OOP Project

Generated by Doxygen 1.9.2

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

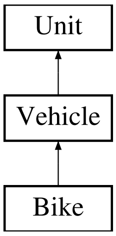
Class Documentation

4.1 Bike Class Reference

This is the class for the Bike type objects.

```
#include <Bike.hpp>
```

Inheritance diagram for Bike:



Public Member Functions

- Bike (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)

  Construct a new Bike:: Bike object.
- void draw (bool flag)

  Draws the Bike type Object.

4.1.1 Detailed Description

This is the class for the Bike type objects.

This class inherits from Vehicle class and provides the sprite and mover coordinates to it.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Bike()

```
Bike::Bike (
            SDL_Renderer * rend,
            SDL_Texture * ast,
            SDL_Rect mov )
```

Construct a new Bike:: Bike object.

Parameters

rend	
ast	
mov	

4.1.3 Member Function Documentation

4.1.3.1 draw()

```
void Bike::draw (
             bool flag )
```

Draws the Bike type Object.

Calls the draw function of Vehicle and passes the flag to the draw function of Vehicle class which then determines whether the Bike will move forward or not.

Parameters

flag	

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

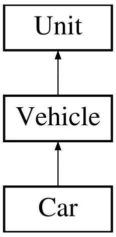
- Bike.hpp
- Bike.cpp

4.2 Car Class Reference

This is the class for the Car type objects.

`#include <Car.hpp>`

Inheritance diagram for Car:



Public Member Functions

- Car (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)

  Construct a new Car:: Car object.
- void draw (bool flag)

  Draws the Car type Object.

4.2.1 Detailed Description

This is the class for the Car type objects.

This class inherits from Vehicle class and provides the sprite and mover coordinates to it.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Car()

```
Car::Car (
             SDL_Renderer ∗ rend,
             SDL_Texture ∗ ast,
             SDL_Rect mov )
```

Construct a new Car:: Car object.

Parameters

rend	
ast	
mov	

4.2.3 Member Function Documentation

4.2.3.1 draw()

```
void Car::draw (
             bool flag )
```

Draws the Car type Object.

Calls the draw function of Vehicle and passes the flag to the draw function of Vehicle class which then determines whether the Car will move forward or not.

Parameters

flag	

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

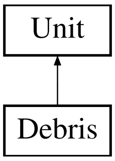
- Car.hpp
- Car.cpp

4.3 Debris Class Reference

This class creates the debris when a Vehicle is destroyed.

```
#include <Debris.hpp>
```

Inheritance diagram for Debris:



Public Member Functions

- void **animate** ()

  Animates and draws the debris.
- Debris (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)

  Construct a new Debris:: Debris object.

4.3.1 Detailed Description

This class creates the debris when a Vehicle is destroyed.

Whenever a Car collides with another Car or with Sheep an explosion animation is displayed, this class handles the animation and drawing of the explosion.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Debris()

```
Debris::Debris (
             SDL_Renderer * rend,
             SDL_Texture * ast,
             SDL_Rect mov )
```

Construct a new Debris:: Debris object.

Parameters

rend	
ast	
mov	

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

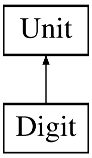
- Debris.hpp
- Debris.cpp

4.4 Digit Class Reference

Digit used in timer.

`#include <Digit.hpp>`

Inheritance diagram for Digit:



Public Member Functions

- Digit (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov, SDL_Rect src)

  Construct a new Digit:: Digit object.
- void draw ()

  Draws the digit.
- void animate (unsigned int res)

  Animates the digit.

4.4.1 Detailed Description

Digit used in timer.

Creates a digit type object to be used in the timer.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Digit()

```
Digit::Digit (
             SDL_Renderer ∗ rend,
             SDL_Texture ∗ ast,
             SDL_Rect mov,
             SDL_Rect src )
```

Construct a new Digit:: Digit object.

