Chor Police

11

Generated by Doxygen 1.8.13

Contents

Chapter 1

README

[ ] Create a to-do List.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 cp::AssetManager Class Reference

Public Member Functions

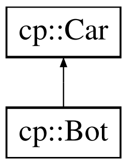
- void **load_texture** (std::string name, std::string file_name)
- void **load_font** (std::string name, std::string file_name)
- sf::Texture & **get_texture** (std::string name)
- sf::Font & **get_font** (std::string name)

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

- include/ResourceManagers/AssetManager.hpp
- libs/AssetManager.cpp

5.2 cp::Bot Class Reference

Inheritance diagram for cp::Bot:



Public Types

- using **input_type** = std::vector< bool >

Public Member Functions

- **Bot** (GameDataRef _data, int car_num)
- void **drawSprite** (const Line &line)
- virtual void **update_car** (float dt, const std::vector< Line > &lines, float segL)
- void **handle_input** (input_type mask, float dt)
- void **handle_input** ()

Public Attributes

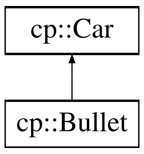
- int **img** = 1

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

- include/Objects/Bot.hpp
- libs/Bot.cpp

5.3 cp::Bullet Class Reference

Inheritance diagram for cp::Bullet:



Public Member Functions

- **Bullet** (GameDataRef _data, int car_num)
- virtual void **init** (sf::Vector3f pos)
- void **drawSprite** (const Line &line)
- virtual void **update_car** (float dt, const std::vector< Line > &lines, float segL)
- void **handle_input** ()

Public Attributes

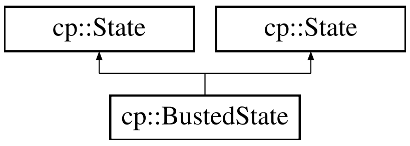
- int **frames** =0

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

- include/Objects/Bullet.hpp
- libs/Bullet.cpp

5.4 cp::BustedState Class Reference

Inheritance diagram for cp::BustedState:



Public Member Functions

- **BustedState** (GameDataRef _data)
- void **init** ()
- void **draw** (float delta)
- void **update** (float delta)
- void **handle_input** (float delta)
- **BustedState** (GameDataRef _data)
- void **init** ()
- void **draw** (float delta)
- void **update** (float delta)
- void **handle_input** (float delta)

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

- include/BustedState.hpp
- libs/BustedState.cpp

5.5 cp::Camera Class Reference

Public Member Functions

- const sf::Vector3f & **getPosition** () const
- const sf::Vector3f & **getSpeed** () const
- void **catch_player** (const Car &player)
- float **getCamD** () const
- void **handle_input** ()

Public Attributes

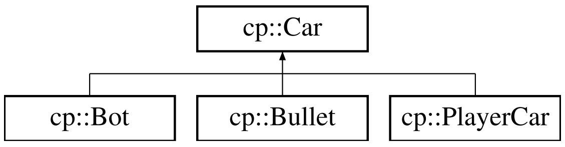
- std::thread::id **id**
- sf::Vector3f **e_position** = sf::Vector3f(0, 1500, 0)

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

- include/Objects/Camera.hpp

5.6 cp::Car Class Reference

Inheritance diagram for cp::Car:



Public Member Functions

- **Car** (GameDataRef _data, int car_num)
- void **draw_car** ()
- virtual void **init** (sf::Vector3f pos)
- void **reset** ()
- virtual void **update_car** (float dt, const std::vector< Line > &lines, float segL)=0
- sf::Vector3f **getPosition** () const
- sf::Vector3f **getSpeed** () const
- void **onCollision** ()

Public Attributes

- int **car_image_num**
- bool **l** = false
- bool **r** = false
- sf::Sprite **sprite**
- GameDataRef **data**
- sf::Vector3f **e_position**
- sf::Vector3f **e_speed**
- sf::Vector3f **e_acceleration**
- sf::Vector3f **e_decleration**
- sf::Vector3f **e_max_speed**
- float **centrifugal** = 0.5
- float **car_mass** =0
- float **health** = 100
- bool **in_use** =false

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

- include/Objects/Car.hpp
- libs/Car.cpp

5.7 cp::Client Class Reference

Public Types

- using **ID** = long long int
- using **IP** = std::string
- using **key_input_type** = std::pair< ID, std::vector< bool > >

Public Member Functions

- Client (ID identity)

  Construct a new Client object.
- sf::TcpSocket & get_socket ()

  Get the socket object.
- ID get_identity () const

  Get the identity of the Client.
- void connect_to (const std::string &ip, int port)

  Utility function to connec to host.
- void send_packet (sf::Packet &packet)

  Utility to send packet over the network to the client connected to other end.
- void recieve_packet (sf::Packet &packet)

  Utility function to recieve a packet from other end.
- sf::Socket::Status getLastStatus () const

  Get the Last Status object.

