The Adventure of Aladdin

Generated by Doxygen 1.8.20

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

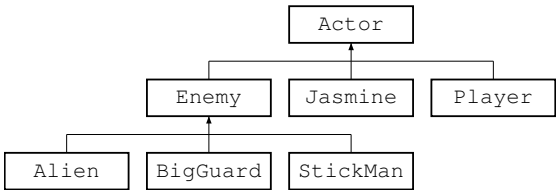
Class Documentation

3.1 Actor Class Reference

Parent class from which all Enemies, Player and Jasmine inherit from.

`#include <Actor.hpp>`

Inheritance diagram for Actor:



Public Member Functions

- **Actor** (SDL_Texture ∗)
- void **draw** (SDL_Renderer ∗)
- SDL_Rect **getMover** ()
- int **getHealth** ()
- void **setHealth** ()

Protected Attributes

- SDL_Rect **mover**
- SDL_Renderer ∗ **gRenderer**
- SDL_Renderer ∗ **assests**
- int **health**

3.1.1 Detailed Description

Parent class from which all Enemies, Player and Jasmine inherit from.

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

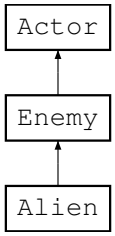
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Actor.hpp

3.2 Alien Class Reference

This class inherits from the Enemy class. It is one of the three antagonists of the game. MoveCount is the integer animation count tied to the movemnet of the Alien. AttackCount is the integer animation count tied to the attack sequence of the Alien. hurtCount is the integer animation count tied to the sequence when the Alien is defeated.

`#include <alien.hpp>`

Inheritance diagram for Alien:



Public Member Functions

- Alien (SDL_Texture ∗asst)
- void draw (SDL_Renderer ∗gRenderer)
- void emove (bool move, bool flag2)
- void move (int mx, bool flag, bool flag2, SDL_RendererFlip flip)
- void attack ()
- void hurt ()
- char getType ()
- void setHealthnMover (int helth, int xmov)
- void createOutFile (std::ostream &myfile)

Additional Inherited Members

3.2.1 Detailed Description

This class inherits from the Enemy class. It is one of the three antagonists of the game. MoveCount is the integer animation count tied to the movemnet of the Alien. AttackCount is the integer animation count tied to the attack sequence of the Alien. hurtCount is the integer animation count tied to the sequence when the Alien is defeated.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Alien()

```
Alien::Alien (
            SDL_Texture * asst )
```

This is the constructor for the Alien class.

Parameters

*SDL_Texture**	which is basically the asset file passed from the game.cpp.

3.2.3 Member Function Documentation

3.2.3.1 attack()

```
void Alien::attack ( )
```

This function cycles through the attack animation of the Alien. Cycling is done through AttackCount.

3.2.3.2 createOutFile()

```
void Alien::createOutFile (
            std::ostream & myfile )  [virtual]
```

This functions writes the type, health value and x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented from Enemy.

3.2.3.3 draw()

```
void Alien::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This draw function renders the Alien object on the screen.

Parameters

*SDL_Renderer**	

Reimplemented from Enemy.

3.2.3.4 emove()

```
void Alien::emove (
            bool move,
            bool flag2 )
```

This function cycles through the move animation of the Alien.

Parameters

move	this is a boolean flag which toggles the x direction movement of the object on the screen. \parm flag2 is a boolean flag which toggles whether or not the animation must play or not.

3.2.3.5 getType()

```
char Alien::getType ( )  [virtual]
```

This function returns the type of Enemy that it is. In this case a Alien

Reimplemented from Enemy.

3.2.3.6 hurt()

```
void Alien::hurt ( )  [virtual]
```

This function cycles through the death animation of the Alien. Cycling is done through hurtCount.

Reimplemented from Enemy.

3.2.3.7 move()

```
void Alien::move (
            int mx,
            bool flag,
            bool flag2,
            SDL_RendererFlip flip )  [virtual]
```

This function determines the static or movable nature of the Alien object and thus appropiately plays the respective move animation called through emove.

Parameters

mx	this is the integer x value of the Player's mover so as to determine the distance between the Alien and Player and thus force Alien movement towards the left.
flag	boolean value which is set true when this enemy is created.
flag2	variable that toggles true when left arrow key is down and false when it is up.
flip	denotes the orientation of the Player.

Reimplemented from Enemy.

3.2.3.8 setHealthnMover()

```
void Alien::setHealthnMover (
            int helth,
            int xmov )  [virtual]
```

This function sets the Alien's health and position when loading the game from the save file. \parm helth is the passed health integer value read from the save file.

Parameters

xmov	is the passed integer x-coordinate read from the save file.

Reimplemented from Enemy.

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

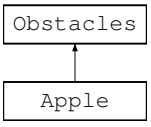
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/alien.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/alien.cpp

3.3 Apple Class Reference

This class inherits from the Obstalces class. It is one of the two static objects which interact with the Player. The specific function of this object is related to the health of the Player.

