
- bool getConnected ()
- void setIpAddress (std::string s)
- void setPort (std::string p)
- bool getConnecting ()
- bool isWorking ()
- void connect ()
- void cancelConnect ()
- void disconnect ()
- void test ()
- void awaitStart (std::atomic< State > &s)
- void join (std::atomic< State > &s)
- void start ()
- sf::TcpSocket & getSocket ()

Public Attributes

- sf::IpAddress ip
- std::string port = "5030"
- std::map< sf::Keyboard::Key, bool > keymap

5.5.1 Constructor & Destructor Documentation

5.5.1.1 networkClient()

```
networkClient::networkClient ( )  [inline]
```

5.5.2 Member Function Documentation

5.5.2.1 awaitStart()

```
void networkClient::awaitStart (
            std::atomic< State > & s )
```

5.5.2.2 cancelConnect()

```
void networkClient::cancelConnect ( )
```

5.5.2.3 connect()

```
void networkClient::connect ( )
```

5.5.2.4 disconnect()

```
void networkClient::disconnect ( )
```

5.5.2.5 getConnected()

```
bool networkClient::getConnected ( )
```

5.5.2.6 getConnecting()

```
bool networkClient::getConnecting ( )
```

5.5.2.7 getSocket()

```
sf::TcpSocket & networkClient::getSocket ( )
```

5.5.2.8 isWorking()

```
bool networkClient::isWorking ( )
```

5.5.2.9 join()

```
void networkClient::join (
            std::atomic< State > & s )
```

5.5.2.10 setIpAddress()

```
void networkClient::setIpAddress (
            std::string s )  [inline]
```

5.5.2.11 setPort()

```
void networkClient::setPort (
            std::string p )  [inline]
```

5.5.2.12 start()

```
void networkClient::start ( )
```

5.5.2.13 test()

```
void networkClient::test ( )
```

5.5.3 Member Data Documentation

5.5.3.1 ip

```
sf::IpAddress networkClient::ip
```

5.5.3.2 keymap

```
std::map<sf::Keyboard::Key, bool> networkClient::keymap
```

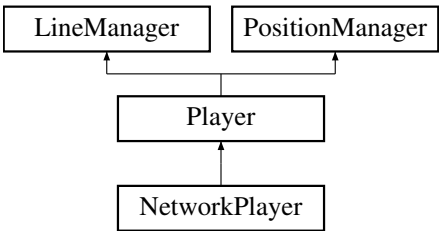
5.5.3.3 port

```
std::string networkClient::port = "5030"
```

5.6 NetworkPlayer Class Reference

```
#include <Network.h>
```

Inheritance diagram for NetworkPlayer:



Public Member Functions

- NetworkPlayer (sf::TcpSocket &co)
- void restart ()
- void processMovement (sf::Packet &p)

Public Attributes

- sf::TcpSocket & socket
- ControlSignals ctrl
- sf::Clock clock
- sf::Time timeSinceLastUpdate
- bool movable {}

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 NetworkPlayer()

```
NetworkPlayer::NetworkPlayer (
            sf::TcpSocket & co )  [inline]
```

5.6.2 Member Function Documentation

5.6.2.1 processMovement()

```
void NetworkPlayer::processMovement (
            sf::Packet & p )
```

5.6.2.2 restart()

```
void NetworkPlayer::restart ( )  [inline]
```

5.6.3 Member Data Documentation

5.6.3.1 clock

```
sf::Clock NetworkPlayer::clock
```

5.6.3.2 ctrl

```
ControlSignals NetworkPlayer::ctrl
```

5.6.3.3 movable

```
bool NetworkPlayer::movable {}
```

5.6.3.4 socket

```
sf::TcpSocket& NetworkPlayer::socket
```

5.6.3.5 timeSinceLastUpdate

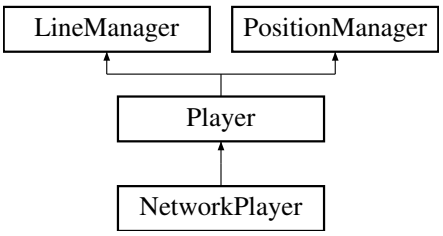
```
sf::Time NetworkPlayer::timeSinceLastUpdate
```

5.7 Player Class Reference

Class for managing players, particularly movement, size, position, collision etc.

```
#include <Player.h>
```

Inheritance diagram for Player:



Public Member Functions

- Player (sf::Vector2f p={ 400, 400 }, int s=5)
- Player (int radius=300, int s=5)
- void restart (sf::Vector2f newPos)
- void restart ()

  restart game
- void setLineMode (LineModes newMode)
- int getId ()
- void moveTo (sf::Vector2f newp)

  Moving player to different position through its movement.
- void moveBy (float distance)

  Function that move line by perticular distance and adds score.
- sf::Vector2f getPosition ()

  Get current position on the map.
- sf::Vector2f getStarting ()

  Get starting position.
- void setPlacesPath (bool v)

  Set whether player places path.
- bool getPlacesPath ()

  Get whether player places path.
- int getSize ()

  Get size of player.
- void setSize (int n)

  Set size of player.
- bool checkForCollision ()
- bool checkForCollision (Player &other)
- void updateCollisionQueue ()

  Function that iterates and checks whether collision occured.
- void setVisualPath ()

  Function that sets visual path to players.
- void setCollisionPath ()

  Function that sets path used to initiate collision.
- void setPath ()

  Setting and initiating both visual and collision lines.
- void chooseWhetherToPlacePathOrNot ()

  Function that randomly delays placing of path.
- ~Player ()

Public Attributes

- ScoreManager score
- int randRGBv0 = rand() % 255
- int randRGBv1 = rand() % 255
- int randRGBv2 = rand() % 255

Friends

- class MyRenderWindow

Additional Inherited Members

5.7.1 Detailed Description

Class for managing players, particularly movement, size, position, collision etc.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Player() [1/2]

```
Player::Player (
            sf::Vector2f p = { 400, 400 },
            int s = 5 )  [inline]
```

5.7.2.2 Player() [2/2]

```
Player::Player (
            int radius = 300,
            int s = 5 )  [inline]
```

5.7.2.3 ~Player()

```
Player::~Player ( )  [inline]
```

5.7.3 Member Function Documentation

5.7.3.1 checkForCollision() [1/2]

```
bool Player::checkForCollision ( )
```

check for collision with screen border, then check for collision with path

5.7.3.2 checkForCollision() [2/2]

```
bool Player::checkForCollision (
            Player & other )
```

check for collision with screen border, then check for collision with path

5.7.3.3 chooseWhetherToPlacePathOrNot()

```
void Player::chooseWhetherToPlacePathOrNot ( )
```

Function that randomly delays placing of path.

5.7.3.4 getId()

```
int Player::getId ( )
```

5.7.3.5 getPlacesPath()

```
bool Player::getPlacesPath ( )
```

Get whether player places path.

5.7.3.6 getPosition()

```
sf::Vector2f Player::getPosition ( )
```

Get current position on the map.

5.7.3.7 getSize()

```
int Player::getSize ( )
```

Get size of player.

5.7.3.8 getStarting()

```
sf::Vector2f Player::getStarting ( )
```

Get starting position.

5.7.3.9 moveBy()

```
void Player::moveBy (
            float distance )
```

Function that move line by perticular distance and adds score.

5.7.3.10 moveTo()

```
void Player::moveTo (
            sf::Vector2f newp )
```

Moving player to different position through its movement.

5.7.3.11 restart() [1/2]

```
void Player::restart ( )  [inline]
```

restart game

5.7.3.12 restart() [2/2]

```
void Player::restart (
            sf::Vector2f newPos )  [inline]
```

5.7.3.13 setCollisionPath()

```
void Player::setCollisionPath ( )
```

Function that sets path used to initiate collision.

5.7.3.14 setLineMode()

```
void Player::setLineMode (
            LineModes newMode )
```

5.7.3.15 setPath()

```
void Player::setPath ( )
```

Setting and initiating both visual and collision lines.

5.7.3.16 setPlacesPath()

```
void Player::setPlacesPath (
            bool v )
```

Set whether player places path.

5.7.3.17 setSize()

```
void Player::setSize (
            int n )
```

Set size of player.

5.7.3.18 setVisualPath()

```
void Player::setVisualPath ( )
```

Function that sets visual path to players.