Friends

- Client & operator<< (Client &client, const GameSimulatorSnap &snap)

  Friend function for operator<< overloaded for sending snap over the network.
- Client & operator>> (Client &client, key_input_type &labelled_input)

  Overloaded operator>> to send labelled input over the network.

5.7.1 Constructor & Destructor Documentation

5.7.1.1 Client()

```
cp::Client::Client (
             ID identity )  [inline]
```

Construct a new Client object.

Parameters

identity	desired id of the client.

5.7.2 Member Function Documentation

5.7.2.1 connect_to()

```
void cp::Client::connect_to (
             const std::string & ip,
             int port )  [inline]
```

Utility function to connec to host.

Parameters

ip	IP of the host.
port	PORT number of the host.

5.7.2.2 get_identity()

```
ID cp::Client::get_identity ( ) const  [inline]
```

Get the identity of the Client.

Returns

ID id of the client.

5.7.2.3 get_socket()

```
sf::TcpSocket& cp::Client::get_socket ( )  [inline]
```

Get the socket object.

Returns

sf::TcpSocket& Internal Socket of the client.

5.7.2.4 getLastStatus()

```
sf::Socket::Status cp::Client::getLastStatus ( ) const  [inline]
```

Get the Last Status object.

Returns

sf::Socket::Status Returns the status of the last call to send/recieve.

5.7.2.5 recieve_packet()

```
void cp::Client::recieve_packet (
             sf::Packet & packet )  [inline]
```

Utility function to recieve a packet from other end.

Parameters

packet	The packet that you want to recieve data into.

5.7.2.6 send_packet()

```
void cp::Client::send_packet (
             sf::Packet & packet )  [inline]
```

Utility to send packet over the network to the client connected to other end.

Parameters

packet	The packet that you want to send .

5.7.3 Friends And Related Function Documentation

5.7.3.1 operator<<

```
Client& operator<< (
            Client & client,
            const GameSimulatorSnap & snap )  [friend]
```

Friend function for operator<< overloaded for sending snap over the network.

Parameters

client	Client that you want to send the snap to.
snap	The snap that you want to send.

Returns

Client& Returns Client object back.

5.7.3.2 operator>>

```
Client& operator>> (
            Client & client,
            key_input_type & labelled_input )  [friend]
```

Overloaded operator>> to send labelled input over the network.

Parameters

client	The client object to which you want to send the labelled input
labelled_input	The labelled input that you want to send.

Returns

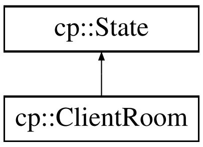
Client& Returns back the Client Reference.

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

- include/Network/Client.hpp

5.8 cp::ClientRoom Class Reference

Inheritance diagram for cp::ClientRoom:



Public Member Functions

- **ClientRoom** (GameDataRef _data)
- void init ()

  Initializing the ClientRoom components.
- void handle_input (float delta)

  THis function provides the interface to handle_input in ClientRoom.
- void update (float delta)

  Provides an interface to update the ClientRoom.
- void draw (float delta)

  Draw the components on the screen.
- void get_notifications ()

  Get the notifications from the server.
- void use_notification ()

  Utility function to use_recieved notification.

5.8.1 Member Function Documentation

5.8.1.1 draw()

```
void cp::ClientRoom::draw (
             float delta )  [inline], [virtual]
```

Draw the components on the screen.

Parameters

delta	

Implements cp::State.

5.8.1.2 handle_input()

```
void cp::ClientRoom::handle_input (
             float delta )  [inline], [virtual]
```

THis function provides the interface to handle_input in ClientRoom.

Parameters

delta	TIme difference between two handle_input calls.

Implements cp::State.

5.8.1.3 update()

```
void cp::ClientRoom::update (
             float delta )  [inline], [virtual]
```

Provides an interface to update the ClientRoom.

Parameters

delta	TIme difference between two update calls

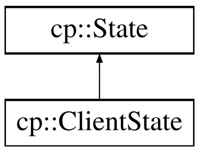
Implements cp::State.