```
#include <apple.hpp>
```

Inheritance diagram for Apple:



Public Member Functions

- Apple (SDL_Texture ∗asst)
- void draw (SDL_Renderer ∗gRenderer)
- void move (SDL_RendererFlip flip, bool flag3)
- char name ()
- void setMove (int x)
- void createOutFile (std::ostream &myfile)

Additional Inherited Members

3.3.1 Detailed Description

This class inherits from the Obstalces class. It is one of the two static objects which interact with the Player. The specific function of this object is related to the health of the Player.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Apple()

```
Apple::Apple (
            SDL_Texture * asst )
```

This is the constructor for the Apple class.

Parameters

SDL_Texture∗	which is basically the asset file passed from the game.cpp.

3.3.3 Member Function Documentation

3.3.3.1 createOutFile()

```
void Apple::createOutFile (
            std::ostream & myfile )  [virtual]
```

This functions writes the type, health integer value and integer x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented from Obstacles.

3.3.3.2 draw()

```
void Apple::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This draw function renders the Apple object on the screen.

Parameters

*SDL_Renderer**	

Reimplemented from Obstacles.

3.3.3.3 move()

```
void Apple::move (
            SDL_RendererFlip flip,
            bool flag3 )  [virtual]
```

This function moves the Apple object onto the screen when appropiate. When the Player is moving to the right the Apple is reducing the x value. When player is moving to the left, we are increasing the x value.

Parameters

flip	provides the class with the orientation of the player which is of type SDL_RendererFlip.
flag3	is a boolean that is toggled true when Player is pressing the key down.

Reimplemented from Obstacles.

3.3.3.4 name()

```
char Apple::name ( )  [virtual]
```

This function provides us with the specific type of object. In this case an Apple.

Reimplemented from Obstacles.

3.3.3.5 setMove()

```
void Apple::setMove (
            int x )  [virtual]
```

This function sets the integer x cooridinate of the Apple when loading from save file.

Reimplemented from Obstacles.

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

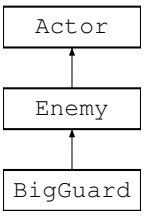
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/apple.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/apple.cpp

3.4 BigGuard Class Reference

This class inherits from the Enemy class. It is one of the three antagonists of the game. MoveCount is the integer animation count tied to the movemnet of the BigGuard. AttackCount is the integer animation count tied to the attack sequence of the BigGuard. hurtCount is the integer animation count tied to the sequence when the BigGuard is defeated.

```
#include <bigguard.hpp>
```

Inheritance diagram for BigGuard:



Public Member Functions

- BigGuard (SDL_Texture ∗asst)
- void draw (SDL_Renderer ∗gRenderer)
- void emove (bool move, bool flag2)
- void move (int mx, bool flag, bool flag2, SDL_RendererFlip flip)
- void attack (SDL_Texture ∗gTexture, SDL_Rect bgSRC)
- void hurt ()
- char getType ()
- void setHealthnMover (int helth, int xmov)
- void createOutFile (std::ostream &myfile)

Additional Inherited Members

3.4.1 Detailed Description

This class inherits from the Enemy class. It is one of the three antagonists of the game. MoveCount is the integer animation count tied to the movemnet of the BigGuard. AttackCount is the integer animation count tied to the attack sequence of the BigGuard. hurtCount is the integer animation count tied to the sequence when the BigGuard is defeated.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 BigGuard()

```
BigGuard::BigGuard (
            SDL_Texture * asst )
```

This is the constructor for the BigGuard class.

Parameters

*SDL_Texture**	which is basically the asset file passed from the game.cpp.

3.4.3 Member Function Documentation

3.4.3.1 attack()

```
void BigGuard::attack (
            SDL_Texture * gTexture,
            SDL_Rect bgSRC )  [virtual]
```

This function cycles through the attack animation of the BigGuard. Cycling is done through AttackCount.

Reimplemented from Enemy.

3.4.3.2 createOutFile()

```
void BigGuard::createOutFile (
            std::ostream & myfile )  [virtual]
```

This functions writes the type, health value and x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented from Enemy.

3.4.3.3 draw()

```
void BigGuard::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This draw function renders the BigGuard object on the screen.

Parameters

*SDL_Renderer**	

Reimplemented from Enemy.

3.4.3.4 emove()

```
void BigGuard::emove (
            bool move,
            bool flag2 )
```

This function cycles through the move animation of the BigGuard.

Parameters

move	this is a boolean flag which toggles the x direction movement of the object on the screen. \parm flag2 is a boolean flag which toggles whether or not the animation must play or not.

3.4.3.5 getType()

```
char BigGuard::getType ( )  [virtual]
```

This function returns the type of Enemy that it is. In this case a BigGuard

Reimplemented from Enemy.

3.4.3.6 hurt()

```
void BigGuard::hurt ( )  [virtual]
```

This function cycles through the death animation of the BigGuard. Cycling is done through hurtCount.

Parameters

bgSRC	is an SDL_Rect which provides background co-ordinates for rendering of scene during Attack.

Reimplemented from Enemy.

3.4.3.7 move()

```
void BigGuard::move (
            int mx,
            bool flag,
            bool flag2,
            SDL_RendererFlip flip )  [virtual]
```

This function determines the static or movable nature of the BigGuard object and thus appropiately plays the respective move animation called through emove.

Parameters

mx	this is the integer x value of the Player's mover so as to determine the distance between the BigGuard and Player and thus force BigGuard movement towards the left.
flag	boolean value which is set true when this enemy is created.
flag2	variable that toggles true when left arrow key is down and false when it is up.
flip	denotes the orientation of the Player and this of type SDL_RendererFlip.

Reimplemented from Enemy.

3.4.3.8 setHealthnMover()

