

The Adventure of Aladdin

Generated by Doxygen 1.8.20

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	5
3.1 Actor Class Reference	5
3.1.1 Detailed Description	6
3.2 Alien Class Reference	6
3.2.1 Detailed Description	6
3.2.2 Constructor & Destructor Documentation	7
3.2.2.1 Alien()	7
3.2.3 Member Function Documentation	7
3.2.3.1 attack()	7
3.2.3.2 createOutFile()	7
3.2.3.3 draw()	7
3.2.3.4 emove()	8
3.2.3.5 getType()	8
3.2.3.6 hurt()	8
3.2.3.7 move()	8
3.2.3.8 setHealthnMover()	9
3.3 Apple Class Reference	9
3.3.1 Detailed Description	10
3.3.2 Constructor & Destructor Documentation	10
3.3.2.1 Apple()	10
3.3.3 Member Function Documentation	10
3.3.3.1 createOutFile()	10
3.3.3.2 draw()	11
3.3.3.3 move()	11
3.3.3.4 name()	11
3.3.3.5 setMove()	12
3.4 BigGuard Class Reference	12
3.4.1 Detailed Description	13
3.4.2 Constructor & Destructor Documentation	13
3.4.2.1 BigGuard()	13
3.4.3 Member Function Documentation	13
3.4.3.1 attack()	13
3.4.3.2 createOutFile()	13
3.4.3.3 draw()	14
3.4.3.4 emove()	14
3.4.3.5 getType()	14
3.4.3.6 hurt()	15

3.4.3.7 move()	15
3.4.3.8 setHealthnMover()	15
3.5 Enemy Class Reference	16
3.5.1 Detailed Description	16
3.5.2 Constructor & Destructor Documentation	16
3.5.2.1 Enemy() [1/2]	16
3.5.2.2 Enemy() [2/2]	17
3.5.3 Member Function Documentation	17
3.5.3.1 attack()	17
3.5.3.2 checkCollison()	17
3.5.3.3 createOutFile()	18
3.5.3.4 draw()	18
3.5.3.5 getType()	18
3.5.3.6 hurt()	18
3.5.3.7 move()	19
3.5.3.8 operator<=()	19
3.5.3.9 reduceHealth()	19
3.5.3.10 setHealthnMover()	20
3.6 FactoryEnemy Class Reference	20
3.6.1 Detailed Description	20
3.6.2 Constructor & Destructor Documentation	20
3.6.2.1 FactoryEnemy()	20
3.6.2.2 ~FactoryEnemy()	21
3.6.3 Member Function Documentation	21
3.6.3.1 getEnemy()	21
3.7 FactoryObstacles Class Reference	21
3.7.1 Detailed Description	21
3.7.2 Constructor & Destructor Documentation	22
3.7.2.1 FactoryObstacles()	22
3.7.2.2 ~FactoryObstacles()	22
3.7.3 Member Function Documentation	22
3.7.3.1 getObstacles()	22
3.8 Game Class Reference	22
3.8.1 Detailed Description	23
3.8.2 Member Function Documentation	23
3.8.2.1 deleteAllObjects()	23
3.8.2.2 drawAllItems()	24
3.8.2.3 drawAllObjects()	24
3.8.2.4 drawJas()	24
3.8.2.5 EnemyAttack()	25
3.8.2.6 gameReset()	25
3.8.2.7 JasmineCollision()	25

3.8.2.8 mouseOnBackgroundButton()	26
3.8.2.9 mouseOnQuitButton() [1/2]	26
3.8.2.10 mouseOnQuitButton() [2/2]	27
3.8.2.11 objectCollision()	27
3.8.2.12 playerAttack()	27
3.8.2.13 readBGFromFile()	28
3.8.2.14 run()	28
3.8.2.15 status()	28
3.9 Jasmine Class Reference	29
3.9.1 Detailed Description	29
3.9.2 Constructor & Destructor Documentation	29
3.9.2.1 Jasmine()	29
3.9.3 Member Function Documentation	30
3.9.3.1 draw()	30
3.9.3.2 Move()	30
3.10 Lamp Class Reference	30
3.10.1 Detailed Description	31
3.10.2 Constructor & Destructor Documentation	31
3.10.2.1 Lamp()	31
3.10.3 Member Function Documentation	31
3.10.3.1 createOutFile()	32
3.10.3.2 draw()	32
3.10.3.3 move()	32
3.10.3.4 name()	33
3.10.3.5 setMove()	33
3.11 Obstacles Class Reference	33
3.11.1 Detailed Description	34
3.11.2 Constructor & Destructor Documentation	34
3.11.2.1 Obstacles() [1/2]	34
3.11.2.2 Obstacles() [2/2]	34
3.11.3 Member Function Documentation	34
3.11.3.1 createOutFile()	34
3.11.3.2 draw()	35
3.11.3.3 getMove()	35
3.11.3.4 move()	35
3.11.3.5 name()	36
3.11.3.6 setMove()	36
3.12 Player Class Reference	36
3.12.1 Detailed Description	37
3.12.2 Constructor & Destructor Documentation	37
3.12.2.1 Player()	37
3.12.3 Member Function Documentation	38