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

- include/Network/ClientRoom.hpp

5.9 cp::ClientState Class Reference

Inheritance diagram for cp::ClientState:



Public Member Functions

- **ClientState** (GameDataRef _data, Server_ptr server, int unique_id)
- virtual void **init** ()
- virtual void **handle_input** (float delta)

- virtual void **update** (float delta)
- virtual void **draw** (float delta)
- virtual void **pause** ()
- virtual void **resume** ()

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

- include/States/ClientState.hpp
- libs/ClientState.cpp

5.10 cp::Collision Class Reference

Public Member Functions

- bool **handle_collision** (Car &car1, Car &car2, GameMap &map, float cor)

Static Public Member Functions

- static void **simulate_physics** (std::vector< Car ∗> &entities, GameMap &map)
- static void **single_entity_check** (std::vector< Car ∗> ∗entites_ptr, int index, GameMap ∗map_ptr, std↩ ::mutex ∗mutex_arr)
- static void **handle_collision** (Car &car1, Car &car2, GameMap &map)
- static void **cover_collided** (Car &car1, Car &car2, int diff, float cor)
- static bool **detect_collision** (const sf::Sprite &s1, const sf::Sprite &s2)

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

- include/Physics/Collision.hpp

5.11 cp::entity_info Class Reference

Public Member Functions

- **entity_info** (cp::PlayerCar &car)

Friends

- class **GameSimulator**
- std::ofstream & **operator**<< (std::ofstream &fout, const entity_info &entity_i)
- sf::Packet & **operator**<< (sf::Packet &fout, const entity_info &entity_i)
- sf::Packet & **operator**>> (sf::Packet &fin, entity_info &entity_i)

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

- include/States/GameSimulator.hpp

5.12 cp::Game Class Reference

Public Member Functions

- Game (int width, int height, std::string title)

    Construct a new Game:: Game object.
- ∼Game ()

    Destroy the Game:: Game object.

5.12.1 Constructor & Destructor Documentation

5.12.1.1 Game()

```
cp::Game::Game (
             int width,
             int height,
             std::string title )
```

Construct a new Game:: Game object.

Parameters

width	Width of the screen requested.
height	Height of the screen requested.
title	Title of the game screen.

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

- include/Game.hpp
- libs/Game.cpp

5.13 cp::GameData Struct Reference

Public Attributes

- StateMachine **machine**
- sf::RenderWindow **window**
- AssetManager **assets**
- InputManager **input**
- NetworkManager **Nmanager**

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

- include/Game.hpp

5.14 cp::GameMap Class Reference

Public Member Functions

- **GameMap** (GameDataRef _data)
- void **init** ()
- void **draw_quad** (sf::Color c, int x1, int y1, int w1, int x2, int y2, int w2)
- void **update** (float delta)
- void **project** (Line &line, float camX, float camY, float camZ, float camD)
- void **draw** (int count, const Camera &main_camera)
- void **drawSprite** (const Line &line)
- int **get_grid_index** (float distance)
- void **bound_entity** (cp::Car &car)
- void **bound_entity** (Camera &camera)
- void **bound_entity** (Bullet &bot)
- void **bound_entity** (Bot &bot)
- int **getRoadWidth** () const
- int **getSegL** () const
- int **getGridCount** () const
- int **getScreenWidth** () const
- int **getScreenHeight** () const

Public Attributes

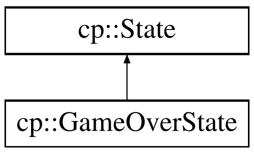
- std::vector< Line > **lines**

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

- include/Objects/GameMap.hpp
- libs/GameMap.cpp

5.15 cp::GameOverState Class Reference

Inheritance diagram for cp::GameOverState:



Public Member Functions

- **GameOverState** (GameDataRef _data)
- void **init** ()
- void **handle_input** (float delta)
- void **draw** (float delta)
- void **update** (float delta)

Public Attributes

- sf::Font **font**
- sf::Text **text**

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

- include/States/GameOverState.hpp
- libs/GameOverState.cpp

5.16 cp::GameSimulationLog Class Reference

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

- include/States/GameSimulator.hpp

5.17 cp::GameSimulator Class Reference

Public Types

- using **ID** = long long int
- using **input_type** = Bot::input_type
- using **input_return_type** = std::pair< ID, input_type >

Public Member Functions

- GameSimulator (GameDataRef res_store)

  Construct a new Game Simulator:: Game Simulator object.
- ∼GameSimulator ()

  Destroy the Game Simulator:: Game Simulator object.
- GameSimulatorSnap get_current_snap (SnapFlag flag)

  Get the current snap object.
- GameSimulationLog use_snap (const GameSimulatorSnap &snap, bool is_forced=true)

  Updates the game simulation with the snap provided.
- float **distance** (entity_info &a, entity_info &b)
- void **output** (entity_info &a, entity_info &b, std::vector< bool > &input)
- void **AI_bot_output** ()
- void **generate_log** ()
- void init ()

  Initializing all the entities in the Game.
- void handle_input (float delta)

  This function provide space for doing handle input on all the entities.
- void draw (float delta)

  This method provide the room for drawing all the elements on the window.
- void update (float delta)

  This method provide room for updating all the entities.
- PlayerCar generate_bot (const entity_info &info)

Utility function to generate the bots.
- bool add_external_player (ID id)

  Adds external player with their id.
- bool add_bot_players ()

  Adds a bot player in the simulation.
- void remove_ext_player (ID id)

  Removes an external player if available.
- bool update_main_player (ID id)

  Update the main player of the simulation.
- bool is_main_player_available ()

  Checks if main player is in the simulation.
- input_return_type **get_input** ()
- void **focus_on** (ID id)

Public Attributes

- std::ofstream **fout**

5.17.1 Constructor & Destructor Documentation

5.17.1.1 GameSimulator()

```
cp::GameSimulator::GameSimulator (
            GameDataRef res_store )
```

Construct a new Game Simulator:: Game Simulator object.