```
void BigGuard::setHealthnMover (
            int helth,
            int xmov )  [virtual]
```

This function sets the BigGuard's health and position when loading the game from the save file. \parm helth is the passed health integer value read from the save file.

Parameters

xmov	is the passed integer x-coordinate read from the save file.

Reimplemented from Enemy.

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

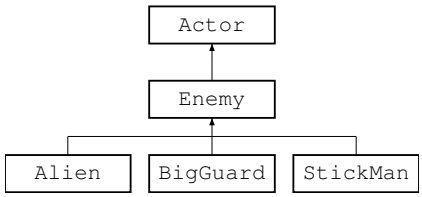
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/bigguard.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/bigguard.cpp

3.5 Enemy Class Reference

This is the Parent class to generate the enemies. This class further inherits from the Actor Class.

`#include <Enemy.hpp>`

Inheritance diagram for Enemy:



Public Member Functions

- Enemy (SDL_Texture ∗asst)
- Enemy ()
- virtual void draw (SDL_Renderer ∗gRenderer)
- virtual void move (int mx, bool flag, bool flag2, SDL_RendererFlip flip)
- virtual void attack (SDL_Texture ∗gTexture, SDL_Rect bgSRC)
- virtual void hurt ()
- virtual char getType ()
- virtual void setHealthnMover (int helth, int xmov)
- void reduceHealth (char ch)
- void checkCollison (SDL_Rect a, SDL_Rect b, SDL_RendererFlip flip, char ch)
- bool operator<= (int temp)
- virtual void createOutFile (std::ostream &myfile)

Additional Inherited Members

3.5.1 Detailed Description

This is the Parent class to generate the enemies. This class further inherits from the Actor Class.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Enemy() [1/2]

```
Enemy::Enemy (
            SDL_Texture * asst )
```

This is the constructor for the Enemy class.

Parameters

*SDL_Texture**	which is basically the asset file passed from the game.cpp.

3.5.2.2 Enemy() [2/2]

```
Enemy::Enemy ( )  [inline]
```

This is the default constructor for the Enemy class that does not take any arguments

3.5.3 Member Function Documentation

3.5.3.1 attack()

```
void Enemy::attack (
            SDL_Texture * gTexture,
            SDL_Rect bgSRC )  [virtual]
```

This virtual function cycles through the attack animation of the Enemy.

Reimplemented in StickMan, and BigGuard.

3.5.3.2 checkCollison()

```
void Enemy::checkCollison (
            SDL_Rect a,
            SDL_Rect b,
            SDL_RendererFlip flip,
            char ch )
```

This function checks for collision between the Enemy and the Player. If true reduceHealth is called.

Parameters

a	is the SDL_Rect mover for the Player.
b	is the SDL_Rect mover for the Enemy.
flip	denotes the orientation of the Player which is of type SDL_RendererFlip.
ch	is a character that denotes the attack performed by the Player.

3.5.3.3 createOutFile()

```
void Enemy::createOutFile (
            std::ostream & myfile )  [virtual]
```

This virtual function writes the type, health value and x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented in StickMan, BigGuard, and Alien.

3.5.3.4 draw()

```
void Enemy::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This virtual draw function renders the Enemy object on the screen.

Parameters

SDL_Renderer∗	

Reimplemented in StickMan, BigGuard, and Alien.

3.5.3.5 getType()

```
char Enemy::getType ( )  [virtual]
```

This virtual function returns the type of Enemy that it is.

Reimplemented in StickMan, BigGuard, and Alien.

3.5.3.6 hurt()

```
void Enemy::hurt ( )  [virtual]
```

This virtual function cycles through the death animation of the Enemy.

Reimplemented in StickMan, BigGuard, and Alien.

3.5.3.7 move()

```
void Enemy::move (
            int mx,
            bool flag,
            bool flag2,
            SDL_RendererFlip flip )  [virtual]
```

This virtual function determines the static or movable nature of the Enemy object and thus appropiately plays the respective move animation.

Parameters

mx	this is the integer x value of the Player's mover so as to determine the distance between the Enemy and Player and thus force Enemy movement towards the left.
flag	boolean value which is set true when this enemy is created.
flag2	variable that toggles true when left arrow key is down and false when it is up.
flip	denotes the orientation of the Player which is of type SDL_RendererFlip.

Reimplemented in StickMan, BigGuard, and Alien.

3.5.3.8 operator<=()

```
bool Enemy::operator<= (
            int temp )
```

This is the less than or equal to operator overloaded.

Parameters

temp	is a constant integer that the Enemy is compared with in the context of its health value.

3.5.3.9 reduceHealth()

```
void Enemy::reduceHealth (
            char ch )
```

This function reduces the health of the Enemy by a certain amount depending upon the type of attack recieved.

Parameters

ch	denotes a one character signifier as to the attack recieved.

3.5.3.10 setHealthnMover()

```
void Enemy::setHealthnMover (
            int health,
            int xmov )  [virtual]
```

This virtual function sets the Enemy's health and position when loading the game from the save file. \parm helth is the passed health integer value read from the save file.

Parameters

xmov	is the passed integer x-coordinate read from the save file.

Reimplemented in StickMan, BigGuard, and Alien.