3.12.3.1 AdvancedSword()	38
3.12.3.2 BasicSword()	38
3.12.3.3 celebration()	38
3.12.3.4 draw()	38
3.12.3.5 HorizontalMove()	38
3.12.3.6 hurt()	39
3.12.3.7 operator++()	39
3.12.3.8 operator--()	39
3.12.3.9 operator=()	39
3.12.3.10 reduceHealth()	39
3.12.3.11 stateLoad()	39
3.12.3.12 stateReset()	40
3.12.3.13 stateWrite()	40
3.13 StickMan Class Reference	40
3.13.1 Detailed Description	41
3.13.2 Constructor & Destructor Documentation	41
3.13.2.1 StickMan()	41
3.13.3 Member Function Documentation	41
3.13.3.1 attack()	41
3.13.3.2 createOutFile()	42
3.13.3.3 draw()	42
3.13.3.4 emove()	42
3.13.3.5 getType()	42
3.13.3.6 hurt()	43
3.13.3.7 move()	43
3.13.3.8 setHealthnMover()	43
Index	45

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Actor	5
Enemy	16
Alien	6
BigGuard	12
StickMan	40
Jasmine	29
Player	36
FactoryEnemy	20
FactoryObstacles	21
Game	22
Obstacles	33
Apple	9
Lamp	30

Chapter 2

Class Index

2.1 Class List

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

Actor	Parent class from which all Enemies, Player and Jasmine inherit from	5
Alien	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. 6	
Apple	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	9
BigGuard	This class inherits from the Enemy class. It is one of the three antagonists of the game. Move↔Count 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. 12	
Enemy	This is the Parent class to generate the enemies. This class further inherits from the Actor Class	16
FactoryEnemy	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	20
FactoryObstacles	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	21
Game	This is the Game Class which runs the game and interacts with all the other classes	22
Jasmine	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	29
Lamp	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	30
Obstacles	This is the Parent class to generate the obstacles	33

Player

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](#) 36

StickMan

This class inherits from the [Enemy](#) class. It is one of the three antagonists of the game. Move↔Count 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.

40

Chapter 3

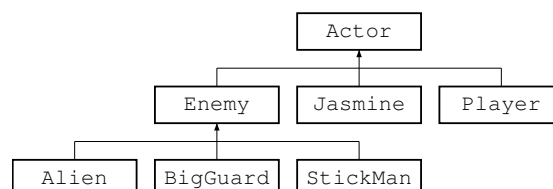
Class Documentation

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:

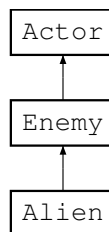
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Actor.hpp

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

<i>SDL_Texture*</i>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

Reimplemented from [Enemy](#).

3.2.3.4 move()

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

This function cycles through the move animation of the [Alien](#).

Parameters

<i>move</i>	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 appropriately plays the respective move animation called through emove.

Parameters

<i>mx</i>	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.
<i>flag</i>	boolean value which is set true when this enemy is created.
<i>flag2</i>	variable that toggles true when left arrow key is down and false when it is up.
<i>flip</i>	denotes the orientation of the Player .

Reimplemented from [Enemy](#).

3.2.3.8 setHealthnMover()

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

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

Parameters

<i>xmov</i>	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:

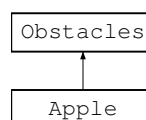
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/alien.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/alien.cpp

3.3 Apple Class Reference

This class inherits from the Obstacles 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 Obstacles 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

<i>SDL_Texture*</i>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

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 appropriate. 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

<i>flip</i>	provides the class with the orientation of the player which is of type SDL_RendererFlip.
<i>flag3</i>	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 coordinate of the [Apple](#) when loading from save file.

Reimplemented from [Obstacles](#).

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

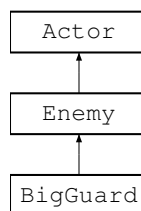
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/apple.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/apple.cpp

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

<i>SDL_Texture*</i>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

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

<i>move</i>	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

<i>bgSRC</i>	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 appropriately plays the respective move animation called through emove.