Parameters

res_store	Contains all resource managers

5.17.2 Member Function Documentation

5.17.2.1 add_bot_players()

```
bool cp::GameSimulator::add_bot_players ( )  [inline]
```

Adds a bot player in the simulation.

Returns

true Returns true if bot addition was a success.
false Returns false otherwise.

5.17.2.2 add_external_player()

```
bool cp::GameSimulator::add_external_player (
             ID id )  [inline]
```

Adds external player with their id.

Parameters

id	This is the id they have requested.

Returns

true if player addition is successful
false if player addition fails

5.17.2.3 draw()

```
void cp::GameSimulator::draw (
             float delta )
```

This method provide the room for drawing all the elements on the window.

Parameters

delta	Time difference between two accumulator

5.17.2.4 generate_bot()

```
PlayerCar cp::GameSimulator::generate_bot (
             const entity_info & info )
```

Utility function to generate the bots.

Parameters

info	uses the info provided in the argument to generate the bot

Returns

PlayerCar Returns the generated object

5.17.2.5 get_current_snap()

```
GameSimulatorSnap cp::GameSimulator::get_current_snap (
              SnapFlag flag )
```

Get the current snap object.

Returns a snap of the game such that the simulation can be recreated.

Parameters

flag	

Returns

GameSimulatorSnap

Parameters

flag	Type of snap that you want (NETWORK/OFFLINE)

Returns

GameSimulatorSnap The current snap of the game.

5.17.2.6 handle_input()

```
void cp::GameSimulator::handle_input (
              float delta )
```

This function provide space for doing handle input on all the entities.

Parameters

delta	The time difference between two frames

5.17.2.7 is_main_player_available()

```
bool cp::GameSimulator::is_main_player_available ( )  [inline]
```

Checks if main player is in the simulation.

Returns

true if main player is found.
false if main player not found.

5.17.2.8 remove_ext_player()

```
void cp::GameSimulator::remove_ext_player (
             ID id )  [inline]
```

Removes an external player if available.

Parameters

id	Id of the external player to remove.

5.17.2.9 update()

```
void cp::GameSimulator::update (
             float delta )
```

This method provide room for updating all the entities.

Parameters

delta	This is the time difference between two frames.

5.17.2.10 update_main_player()

```
bool cp::GameSimulator::update_main_player (
             ID id )  [inline]
```

Update the main player of the simulation.

Parameters

id	Update the main player with ID

Returns

true If operation is succesfull.
false If operation is unsuccessfull.

5.17.2.11 use_snap()

```
GameSimulationLog cp::GameSimulator::use_snap (
             const GameSimulatorSnap & snap,
             bool is_forced = true )
```

Updates the game simulation with the snap provided.

Calling this function will replace all the entities and their info with info in snap argument.

Parameters

snap	Refers to the snap of the game to update the simulation with.
is_forced	If set then forcefully overwrites the snap provided.

Returns

GameSimulationLog Returns a log describing whether the replacment was partial/discarded/sucess.

Parameters

snap	Snap that you want to replace the GameInfo with
is_forced	Forcefully replace all the info with the snap info

Returns

GameSimulationLog Returns a log file illustrating the success of the operation.

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

- include/States/GameSimulator.hpp
- libs/GameSimulator.cpp

5.18 cp::GameSimulatorSnap Class Reference

Public Member Functions

- **GameSimulatorSnap** (int a, int b, int c, int d, std::map< ID, PlayerCar > &players_map)

Friends

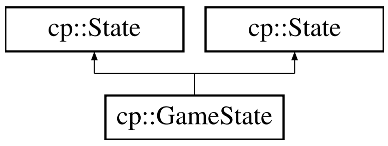
- class **GameSimulator**
- std::ofstream & **operator**<< (std::ofstream &fout, const GameSimulatorSnap &snap)
- sf::Packet & **operator**<< (sf::Packet &fout, const GameSimulatorSnap &snap)
- sf::Packet & **operator**>> (sf::Packet &fin, GameSimulatorSnap &snap)

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

- include/States/GameSimulator.hpp

5.19 cp::GameState Class Reference

Inheritance diagram for cp::GameState:



Public Types

- typedef std::shared_ptr< PlayerCar > **CarRef**

Public Member Functions

- **GameState** (GameDataRef _data)
- void **init** ()
- void **handle_input** (float delta)
- void **draw** (float delta)
- void **update** (float delta)
- void **drawSprite** (Line &line)
- **GameState** (GameDataRef _data)
- virtual void **init** ()
- virtual void **handle_input** (float delta)

Static Public Member Functions

- static void **network_handler** (GameDataRef data, std::shared_ptr< PlayerCar > car, std::shared_ptr< Bot > bot)

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

- include/GameState.hpp
- libs/GameState.cpp

5.20 cp::InputManager Class Reference

Public Member Functions

- bool **is_sprite_clicked** (sf::Sprite sprite, sf::Mouse::Button button, sf::RenderWindow &window)
- sf::Vector2i **get_mouse_position** (sf::RenderWindow &window)
- void **register_input** (register_input_type input_pair)
- input_type **get_mask** (ID id)

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

- include/ResourceManagers/InputManager.hpp
- libs/InputManager.cpp

5.21 cp::Line Class Reference

Public Attributes

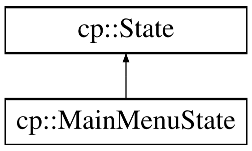
- float **x** = 0
- float **y** = 0
- float **z** = 0
- float **no_curve_Y** =0
- float **no_curve_X** =0
- float **X** = 0
- float **Y** = 0
- float **W** = 0
- float **curve** = 0
- float **spriteX** = 0
- float **clip** = 0
- float **scale** = 0
- sf::Sprite **sprite**

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

- include/Objects/Line.hpp

5.22 cp::MainMenuState Class Reference

Inheritance diagram for cp::MainMenuState:



Public Member Functions

- MainMenuState (GameDataRef _data)

  Construct a new Main Menu State:: Main Menu State object.
- void init ()

  This function initializes all the components of MainMenu state.
- void handle_input (float delta)

  Provide interface to handle inputs in the MainMenu State.
- void draw (float delta)

  Function to draw all the components of MainMenu state on the screen.
- void **update** (float delta)

Public Attributes

- bool **update_required** = true

5.22.1 Constructor & Destructor Documentation

5.22.1.1 MainMenuState()

```
cp::MainMenuState::MainMenuState (
             GameDataRef _data )
```

Construct a new Main Menu State:: Main Menu State object.

Parameters

_data	Pointer to all the resource managers and window

5.22.2 Member Function Documentation

5.22.2.1 draw()

```
void cp::MainMenuState::draw (
             float delta )  [virtual]
```

Function to draw all the components of MainMenu state on the screen.

Parameters

delta	

Implements cp::State.

5.22.2.2 handle_input()

```
void cp::MainMenuState::handle_input (
             float delta )  [virtual]
```

Provide interface to handle inputs in the MainMenu State.

Parameters

delta	Time difference between two handle_input call

Implements cp::State.

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

- include/States/MainMenuState.hpp
- libs/MainMenuState.cpp

5.23 cp::NetworkManager Class Reference

Static Public Member Functions

- static void **createServer** ()
- static void **createClient** ()
- static void **run** (int type)
- static void **sendData** (sf::Vector3f pos)
- static void **send** (sf::Vector3f pos)

Public Attributes

- sf::Socket::Status **s_status**
- sf::Socket::Status **c_status**
- std::thread **n_thread**

Static Public Attributes

- static sf::TcpSocket **client**

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

- include/NetworkManager.hpp
- libs/NetworkManager.cpp

5.24 cp::ObjectPool< T > Class Template Reference

Public Member Functions

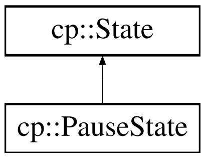
- **ObjectPool** (size_t size)
- T ∗ **getObject** (GameDataRef _data, int car_num)
- void **returnObject** (T ∗obj)

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

- include/ObjectPool.hpp

5.25 cp::PauseState Class Reference

Inheritance diagram for cp::PauseState:



Public Member Functions

- **PauseState** (GameDataRef _data)
- void **init** ()
- void **handle_input** (float delta)
- void **draw** (float delta)
- void **update** (float delta)

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

- include/States/PauseState.hpp
- libs/PauseState.cpp

5.26 cp::PercentageBar Class Reference

Public Member Functions

- **PercentageBar** (GameDataRef _data)
- void **init** (sf::Vector2f size, sf::Vector2f position, sf::Color c1, sf::Color c2)
- void **draw** ()

Public Attributes

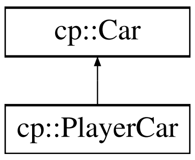
- float **percentage** = 100

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

- include/PercentageBar.hpp

5.27 cp::PlayerCar Class Reference

Inheritance diagram for cp::PlayerCar:



Public Member Functions

- PlayerCar (GameDataRef _data, int _car_num)

  Construct a new Player Car:: Player Car object.
- ∼PlayerCar ()

  Destroy the Player Car:: Player Car object.
- void update_car (float dt, const std::vector< Line > &lines, float segL)

  Update the car according to it's position and the map.
- void drawSprite (const Line &line)

  Utility function to draw sprites of the car.
- void **drawUsingCamera** (const Camera &main_camera)
- void **project** (Line &line, float camX, float camY, float camZ, float camD)
- void handle_input (std::vector< bool > mask, float dt)

  Provides interface to control the car.