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

- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Enemy.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Enemy.cpp

3.6 FactoryEnemy Class Reference

This class implements the Factory design pattern and returns a pointer of Enemy object depending upon the type passed in the constructor of this class.

`#include <factoryEnemy.hpp>`

Public Member Functions

- FactoryEnemy (int type, SDL_Texture *asst)
- ~FactoryEnemy ()
- Enemy * getEnemy ()

3.6.1 Detailed Description

This class implements the Factory design pattern and returns a pointer of Enemy object depending upon the type passed in the constructor of this class.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 FactoryEnemy()

```
FactoryEnemy::FactoryEnemy (
            int type,
            SDL_Texture * asst )
```

This constructor creates the Enemy object depending upon the type.

Parameters

type	is an integer variable to determine which Enemy object is to be produced.
SDL_Texture∗	is the asset sheet passed from game.cpp.

3.6.2.2 ∼FactoryEnemy()

```
FactoryEnemy::∼FactoryEnemy ( )
```

This destructor is clearing the dynamically allocated memory for the objects.

3.6.3 Member Function Documentation

3.6.3.1 getEnemy()

```
Enemy * FactoryEnemy::getEnemy ( )
```

This function is just returning the pointer to the Enemy object once it is created. Return type is Enemy∗.

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

- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/factoryEnemy.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/factoryEnemy.cpp

3.7 FactoryObstacles Class Reference

This class implements the Factory design pattern and returns a pointer of Obstacles object depending upon the type passed in the constructor of this class.

```
#include <factoryObstacles.hpp>
```

Public Member Functions

- FactoryObstacles (int type, SDL_Texture ∗asst)
- ∼FactoryObstacles ()
- Obstacles ∗ getObstacles ()

3.7.1 Detailed Description

This class implements the Factory design pattern and returns a pointer of Obstacles object depending upon the type passed in the constructor of this class.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 FactoryObstacles()

```
FactoryObstacles::FactoryObstacles (
            int type,
            SDL_Texture * asst )
```

This constructor creates the Obstacles object depending upon the type.

Parameters

type	is an integer variable to determine which Obstacles object is to be produced.
SDL_Texture∗	is the asset sheet passed from game.cpp.

3.7.2.2 ∼FactoryObstacles()

```
FactoryObstacles::~FactoryObstacles ( )
```

This destructor is clearing the dynamically allocated memory for the objects.

3.7.3 Member Function Documentation

3.7.3.1 getObstacles()

```
Obstacles * FactoryObstacles::getObstacles ( )
```

This function is just returning the pointer to the Obstacles object once it is created. Return type is Obstacles∗.

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

- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/factoryObstacles.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/factoryObstacles.cpp

3.8 Game Class Reference

This is the Game Class which runs the game and interacts with all the other classes.

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

Public Member Functions

- bool **init** ()
- bool **loadMedia** ()
- void **close** ()
- SDL_Texture * **loadTexture** (std::string path)
- void run ()
- void drawAllObjects (int mx, bool flag, bool flag2, SDL_RendererFlip flip, bool flag3)
- void EnemyAttack (SDL_Rect bgSrc, Player *ptr, SDL_RendererFlip flip, bool flag3)
- void playerAttack (Player *ptr, SDL_RendererFlip flip, char ch, SDL_Rect bgSRC, bool flag3)
- void status (SDL_Renderer *gRenderer, SDL_Texture *assets)
- void drawAllItems (SDL_RendererFlip flip, bool flag3)
- void drawJas (bool flag4, SDL_RendererFlip flip)
- void mouseOnBackgroundButton (SDL_Event *e, Player &p, SDL_Rect &bgSRC, int &px, int &enyval, int &appleval, int &lampval)

  Checks mouse input on main menu screen.
- void mouseOnQuitButton (SDL_Event *e, Player &p, SDL_Rect &bgSRC)

  Checks mouse input on pause screen.
- void mouseOnQuitButton (SDL_Event *e, Player &p, SDL_Rect &bgSRC, bool low, int &px)

  Checks mouse input on quit button on victory or lose screen.
- void gameReset (Player &p, SDL_Rect &bgSRC)

  resets the game and deletes all objects
- void readBGFromFile (SDL_Rect &r, int &points)

  loads from file upon the load game button press
- void objectCollision (Player *ptr)
- void JasmineCollision (Player *ptr, SDL_Rect &bgSRC, SDL_RendererFlip flip)
- void deleteAllObjects (list< Enemy * > &enemys, list< Obstacles * > &items)

3.8.1 Detailed Description

This is the Game Class which runs the game and interacts with all the other classes.

3.8.2 Member Function Documentation

3.8.2.1 deleteAllObjects()

```
void Game::deleteAllObjects (
            list< Enemy * > & enemys,
            list< Obstacles * > & items )
```

This function deletes all Enemies and Obstacles.

Parameters

enemys	is the list of type Enemy*. It houses all the enemies.
items	is the list of type Obstacles*. It houses all the obstacles.

3.8.2.2 drawAllItems()

```
void Game::drawAllItems (
            SDL_RendererFlip flip,
            bool flag3 )
```

This function draws all the Obstacles that are currently within the items list.

Parameters

flip	provides the orientation of the Player which is of type SDL_RendererFlip.
flag3	is a boolean that is toggled true when the Player is pressing the move key down. This is related to Obstacles movement.