Parameters

rend	
ast	
mov	
src	

4.4.3 Member Function Documentation

4.4.3.1 animate()

```
void Digit::animate (
             unsigned int res )
```

Animates the digit.

Changes the src of the digits based on the counter

Parameters

res	

4.4.3.2 draw()

```
void Digit::draw ( )
```

Draws the digit.

Calls the draw function of the Unit class to draw the digit.

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

- Digit.hpp
- Digit.cpp

4.5 Game Class Reference

Game class is the container for Jamalo class and handles input and output interfacing.

```
#include <game.hpp>
```

Public Member Functions

- bool **init** ()
- bool **loadMedia** ()
- void **close** ()
- SDL_Texture ∗ **loadTexture** (std::string path)
- void **run** ()

4.5.1 Detailed Description

Game class is the container for Jamalo class and handles input and output interfacing.

game class takes the inputs from the users and provides it to the jamalo class to be processed. It also takes care of the start and the end screens.

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

- game.hpp
- game.cpp

4.6 Jamalo Class Reference

This is the Level Design class, and contains all other game related classes.

```
#include <Jamalo.hpp>
```

Public Member Functions

- Jamalo (SDL_Renderer ∗, SDL_Texture ∗)

  Construct a new Jamalo:: Jamalo object.
- void drawObjects ()

  Draws all objects.
- void createObjects ()

  Generates Vehicle type objects.
- void move (char x)

  Calls the move function of the Sheep and passes it the user input.
- bool hitRegistered (Vehicle v)

  Checks if vehicle is colliding with sheep.
- bool hitRegistered (Vehicle v, list< Vehicle > &vehicles)

  Checks if any vehicles are currently colliding.
- void radar (Vehicle &currentV)

  Implementation of radar system in the Vehicles.
- bool isOver ()

  Checks game state.

4.6.1 Detailed Description

This is the Level Design class, and contains all other game related classes.

The Jamalo class aggregates all other game related classes, namely Sheep, Vehicle and its childs, Timer and Lives. All top-level functionality is implemented in this class, the purpose of this class is to connect all objects in a proper manner, and to carry out class operations over them. Majority of the game mechanics are implemented here. This class: -Draws all objects -Creates all objects -Contains the radar system through which vehicles slow down if there is a vehicle in front of them -Checks for collisions between objects

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Jamalo()

```
Jamalo::Jamalo (
             SDL_Renderer * renderer,
             SDL_Texture * asst )
```

Construct a new Jamalo:: Jamalo object.

Creates a Jamalo type object and initializes values for all other objects in the game.

Parameters

renderer	
asst	

4.6.3 Member Function Documentation

4.6.3.1 createObjects()

```
void Jamalo::createObjects ( )
```

Generates Vehicle type objects.

Randomly generates Vehicle type objects and randomly selects a lane for them to spawn in.

4.6.3.2 drawObjects()

```
void Jamalo::drawObjects ( )
```

Draws all objects.

Iterates over the list of all vehicles and calls their draw function. It also calls the draw function of all other objects.

4.6.3.3 hitRegistered() [1/2]

```
bool Jamalo::hitRegistered (
            Vehicle vehicle_hit )
```

Checks if vehicle is colliding with sheep.

The function is an overloaded instance which checks if the given vehicle is colliding with the sheep or not.

Parameters

vehicle_hit	

Returns

true

false

4.6.3.4 hitRegistered() [2/2]

```
bool Jamalo::hitRegistered (
            Vehicle v,
            list< Vehicle > & vehicles )
```

Checks if any vehicles are currently colliding.

The function takes a Vehicle type object and compares it with all other Vehicles to see if their mover rectangles are overlapping or not, this is to check if the vehicles are colliding with each other or not.

Parameters

v	
vehicles	

Returns

true

false

4.6.3.5 isOver()

```
bool Jamalo::isOver ( )
```

Checks game state.

Returns true if game is over i.e all 3 lives have been lost.

Returns

true

false

4.6.3.6 move()

```
void Jamalo::move (
            char x )
```

Calls the move function of the Sheep and passes it the user input.

