Micro Machine

1

Generated by Doxygen 1.9.1

Chapter 1

Source content

This folder should contain only hpp/cpp files of your implementation. You can also place hpp files in a separate directory `include`.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Namespace Documentation

6.1 BodyType Namespace Reference

Provides functions to check the type of user data.

Functions

- bool IsVehicle (UserData *userData)

  Checks if the user data represents a Vehicle.
- bool IsCollectable (UserData *userData)

  Checks if the user data represents a Collectable.
- bool IsObstacle (UserData *userData)

  Checks if the user data represents an Obstacle.
- bool IsOutsideArea (UserData *userData)

  Checks if the user data represents an OutsideArea.

6.1.1 Detailed Description

Provides functions to check the type of user data.

6.1.2 Function Documentation

6.1.2.1 IsCollectable()

```
bool BodyType::IsCollectable (
            UserData * userData )
```

Checks if the user data represents a Collectable.

Parameters

userData	Pointer to user data.

Returns

True if the user data is of type Collectable, false otherwise.

6.1.2.2 IsObstacle()

```
bool BodyType::IsObstacle (
            UserData * userData )
```

Checks if the user data represents an Obstacle.

Parameters

userData	Pointer to user data.

Returns

True if the user data is of type Obstacle, false otherwise.

6.1.2.3 IsOutsideArea()

```
bool BodyType::IsOutsideArea (
            UserData * userData )
```

Checks if the user data represents an OutsideArea.

Parameters

userData	Pointer to user data.

Returns

True if the user data is of type OutsideArea, false otherwise.

6.1.2.4 IsVehicle()

```
bool BodyType::IsVehicle (
            UserData * userData )
```

Checks if the user data represents a Vehicle.

Parameters

userData	Pointer to user data.

Returns

True if the user data is of type Vehicle, false otherwise.

Chapter 7

Class Documentation

7.1 BaseMenu Class Reference

A base class for creating menus in an SFML application.

`#include <baseMenu.hpp>`

Inheritance diagram for BaseMenu:

7.2 Buff Class Reference

Base class for Buffs, derived from Timer.

`#include <buff.hpp>`

Inheritance diagram for Buff:

Collaboration diagram for Buff:

Public Member Functions

- Buff (int duration, BuffType type, BuffEffect effectType)

  Constructor for Buff class.
- ~Buff ()

  Destructor for Buff class.
- virtual void ApplyEffect (Vehicle *vehicle)=0

  Applies the buff effect to a Vehicle.
- virtual void ReverseEffect (Vehicle *vehicle)=0

  Reverses the buff effect applied to a Vehicle.
- bool IsContinuous ()

  Checks if the buff effect is continuous.
- bool IsNegativeBuff ()

  Checks if the buff is a negative buff.

Protected Attributes

- std::string id

  Unique identifier for the buff.
- BuffType type_

  Type of the buff (Positive or Negative).
- BuffEffect effectType_

  Type of the buff effect (onetime or continuous).

7.2.1 Detailed Description

Base class for Buffs, derived from Timer.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Buff()

```
Buff::Buff (
            int duration,
            BuffType type,
            BuffEffect effectType )  [inline]
```

Constructor for Buff class.

Parameters

duration	The duration of the buff.
type	Type of the buff (Positive or Negative).
effectType	Type of the buff effect (onetime or continuous).

7.2.3 Member Function Documentation

7.2.3.1 ApplyEffect()

```
virtual void Buff::ApplyEffect (
            Vehicle * vehicle )  [pure virtual]
```

Applies the buff effect to a Vehicle.

Parameters

vehicle	The Vehicle to which the buff effect is applied.

Implemented in OxSuperSkillBuff, NegativeBuff::CrazyRotate, NegativeBuff::ReverseMushroom, HorseSuperSkillBuff, PositiveBuff::Magnetic, PositiveBuff::MaxSpeed, and PositiveBuff::OxSuperSkillBuff.

7.2.3.2 IsContinuous()

```
bool Buff::IsContinuous ( )  [inline]
```

Checks if the buff effect is continuous.

Returns

True if the buff effect is continuous, false otherwise.

7.2.3.3 IsNegativeBuff()

```
bool Buff::IsNegativeBuff ( )  [inline]
```

Checks if the buff is a negative buff.

Returns

True if the buff is negative, false otherwise.

7.2.3.4 ReverseEffect()

```
virtual void Buff::ReverseEffect (
            Vehicle * vehicle )  [pure virtual]
```

Reverses the buff effect applied to a Vehicle.

Parameters

vehicle	The Vehicle to which the buff effect is reversed.

Implemented in OxSuperSkillBuff, NegativeBuff::CrazyRotate, NegativeBuff::ReverseMushroom, HorseSuperSkillBuff, PositiveBuff::Magnetic, PositiveBuff::MaxSpeed, and PositiveBuff::OxSuperSkillBuff.

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

- src/include/buff.hpp

7.3 Cat Class Reference

Class representing a Cat, derived from Vehicle.

`#include <cat.hpp>`

Inheritance diagram for Cat:

Collaboration diagram for Cat:

Public Member Functions

- Cat (b2World *world, float x, float y, const sf::Texture &texture)
  Constructor for the Cat class.
- ~Cat ()
  Destructor for the Cat class.
- void SuperSkill () override
  Overrides the super skill function from the base class (Vehicle).

Additional Inherited Members

7.3.1 Detailed Description

Class representing a Cat, derived from Vehicle.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Cat()

```
Cat::Cat (
            b2World * world,
            float x,
            float y,
            const sf::Texture & texture )
```

Constructor for the Cat class.

Parameters

world	Pointer to the Box2D world.
x	Initial x-coordinate of the Cat.
y	Initial y-coordinate of the Cat.
texture	Reference to the SFML texture for the Cat.

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

- src/include/cat.hpp
- src/cat.cpp

7.4 CheckPoint Class Reference

Represents a checkpoint in the game world.

`#include <checkpoint.hpp>`

Inheritance diagram for CheckPoint:

Collaboration diagram for CheckPoint:

Public Member Functions

- CheckPoint (b2World ∗world, b2Vec2 position, float height, float width)

  Constructor for CheckPoint.
- virtual void OnContact (Vehicle ∗car) override

  Handle the event when a vehicle contacts the checkpoint.
- bool IsVisitedBy (Vehicle ∗car) const

  Check if a vehicle has visited this checkpoint.
- void RemoveVisitedCar (Vehicle ∗car)

  Remove a vehicle from the list of those who have visited this checkpoint.

7.4.1 Detailed Description

Represents a checkpoint in the game world.

The CheckPoint class is used to create checkpoints in the game world. Vehicle need to visit all hidden checkpoints to consider as finishing one round

7.4.2 Constructor & Destructor Documentation

7.4.2.1 CheckPoint()

```
CheckPoint::CheckPoint (
            b2World * world,
            b2Vec2 position,
            float height,
            float width )
```

Constructor for CheckPoint.

Parameters

world	The Box2D world where the checkpoint will exist.
position	The position of the checkpoint in the world.
height	The height of the checkpoint area.
width	The width of the checkpoint area.