5.7.3.19 updateCollisionQueue()

```
void Player::updateCollisionQueue ( )
```

Function that iterates and checks whether collision occured.

rescure p address;

increment p address

5.7.4 Friends And Related Function Documentation

5.7.4.1 MyRenderWindow

```
friend class MyRenderWindow  [friend]
```

5.7.5 Member Data Documentation

5.7.5.1 randRGBv0

```
int Player::randRGBv0 = rand() % 255
```

5.7.5.2 randRGBv1

```
int Player::randRGBv1 = rand() % 255
```

5.7.5.3 randRGBv2

```
int Player::randRGBv2 = rand() % 255
```

5.7.5.4 score

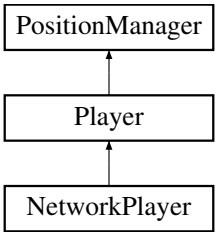
```
ScoreManager Player::score
```

5.8 PositionManager Class Reference

Class for managing position of players.

```
#include <Player.h>
```

Inheritance diagram for PositionManager:



Public Member Functions

- PositionManager (sf::Vector2f p)
- void restart ()

  Function that restarts position of players after collision.
- void pickNewPosition (float radius)
- PositionManager (float radius=300)
- float getDistanceFromPrevious ()
- float getAngleFromPrevious ()
- void setAngle (float a)

  Setting angle.
- float getAngle ()
- void changeAngle (float a)

  Changing angle.
- void applyDisplacement (float distance, float angle)

  Function that applies change of position, including angles.
- void setPosition (const sf::Vector2f &n)

  Setting position.
- void applyDisplacement (Vector &v)

  Applying displacement.

Friends

- class Player

5.8.1 Detailed Description

Class for managing position of players.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 PositionManager() [1/2]

```
PositionManager::PositionManager (
            sf::Vector2f p )  [inline]
```

5.8.2.2 PositionManager() [2/2]

```
PositionManager::PositionManager (
            float radius = 300 )  [inline]
```

5.8.3 Member Function Documentation

5.8.3.1 applyDisplacement() [1/2]

```
void PositionManager::applyDisplacement (
            float distance,
            float angle )
```

Function that applies change of position, including angles.

5.8.3.2 applyDisplacement() [2/2]

```
void PositionManager::applyDisplacement (
            Vector & v )
```

Applying displacement.

5.8.3.3 changeAngle()

```
void PositionManager::changeAngle (
            float a )
```

Changing angle.

5.8.3.4 getAngle()

```
float PositionManager::getAngle ( )
```

5.8.3.5 getAngleFromPrevious()

```
float PositionManager::getAngleFromPrevious ( )
```

5.8.3.6 getDistanceFromPrevious()

```
float PositionManager::getDistanceFromPrevious ( )
```

5.8.3.7 pickNewPosition()

```
void PositionManager::pickNewPosition (
            float radius )  [inline]
```

5.8.3.8 restart()

```
void PositionManager::restart ( )
```

Function that restarts position of players after collision.

5.8.3.9 setAngle()

```
void PositionManager::setAngle (
            float a )
```

Setting angle.

5.8.3.10 setPosition()

```
void PositionManager::setPosition (
            const sf::Vector2f & n )
```

Setting position.

5.8.4 Friends And Related Function Documentation

5.8.4.1 Player

```
friend class Player  [friend]
```

5.9 ScoreManager Class Reference

Class for managing the score.

```
#include <Player.h>
```

Public Member Functions

- ScoreManager ()
- void nextRound ()
- int getCurrentScore ()
- int getScore ()
- void addScore ()
- void restart ()
- void addScore (int add)

5.9.1 Detailed Description

Class for managing the score.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ScoreManager()

```
ScoreManager::ScoreManager ( )  [inline]
```

5.9.3 Member Function Documentation

5.9.3.1 addScore() [1/2]

```
void ScoreManager::addScore ( )  [inline]
```

5.9.3.2 addScore() [2/2]

```
void ScoreManager::addScore (
            int add )  [inline]
```

5.9.3.3 getCurrentScore()

```
int ScoreManager::getCurrentScore ( )  [inline]
```

5.9.3.4 getScore()

```
int ScoreManager::getScore ( )  [inline]
```

5.9.3.5 nextRound()

```
void ScoreManager::nextRound ( )  [inline]
```

5.9.3.6 restart()

```
void ScoreManager::restart ( )  [inline]
```

5.10 Server2ndTry Class Reference

```
#include <Network.h>
```

Public Member Functions

- void start ()
- bool isVictoryConditionMet ()
- std::string serializePlayerData ()
- void handleStart (sf::Packet &p, sf::TcpSocket &s)
- void startPathsIf (bool conditions)
- void handleUpdate (sf::Packet &incomingMessage, sf::TcpSocket &socket)
- void updateLoop ()
- void handleJoin (sf::Packet &IncomingMessage, sf::TcpSocket &socket)
- void parseRecieved (sf::Packet &incomingMessage, sf::TcpSocket &socket)
- void acceptLoop ()
- void restart ()
- void restartRound ()
- void recvLoop ()
- void stopServer ()
- void setRunning (bool newState)
- bool getRunning ()