Parameters

x	

4.6.3.7 radar()

```
void Jamalo::radar (
            Vehicle & current )
```

Implementation of radar system in the Vehicles.

The function a vehicle and determines the distance between that vehicle and the vehicle in front of it, if there is a vehicle in front of the current vehicle then it gives a signal for the current vehicle to stop. It is also responsity for the drawing the vehicles.

Parameters

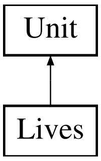
current	

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

- Jamalo.hpp
- Jamalo.cpp

4.7 Lives Class Reference

Inheritance diagram for Lives:



Public Member Functions

- **Lives** (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)
- void **draw** ()

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

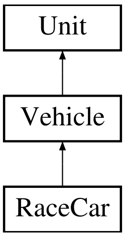
- Lives.hpp
- Lives.cpp

4.8 RaceCar Class Reference

This is the class for the Racecar type objects.

```
#include <Racecar.hpp>
```

Inheritance diagram for RaceCar:



Public Member Functions

- RaceCar (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)

  Construct a new Race Car:: Race Car object.
- void draw (bool flag)

  Draws the Racecar type Object.

4.8.1 Detailed Description

This is the class for the Racecar type objects.

This class inherits from Vehicle class and provides the sprite and mover coordinates to it.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 RaceCar()

```
RaceCar::RaceCar (
             SDL_Renderer * rend,
             SDL_Texture * ast,
             SDL_Rect mov )
```

Construct a new Race Car:: Race Car object.

Parameters

rend	
ast	
mov	

4.8.3 Member Function Documentation

4.8.3.1 draw()

```
void RaceCar::draw (
             bool flag )
```

Draws the Racecar type Object.

Calls the draw function of Vehicle and passes the flag to the draw function of Vehicle class which then determines whether the Racecar will move forward or not.

Parameters

flag	

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

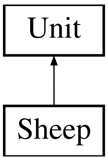
- Racecar.hpp
- Racecar.cpp

4.9 Sheep Class Reference

This class generates the Sheep.

```
#include <Sheep.hpp>
```

Inheritance diagram for Sheep:



Public Member Functions

- Sheep (SDL_Renderer ∗rend, SDL_Texture ∗ast)

  Construct a new Sheep:: Sheep object.
- SDL_Rect getter () const

  The getter function for the mover rectangle of the Sheep.
- void draw ()

  Draws the sheep.
- void move (char x)

  Moves the sheep.
- void **getHit** ()

  A function to display effects when sheep gets hit.

4.9.1 Detailed Description

This class generates the Sheep.

The protagonist sheep and functionality related to its movement, drawing and the hit reactions are implemented here.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Sheep()

```
Sheep::Sheep (
            SDL_Renderer ∗ rend,
            SDL_Texture ∗ ast )
```

Construct a new Sheep:: Sheep object.

Parameters

rend	
ast	

4.9.3 Member Function Documentation

4.9.3.1 draw()

```
void Sheep::draw ( )
```

Draws the sheep.

Calls the draw function from the Unit class and passes the relevant functions to it.

4.9.3.2 getter()

```
SDL_Rect Sheep::getter ( ) const
```

The getter function for the mover rectangle of the Sheep.

Returns

SDL_Rect

4.9.3.3 move()

```
void Sheep::move (
              char x )
```

Moves the sheep.

The function takes input from the Jamalo class and moves the sheep based on the key pressed. If sheep is at the border then it prevents it from moving any further.

Parameters

x	

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

- Sheep.hpp
- Sheep.cpp

4.10 Timer Class Reference

Public Member Functions

- **Timer** (SDL_Renderer ∗rend, SDL_Texture ∗ast)
- void **draw** ()
- void **count** ()

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

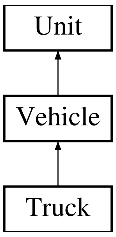
- Timer.hpp
- Timer.cpp

4.11 Truck Class Reference

This is the class for the Truck type objects.

```
#include <Truck.hpp>
```

Inheritance diagram for Truck:



Public Member Functions

- Truck (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)

  Construct a new Truck:: Truck object.
- void draw (bool flag)

  Draws the Truck type Object.