7.4.3 Member Function Documentation

7.4.3.1 IsVisitedBy()

```
bool CheckPoint::IsVisitedBy (
            Vehicle * car ) const
```

Check if a vehicle has visited this checkpoint.

Parameters

car	A pointer to the vehicle to check.

Returns

True if the vehicle has visited, false otherwise.

7.4.3.2 OnContact()

```
void CheckPoint::OnContact (
            Vehicle * car )  [override], [virtual]
```

Handle the event when a vehicle contacts the checkpoint.

Parameters

car	A pointer to the vehicle that made contact with the checkpoint.

Reimplemented from OutsideArea.

Reimplemented in StartLine.

7.4.3.3 RemoveVisitedCar()

```
void CheckPoint::RemoveVisitedCar (
            Vehicle * car )
```

Remove a vehicle from the list of those who have visited this checkpoint.

Parameters

car	A pointer to the vehicle to be removed.

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

- src/include/checkpoint.hpp
- src/checkpoint.cpp

7.5 Collectable Class Reference

Represents a collectable item in the game.

`#include <collectable.hpp>`

Inheritance diagram for Collectable:

Collaboration diagram for Collectable:

Public Member Functions

- Collectable (b2World ∗world, b2Vec2 position, float radius, Buff ∗buff, const sf::Texture &texture)
  Constructs a Collectable object.
- std::pair< float, float > GetPosition () const
  Gets the current position of the collectable.
- void DeleteBody ()
  Deletes the Box2D body associated with the collectable.
- float GetRadius ()
  Gets the radius of the collectable.
- bool IsNullBody ()
  Checks if the collectable has a null body.
- void setDelete ()
  Marks the collectable to be deleted.
- bool getDelete ()
  Gets the delete status of the collectable.
- void OnContact (Vehicle ∗car)
  Handles the contact event with a Vehicle.
- void draw (sf::RenderTarget &target, sf::RenderStates states) const
  Draws the collectable on the target.

7.5.1 Detailed Description

Represents a collectable item in the game.

Inherits from sf::Drawable and provides functionality for handling collectables.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Collectable()

```
Collectable::Collectable (
            b2World * world,
            b2Vec2 position,
            float radius,
            Buff * buff,
            const sf::Texture & texture )
```

Constructs a Collectable object.

Parameters

world	The Box2D world in which the collectable exists.
position	The initial position of the collectable.
radius	The radius of the collectable.
buff	A pointer to the Buff associated with the collectable.
texture	The texture used for rendering the collectable.

7.5.3 Member Function Documentation

7.5.3.1 draw()

```
void Collectable::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const
```

Draws the collectable on the target.

Parameters

target	The rendering target.
states	The rendering states.

7.5.3.2 getDelete()

```
bool Collectable::getDelete ( )
```

Gets the delete status of the collectable.

Returns

True if the collectable is marked to be deleted, false otherwise.

7.5.3.3 GetPosition()

```
std::pair< float, float > Collectable::GetPosition ( ) const
```

Gets the current position of the collectable.

Returns

A pair representing the x and y coordinates of the collectable's position.

7.5.3.4 GetRadius()

```
float Collectable::GetRadius ( )
```

Gets the radius of the collectable.

Returns

The radius of the collectable.

7.5.3.5 IsNullBody()

```
bool Collectable::IsNullBody ( )
```

Checks if the collectable has a null body.

Returns

True if the collectable has a null body, false otherwise.

7.5.3.6 OnContact()

```
void Collectable::OnContact (
            Vehicle * car )
```

Handles the contact event with a Vehicle.

Parameters

car	A pointer to the Vehicle involved in the contact.

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

- src/include/collectable.hpp
- src/colletable.cpp

7.6 NegativeBuff::CrazyRotate Class Reference

A class representing a crazy rotate negative buff.

`#include <negativebuff.hpp>`

Inheritance diagram for NegativeBuff::CrazyRotate:

Collaboration diagram for NegativeBuff::CrazyRotate:

Public Member Functions

- CrazyRotate (int duration, float degree, float intensity)
  Constructor for CrazyRotate.
- void ApplyEffect (Vehicle *vehicle) override
  Applies the rotation effect on a Vehicle.
- void ReverseEffect (Vehicle *vehicle) override
  Reverses the rotation effect on a Vehicle.
- ~CrazyRotate () override
  Destructor for CrazyRotate.

Additional Inherited Members

7.6.1 Detailed Description

A class representing a crazy rotate negative buff.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 CrazyRotate()

```
CrazyRotate::CrazyRotate (
            int duration,
            float degree,
            float intensity )
```

Constructor for CrazyRotate.

Parameters

id	The identifier for the buff.
duration	The duration of the buff effect.
torque	The torque applied during rotation.
intensity	The intensity of the rotation effect.

7.6.3 Member Function Documentation

7.6.3.1 ApplyEffect()

```
void CrazyRotate::ApplyEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Applies the rotation effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is applied.

Implements Buff.

7.6.3.2 ReverseEffect()

```
void CrazyRotate::ReverseEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Reverses the rotation effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is reversed.

Implements Buff.

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

- src/include/negativebuff.hpp
- src/negativeBuff.cpp

7.7 Dog Class Reference

Class representing a Dog, derived from Vehicle.

`#include <dog.hpp>`

Inheritance diagram for Dog:

Collaboration diagram for Dog:

Public Member Functions

- Dog (b2World *world, float x=0, float y=0, const std::string &imagePath="../img/buffalo.png")
  Constructor for the Dog class.
- ~Dog ()
  Destructor for the Dog class.
- void SuperSkill () override
  Overrides the super skill function from the base class (Vehicle).

Additional Inherited Members

7.7.1 Detailed Description

Class representing a Dog, derived from Vehicle.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Dog()

```
Dog::Dog (
            b2World * world,
            float x = 0,
            float y = 0,
            const std::string & imagePath = "../img/buffalo.png" )
```

Constructor for the Dog class.

Parameters

world	Pointer to the Box2D world.
x	Initial x-coordinate of the Dog (default: 0).
y	Initial y-coordinate of the Dog (default: 0).
imagePath	Path to the image file for the Dog (default: "../img/buffalo.png").

7.7.2.2 ~Dog()

```
Dog::~Dog ( )
```

Destructor for the Dog class.

Destroys the Box2D body associated with the Dog.

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

- src/include/dog.hpp

7.8 Game Class Reference

Manages the main aspects of the game including rendering, updating, and game state management.

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

Public Member Functions

- Game ()
  Constructor.
- ~Game ()
  Destructor.
- void Initialize ()
  Initializes the game (create window, world, etc.)
- bool Run ()
  Main game loop.
- void HandleInput ()
  Handles user input.
- void Update (sf::Time deltaTime)
  Updates game state.
- void AddBoundaries ()
  Adds boundaries to the game window.
- void CreateWall (const b2Vec2 &position, const b2Vec2 &size)
  Creates an impassable wall.
- void RenderGame ()
  Renders the game.
- sf::Vector2f ClampViewCenter (const sf::Vector2f &center, const sf::Vector2f &viewSize, const sf::Vector2f &mapSize)
  Helper function to clamp camera view.
- void DrawGameWorld ()
  Draws the game world.
- void HandleMenuInput ()
  Handles menu input.
- void **HandleMenuInput2** ()
- void **HandleWinningBoard** ()
- void RenderBaseMenu (BaseMenu &menu)
  Renders the menu.