Public Attributes

- float **friction** = e_max_speed.z/5

5.27.1 Constructor & Destructor Documentation

5.27.1.1 PlayerCar()

```
cp::PlayerCar::PlayerCar (
            GameDataRef _data,
            int car_num )
```

Construct a new Player Car:: Player Car object.

Parameters

_data	Pointer Reference to the resources and state machines.
car_num	The sprite number for the car object

5.27.2 Member Function Documentation

5.27.2.1 drawSprite()

```
void cp::PlayerCar::drawSprite (
            const Line & line )
```

Utility function to draw sprites of the car.

Parameters

line	Provides info regarding map scale at the current grid the car is positioned at

5.27.2.2 handle_input()

```
void cp::PlayerCar::handle_input (
             std::vector< bool > mask,
             float dt )
```

Provides interface to control the car.

Parameters

mask	Boolean mask indicating the Keyboard inputs.
dt	Time difference between two handle_input call.

5.27.2.3 update_car()

```
void cp::PlayerCar::update_car (
             float dt,
             const std::vector< Line > & lines,
             float segL ) [virtual]
```

Update the car according to it's position and the map.

Parameters

dt	Time difference between two update frames
lines	provides the map grid info
segL	Segment line between two grid in the map

Implements cp::Car.

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

- include/Objects/PlayerCar.hpp
- libs/PlayerCar.cpp

5.28 cp::Server Class Reference

Public Member Functions

- Server ()

Construct a new Server object.
- ID get_identity () const
  Get the identity object.
- sf::TcpSocket & get_socket ()
  Get the socket object.
- sf::Socket::Status getLastStatus () const
  Get the Last Status object.
- void connect_to (const std::string &ip, int port)
  connect to the specified port and ip
- void send_packet (sf::Packet &packet)
  send packet
- void **recieve_packet** (sf::Packet &packet)

Friends

- Server & operator<< (Server &server, const key_input_type &labelled_input)
  Overloaded operator<< to send the labelled input.
- Server & operator>> (Server &server, GameSimulatorSnap &snap)
  Overloaded operator>> to recieve GameSnaps.

5.28.1 Member Function Documentation

5.28.1.1 connect_to()

```
void cp::Server::connect_to (
             const std::string & ip,
             int port )  [inline]
```

connect to the specified port and ip

Parameters

ip	IP of the server.
port	PORT of the server.

5.28.1.2 get_identity()

```
ID cp::Server::get_identity ( ) const  [inline]
```

Get the identity object.

Returns

ID id of the Server.

5.28.1.3 get_socket()

```
sf::TcpSocket& cp::Server::get_socket ( )  [inline]
```

Get the socket object.

Returns

sf::TcpSocket& INternal handle of the socket.

5.28.1.4 getLastStatus()

```
sf::Socket::Status cp::Server::getLastStatus ( ) const  [inline]
```

Get the Last Status object.

Returns

sf::Socket::Status Status of the last function call to recieve/send.

5.28.1.5 send_packet()

```
void cp::Server::send_packet (
            sf::Packet & packet )  [inline]
```

send packet

Parameters

packet	

5.28.2 Friends And Related Function Documentation

5.28.2.1 operator<<

```
Server& operator<< (
            Server & server,
            const key_input_type & labelled_input )  [friend]
```

Overloaded operator<< to send the labelled input.

Parameters

server	The server that you want to send the labelled input to.
labelled_input	The labelled input to send.

Returns

Server& Returns back the server reference.

5.28.2.2 operator>>

```
Server& operator>> (
            Server & server,
            GameSimulatorSnap & snap )  [friend]
```

Overloaded operator>> to recieve GameSnaps.

Parameters

server	The server that you want to recieve snap from.
snap	The snap reference for getting incoming snap.

Returns

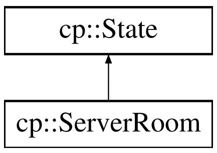
Server& Returns back the server.