3.8.2.3 drawAllObjects()

```
void Game::drawAllObjects (
            int mx,
            bool flag,
            bool flag2,
            SDL_RendererFlip flip,
            bool flag3 )
```

This function draws all the Enemies that are currently within the enemys list.

Parameters

mx	this provides the Player's integer x coordinate.
flag	is a boolean related to creation of the Enemies.
flag2	is a boolean that instigates Enemy move animations but depending upon the value toggles the actual movement across the screen.
flip	provides the orientation of the Player which is of type SDL_RendererFlip.
flag3	is a boolean that is toggled true when the Player is pressing the move key down. This is related to Obstacles movement.

3.8.2.4 drawJas()

```
void Game::drawJas (
            bool flag4,
            SDL_RendererFlip flip )
```

This function draws the Jasmine character when she exists provided the required condition for her creation is met.

Parameters

flag4	is a boolean related to the Move function of Jasmine so as to have her move across the screen with the animation only when the Player is moving to the right.
flip	provides the orientation of the Player which is of type SDL_RendererFlip.

3.8.2.5 EnemyAttack()

```
void Game::EnemyAttack (
            SDL_Rect bgSrc,
            Player * ptr,
            SDL_RendererFlip flip,
            bool flag3 )
```

This function calls the attack fucntion for all the enemies present in the scene if a certain probability is met.

Parameters

bgSrc	provides SDL_Rect src rectangle for the background - this is to generate the scene when dealing with the attack animation.
*Player**	is a pointer to the Player object.
flip	provides the orientation of the Player.
flag3	is toggled true when the Player is pressing the move key down. This is related to Obstacles movement.

3.8.2.6 gameReset()

```
void Game::gameReset (
            Player & p,
            SDL_Rect & bgSRC )
```

resets the game and deletes all objects

Parameters

Player	&p The player generated during game, in case of load game, this needs to be written to file.
SDL_Rect	&bgSRC this is the background mover that makes it scrollable. It needs to be reset, or read from file.

3.8.2.7 JasmineCollision()

```
void Game::JasmineCollision (
            Player * ptr,
```

```
            SDL_Rect & bgSRC,
            SDL_RendererFlip flip )
```

This function checks for collisions of the Player with Jasmine. If collision occurs then the end game scenario takes place.

Parameters

Player∗,this	is the Player object's pointer.
bgSRC	provides SDL_Rect src rectangle for the background - this is to generate the scene when dealing with the end game scenario.
flip	provides the Player's orientation which is of type SDL_RendererFlip.

3.8.2.8 mouseOnBackgroundButton()

```
void Game::mouseOnBackgroundButton (
            SDL_Event * e,
            Player & p,
            SDL_Rect & bgSRC,
            int & px,
            int & enyval,
            int & appleval,
            int & lampval )
```

Checks mouse input on main menu screen.

Parameters

SDL_Event∗	e SDL Event pointer that tells us when the mouse is pressed.
Player	&p The player generated during game, in case of load game, this needs to be written to file.
SDL_Rect	&bgSRC this is the background mover that makes it scrollable. It needs to be reset, or read from file.
int	&px This is the players (ajlal pls help)

3.8.2.9 mouseOnQuitButton() [1/2]

```
void Game::mouseOnQuitButton (
            SDL_Event * e,
            Player & p,
            SDL_Rect & bgSRC )
```

Checks mouse input on pause screen.

Parameters

SDL_Event∗	e SDL Event pointer that tells us when the mouse is pressed.
Player	&p The player generated during game, in case of load game, this needs to be written to file.
SDL_Rect	&bgSRC this is the background mover that makes it scrollable. It needs to be reset, or read from file.

3.8.2.10 mouseOnQuitButton() [2/2]

```
void Game::mouseOnQuitButton (
            SDL_Event * e,
            Player & p,
            SDL_Rect & bgSRC,
            bool low,
            int & px )
```

Checks mouse input on quit button on victory or lose screen.

Parameters

*SDL_Event**	e SDL Event pointer that tells us when the mouse is pressed.
Player	&p The player generated during game, in case of load game, this needs to be written to file.
SDL_Rect	&bgSRC this is the background mover that makes it scrollable. It needs to be reset, or read from file.
bool	low this is to make clear this function is overloaded and is for victory screen, has no use
int	&px this is the players virtual position and needs to be reset

3.8.2.11 objectCollision()

```
void Game::objectCollision (
            Player * ptr )
```

This function checks for collisions of the Player with the respective objects - either health incremnet or score supplement.

Parameters

Player,this*	is the Player object's pointer.

3.8.2.12 playerAttack()

```
void Game::playerAttack (
            Player * ptr,
            SDL_RendererFlip flip,
            char ch,
            SDL_Rect bgSRC,
            bool flag3 )
```

This function calls the attack function of the Player and decreases the health of the Enemy provided there is a collision.

Parameters

Player∗,this	is the Player object's pointer.
flip	provides the orientation of the Player which is of type SDL_RendererFlip.
ch	is a character that provides the attack performed by the Player.
bgSRC	provides SDL_Rect src rectangle for the background - this is to generate the scene when dealing with the attack animation.
flag3	is a boolean that is toggled true when the Player is pressing the move key down. This is related to Obstacles movement.