7.8.1 Detailed Description

Manages the main aspects of the game including rendering, updating, and game state management.

The Game class handles the main game loop, event processing, and switching between different states of the game. It is responsible for initializing and managing game resources, rendering game elements, and updating game logic.

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

- src/include/game.hpp
- src/game.cpp

7.9 GameMenu Class Reference

Manages the main game menu interface.

`#include <menu.hpp>`

Inheritance diagram for GameMenu:

Collaboration diagram for GameMenu:

Public Types

- enum MenuOption { **ONE_PLAYER** , **TWO_PLAYER** , **EXIT** , **NUM_ITEMS** }
  Defines menu options available in the game menu.

Public Member Functions

- GameMenu (sf::RenderWindow &window, const sf::Font &font, const sf::Texture &texture)
  Constructs a GameMenu object with a given window, font, and texture.

Additional Inherited Members

7.9.1 Detailed Description

Manages the main game menu interface.

The GameMenu class extends the BaseMenu class to provide a specific interface for the main game menu. It includes options for starting a single player or two-player game, or exiting the game.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 GameMenu()

```
GameMenu::GameMenu (
            sf::RenderWindow & window,
            const sf::Font & font,
            const sf::Texture & texture )
```

Constructs a GameMenu object with a given window, font, and texture.

Initializes the game menu with specified appearance and functionality, setting up the menu items and their properties.

Parameters

window	The SFML RenderWindow object where the menu will be drawn.
font	The SFML Font object used to render text in the menu.
texture	The SFML Texture object used for the menu's background.

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

- src/include/menu.hpp
- src/menu.cpp

7.10 GameMenu2 Class Reference

Manages the secondary game menu interface.

`#include <menu2.hpp>`

Inheritance diagram for GameMenu2:

Collaboration diagram for GameMenu2:

Public Types

- enum MenuOption { **FOREST** , **OCEAN** , **EXIT** , **NUM_ITEMS** }

  Defines menu options available in the secondary game menu.

Public Member Functions

- GameMenu2 (sf::RenderWindow &window, const sf::Font &font, const sf::Texture &texture)

  Constructs a GameMenu2 object with a given window, font, and texture.

Additional Inherited Members

7.10.1 Detailed Description

Manages the secondary game menu interface.

The GameMenu2 class extends the BaseMenu class to provide a specific interface for the secondary game menu. It includes options for choosing different game environments or exiting the game.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 GameMenu2()

```
GameMenu2::GameMenu2 (
            sf::RenderWindow & window,
            const sf::Font & font,
            const sf::Texture & texture )
```

Constructs a GameMenu2 object with a given window, font, and texture.

Initializes the secondary game menu with specified appearance and functionality, setting up the menu items and their properties specific to game environment selection.

Parameters

window	The SFML RenderWindow object where the menu will be drawn.
font	The SFML Font object used to render text in the menu.
texture	The SFML Texture object used for the menu's background.

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

- src/include/menu2.hpp
- src/menu2.cpp

7.11 Horse Class Reference

Class representing a Horse, derived from Vehicle.

`#include <horse.hpp>`

Inheritance diagram for Horse:

Collaboration diagram for Horse:

Public Member Functions

- Horse (b2World ∗world, float x, float y, const sf::Texture &texture)

  Constructor for the Horse class.
- ∼Horse ()

  Destructor for the Horse class.
- void SuperSkill () override

  Overrides the super skill function from the base class (Vehicle).

Additional Inherited Members

7.11.1 Detailed Description

Class representing a Horse, derived from Vehicle.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Horse()

```
Horse::Horse (
            b2World * world,
            float x,
            float y,
            const sf::Texture & texture )
```

Constructor for the Horse class.

Parameters

world	Pointer to the Box2D world.
x	Initial x-coordinate of the Horse.
y	Initial y-coordinate of the Horse.
texture	Reference to the SFML texture for the Horse.

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

- src/include/horse.hpp
- src/horse.cpp

7.12 HorseSuperSkillBuff Class Reference

Inheritance diagram for HorseSuperSkillBuff:

Collaboration diagram for HorseSuperSkillBuff:

Public Member Functions

- **HorseSuperSkillBuff** (int duration, float Intensity)
- void ApplyEffect (Vehicle *vehicle) override

  Applies the buff effect to a Vehicle.
- void ReverseEffect (Vehicle *vehicle) override

  Reverses the buff effect applied to a Vehicle.

Additional Inherited Members

7.12.1 Member Function Documentation

7.12.1.1 ApplyEffect()

```
void HorseSuperSkillBuff::ApplyEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Applies the buff effect to a Vehicle.

Parameters

vehicle	The Vehicle to which the buff effect is applied.

Implements Buff.

7.12.1.2 ReverseEffect()

```
void HorseSuperSkillBuff::ReverseEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Reverses the buff effect applied to a Vehicle.

Parameters

vehicle	The Vehicle to which the buff effect is reversed.

Implements Buff.

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

- src/horse.cpp

7.13 PositiveBuff::Magnetic Class Reference

A class representing a magnetic positive buff.

`#include <positivebuff.hpp>`

Inheritance diagram for PositiveBuff::Magnetic:

Collaboration diagram for PositiveBuff::Magnetic:

Public Member Functions

- Magnetic (int duration, float radius)
  Constructor for Magnetic.
- void ApplyEffect (Vehicle *vehicle)
  Applies the magnetic effect on a Vehicle.
- void ReverseEffect (Vehicle *vehicle)
  Reverses the magnetic effect on a Vehicle.
- ~Magnetic ()
  Destructor for Magnetic.

Additional Inherited Members

7.13.1 Detailed Description

A class representing a magnetic positive buff.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Magnetic()

```
Magnetic::Magnetic (
            int duration,
            float radius )
```

Constructor for Magnetic.

Parameters

duration	The duration of the buff effect.
radius	The radius of the magnetic effect.

7.13.3 Member Function Documentation

7.13.3.1 ApplyEffect()

```
void Magnetic::ApplyEffect (
            Vehicle * vehicle )  [virtual]
```

Applies the magnetic effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is applied.

Implements Buff.

7.13.3.2 ReverseEffect()

```
void Magnetic::ReverseEffect (
            Vehicle * vehicle )  [virtual]
```

Reverses the magnetic effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is reversed.

Implements Buff.

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

- src/include/positivebuff.hpp
- src/positiveBuff.cpp

7.14 Map Class Reference

Manages the game map's graphical representation.

`#include <map.hpp>`

Public Member Functions

- Map (const sf::Texture &texture)

  Constructs a Map object with a given texture.
- void Draw (sf::RenderWindow &window)

  Draws the map onto the game window.

7.14.1 Detailed Description

Manages the game map's graphical representation.

The Map class is responsible for handling the visual aspects of the game map. It loads the map texture, handles its scaling, and provides functionality for rendering the map on the game window.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Map()

```
Map::Map (
            const sf::Texture & texture )
```

Constructs a Map object with a given texture.

Loads the texture for the map and prepares it for rendering.

Parameters

texture	The SFML texture object representing the game map.