4.11.1 Detailed Description

This is the class for the Truck type objects.

This class inherits from Vehicle class and provides the sprite and mover coordinates to it.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Truck()

```
Truck::Truck (
             SDL_Renderer ∗ rend,
             SDL_Texture ∗ ast,
             SDL_Rect mov )
```

Construct a new Truck:: Truck object.

Parameters

rend	
ast	
mov	

4.11.3 Member Function Documentation

4.11.3.1 draw()

```
void Truck::draw (
            bool flag )
```

Draws the Truck type Object.

Calls the draw function of Vehicle and passes the flag to the draw function of Vehicle class which then determines whether the Truck will move forward or not.

Parameters

flag	

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

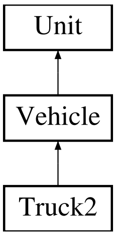
- Truck.hpp
- Truck.cpp

4.12 Truck2 Class Reference

This is the class for the Truck2 type objects.

`#include <Truck2.hpp>`

Inheritance diagram for Truck2:



Public Member Functions

- Truck2 (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov)

  Construct a new Truck 2:: Truck 2 object.
- void draw (bool flag)

  Draws the Truck2 type Object.

4.12.1 Detailed Description

This is the class for the Truck2 type objects.

This class inherits from Vehicle class and provides the sprite and mover coordinates to it.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Truck2()

```
Truck2::Truck2 (
             SDL_Renderer * rend,
             SDL_Texture * ast,
             SDL_Rect mov )
```

Construct a new Truck 2:: Truck 2 object.

Parameters

rend	
ast	
mov	

4.12.3 Member Function Documentation

4.12.3.1 draw()

```
void Truck2::draw (
             bool flag )
```

Draws the Truck2 type Object.

Calls the draw function of Vehicle and passes the flag to the draw function of Vehicle class which then determines whether the Truck will move forward or not.

Parameters

flag	

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

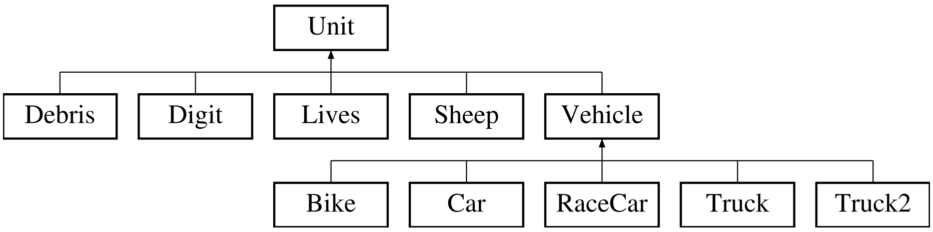
- Truck2.hpp
- Truck2.cpp

4.13 Unit Class Reference

The unit class from which all game related classes are inherited.

```
#include <Unit.hpp>
```

Inheritance diagram for Unit:



Public Member Functions

- Unit (SDL_Renderer ∗, SDL_Texture ∗)

  Construct a new Unit:: Unit object.
- void draw (SDL_Rect srcRect, SDL_Rect moverRect)

  Draws an object.

4.13.1 Detailed Description

The unit class from which all game related classes are inherited.

This class provides the drawing functionality to all other classes that need to draw a sprite on the screen.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Unit()

```
Unit::Unit (
             SDL_Renderer ∗ rend,
             SDL_Texture ∗ ast )
```

Construct a new Unit:: Unit object.

Parameters

rend	
ast	

4.13.3 Member Function Documentation

4.13.3.1 draw()

```
void Unit::draw (
             SDL_Rect srcRect,
             SDL_Rect moverRect )
```

Draws an object.

This function draws an object based on the src and mover rectangles passes to it.