5.10.1 Member Function Documentation

5.10.1.1 acceptLoop()

```
void Server2ndTry::acceptLoop ( )
```

5.10.1.2 getRunning()

```
bool Server2ndTry::getRunning ( )  [inline]
```

5.10.1.3 handleJoin()

```
void Server2ndTry::handleJoin (
            sf::Packet & IncomingMessage,
            sf::TcpSocket & socket )
```

5.10.1.4 handleStart()

```
void Server2ndTry::handleStart (
            sf::Packet & p,
            sf::TcpSocket & s )
```

5.10.1.5 handleUpdate()

```
void Server2ndTry::handleUpdate (
            sf::Packet & incomingMessage,
            sf::TcpSocket & socket )
```

5.10.1.6 isVictoryConditionMet()

```
bool Server2ndTry::isVictoryConditionMet ( )
```

5.10.1.7 parseRecieved()

```
void Server2ndTry::parseRecieved (
            sf::Packet & incomingMessage,
            sf::TcpSocket & socket )
```

5.10.1.8 recvLoop()

```
void Server2ndTry::recvLoop ( )
```

5.10.1.9 restart()

```
void Server2ndTry::restart ( )
```

5.10.1.10 restartRound()

```
void Server2ndTry::restartRound ( )
```

5.10.1.11 serializePlayerData()

```
std::string Server2ndTry::serializePlayerData ( )
```

5.10.1.12 setRunning()

```
void Server2ndTry::setRunning (
            bool newState )  [inline]
```

5.10.1.13 start()

```
void Server2ndTry::start ( )
```

5.10.1.14 startPathsIf()

```
void Server2ndTry::startPathsIf (
            bool conditions )
```

5.10.1.15 stopServer()

```
void Server2ndTry::stopServer ( )
```

5.10.1.16 updateLoop()

```
void Server2ndTry::updateLoop ( )
```

5.11 Vector Class Reference

Class for managing vectors.

```
#include <Player.h>
```

Public Member Functions

- Vector (float l, float a)
- sf::Vector2f getDisplacement ()

5.11.1 Detailed Description

Class for managing vectors.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Vector()

```
Vector::Vector (
            float l,
            float a )  [inline]
```

5.11.3 Member Function Documentation

5.11.3.1 getDisplacement()

```
sf::Vector2f Vector::getDisplacement ( )
```

Chapter 6

File Documentation

6.1 C:/Users/barti/Documents/CurveFeverClone/CurveFever/headers/↩ Network.h File Reference

```
#include "Player.h"
#include <thread>
#include <SFML/Graphics.hpp>
#include <SFML/Network.hpp>
#include <mutex>
#include <sstream>
```

Classes

- class ControlSignals
- class NetworkPlayer
- class networkClient
- class Server2ndTry

Enumerations

- enum class State {
  singleplayer = 1 , multiplayerMenu , menu , multiplayer ,
  serverHost }

Functions

- bool compareHosts (sf::TcpSocket &c, sf::TcpSocket &s)

Variables

- const std::string defaultPort = "5030"

6.1.1 Enumeration Type Documentation

6.1.1.1 State

```
enum class State  [strong]
```

Enumerator

singleplayer	
multiplayerMenu	
menu	
multiplayer	
serverHost	

6.1.2 Function Documentation

6.1.2.1 compareHosts()

```
bool compareHosts (
            sf::TcpSocket & c,
            sf::TcpSocket & s )
```