7.14.3 Member Function Documentation

7.14.3.1 Draw()

```
void Map::Draw (
            sf::RenderWindow & window )
```

Draws the map onto the game window.

Renders the map onto the provided SFML RenderWindow object. This method should be called every frame to display the map in the game window.

Parameters

window	The SFML RenderWindow object where the map will be drawn.

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

- src/include/map.hpp
- src/map.cpp

7.15 PositiveBuff::MaxSpeed Class Reference

A class representing a maximum speed positive buff.

`#include <positivebuff.hpp>`

Inheritance diagram for PositiveBuff::MaxSpeed:

Collaboration diagram for PositiveBuff::MaxSpeed:

Public Member Functions

- MaxSpeed (int duration, float boost)
  Constructor for MaxSpeed.
- void ApplyEffect (Vehicle ∗vehicle)
  Applies the maximum speed effect on a Vehicle.
- void ReverseEffect (Vehicle ∗vehicle)
  Reverses the maximum speed effect on a Vehicle.
- ∼MaxSpeed ()
  Destructor for MaxSpeed.

Additional Inherited Members

7.15.1 Detailed Description

A class representing a maximum speed positive buff.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 MaxSpeed()

```
MaxSpeed::MaxSpeed (
            int duration,
            float boost )
```

Constructor for MaxSpeed.

Parameters

duration	The duration of the buff effect.
boost	The boost applied to the maximum speed.

7.15.3 Member Function Documentation

7.15.3.1 ApplyEffect()

```
void MaxSpeed::ApplyEffect (
            Vehicle * vehicle )  [virtual]
```

Applies the maximum speed effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is applied.

Implements Buff.

7.15.3.2 ReverseEffect()

```
void MaxSpeed::ReverseEffect (
            Vehicle * vehicle )  [virtual]
```

Reverses the maximum speed effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is reversed.

Implements Buff.

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

- src/include/positivebuff.hpp
- src/positiveBuff.cpp

7.16 MyContactListener Class Reference

A custom contact listener class for handling Box2D contact events.

`#include <ContactListener.hpp>`

Inheritance diagram for MyContactListener:

Collaboration diagram for MyContactListener:

Public Member Functions

- MyContactListener ()
  Default constructor for MyContactListener.
- void BeginContact (b2Contact ∗contact) override
  Called when two fixtures start to touch.
- void EndContact (b2Contact ∗contact) override
  Called when two fixtures cease to touch.
- ∼MyContactListener ()
  Destructor for MyContactListener.

7.16.1 Detailed Description

A custom contact listener class for handling Box2D contact events.

7.16.2 Member Function Documentation

7.16.2.1 BeginContact()

```
void MyContactListener::BeginContact (
            b2Contact * contact )  [override]
```

Called when two fixtures start to touch.

Parameters

contact	The contact object containing information about the contact.

7.16.2.2 EndContact()

```
void MyContactListener::EndContact (
            b2Contact * contact )  [override]
```

Called when two fixtures cease to touch.

Parameters

contact	The contact object containing information about the contact.

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

- src/include/ContactListener.hpp
- src/ContactListener.cpp

7.17 Obstacle Class Reference

Class representing an Obstacle that can interact with vehicles.

`#include <obstacle.hpp>`

Inheritance diagram for Obstacle:

Collaboration diagram for Obstacle:

Public Member Functions

- Obstacle (b2World *world, b2Vec2 position, float radius, const sf::Texture &texture)
  Constructor for the Obstacle class.
- std::pair< float, float > GetPosition () const
  Get the position of the obstacle.
- void DeleteBody ()
  Delete the Box2D body associated with the obstacle.
- float GetRadius ()
  Get the radius of the obstacle.
- bool IsNullBody ()
  Check if the Box2D body associated with the obstacle is null.
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override
  Implementation of the draw function for rendering the obstacle.
- void OnContact (Vehicle *car)
  Function called when the obstacle comes into contact with a vehicle.

7.17.1 Detailed Description

Class representing an Obstacle that can interact with vehicles.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Obstacle()

```
Obstacle::Obstacle (
            b2World * world,
            b2Vec2 position,
            float radius,
            const sf::Texture & texture )
```

Constructor for the Obstacle class.

Parameters

world	Pointer to the Box2D world.
position	Initial position of the obstacle in the world.
radius	Radius of the obstacle.
texture	Reference to the SFML texture for the obstacle.

7.17.3 Member Function Documentation

7.17.3.1 draw()

```
void Obstacle::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override]
```

Implementation of the draw function for rendering the obstacle.

Parameters

target	Render target.
states	Render states.

7.17.3.2 GetPosition()

```
std::pair< float, float > Obstacle::GetPosition ( ) const
```

Get the position of the obstacle.

Returns

A pair representing the x and y coordinates of the obstacle.

7.17.3.3 GetRadius()

```
float Obstacle::GetRadius ( )
```

Get the radius of the obstacle.

Returns

The radius of the obstacle.

7.17.3.4 IsNullBody()

```
bool Obstacle::IsNullBody ( )
```

Check if the Box2D body associated with the obstacle is null.

Returns

True if the body is null, false otherwise.

7.17.3.5 OnContact()

```
void Obstacle::OnContact (
            Vehicle * car )
```

Function called when the obstacle comes into contact with a vehicle.

Parameters

car	Pointer to the vehicle in contact with the obstacle.

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

- src/include/obstacle.hpp
- src/obstacle.cpp

7.18 OutsideArea Class Reference

Represents areas outside the main playable region in the game.

`#include <outsideArea.hpp>`

Inheritance diagram for OutsideArea:

Public Member Functions

- OutsideArea (b2World *world, b2Vec2 position, float height, float width)
  Constructs an OutsideArea object.
- std::pair< float, float > GetPosition () const
  Gets the position of the outside area.
- void DeleteBody ()
  Deletes the physical body associated with the area.
- std::pair< float, float > GetDimension ()
  Gets the dimensions of the outside area.
- bool IsNullBody ()
  Checks if the physical body of the area is null.
- virtual void OnContact (Vehicle *car)
  Virtual function to handle contact with a vehicle.

7.18.1 Detailed Description

Represents areas outside the main playable region in the game.

The OutsideArea class is used to create and manage areas in the game world that are not part of the central gameplay but are significant for game mechanics, such as checkpoints and start lines. It handles their physical representation and interactions.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 OutsideArea()

```
OutsideArea::OutsideArea (
            b2World * world,
            b2Vec2 position,
            float height,
            float width )
```

Constructs an OutsideArea object.

Initializes an area outside the main playable region with a position, height, and width.

Parameters

world	The Box2D world where the outside area exists.
position	The position of the area in the game world.
height	The height of the area.
width	The width of the area.

7.18.3 Member Function Documentation

7.18.3.1 GetDimension()

```
std::pair< float, float > OutsideArea::GetDimension ( )
```

Gets the dimensions of the outside area.

Returns

A pair of floats representing the height and width of the area.

7.18.3.2 GetPosition()

```
std::pair< float, float > OutsideArea::GetPosition ( ) const
```

Gets the position of the outside area.

Returns

A pair of floats representing the x and y coordinates of the area.