3.8.2.13 readBGFromFile()

```
void Game::readBGFromFile (
            SDL_Rect & r,
            int & points )
```

loads from file upon the load game button press

Parameters

SDL_Rect	&bgSRC this is the background mover that makes it scrollable. It needs to be reset, or read from file.r This is the SDL Rectangle that
int	&points This is the variable holding the amount of points a player has, and when loading, it needs to be set.

3.8.2.14 run()

```
void Game::run ( )
```

This functions houses a while loop wherein the game continously runs until the quit condition is not met.

3.8.2.15 status()

```
void Game::status (
            SDL_Renderer * gRenderer,
            SDL_Texture * assets )
```

This function is to relagate the fucntionality of the game bar at the top of the screen - the health and point meter.

Parameters

SDL_Renderer∗	to render the apple and lamp sprites.
SDL_Texture∗	for the passing of the appropiate spritesheet.

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

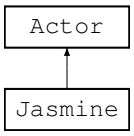
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/game.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/game.cpp

3.9 Jasmine Class Reference

This class provides the move and animation functionality for the Jasmine character. It inherits from the Actor class. This class is implemented using Singleton Design Pattern. MoveCount is tied to the move animation of the character.

`#include <jasmine.hpp>`

Inheritance diagram for Jasmine:



Public Member Functions

- Jasmine (SDL_Texture ∗asst)
- void Move (bool flag4, SDL_RendererFlip flip)
- void draw (SDL_Renderer ∗gRenderer)

Static Public Member Functions

- static Jasmine ∗ **getInstance** (SDL_Texture ∗Jassets)

Additional Inherited Members

3.9.1 Detailed Description

This class provides the move and animation functionality for the Jasmine character. It inherits from the Actor class. This class is implemented using Singleton Design Pattern. MoveCount is tied to the move animation of the character.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Jasmine()

```
Jasmine::Jasmine (
            SDL_Texture * asst )
```

This is the constructor for the Jasmine class.

Parameters

*SDL_Texture**	this provides the asset sheet from game.cpp.

3.9.3 Member Function Documentation

3.9.3.1 draw()

```
void Jasmine::draw (
            SDL_Renderer * gRenderer )
```

This function draws the Jasmine character on the screen.

Parameters

*SDL_Renderer**	is the window renderer.

3.9.3.2 Move()

```
void Jasmine::Move (
            bool flag4,
            SDL_RendererFlip flip )
```

This function calls the move animation of the Jasmine character.

Parameters

flag4	this boolean variable is tied to Player movement. It is toggled so as to allow movement and animation when Player is moving towards the right.
flip	provides the orientation of the Player.

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

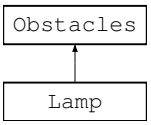
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/jasmine.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/jasmine.cpp

3.10 Lamp Class Reference

This class inherits from the Obstalces class. It is one of the two static objects which interact with the Player. The specific function of this object is related to the points of the Player.

`#include <lamp.hpp>`

Inheritance diagram for Lamp:



Public Member Functions

- Lamp (SDL_Texture *asst)
- void draw (SDL_Renderer *gRenderer)
- void move (SDL_RendererFlip flip, bool flag3)
- char name ()
- void setMove (int x)
- void createOutFile (std::ostream &myfile)

Additional Inherited Members

3.10.1 Detailed Description

This class inherits from the Obstalces class. It is one of the two static objects which interact with the Player. The specific function of this object is related to the points of the Player.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Lamp()

```
Lamp::Lamp (
            SDL_Texture * asst )
```

This is the constructor for the Lamp class.

Parameters

*SDL_Texture**	which is basically the asset file passed from the game.cpp.

3.10.3 Member Function Documentation

3.10.3.1 createOutFile()

```
void Lamp::createOutFile (
            std::ostream & myfile )  [virtual]
```

This functions writes the type, integer health value and integer x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented from Obstacles.

3.10.3.2 draw()

```
void Lamp::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This draw function renders the Lamp object on the screen.

Parameters

SDL_Renderer∗	

Reimplemented from Obstacles.

3.10.3.3 move()

```
void Lamp::move (
            SDL_RendererFlip flip,
            bool flag3 )  [virtual]
```

This function moves the Lamp object onto the screen when appropiate. When the Player is moving to the right the Lamp is reducing the x value. When player is moving to the left, we are increasing the x value.

Parameters

flip	provides the class with the orientation of the player which is of type SDL_RendererFlip..
flag3	is a boolean that is toggled true when Player is pressing the key down.

Reimplemented from Obstacles.

3.10.3.4 name()

```
char Lamp::name ( )  [virtual]
```

This function provides us with the specific type of Obstacle. In this case an Lamp.

Reimplemented from Obstacles.

3.10.3.5 setMove()

```
void Lamp::setMove (
            int x )  [virtual]
```

This function sets the integer x cooridinate of the Lamp when loading from save file.

Reimplemented from Obstacles.

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

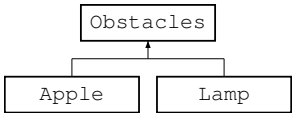
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/lamp.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/lamp.cpp

3.11 Obstacles Class Reference

This is the Parent class to generate the obstacles.