6.1.3 Variable Documentation

6.1.3.1 defaultPort

```
const std::string defaultPort = "5030"
```

6.2 Network.h

Go to the documentation of this file.

```
1 #pragma once
2 #include "Player.h"
3 #include <thread>
4 #include <SFML/Graphics.hpp>
5 #include <SFML/Network.hpp>
6 #include <mutex>
7 #include <sstream>
8 const std::string defaultPort = "5030";
9
10 enum class State {
11     singleplayer = 1,
12     multiplayerMenu,
13     menu,
14     multiplayer,
15     serverHost
16 };
17
18 bool compareHosts(sf::TcpSocket& c, sf::TcpSocket& s);
19
20 class ControlSignals {
21 public:
22     bool left, right, space;
23     ControlSignals(bool l = false, bool r = false, bool s = false) {
24         left = l;
25         right = r;
26         space = s;
```

```
27     }
28 };
29
30 class NetworkPlayer : public Player {
31 public:
32     sf::TcpSocket& socket;
33     ControlSignals ctrl;
34     sf::Clock clock;
35     sf::Time timeSinceLastUpdate;
36     bool movable{};
37     NetworkPlayer(sf::TcpSocket& co) : Player(300), socket(co), ctrl() {
38         clock.restart();
39         movable = true;
40     };
41     void restart() {
42         movable = true;
43         clock.restart();
44         Player::restart();
45     }
46     void processMovement(sf::Packet& p);
47 };
48
49
50 class networkClient {
51     std::size_t recsize{};
52     bool isConnected = false;
53     bool isConnecting = false;
54     sf::TcpSocket socket;
55     std::string w = "welcome";
56     std::unique_ptr<std::thread> waiterThread;
57 public:
58     sf::IpAddress ip;
59     std::string port = "5030";
60     std::map<sf::Keyboard::Key, bool> keymap;
61     networkClient() {
62         //ip = sf::IpAddress::getLocalAddress();
63         port = defaultPort;
64         ip = sf::IpAddress::getPublicAddress(sf::seconds(5.0f));
65     }
66     bool getConnected();
67     void setIpAddress(std::string s) {
68         ip = sf::IpAddress(s);
69     }
70     void setPort(std::string p) {
71         port = p;
72     }
73     bool getConnecting();
74     bool isWorking();
75     void connect();
76     void cancelConnect();
77     void disconnect();
78     void test();
79     void awaitStart(std::atomic<State>& s);
80     void join(std::atomic<State>& s);
81     void start();
82     sf::TcpSocket& getSocket();
83 };
84
85
86 class Server2ndTry {
87     sf::TcpListener listener;
88     sf::SocketSelector selector;
89     std::mutex socketMutex;
90     std::vector<std::shared_ptr<sf::TcpSocket» sockets;
91     std::vector<std::shared_ptr<NetworkPlayer» players;
92     sf::Clock clock;
93     bool started = false;
94     bool pathsStarted = false;
95     std::atomic<bool> isRunning;
96     std::thread loops;
97
98     int currentRound = 0, roundLimit = 4;
99     void startNewRound() {
100         pathsStarted = false;
101         clock.restart();
102         sf::Packet message;
103         std::stringstream stream;
104         message « "RESTART";
105         for (auto& p : players) {
106             p->score.nextRound();
107             p->restart();
108             p->movable = true;
109             p->setPlacesPath(false);
110         }
111         for (auto& p : players) {
112             stream « p->getPosition().x « " " « p->getPosition().y « " " « p->movable « "|";
113         }
```

```
114         std::string temp = stream.str();
115         temp.pop_back(); // removing the last |
116         message << temp;
117         for (auto& p : players) {
118             p->socket.send(message);
119         }
120     }
121     void endGame() {
122         sf::Packet message;
123         std::stringstream stream;
124         message << "SCORE";
125         for (auto& p : players) {
126             stream << p->score.getScore() << " ";
127         }
128 
129         std::string temp = stream.str();
130         temp.pop_back();
131         message << temp;
132         for (auto& p : players) {
133             p->socket.send(message);
134         }
135 
136     }
137 public:
138     void start();
139     bool isVictoryConditionMet();
140     std::string serializePlayerData();
141     void handleStart(sf::Packet& p, sf::TcpSocket& s);
142     void startPathsIf(bool conditions);
143     void handleUpdate(sf::Packet& incomingMessage, sf::TcpSocket& socket);
144     void updateLoop();
145     void handleJoin(sf::Packet& IncomingMessage, sf::TcpSocket& socket);
146     void parseRecieved(sf::Packet& incomingMessage, sf::TcpSocket& socket);
147     void acceptLoop();
148     void restart();
149     void restartRound();
150     void recvLoop();
151     void stopServer();
152     void setRunning(bool newState) {
153         return isRunning.store(newState);
154     }
155     bool getRunning() {
156         return isRunning.load();
157     }
158 };
```

6.3 C:/Users/barti/Documents/CurveFeverClone/CurveFever/headers/Player.h File Reference

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

Classes

- class Vector

  Class for managing vectors.
- class PositionManager

  Class for managing position of players.
- class LineManager

  Class for managing line indexes, line placement.
- class ScoreManager

  Class for managing the score.
- class Player

  Class for managing players, particularly movement, size, position, collision etc.