7.18.3.3 IsNullBody()

```
bool OutsideArea::IsNullBody ( )
```

Checks if the physical body of the area is null.

Returns

True if the body is null, false otherwise.

7.18.3.4 OnContact()

```
void OutsideArea::OnContact (
            Vehicle * car )  [virtual]
```

Virtual function to handle contact with a vehicle.

This function can be overridden in derived classes to define specific behavior when a vehicle comes into contact with the area.

Parameters

car	A pointer to the vehicle that made contact with the area.

Reimplemented in StartLine, and CheckPoint.

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

- src/include/outsideArea.hpp
- src/outsideArea.cpp

7.19 Ox Class Reference

Class representing an Ox, derived from Vehicle.

`#include <ox.hpp>`

Inheritance diagram for Ox:

Collaboration diagram for Ox:

Public Member Functions

- Ox (b2World ∗world, float x, float y, const sf::Texture &texture)
  Constructor for the Ox class.
- ∼Ox ()
  Destructor for the Ox class.
- void SuperSkill () override
  Overrides the super skill function from the base class (Vehicle).

Additional Inherited Members

7.19.1 Detailed Description

Class representing an Ox, derived from Vehicle.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Ox()

```
Ox::Ox (
            b2World * world,
            float x,
            float y,
            const sf::Texture & texture )
```

Constructor for the Ox class.

Parameters

world	Pointer to the Box2D world.
x	Initial x-coordinate of the Ox.
y	Initial y-coordinate of the Ox.
texture	Reference to the SFML texture for the Ox.

7.19.2.2 ~Ox()

```
Ox::~Ox ( )
```

Destructor for the Ox class.

Destroys the Box2D body associated with the Ox.

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

- src/include/ox.hpp
- src/ox.cpp

7.20 OxSuperSkillBuff Class Reference

Inheritance diagram for OxSuperSkillBuff:

Collaboration diagram for OxSuperSkillBuff:

Public Member Functions

- **OxSuperSkillBuff** (int duration, float Intensity)
- void ApplyEffect (Vehicle *vehicle) override

  Applies the buff effect to a Vehicle.
- void ReverseEffect (Vehicle *vehicle) override

  Reverses the buff effect applied to a Vehicle.

Additional Inherited Members

7.20.1 Member Function Documentation

7.20.1.1 ApplyEffect()

```
void OxSuperSkillBuff::ApplyEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Applies the buff effect to a Vehicle.

Parameters

vehicle	The Vehicle to which the buff effect is applied.

Implements Buff.

7.20.1.2 ReverseEffect()

```
void OxSuperSkillBuff::ReverseEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Reverses the buff effect applied to a Vehicle.

Parameters

vehicle	The Vehicle to which the buff effect is reversed.

Implements Buff.

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

- src/ox.cpp

7.21 PositiveBuff::OxSuperSkillBuff Class Reference

A class representing an Ox super skill positive buff.

`#include <positivebuff.hpp>`

Inheritance diagram for PositiveBuff::OxSuperSkillBuff:

Collaboration diagram for PositiveBuff::OxSuperSkillBuff:

Public Member Functions

- OxSuperSkillBuff (int duration, float intensity)
  Constructor for OxSuperSkillBuff.
- void ApplyEffect (Vehicle *vehicle)
  Applies the super skill effect on a Vehicle.
- void ReverseEffect (Vehicle *vehicle)
  Reverses the super skill effect on a Vehicle.
- ~OxSuperSkillBuff ()
  Destructor for OxSuperSkillBuff.

Additional Inherited Members

7.21.1 Detailed Description

A class representing an Ox super skill positive buff.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 OxSuperSkillBuff()

```
OxSuperSkillBuff::OxSuperSkillBuff (
            int duration,
            float intensity )
```

Constructor for OxSuperSkillBuff.

Parameters

duration	The duration of the buff effect.
intensity	The intensity of the super skill effect.

7.21.3 Member Function Documentation

7.21.3.1 ApplyEffect()

```
void OxSuperSkillBuff::ApplyEffect (
            Vehicle * vehicle )  [virtual]
```

Applies the super skill effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is applied.

Implements Buff.

7.21.3.2 ReverseEffect()

```
void OxSuperSkillBuff::ReverseEffect (
            Vehicle * vehicle )  [virtual]
```

Reverses the super skill effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is reversed.

Implements Buff.

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

- src/include/positivebuff.hpp
- src/positiveBuff.cpp

7.22 RealTime Class Reference

A class representing a real-time countdown timer.

`#include <realTime.hpp>`

Public Member Functions

- RealTime (int duration, const sf::Font &font, sf::Color color=sf::Color::White, sf::Vector2f position=sf::↩ Vector2f(10, 10))

  Constructor for RealTime.
- void SetUp ()

  Sets up the initial state of the countdown timer.
- bool IsTimeUp ()

  Checks if the time has run out.
- void Update (int player1Rounds, int player2Rounds)

  Updates the countdown timer.
- void SetGameMode (int mode)

  Sets the game mode (1 for one player, 2 for two players).
- void SetPlayerRounds (int player1Rounds, int player2Rounds)

  Sets the number of rounds for both players.
- void Draw (sf::RenderWindow &window)

  Draws the countdown timer on the provided SFML window.

7.22.1 Detailed Description

A class representing a real-time countdown timer.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 RealTime()

```
RealTime::RealTime (
            int duration,
            const sf::Font & font,
            sf::Color color = sf::Color::White,
            sf::Vector2f position = sf::Vector2f(10, 10) )
```

Constructor for RealTime.

Parameters

duration	The duration of the countdown timer.
font	The font used for rendering the time text.
color	The color of the time text (default is sf::Color::White).
position	The position of the time text (default is sf::Vector2f(10, 10)).

7.22.3 Member Function Documentation

7.22.3.1 Draw()

```
void RealTime::Draw (
            sf::RenderWindow & window )
```

Draws the countdown timer on the provided SFML window.

Parameters

window	The SFML window on which to draw the countdown timer.

7.22.3.2 IsTimeUp()

```
bool RealTime::IsTimeUp ( )
```

Checks if the time has run out.

Returns

True if the time is up, false otherwise.

7.22.3.3 SetGameMode()

```
void RealTime::SetGameMode (
            int mode )
```

Sets the game mode (1 for one player, 2 for two players).

Parameters

mode	The game mode to set.

7.22.3.4 SetPlayerRounds()

```
void RealTime::SetPlayerRounds (
            int player1Rounds,
            int player2Rounds )
```

Sets the number of rounds for both players.

Parameters

player1Rounds	Number of rounds for player 1.
player2Rounds	Number of rounds for player 2.

7.22.3.5 Update()

```
void RealTime::Update (
            int player1Rounds,
            int player2Rounds )
```

Updates the countdown timer.

Parameters

player1Rounds	Number of rounds for player 1.
player2Rounds	Number of rounds for player 2.

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

- src/include/realTime.hpp
- src/realTime.cpp

7.23 ResourceManager Class Reference

Manages game resources including images, fonts, and sounds.

```
#include <resourceManager.hpp>
```

Public Member Functions

- ResourceManager ()

  Constructs a ResourceManager object.
- ~ResourceManager ()

Destroys the ResourceManager object.