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

- include/Network/Server.hpp

5.29 cp::ServerRoom Class Reference

Inheritance diagram for cp::ServerRoom:



Public Member Functions

- **ServerRoom** (GameDataRef _data)
- void **init** ()
- virtual void **handle_input** (float delta)
- virtual void **update** (float delta)
- virtual void **draw** (float delta)
- virtual void **pause** ()
- virtual void **resume** ()

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

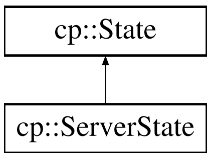
- include/Network/ServerRoom.hpp

5.30 cp::ServerState Class Reference

ServerState is the state where network updates takes place. It is the game simulator which simulates the game and then update the relevant data over the network.

```
#include <ServerState.hpp>
```

Inheritance diagram for cp::ServerState:



Public Member Functions

- **ServerState** (GameDataRef _data, std::set< TcpClient_ptr > clients)
- virtual void **handle_input** (float delta)
- virtual void **update** (float delta)
- virtual void **draw** (float delta)
- virtual void **pause** ()
- virtual void **resume** ()
- virtual void **init** ()

5.30.1 Detailed Description

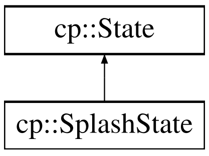
ServerState is the state where network updates takes place. It is the game simulator which simulates the game and then update the relevant data over the network.

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

- include/States/ServerState.hpp
- libs/ServerState.cpp

5.31 cp::SplashState Class Reference

Inheritance diagram for cp::SplashState:



Public Member Functions

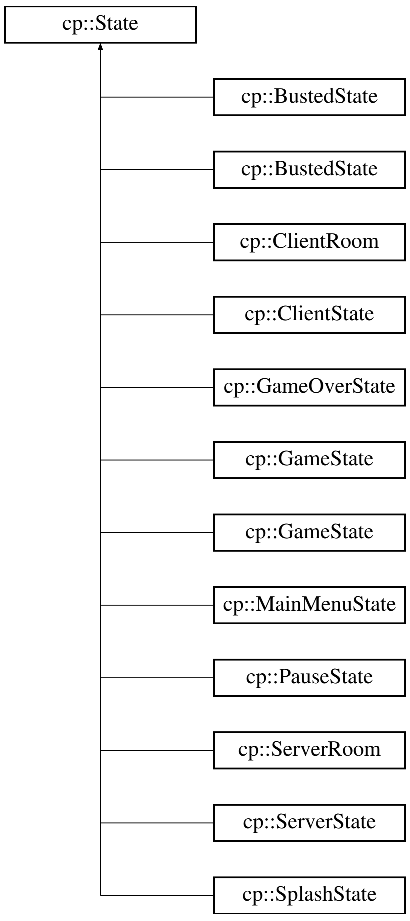
- **SplashState** (GameDataRef _data)
- void **init** ()
- void **handle_input** (float delta)
- void **draw** (float delta)
- void **update** (float delta)

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

- include/States/SplashState.hpp
- libs/SplashState.cpp

5.32 cp::State Class Reference

Inheritance diagram for cp::State:



Public Member Functions

- virtual void **init** ()=0
- virtual void **handle_input** (float delta)=0
- virtual void **update** (float delta)=0
- virtual void **draw** (float delta)=0
- virtual void **pause** ()
- virtual void **resume** ()

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

- include/States/State.hpp

5.33 cp::StateMachine Class Reference

Public Member Functions

- void **add_state** (StateRef new_state, bool is_replacing=true)
- void **remove_state** ()
- void **process_state_change** ()
- StateRef & **get_active_state** ()

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

- include/States/StateMachine.hpp
- libs/StateMachine.cpp

Chapter 6

File Documentation

6.1 include/Network/Client.hpp File Reference

Client class refers to a another pc connected over network.

```
#include <string>
#include "States/GameSimulator.hpp"
#include "SFML/Network.hpp"
```

Classes

- class cp::Client

6.1.1 Detailed Description

Client class refers to a another pc connected over network.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-26

Copyright

Copyright (c) 2019

6.2 include/Network/ClientRoom.hpp File Reference

The client room that is displayed on client's computer just before online game play.

```
#include "States/State.hpp"
#include "Game.hpp"
#include "Network/Server.hpp"
#include <iostream>
#include <fstream>
#include <cstring>
#include <memory>
#include "States/ClientState.hpp"
```

Classes

- class cp::ClientRoom

6.2.1 Detailed Description

The client room that is displayed on client's computer just before online game play.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-28

Copyright

Copyright (c) 2019

6.3 include/Network/Server.hpp File Reference

Server class that handles the data sending over the network.

```
#include "States/GameSimulator.hpp"
```

Classes

- class cp::Server

6.3.1 Detailed Description

Server class that handles the data sending over the network.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-26

Copyright

Copyright (c) 2019

6.4 include/Network/ServerRoom.hpp File Reference

Simple Server room displayed before Online Play Starts.

```
#include "States/State.hpp"
#include "Game.hpp"
#include "Network/Client.hpp"
#include <iostream>
#include <cstring>
#include "States/ServerState.hpp"
#include <set>
#include <fstream>
#include <memory>
```

Classes

- class cp::ServerRoom

6.4.1 Detailed Description

Simple Server room displayed before Online Play Starts.