Macros

- #define PI std::acos(0) ∗ 2

Enumerations

- enum class Inputs { Left = 1 , Right = 2 , Space = 3 }

Functions

- const sf::Vector2u screenSize (1000, 1000)
- sf::Vector2f midpoint (sf::Vector2f v1, sf::Vector2f v2)
- float distance (sf::Vector2f a, sf::Vector2f b)

6.3.1 Macro Definition Documentation

6.3.1.1 PI

```
#define PI std::acos(0) * 2
```

6.3.2 Enumeration Type Documentation

6.3.2.1 Inputs

```
enum class Inputs  [strong]
```

Enumerator

Left	
Right	
Space	

6.3.3 Function Documentation

6.3.3.1 distance()

```
float distance (
            sf::Vector2f a,
```

```
            sf::Vector2f b )
```

6.3.3.2 midpoint()

```
sf::Vector2f midpoint (
            sf::Vector2f v1,
            sf::Vector2f v2 )
```

6.3.3.3 screenSize()

```
const sf::Vector2u screenSize (
            1000 ,
            1000  )
```

6.4 Player.h

Go to the documentation of this file.

```
1 #pragma once
2 #include <SFML/Graphics.hpp>
3 #include <iostream>
4 #define PI std::acos(0) * 2
5 const sf::Vector2u screenSize(1000, 1000);
6
7 sf::Vector2f midpoint(sf::Vector2f v1, sf::Vector2f v2);
8
10 class Vector {
11     float length{};
12     float angle{};
13 public:
14     Vector(float l, float a) : length(l), angle(a) {};
15     sf::Vector2f getDisplacement();
16 };
17
18 float distance(sf::Vector2f a, sf::Vector2f b);
20 class PositionManager {
21     float angle{};
22     sf::Vector2f current;
23     sf::Vector2f previous;
24     sf::Vector2f starting;
25     float startingDistance{};
26     friend class Player;
27 public:
28     PositionManager(sf::Vector2f p) : current(p), starting(p), previous(p) {}
29     void restart();
30     void pickNewPosition(float radius) {
31         angle = rand() % 360;
32         sf::Vector2f r = { radius, 0 };
33         sf::Vector2f secr = { screenSize.x / 2 + (r.x * cos(angle) - r.y * sin(angle)),screenSize.y / 2 +
       (r.x * sin(angle) + r.y * cos(angle)) };
34         current = secr, starting = secr, previous = secr;
35     }
36     PositionManager(float radius = 300) : startingDistance(radius) {
37         pickNewPosition(startingDistance);
38
39     }
40     float getDistanceFromPrevious();
41     float getAngleFromPrevious();
42     void setAngle(float a);
43     float getAngle();
44     void changeAngle(float a);
45     void applyDisplacement(float distance, float angle);
46     void setPosition(const sf::Vector2f& n);
47     void applyDisplacement(Vector& v);
48 };
```

```
49
51 class LineManager {
52     const int startingLineIndex = 0 - 1; // because it's always incremented by one whenever path is
       placed;
53     int lineIndex = startingLineIndex;
54 public:
55     enum class LineModes {
56         both=1,
57         visual=2,
58         collision=3
59     };
60     std::vector<sf::VertexArray*> linesArray;
61     std::vector<std::pair<float, float» collisionPointMap;
62     std::vector<std::pair<float, float» collisionPointQueue;
63     LineManager() {
64         linesArray.reserve(100);
65         collisionPointMap.reserve(10000);
66         collisionPointQueue.reserve(100);
67     }
68     void initiateLine();
69     void restart();
70     int getLineIndex();
71     void setLineindex(int i);
72     virtual ~LineManager() {
73         collisionPointQueue.clear();
74         collisionPointMap.clear();
75         linesArray.clear();
76     }
77 };
78
80 class ScoreManager {
81     int startingScore = 0;
82     std::vector<int> scoreBoard;
83     int round = 0;
84 public:
85     ScoreManager() {
86         scoreBoard.push_back(startingScore);
87     }
88     void nextRound() {
89         scoreBoard.push_back(startingScore);
90         round += 1;
91     }
92     int getCurrentScore() {
93         return scoreBoard[round];
94     }
95     int getScore() {
96         int result{};
97         for (auto x : scoreBoard) {
98             result += x;
99         }
100         return result;
101     }
102     void addScore() {
103         scoreBoard[round] += 1;
104
105     }
106     void restart() {
107         scoreBoard.clear();
108         scoreBoard.emplace_back(startingScore);
109         round = 0;
110     }
111     void addScore(int add) {
112         scoreBoard[round] += add;
113     }
114 };
115
116 enum class Inputs {
117     Left = 1,
118     Right = 2,
119     Space = 3,
120 };
121
123 class Player : public LineManager, public PositionManager {
124     bool placesPath = true;
125     int size{};
126     int id{};
127     sf::Clock linerestart;
128     LineModes lineMode = LineModes::both;
129     sf::Clock clock1;
130 public:
131     ScoreManager score;
132     friend class MyRenderWindow;
133     Player(sf::Vector2f p = { 400, 400 }, int s = 5) : PositionManager(p), LineManager() {
134         size = s;
135         placesPath = true;
136         initiateLine();
137     }
```

```
138     Player(int radius = 300, int s = 5) : PositionManager(radius), LineManager() {
139         size = s;
140         placesPath = true;
141         initiateLine();
142     }
143     void restart(sf::Vector2f newPos) {
144         restart();
145         setPosition(newPos);
146     }
148     void restart() {
149         placesPath = false;
150         LineManager::restart();
151         PositionManager::restart();
152         score.restart();
153         placesPath = true;
154         linerestart.restart();
155     }
156     void setLineMode(LineModes newMode);
157     int getId();
158     void moveTo(sf::Vector2f newp);
159     void moveBy(float distance);
160     // Get current position on the map
161     sf::Vector2f getPosition();
162     // Get starting position
163     sf::Vector2f getStarting();
164     // Set whether player places path
165     void setPlacesPath(bool v);
166     // Get whether player places path
167     bool getPlacesPath();
168     // Get size of player
169     int getSize();
170     // Set size of player
171     void setSize(int n);
172     bool checkForCollision();
173     bool checkForCollision(Player& other);
174     void updateCollisionQueue();
175     int randRGBv0 = rand() % 255;
176     int randRGBv1 = rand() % 255;
177     int randRGBv2 = rand() % 255;
178
179     void setVisualPath();
180     void setCollisionPath();
181     void setPath();
182     void chooseWhetherToPlacePathOrNot();
183     ~Player() {}
184 };
```