- void LoadImage (const std::string &key, const std::string &filename)

  Loads an image from a file and stores it with a given key.
- const sf::Texture & GetImage (const std::string &key) const

  Retrieves an image resource by its key.
- void LoadFont (const std::string &key, const std::string &filename)

  Loads a font from a file and stores it with a given key.
- const sf::Font & GetFont (const std::string &key) const

  Retrieves a font resource by its key.
- void LoadSoundBackground (const std::string &key, const std::string &filename)

  Loads a background sound from a file and stores it with a given key.
- const sf::SoundBuffer & GetSoundBackground (const std::string &key) const

  Retrieves a background sound resource by its key.
- void LoadSoundStep (const std::string &key, const std::string &filename)

  Loads a step sound from a file and stores it with a given key.
- const sf::SoundBuffer & GetSoundStep (const std::string &key) const

  Retrieves a step sound resource by its key.
- void LoadFromJson (const std::string &jsonFilePath)

  Loads resources defined in a JSON configuration file.

7.23.1 Detailed Description

Manages game resources including images, fonts, and sounds.

The ResourceManager class is responsible for loading, storing, and providing access to various game resources. It uses an internal mapping to link resource keys to their corresponding loaded resources.

7.23.2 Member Function Documentation

7.23.2.1 GetFont()

```
const sf::Font & ResourceManager::GetFont (
            const std::string & key ) const
```

Retrieves a font resource by its key.

Parameters

key	The string key of the font resource.

Returns

A reference to the sf::Font object.

7.23.2.2 GetImage()

```
const sf::Texture & ResourceManager::GetImage (
            const std::string & key ) const
```

Retrieves an image resource by its key.

Parameters

key	The string key of the image resource.

Returns

A reference to the sf::Texture object.

7.23.2.3 GetSoundBackground()

```
const sf::SoundBuffer & ResourceManager::GetSoundBackground (
            const std::string & key ) const
```

Retrieves a background sound resource by its key.

Parameters

key	The string key of the sound resource.

Returns

A reference to the sf::SoundBuffer object.

7.23.2.4 GetSoundStep()

```
const sf::SoundBuffer & ResourceManager::GetSoundStep (
            const std::string & key ) const
```

Retrieves a step sound resource by its key.

Parameters

key	The string key of the step sound resource.

Returns

A reference to the sf::SoundBuffer object.

7.23.2.5 LoadFont()

```
void ResourceManager::LoadFont (
            const std::string & key,
            const std::string & filename )
```

Loads a font from a file and stores it with a given key.

Parameters

key	A string to identify the font resource.
filename	The file path of the font to load.

7.23.2.6 LoadFromJson()

```
void ResourceManager::LoadFromJson (
            const std::string & jsonFilePath )
```

Loads resources defined in a JSON configuration file.

This method loads various resources specified in a JSON file, such as textures and sounds.

Parameters

jsonFilePath	Path to the JSON file containing resource definitions.

7.23.2.7 LoadImage()

```
void ResourceManager::LoadImage (
            const std::string & key,
            const std::string & filename )
```

Loads an image from a file and stores it with a given key.

Parameters

key	A string to identify the image resource.
filename	The file path of the image to load.

7.23.2.8 LoadSoundBackground()

```
void ResourceManager::LoadSoundBackground (
            const std::string & key,
            const std::string & filename )
```

Loads a background sound from a file and stores it with a given key.

Parameters

key	A string to identify the sound resource.
filename	The file path of the sound to load.

7.23.2.9 LoadSoundStep()

```
void ResourceManager::LoadSoundStep (
            const std::string & key,
            const std::string & filename )
```

Loads a step sound from a file and stores it with a given key.

Parameters

key	A string to identify the step sound resource.
filename	The file path of the sound to load.

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

- src/include/resourceManager.hpp
- src/resourceManager.cpp

7.24 NegativeBuff::ReverseMushroom Class Reference

A class representing a reverse mushroom negative buff.

`#include <negativebuff.hpp>`

Inheritance diagram for NegativeBuff::ReverseMushroom:

Collaboration diagram for NegativeBuff::ReverseMushroom:

Public Member Functions

- ReverseMushroom (int duration, float intensity)
  Constructor for ReverseMushroom.
- void ApplyEffect (Vehicle *vehicle) override
  Applies the reverse effect on a Vehicle.
- void ReverseEffect (Vehicle *vehicle) override
  Reverses the effect on a Vehicle.
- ~ReverseMushroom () override
  Destructor for ReverseMushroom.

Additional Inherited Members

7.24.1 Detailed Description

A class representing a reverse mushroom negative buff.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 ReverseMushroom()

```
ReverseMushroom::ReverseMushroom (
            int duration,
            float intensity )
```

Constructor for ReverseMushroom.

Parameters

duration	The duration of the buff effect.
intensity	The intensity of the reverse effect.

7.24.3 Member Function Documentation

7.24.3.1 ApplyEffect()

```
void ReverseMushroom::ApplyEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Applies the reverse effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is applied.

Implements Buff.

7.24.3.2 ReverseEffect()

```
void ReverseMushroom::ReverseEffect (
            Vehicle * vehicle )  [override], [virtual]
```

Reverses the effect on a Vehicle.

Parameters

vehicle	The Vehicle on which the effect is reversed.

Implements Buff.

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

- src/include/negativebuff.hpp
- src/negativeBuff.cpp

7.25 StartLine Class Reference

Represents the starting line and manages checkpoints in the game.

`#include <checkpoint.hpp>`

Inheritance diagram for StartLine:

Collaboration diagram for StartLine:

Public Member Functions

- StartLine (b2World *world, b2Vec2 position, float height, float width)
  Constructor for StartLine.
- virtual void OnContact (Vehicle *car) override
  Handle the event when a vehicle contacts the start line.
- void AddCheckPoint (CheckPoint *checkpoint)
  Add a checkpoint to be managed by this start line.
- ~StartLine ()
  Destructor for StartLine.
- std::map< Vehicle *, int > GetPoints ()
  Get the points of each vehicle based on checkpoints visited.

7.25.1 Detailed Description

Represents the starting line and manages checkpoints in the game.

The StartLine class extends CheckPoint and adds functionality to manage multiple checkpoints in the game, tracking the progress of vehicles through these checkpoints.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 StartLine()

```
StartLine::StartLine (
            b2World * world,
            b2Vec2 position,
            float height,
            float width )
```

Constructor for StartLine.

Parameters

world	The Box2D world where the start line will exist.
position	The position of the start line in the world.
height	The height of the start line area.
width	The width of the start line area.

7.25.3 Member Function Documentation

7.25.3.1 AddCheckPoint()

```
void StartLine::AddCheckPoint (
            CheckPoint * checkpoint )
```

Add a checkpoint to be managed by this start line.

Parameters

checkpoint	A pointer to the checkpoint to be added.

7.25.3.2 GetPoints()

```
std::map< Vehicle *, int > StartLine::GetPoints ( )
```

Get the points of each vehicle based on checkpoints visited.

Returns

A map of vehicles to their respective points.

7.25.3.3 OnContact()