Parameters

srcRect	
moverRect	

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

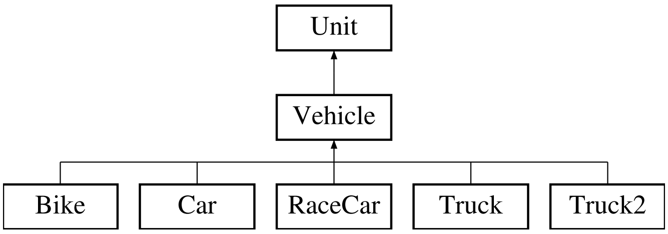
- Unit.hpp
- Unit.cpp

4.14 Vehicle Class Reference

Vehicle class, the parent for all vehicles.

`#include <Vehicle.hpp>`

Inheritance diagram for Vehicle:



Public Member Functions

- Vehicle (SDL_Renderer ∗rend, SDL_Texture ∗ast, SDL_Rect mov, SDL_Rect src, int s)

  Construct a new Vehicle:: Vehicle object.
- void draw (bool flag)

  Draws the Vehicles.
- SDL_Rect getter () const

  Returns mover rect of the Vehicle object.
- void changeLaneUp ()

  Moves the car to the upper lane if any.
- void changeLaneDown ()

  Moves the car to the lower lane if any.

4.14.1 Detailed Description

Vehicle class, the parent for all vehicles.

This class is inherited by Bike, Car, Racecar, Truck and Truck2. It provides all the functionality relating to the movement of the cars and their speed.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Vehicle()

```
Vehicle::Vehicle (
             SDL_Renderer * rend,
             SDL_Texture * ast,
             SDL_Rect mov,
             SDL_Rect src,
             int s )
```

Construct a new Vehicle:: Vehicle object.

Parameters

rend	
ast	
mov	
src	
s	

4.14.3 Member Function Documentation

4.14.3.1 changeLaneDown()

```
void Vehicle::changeLaneDown ( )
```

Moves the car to the lower lane if any.

The function carries out the process of making the car move to the lower lane, it is not being called as it contains bugs at the moment.

4.14.3.2 changeLaneUp()

```
void Vehicle::changeLaneUp ( )
```

Moves the car to the upper lane if any.

The function carries out the process of making the car move to the upper lane, it is not being called as it contains bugs at the moment.

4.14.3.3 draw()

```
void Vehicle::draw (
            bool flag )
```

Draws the Vehicles.

The function calls the draw function of the Unit class, it also changes the mover rectangle based on the flag passed to it.

Parameters

flag	

4.14.3.4 getter()

```
SDL_Rect Vehicle::getter ( ) const
```

Returns mover rect of the Vehicle object.

Returns

SDL_Rect

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

- Vehicle.hpp
- Vehicle.cpp

Chapter 5

File Documentation

5.1 Bike.hpp

```
#include "Vehicle.hpp"

class Bike : public Vehicle
{

public:
    Bike(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);
    void draw(bool flag);
};
```

5.2 Car.hpp

```
#include "Vehicle.hpp"

class Car : public Vehicle
{

public:
    Car(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);
    void draw(bool flag);
};
```

5.3 Debris.hpp

```
#include "Unit.hpp"

class Debris : public Unit
{

    SDL_Rect mover;
    SDL_Rect src = {551, 9, 24, 25};

public:
    void
    animate();

    Debris(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);
};
```

5.4 Digit.hpp

```
#include "Unit.hpp"
#include <SDL.h>

class Digit : public Unit
```

```
{
private:
    SDL_Rect src, mover;

public:
    Digit(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov, SDL_Rect src);
    void draw();
    void animate(unsigned int res);
};
```