6.5 C:/Users/barti/Documents/CurveFeverClone/CurveFever/headers/Window.h File Reference

```
#include "Player.h"
#include <SFML/Graphics.hpp>
```

Classes

- class BackgroundImage

  Class for managing background of the main screen.
- class MyRenderWindow

  Class for initializing window.

6.6 Window.h

Go to the documentation of this file.

```
1 #pragma once
2 #include "Player.h"
3 #include <SFML/Graphics.hpp>
```

```
4
6 class BackgroundImage {
7 public:
8     std::string pngname;
9     sf::Vector2u scrres;
10     sf::Texture imag;
11     sf::Sprite sprit;
12
13     BackgroundImage(std::string pngname = "Texture/gg2.png") {
14         imag.loadFromFile(pngname);
15         scrres = imag.getSize();
16         sprit.setTexture(imag);
17         sprit.setScale(sf::Vector2f(screenSize.x / (float)scrres.x, screenSize.x / (float)scrres.y));
18     };
19     ~BackgroundImage() {};
20 };
21
23 class MyRenderWindow : public sf::RenderWindow {
24 public:
25     sf::Clock guiClock;
26     MyRenderWindow(sf::VideoMode v, std::string title, sf::ContextSettings c) : sf::RenderWindow(v,
      title, sf::Style::Close, c)
27     {
28         guiClock = sf::Clock();
29     }
30     void draw(Player& player);
31     void draw(sf::Text& text) {
32         sf::RenderWindow::draw(text);
33     };
34     void draw(BackgroundImage& bcg);
35     void draw(sf::Shape& s);
36     void draw(sf::VertexArray& v);
37 };
```

6.7 C:/Users/barti/Documents/CurveFeverClone/CurveFever/Source.cpp File Reference

```
#include <iostream>
#include <thread>
#include <mutex>
#include <chrono>
#include <vector>
#include <sstream>
#include "imgui/imgui.h"
#include "imgui/imgui-SFML.h"
#include <SFML/Graphics.hpp>
#include <SFML/Network.hpp>
#include <Player.h>
#include <Window.h>
#include <Network.h>
#include "sources/Player.cpp"
#include "sources/Window.cpp"
#include "sources/Network.cpp"
```