```
void StartLine::OnContact (
            Vehicle * car ) [override], [virtual]
```

Handle the event when a vehicle contacts the start line.

Parameters

car	A pointer to the vehicle that made contact with the start line.

Reimplemented from CheckPoint.

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

- src/include/checkpoint.hpp
- src/checkpoint.cpp

7.26 Timer Class Reference

Represents a simple countdown timer.

`#include <timer.hpp>`

Inheritance diagram for Timer:

Public Member Functions

- Timer (int timeLeft)

  Constructor for Timer.
- virtual ~Timer ()

  Destructor for Timer.
- bool Tick ()

  Decreases the time remaining on the timer by one tick.

7.26.1 Detailed Description

Represents a simple countdown timer.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Timer()

```
Timer::Timer (
            int timeLeft )
```

Constructor for Timer.

Parameters

timeLeft	The initial time left on the timer.

7.26.3 Member Function Documentation

7.26.3.1 Tick()

```
bool Timer::Tick ( )
```

Decreases the time remaining on the timer by one tick.

Returns

True if the object should be removed, false otherwise.

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

- src/include/timer.hpp
- src/timer.cpp

7.27 UserData Union Reference

Represents a union of data or a combination of type and pointer for user data.

```
#include <userDataPointer.hpp>
```

Public Attributes

- uintptr_t data
  Raw data value.
- struct {
  UserType type
  Type of user data.
  void * pointer
  Pointer to user data.
  } info

  Structure containing type and pointer information.

7.27.1 Detailed Description

Represents a union of data or a combination of type and pointer for user data.

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

- src/include/userDataPointer.hpp

7.28 Vehicle Class Reference

Class representing a simple vehicle in a 2D physics world using Box2D.

`#include <vehicle.hpp>`

Inheritance diagram for Vehicle:

Collaboration diagram for Vehicle:

Public Member Functions

- Vehicle (b2World *world, float x, float y, const sf::Texture &texture)
  Constructor for Vehicle.
- ~Vehicle ()
  Destructor for Vehicle.
- void Update ()
  Update function for the vehicle.
- void UpdateLateralVelocity ()
  Update the lateral velocity of the vehicle.
- void UpdateSpeed ()
  Update the speed of the vehicle based on applied force.
- void Rotate (float angle=1)
  Rotate the vehicle by applying angular impulse.
- void UpdateCoolDown ()
  Update the cooldown for the super skill.
- std::pair< float, float > GetPosition () const
  Get the current position of the vehicle.
- void ToggleForce (bool value)
  Toggle the force applied to the vehicle on/off.
- float GetAngle ()
  Get the current angle of the vehicle.
- void ProcessItem ()
  Placeholder function for processing items (to be implemented as needed).
- virtual void draw (sf::RenderTarget &target, sf::RenderStates states) const
  Draw the vehicle.
- void CrazyRotate (float degree, float intensity)
  Apply a rotating buff to the vehicle.
- void MagneticPull (float radius)
  Apply a magnetic pull buff to the vehicle.
- void ReverseMagneticPull ()
  Reverse the magnetic pull buff.
- void ApplyBuff (float forceMu=1.0f, float MaxSpeedMul=1.0f, float SizeMul=1.0f, float TorqueMul=1.0f)
  Multiply various attributes of the vehicle by the specified factors.
- void AddBuff (Buff *buff)
  Add a buff to the vehicle.
- void UpdateBuff ()
  Update active buffs on the vehicle.
- virtual void SuperSkill ()
  Virtual function representing the super skill of the vehicle.
- bool GetForce ()
  Get the state of the force applied to the vehicle.

Protected Attributes

- bool forceOn

  Flag indicating whether a force is applied to the vehicle.
- b2Body ∗ m_body

  Box2D body representing the vehicle.
- float maxSpeed

  Maximum speed of the vehicle.
- sf::Sprite **sprite_**
- sf::Texture **texture_**
- int **superSkillCoolDown** = 0
- b2PolygonShape **dynamicBox**
- b2Body ∗ **m_frontTire**
- b2Body ∗ **m_rearTire**
- float **forceBuff** = 1.0f
- float **MaxSpeedBuff** = 1.0f
- float **SizeBuff** = 1.0f
- float **TorqueBuff** = 1.0f
- std::vector< Buff ∗ > **buffs**
- sf::SoundBuffer **runBuffer**
- sf::Sound **run**

7.28.1 Detailed Description

Class representing a simple vehicle in a 2D physics world using Box2D.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Vehicle()

```
Vehicle::Vehicle (
            b2World * world,
            float x,
            float y,
            const sf::Texture & texture )
```

Constructor for Vehicle.

Parameters

world	Pointer to the Box2D world.
x	Initial x-coordinate of the vehicle.
y	Initial y-coordinate of the vehicle.
texture	Reference to the SFML texture for the vehicle.

7.28.2.2 ~Vehicle()

```
Vehicle::~Vehicle ( )
```

Destructor for Vehicle.

Destroys the Box2D body associated with the vehicle.

7.28.3 Member Function Documentation

7.28.3.1 AddBuff()

```
void Vehicle::AddBuff (
            Buff * buff )
```

Add a buff to the vehicle.

Parameters

buff	Pointer to the buff to be added.

7.28.3.2 ApplyBuff()

```
void Vehicle::ApplyBuff (
            float forceMu = 1.0f,
            float MaxSpeedMul = 1.0f,
            float SizeMul = 1.0f,
            float TorqueMul = 1.0f )
```

Multiply various attributes of the vehicle by the specified factors.

Parameters

forceMu	Force multiplication factor.
MaxSpeedMul	Maximum speed multiplication factor.
SizeMul	Size multiplication factor.
TorqueMul	Torque multiplication factor.

7.28.3.3 CrazyRotate()

```
void Vehicle::CrazyRotate (
```

```
            float degree,
            float intensity )
```

Apply a rotating buff to the vehicle.

Parameters

degree	The rotation degree.
intensity	The intensity of the rotation.

7.28.3.4 draw()

```
void Vehicle::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [virtual]
```

Draw the vehicle.

Parameters

target	Render target.
states	Render states.

7.28.3.5 GetAngle()

```
float Vehicle::GetAngle ( )
```

Get the current angle of the vehicle.

Returns

The current angle of the vehicle.

7.28.3.6 GetForce()

```
bool Vehicle::GetForce ( )
```

Get the state of the force applied to the vehicle.

Returns

True if force is applied, false otherwise.

7.28.3.7 GetPosition()

```
std::pair< float, float > Vehicle::GetPosition ( ) const
```

Get the current position of the vehicle.

Returns

A pair representing the x and y coordinates of the vehicle.

7.28.3.8 MagneticPull()

```
void Vehicle::MagneticPull (
            float radius )
```

Apply a magnetic pull buff to the vehicle.

Parameters

radius	The radius of the magnetic pull.

7.28.3.9 Rotate()

```
void Vehicle::Rotate (
            float angle = 1 )
```

Rotate the vehicle by applying angular impulse.

Parameters

angle	The angular impulse (default is 1).

7.28.3.10 ToggleForce()