```
#include <obstacles.hpp>
```

Inheritance diagram for Obstacles:



Public Member Functions

- Obstacles (SDL_Texture *asst)
- Obstacles ()
- virtual void draw (SDL_Renderer *)
- virtual void move (SDL_RendererFlip flip, bool flag3)
- SDL_Rect getMove ()
- virtual char name ()
- virtual void setMove (int x)
- virtual void createOutFile (std::ostream &myfile)

Protected Attributes

- SDL_Rect **mover**
- SDL_Renderer * **gRenderer**
- SDL_Renderer * **assets**

3.11.1 Detailed Description

This is the Parent class to generate the obstacles.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Obstacles() [1/2]

```
Obstacles::Obstacles (
            SDL_Texture * asst )
```

This is the constructor for the Obstacles class.

Parameters

*SDL_Texture**	which is basically the asset file passed from the game.cpp.

3.11.2.2 Obstacles() [2/2]

```
Obstacles::Obstacles ( )  [inline]
```

This is the default constructor for the Obstacles class that does not take any arguments

3.11.3 Member Function Documentation

3.11.3.1 createOutFile()

```
void Obstacles::createOutFile (
            std::ostream & myfile )  [virtual]
```

This functions writes the type, integer health value and integer x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented in Lamp, and Apple.

3.11.3.2 draw()

```
void Obstacles::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This virtual draw function renders the Obstacles object on the screen.

Parameters

*SDL_Renderer**	

Reimplemented in Lamp, and Apple.

3.11.3.3 getMove()

```
SDL_Rect Obstacles::getMove ( )
```

This function returns the mover for the obstacle.

3.11.3.4 move()

```
void Obstacles::move (
            SDL_RendererFlip flip,
            bool flag3 )  [virtual]
```

This function moves the Obstalces object onto the screen when appropiate. When the Player is moving to the right the Obstacle is reducing the x value. When player is moving to the left, we are increasing the x value.

Parameters

flip	provides the class with the orientation of the player which is of type SDL_RendererFlip..
flag3	is a boolean that is toggled true when Player is pressing the key down.

Reimplemented in Lamp, and Apple.

3.11.3.5 name()

```
char Obstacles::name ( )  [virtual]
```

This function returns the specific type of Obstacle.

Reimplemented in Lamp, and Apple.

3.11.3.6 setMove()

```
void Obstacles::setMove (
            int x )  [virtual]
```

This function sets the x cooridinate of the Obstacle when loading from save file.

Reimplemented in Lamp, and Apple.

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

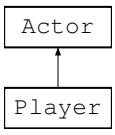
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/obstacles.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/obstacles.cpp

3.12 Player Class Reference

This is the Player class (i.e. Aladdin). This class is implemented using Singleton Design Pattern. MoveCount is an integer tied to the movement animation of the Player. SwordCount is an integer tied to the basic sword attack animation of the Player. AdSword is an integer tied the advanced sword attack animation of the Player. flip is a boolean variable which houses the orientation of the Player which is of type SDL_RendererFlip.. hurtCount is an integer tied to the hurt animation of the Player. celebCount is an integer tied to the end game animation of the Player.

```
#include <Player.hpp>
```

Inheritance diagram for Player:



Public Member Functions

- Player (SDL_Texture ∗asst)
- void HorizontalMove (bool move, int &x, int &px, SDL_RendererFlip flip)
- void BasicSword ()
- void AdvancedSword ()
- void hurt ()
- void draw (SDL_Renderer ∗gRenderer, SDL_RendererFlip flip)
- void reduceHealth ()
- void celebration ()
- void stateReset ()
- void stateLoad ()
- void stateWrite ()
- void operator++ ()
- void operator-- ()
- void operator= (int &applesLeft)

Static Public Member Functions

- static Player ∗ **getInstance** (SDL_Texture ∗assets)

Additional Inherited Members

3.12.1 Detailed Description

This is the Player class (i.e. Aladdin). This class is implemented using Singleton Design Pattern. MoveCount is an integer tied to the movement animation of the Player. SwordCount is an integer tied to the basic sword attack animation of the Player. AdSword is an integer tied the advanced sword attack animation of the Player. flip is a boolean variable which houses the orientation of the Player which is of type SDL_RendererFlip.. hurtCount is an integer tied to the hurt animation of the Player. celebCount is an integer tied to the end game animation of the Player.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Player()

```
Player::Player (
            SDL_Texture * asst )
```

This is the constructor for the Player class.

Parameters

SDL_Texture∗	this provides the asset sheet from game.cpp.

3.12.3 Member Function Documentation

3.12.3.1 AdvancedSword()

```
void Player::AdvancedSword ( )
```

This function handles the advanced sword animation.

3.12.3.2 BasicSword()

```
void Player::BasicSword ( )
```

This function handles the basic sword animation.

3.12.3.3 celebration()

```
void Player::celebration ( )
```

This function handles the end game animation.

3.12.3.4 draw()

```
void Player::draw (
            SDL_Renderer * gRenderer,
            SDL_RendererFlip flip )
```

This function draws the Player character on the screen.

Parameters

*SDL_Renderer**	is the window renderer.
flip	gives the Player's orientation which is of type SDL_RendererFlip.

3.12.3.5 HorizontalMove()

```
void Player::HorizontalMove (
            bool move,
            int & x,
            int & px,
            SDL_RendererFlip flip )
```

This function is for the movement of the Player object.

Parameters

move	is a boolean that denotes whether or not the Player is actually moving which is boolean.
x	is an integer used to move the background to help in creating a scrollable background that is a integer.
px	is the integer distance of the Player with respect to the starting point that is a integer
flip	provides the orientation of the Player which is of type SDL_RendererFlip.