Macros

- #define PI std::acos(0) ∗ 2

Enumerations

- enum class networkFlags { JOIN = 1 , DISCONNECT = 2 , TEST = 3 }

Functions

- sf::Vector2f mmul (float **matrix2, sf::Vector2f v)
- float ** getRotationMatrix (float angle)
- float ** transpose (float **matrix2)
- void printv (sf::Vector2f v)
- void singleplayer (MyRenderWindow &window)
- void menu (MyRenderWindow &window, std::atomic< State > &s, BackgroundImage &bcgg)
- void emptyFn ()
- void splitTo (std::string str, const char seperator, std::vector< std::string > &cont)
- std::pair< std::string, std::string > splitOnceBy (std::string str, const char seperator)
- void multiplayer (MyRenderWindow &window, std::atomic< State > &s, networkClient &net)
- void multiplayerMenu (MyRenderWindow &window, std::atomic< State > &s, networkClient &net)
- void multiplayerConnecting (MyRenderWindow &window, std::atomic< State > &s, networkClient &net)
- void server2ndTry (MyRenderWindow &window, std::atomic< State > &gameState)
- void client ()
- int main ()

Variables

- sf::Font font
- const sf::Vector2u screenSize
- std::mutex tempMut

6.7.1 Macro Definition Documentation

6.7.1.1 PI

```
#define PI std::acos(0) * 2
```

6.7.2 Enumeration Type Documentation

6.7.2.1 networkFlags

```
enum class networkFlags  [strong]
```

Enumerator

JOIN	
DISCONNECT	
TEST	

6.7.3 Function Documentation

6.7.3.1 client()

```
void client ( )
```

6.7.3.2 emptyFn()

```
void emptyFn ( )
```

6.7.3.3 getRotationMatrix()

```
float ** getRotationMatrix (
            float angle )
```

6.7.3.4 main()

```
int main ( )
```

6.7.3.5 menu()

```
void menu (
            MyRenderWindow & window,
            std::atomic< State > & s,
            BackgroundImage & bcgg )
```

6.7.3.6 mmul()

```
sf::Vector2f mmul (
            float ** matrix2,
            sf::Vector2f v )
```

6.7.3.7 multiplayer()

```
void multiplayer (
            MyRenderWindow & window,
            std::atomic< State > & s,
            networkClient & net )
```

6.7.3.8 multiplayerConnecting()

```
void multiplayerConnecting (
            MyRenderWindow & window,
            std::atomic< State > & s,
            networkClient & net )
```

6.7.3.9 multiplayerMenu()

```
void multiplayerMenu (
            MyRenderWindow & window,
            std::atomic< State > & s,
            networkClient & net )
```

6.7.3.10 printv()

```
void printv (
            sf::Vector2f v )
```

6.7.3.11 server2ndTry()

```
void server2ndTry (
            MyRenderWindow & window,
            std::atomic< State > & gameState )
```

6.7.3.12 singleplayer()

```
void singleplayer (
            MyRenderWindow & window )
```

6.7.3.13 splitOnceBy()

```
std::pair< std::string, std::string > splitOnceBy (
            std::string str,
            const char seperator )
```

6.7.3.14 splitTo()

```
void splitTo (
            std::string str,
            const char seperator,
            std::vector< std::string > & cont )
```

6.7.3.15 transpose()

```
float ** transpose (
            float ** matrix2 )
```

6.7.4 Variable Documentation

6.7.4.1 font

```
sf::Font font
```

6.7.4.2 screenSize

```
const sf::Vector2u screenSize  [extern]
```

6.7.4.3 tempMut

```
std::mutex tempMut
```

6.8 C:/Users/barti/Documents/CurveFeverClone/CurveFever/sources/Network.cpp File Reference

```
#include "Network.h"
```

Functions

- bool compareHosts (sf::TcpSocket &c, sf::TcpSocket &s)

6.8.1 Function Documentation

6.8.1.1 compareHosts()

```
bool compareHosts (
            sf::TcpSocket & c,
            sf::TcpSocket & s )
```

6.9 C:/Users/barti/Documents/CurveFeverClone/CurveFever/sources/Player.cpp File Reference

```
#include "Player.h"
```

Functions

- sf::Vector2f midpoint (sf::Vector2f v1, sf::Vector2f v2)
- float distance (sf::Vector2f a, sf::Vector2f b)

6.9.1 Function Documentation

6.9.1.1 distance()

```
float distance (
            sf::Vector2f a,
            sf::Vector2f b )
```

6.9.1.2 midpoint()

```
sf::Vector2f midpoint (
            sf::Vector2f v1,
            sf::Vector2f v2 )
```

6.10 C:/Users/barti/Documents/CurveFeverClone/CurveFever/sources/Window.cpp File Reference

```
#include "Window.h"
```

6.11 C:/Users/barti/Documents/CurveFeverClone/README.md File Reference

Index