```
void Vehicle::ToggleForce (
            bool value )
```

Toggle the force applied to the vehicle on/off.

Parameters

value	True to enable the force, false to disable.

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

- src/include/vehicle.hpp
- src/vehicle.cpp

7.29 WinnerBoard Class Reference

A class to manage and display the winning screen of the game.

`#include <winnerBoard.hpp>`

Inheritance diagram for WinnerBoard:

Collaboration diagram for WinnerBoard:

Public Types

- enum MenuOption { **REPLAY** , **EXIT** , **NUM_ITEMS** }

  Menu options for the WinnerBoard.

Public Member Functions

- WinnerBoard (sf::RenderWindow &window, const sf::Font &font, const sf::Texture &texture)

  Constructs a WinnerBoard object.
- void SetWinner (int playerNumber)

  Sets the winner's information.
- void draw () override

  Draws the winner board on the screen.

Additional Inherited Members

7.29.1 Detailed Description

A class to manage and display the winning screen of the game.

WinnerBoard is a subclass of BaseMenu. It manages the display of the winner's information and provides menu options for replaying the game or exiting.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 WinnerBoard()

```
WinnerBoard::WinnerBoard (
            sf::RenderWindow & window,
            const sf::Font & font,
            const sf::Texture & texture )
```

Constructs a WinnerBoard object.

Parameters

window	Reference to the SFML RenderWindow.
font	The font used for displaying text.
texture	The background texture for the winner board.

7.29.3 Member Function Documentation

7.29.3.1 draw()

```
void WinnerBoard::draw ( )  [override], [virtual]
```

Draws the winner board on the screen.

Overrides the draw method from BaseMenu to include the winner's information and options.

Reimplemented from BaseMenu.

7.29.3.2 SetWinner()

```
void WinnerBoard::SetWinner (
            int playerNumber )
```

Sets the winner's information.

Parameters

playerNumber	The player number of the winner.

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

- src/include/winnerBoard.hpp
- src/winnerBoard.cpp

7.30 World Class Reference

Manages the physics world and game entities like vehicles, collectables, and obstacles.

```
#include <world.hpp>
```

Public Member Functions

- World (b2Vec2 gravity)

  Constructs a World object with a given gravity.
- ~World ()

  Destructor for World class.
- void Update (float timeStep, int velocityIterations, int positionIterations)

  Updates the physics world over a given timestep.
- Vehicle * GetWinner ()

  Retrieves the winning vehicle.
- void AddVehicle (Vehicle *vehicle)

  Adds a vehicle to the world.
- void AddCollectable (Collectable *collectable)

  Adds a collectable to the world.
- void SetRacingTrack (StartLine *startLine)

  Sets the racing track for the world.
- void AddObstacle (Obstacle *obstacle)

  Adds an obstacle to the world.
- std::map< Vehicle *, int > GetPoints ()

  Retrieves the points scored by each vehicle.
- int GetPoint (Vehicle *car)

  Gets the points scored by a specific vehicle.
- bool HaveAnyOneWin ()

  Checks if any vehicle has won the game.
- b2World * GetPhysicWorld () const

  Gets the physics world.
- std::vector< Vehicle * > & GetVehicle ()

  Gets the list of vehicles in the world.
- std::vector< Collectable * > & GetCollectable ()

  Gets the list of collectables in the world.
- std::vector< Obstacle * > & GetObstacle ()

  Gets the list of obstacles in the world.

7.30.1 Detailed Description

Manages the physics world and game entities like vehicles, collectables, and obstacles.

The World class is responsible for updating the physics world, keeping track of game entities, and determining the winner of the game.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 World()

```
World::World (
            b2Vec2 gravity )
```

Constructs a World object with a given gravity.

Parameters

gravity	The gravitational force applied in the world.

7.30.3 Member Function Documentation

7.30.3.1 AddCollectable()

```
void World::AddCollectable (
            Collectable * collectable )
```

Adds a collectable to the world.

Parameters

collectable	Pointer to the collectable to be added.

7.30.3.2 AddObstacle()

```
void World::AddObstacle (
            Obstacle * obstacle )
```

Adds an obstacle to the world.

Parameters

obstacle	Pointer to the obstacle to be added.

7.30.3.3 AddVehicle()

```
void World::AddVehicle (
            Vehicle * vehicle )
```

Adds a vehicle to the world.

Parameters

vehicle	Pointer to the vehicle to be added.

7.30.3.4 GetCollectable()

```
std::vector< Collectable * > & World::GetCollectable ( )
```

Gets the list of collectables in the world.

Returns

Reference to the vector of collectables.

7.30.3.5 GetObstacle()

```
std::vector< Obstacle * > & World::GetObstacle ( )
```

Gets the list of obstacles in the world.

Returns

Reference to the vector of obstacles.

7.30.3.6 GetPhysicWorld()

```
b2World * World::GetPhysicWorld ( ) const
```

Gets the physics world.

Returns

Pointer to the Box2D physics world.

7.30.3.7 GetPoint()

```
int World::GetPoint (
            Vehicle * car )
```

Gets the points scored by a specific vehicle.

Parameters

car	Pointer to the vehicle.

Returns

The points scored by the vehicle.

7.30.3.8 GetPoints()

```
std::map< Vehicle *, int > World::GetPoints ( )
```

Retrieves the points scored by each vehicle.

Returns

Map of vehicles and their corresponding points.

7.30.3.9 GetVehicle()

```
std::vector< Vehicle * > & World::GetVehicle ( )
```

Gets the list of vehicles in the world.

Returns

Reference to the vector of vehicles.

7.30.3.10 GetWinner()

```
Vehicle * World::GetWinner ( )
```

Retrieves the winning vehicle.

Returns

Pointer to the winning vehicle.

7.30.3.11 HaveAnyOneWin()

```
bool World::HaveAnyOneWin ( )
```

Checks if any vehicle has won the game.

Returns

True if there is a winner, false otherwise.

7.30.3.12 SetRacingTrack()

```
void World::SetRacingTrack (
            StartLine * startLine )
```

Sets the racing track for the world.

Parameters

startLine	Pointer to the start line of the racing track.

7.30.3.13 Update()

```
void World::Update (
            float timeStep,
            int velocityIterations,
            int positionIterations )
```

Updates the physics world over a given timestep.

Parameters

timeStep	The timestep over which the world is updated.
velocityIterations	The number of velocity iterations for the physics calculations.
positionIterations	The number of position iterations for the physics calculations.

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

- src/include/world.hpp
- src/world.cpp

Chapter 8

File Documentation

8.1 src/include/menu.hpp File Reference

GameMenu class header.

```
#include <SFML/Graphics.hpp>
#include "baseMenu.hpp"
```

Include dependency graph for menu.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class GameMenu

  Manages the main game menu interface.

8.1.1 Detailed Description

GameMenu class header.

This file contains the definition of the GameMenu class, which is used for handling the main game menu interface. It extends from the BaseMenu class, providing specific options for the main game menu.

8.2 src/include/menu2.hpp File Reference

GameMenu2 class header.

```
#include <SFML/Graphics.hpp>
#include "baseMenu.hpp"
```

Include dependency graph for menu2.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class GameMenu2

  Manages the secondary game menu interface.

8.2.1 Detailed Description

GameMenu2 class header.

This file contains the definition of the GameMenu2 class, which is a specialized version of the BaseMenu for a secondary game menu. This menu offers different environment choices for the game, like forest and ocean themes.