3.12.3.6 hurt()

```
void Player::hurt ( )
```

This function handles the hurt animation.

3.12.3.7 operator++()

```
void Player::operator++ ( )
```

This is the increment operator overloaded. This is tied to the increment of Player health.

3.12.3.8 operator--()

```
void Player::operator-- ( )
```

This is the decrement operator overloaded. This is tied to the decrement of Player health.

3.12.3.9 operator=()

```
void Player::operator= (
            int & applesLeft )
```

This is the assignment operator overloaded. This is tied to gamebar management of Player health.

3.12.3.10 reduceHealth()

```
void Player::reduceHealth ( )
```

This function reduces Player health.

3.12.3.11 stateLoad()

```
void Player::stateLoad ( )
```

loads all the players stats from file to member variables

3.12.3.12 stateReset()

```
void Player::stateReset ( )
```

resets player stats when there is a new game

3.12.3.13 stateWrite()

```
void Player::stateWrite ( )
```

writes all player stats to game file upon exit

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

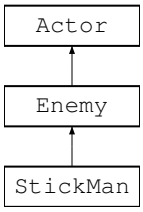
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Player.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Player.cpp

3.13 StickMan Class Reference

This class inherits from the Enemy class. It is one of the three antagonists of the game. MoveCount is the integer animation count tied to the movemnet of the StickMan. AttackCount is the integer animation count tied to the attack sequence of the StickMan. hurtCount is the integer animation count tied to the sequence when the StickMan is defeated.

```
#include <stickman.hpp>
```

Inheritance diagram for StickMan:



Public Member Functions

- StickMan (SDL_Texture ∗asst)
- void draw (SDL_Renderer ∗gRenderer)
- void emove (bool move, bool flag2)
- void move (int mx, bool flag, bool flag2, SDL_RendererFlip flip)
- void attack (SDL_Texture ∗gTexture, SDL_Rect bgSRC)
- void hurt ()
- char getType ()
- void setHealthnMover (int helth, int xmov)
- void createOutFile (std::ostream &myfile)

Additional Inherited Members

3.13.1 Detailed Description

This class inherits from the Enemy class. It is one of the three antagonists of the game. MoveCount is the integer animation count tied to the movemnet of the StickMan. AttackCount is the integer animation count tied to the attack sequence of the StickMan. hurtCount is the integer animation count tied to the sequence when the StickMan is defeated.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 StickMan()

```
StickMan::StickMan (
            SDL_Texture * asst )
```

This is the constructor for the StickMan class.

Parameters

*SDL_Texture**	which is basically the asset file passed from the game.cpp.

3.13.3 Member Function Documentation

3.13.3.1 attack()

```
void StickMan::attack (
            SDL_Texture * gTexture,
            SDL_Rect bgSRC )  [virtual]
```

This function cycles through the attack animation of the StickMan. Cycling is done through AttackCount.

Parameters

bgSRC	is an SDL_Rect which provides background co-ordinates for rendering of scene during Attack.

Reimplemented from Enemy.

3.13.3.2 createOutFile()

```
void StickMan::createOutFile (
            std::ostream & myfile )  [virtual]
```

This functions writes the type, health value and x-coordinate to our save file.

Parameters

myfile	is the ostream object passed by reference from game.cpp when creating the save file.

Reimplemented from Enemy.

3.13.3.3 draw()

```
void StickMan::draw (
            SDL_Renderer * gRenderer )  [virtual]
```

This draw function renders the StickMan object on the screen.

Parameters

SDL_Renderer∗	

Reimplemented from Enemy.

3.13.3.4 emove()

```
void StickMan::emove (
            bool move,
            bool flag2 )
```

This function cycles through the move animation of the StickMan.

Parameters

move	this is a boolean flag which toggles the x direction movement of the object on the screen. \parm flag2 is a boolean flag which toggles whether or not the animation must play or not.

3.13.3.5 getType()

```
char StickMan::getType ( )  [virtual]
```

This function returns the type of Enemy that it is. In this case a StickMan

Reimplemented from Enemy.

3.13.3.6 hurt()

```
void StickMan::hurt ( )  [virtual]
```

This function cycles through the death animation of the StickMan. Cycling is done through hurtCount.

Reimplemented from Enemy.

3.13.3.7 move()

```
void StickMan::move (
            int mx,
            bool flag,
            bool flag2,
            SDL_RendererFlip flip )  [virtual]
```

This function determines the static or movable nature of the StickMan object and thus appropiately plays the respective move animation called through emove.

Parameters

mx	this is the integer x value of the Player's mover so as to determine the distance between the StickMan and Player and thus force StickMan movement towards the left.
flag	boolean value which is set true when this enemy is created.
flag2	is a boolean variable that toggles true when left arrow key is down and false when it is up.
flip	denotes the orientation of the Player and this is of type SDL_RendererFlip.

Reimplemented from Enemy.

3.13.3.8 setHealthnMover()

```
void StickMan::setHealthnMover (
            int helth,
            int xmov )  [virtual]
```

This function sets the StickMan's health and position when loading the game from the save file. \parm helth is the passed health integer value read from the save file.

Parameters

xmov	is the passed integer x-coordinate read from the save file.

Reimplemented from Enemy.

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

- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/stickman.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/stickman.cpp

Index