Author

Anjani (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-27

Copyright

Copyright (c) 2019

6.5 include/ObjectPool.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <list>
#include <iostream>
#include <memory>
#include "Objects/Car.hpp"
#include "Objects/Bot.hpp"
#include "Game.hpp"
#include "Objects/Bullet.hpp"
```

Classes

- class cp::ObjectPool< T >

6.5.1 Detailed Description

Author

your name (`you@domain.com`)

Version

0.1

Date

2019-02-28

Copyright

Copyright (c) 2019

6.6 include/States/ClientState.hpp File Reference

Handles communication between the computer and the server.

```
#include "States/State.hpp"
#include "Game.hpp"
#include "States/GameSimulator.hpp"
#include "Network/Server.hpp"
```

Classes

- class cp::ClientState

6.6.1 Detailed Description

Handles communication between the computer and the server.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-26

Copyright

Copyright (c) 2019

6.7 include/States/GameSimulator.hpp File Reference

A game simulator just like Game class but it get's its clock sync and resource manager from object owner.

```
#include "Game.hpp"
#include <SFML/Graphics.hpp>
#include "GameOverState.hpp"
#include "DEFINITIONS.hpp"
#include "Objects/Bot.hpp"
#include "Objects/PlayerCar.hpp"
#include "Objects/Line.hpp"
#include "Physics/Collision.hpp"
#include "Objects/Camera.hpp"
#include "Objects/GameMap.hpp"
#include <memory>
#include <fstream>
#include <set>
#include <SFML/Network.hpp>
#include "PercentageBar.hpp"
#include "Objects/Bullet.hpp"
#include "ObjectPool.hpp"
#include "States/MainMenuState.hpp"
```

Classes

- class cp::entity_info
- class cp::GameSimulatorSnap
- class cp::GameSimulationLog
- class cp::GameSimulator

Macros

- #define **lli** long long int

Enumerations

- enum **SnapFlag** { **NETWORK_SNAP**, **OFFLINE_SNAP** }

6.7.1 Detailed Description

A game simulator just like Game class but it get's its clock sync and resource manager from object owner.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-26

Copyright

Copyright (c) 2019

6.8 include/States/ServerState.hpp File Reference

The ServerState maintains and simulate online game play.

```
#include "States/State.hpp"
#include "Game.hpp"
#include "States/GameSimulator.hpp"
#include "Network/Client.hpp"
#include <map>
#include <vector>
#include <fstream>
#include <set>
```

Classes

- class cp::ServerState

  ServerState is the state where network updates takes place. It is the game simulator which simulates the game and then update the relevant data over the network.

6.8.1 Detailed Description

The ServerState maintains and simulate online game play.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-02-28

Copyright

Copyright (c) 2019

6.9 libs/Game.cpp File Reference

Provide all the game play resources to play the game and all provides cpu time to different states.

```
#include "Game.hpp"
#include "States/SplashState.hpp"
```

6.9.1 Detailed Description

Provide all the game play resources to play the game and all provides cpu time to different states.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-03-01

Copyright

Copyright (c) 2019

6.10 libs/GameSimulator.cpp File Reference

GameSimulator.cpp file contains the implementations of methods in GameSimulator.hpp.

```
#include "States/GameSimulator.hpp"
```

6.10.1 Detailed Description

GameSimulator.cpp file contains the implementations of methods in GameSimulator.hpp.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-03-01

Copyright

Copyright (c) 2019

6.11 libs/MainMenuState.cpp File Reference

State that represents the MainMenu in the game.

```
#include "States/MainMenuState.hpp"
#include "States/GameState.hpp"
#include "DEFINITIONS.hpp"
#include "NetworkManager.hpp"
#include <iostream>
#include <thread>
#include <sstream>
#include "Network/ServerRoom.hpp"
#include "Network/ClientRoom.hpp"
```

6.11.1 Detailed Description

State that represents the MainMenu in the game.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-03-01

Copyright

Copyright (c) 2019

6.12 libs/PlayerCar.cpp File Reference

PlayerCar.cpp provides Car object.

```
#include <iostream>
#include <cmath>
#include <sstream>
#include "Objects/PlayerCar.hpp"
#include "DEFINITIONS.hpp"
```

6.12.1 Detailed Description

PlayerCar.cpp provides Car object.

Author

Anjani Kumar (cs17btech11002@iith.ac.in)

Version

0.1

Date

2019-03-01

Copyright

Copyright (c) 2019

Index