5.5 game.hpp

```
#include <SDL.h>
#include <SDL_image.h>
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Jamalo.hpp"
#include <SDL_mixer.h>

class Game
{
    // Game state
    bool start = false;
    bool end = false;

    //Screen dimension constants
    const int SCREEN_WIDTH = 1000;
    const int SCREEN_HEIGHT = 600;

    //The window we'll be rendering to
    SDL_Window *gWindow = NULL;

    //The window renderer
    SDL_Renderer *gRenderer = NULL;

    //Current displayed texture
    SDL_Texture *gTexture = NULL;
    //global reference to png image sheets
    SDL_Texture *assets = NULL;
    Mix_Music *bgMusic = NULL;

public:
    bool init();
    bool loadMedia();
    void close();
    SDL_Texture *loadTexture(std::string path);
    void run();
};
```

5.6 Jamalo.hpp

```
#pragma once
#include <SDL.h>
#include "Vehicle.hpp"
#include "Sheep.hpp"
#include "Timer.hpp"
#include "Lives.hpp"
#include <list>
using namespace std;

class Jamalo
{

    Sheep j;
    list<Vehicle> vehicles;
    SDL_Renderer *gRenderer;
    SDL_Texture *assets;
    int nLives;
    Timer t;
    Lives l1;
    Lives l2;
    Lives l3;

public:
    Jamalo(SDL_Renderer *, SDL_Texture *);
    void drawObjects();
    void createObjects();
    void move(char x);
    bool hitRegistered(Vehicle v);
    bool hitRegistered(Vehicle v, list<Vehicle> &vehicles);
    // bool hitRegistered()
    void radar(Vehicle &currentV);
    bool isOver();
}

;
```

5.7 Lives.hpp

```
#include "Unit.hpp"
#include <SDL.h>

class Lives : public Unit
{
    SDL_Rect src = {12, 124, 55, 54};
    SDL_Rect mover;

public:
    Lives(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);

    void draw();
};
```

5.8 Racecar.hpp

```
#include "Vehicle.hpp"

class RaceCar : public Vehicle
{

public:
    RaceCar(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);
    void draw(bool flag);
};
```

5.9 Sheep.hpp

```
#pragma once
#include "Unit.hpp"
using namespace std;

class Sheep : public Unit
{

private:
    SDL_Rect src = {16, 23, 43, 42};
    SDL_Rect mover = {10, 430, 50, 50};
    int nLives;

public:
    // SDL_Rect mover;
    Sheep(SDL_Renderer *rend, SDL_Texture *ast);

    SDL_Rect getter() const;
    void draw();
    void move(char x);
    void getHit();
    // friend bool Jamalo::hitRegistered(Vehicle v);
};
```

5.10 Timer.hpp

```
#include "Digit.hpp"

class Timer
{
    private:
```

```
    Digit minutes;
    Digit second1;
    Digit second2;
    int starttime;

public:
    Timer(SDL_Renderer *rend, SDL_Texture *ast);
    // int start_timer();
    void draw();

    void count();
};

```

5.11 Truck.hpp

```
#include "Vehicle.hpp"

class Truck : public Vehicle
{

public:
    Truck(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);
    void draw(bool flag);
};
```

5.12 Truck2.hpp

```
#include "Vehicle.hpp"

class Truck2 : public Vehicle
{

public:
    Truck2(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov);
    void draw(bool flag);
};
```

5.13 Unit.hpp

```
#pragma once
#include <SDL.h>

class Unit
{
    SDL_Renderer *gRenderer;
    SDL_Texture *assets;

public:
    Unit(SDL_Renderer *, SDL_Texture *);
    void draw(SDL_Rect srcRect, SDL_Rect moverRect);
};
```

5.14 Vehicle.hpp

```
#pragma once
#include "Unit.hpp"
using namespace std;

class Vehicle : public Unit
{

private:
    SDL_Rect src, mover;
    int speed;
    bool changing_lane_down;
    bool changing_lane_up;

public:
    Vehicle(SDL_Renderer *rend, SDL_Texture *ast, SDL_Rect mov, SDL_Rect src, int s);
    void draw(bool flag);
    // friend bool Jamalo::hitRegistered(Vehicle v);
    SDL_Rect getter() const;
    void changeLaneUp();
    void changeLaneDown();
};
```

Index