Parameters

<i>mx</i>	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.
<i>flag</i>	boolean value which is set true when this enemy is created.
<i>flag2</i>	variable that toggles true when left arrow key is down and false when it is up.
<i>flip</i>	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

<i>xmov</i>	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:

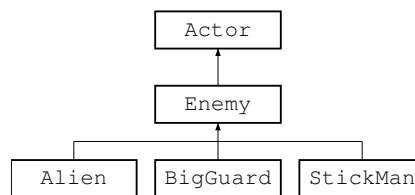
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/bigguard.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/bigguard.cpp

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

<i>SDL_Texture*</i>	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

<i>a</i>	is the <code>SDL_Rect</code> mover for the Player .
<i>b</i>	is the <code>SDL_Rect</code> mover for the Enemy .
<i>flip</i>	denotes the orientation of the Player which is of type <code>SDL_RendererFlip</code> .
<i>ch</i>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

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 appropriately plays the respective move animation.

Parameters

<i>mx</i>	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.
<i>flag</i>	boolean value which is set true when this enemy is created.
<i>flag2</i>	variable that toggles true when left arrow key is down and false when it is up.
<i>flip</i>	denotes the orientation of the Player which is of type <code>SDL_RendererFlip</code> .

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

<i>temp</i>	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

<i>ch</i>	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

<i>xmov</i>	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

<i>type</i>	is an integer variable to determine which Enemy object is to be produced.
<i>SDL_Texture*</i>	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

<i>type</i>	is an integer variable to determine which Obstacles object is to be produced.
<i>SDL_Texture*</i>	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

<i>enemys</i>	is the list of type Enemy*. It houses all the enemies.
<i>items</i>	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

<i>flip</i>	provides the orientation of the Player which is of type <code>SDL_RendererFlip</code> .
<i>flag3</i>	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

<i>mx</i>	this provides the Player 's integer x coordinate.
<i>flag</i>	is a boolean related to creation of the Enemies.
<i>flag2</i>	is a boolean that instigates Enemy move animations but depending upon the value toggles the actual movement across the screen.
<i>flip</i>	provides the orientation of the Player which is of type <code>SDL_RendererFlip</code> .
<i>flag3</i>	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

<i>flag4</i>	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.
<i>flip</i>	provides the orientation of the Player which is of type <code>SDL_RendererFlip</code> .

3.8.2.5 EnemyAttack()

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

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

Parameters

<i>bgSrc</i>	provides <code>SDL_Rect</code> src rectangle for the background - this is to generate the scene when dealing with the attack animation.
<i>Player*</i>	is a pointer to the Player object.
<i>flip</i>	provides the orientation of the Player .
<i>flag3</i>	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

<i>Player</i>	&p The player generated during game, in case of load game, this needs to be written to file.
<i>SDL_Rect</i>	&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

<i>Player*,this</i>	is the Player object's pointer.
<i>bgSRC</i>	provides SDL_Rect src rectangle for the background - this is to generate the scene when dealing with the end game scenario.
<i>flip</i>	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

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

<i>SDL_Event*</i>	e SDL Event pointer that tells us when the mouse is pressed.
<i>Player</i>	&p The player generated during game, in case of load game, this needs to be written to file.
<i>SDL_Rect</i>	&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

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

Parameters

<i>Player*,this</i>	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

<i>Player*,this</i>	is the Player object's pointer.
<i>flip</i>	provides the orientation of the Player which is of type <code>SDL_RendererFlip</code> .
<i>ch</i>	is a character that provides the attack performed by the Player .
<i>bgSRC</i>	provides <code>SDL_Rect</code> src rectangle for the background - this is to generate the scene when dealing with the attack animation.
<i>flag3</i>	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

<i>SDL_Rect</i>	&bgSRC this is the background mover that makes it scrollable. It needs to be reset, or read from file.r This is the <code>SDL_Rect</code> that
<i>int</i>	&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

<i>SDL_Renderer*</i>	to render the apple and lamp sprites.
<i>SDL_Texture*</i>	for the passing of the appropriate spritesheet.

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

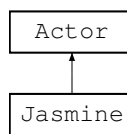
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/game.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/game.cpp

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

<i>SDL_Texture*</i>	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

<i>SDL_Renderer*</i>	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

<i>flag4</i>	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.
<i>flip</i>	provides the orientation of the Player .

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

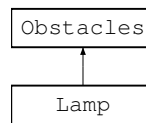
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/jasmine.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/jasmine.cpp

3.10 Lamp Class Reference

This class inherits from the Obstacles 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 Obstacles 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

<i>SDL_Texture*</i>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

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 appropriate. 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

<i>flip</i>	provides the class with the orientation of the player which is of type SDL_RendererFlip..
<i>flag3</i>	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 coordinate of the [Lamp](#) when loading from save file.

Reimplemented from [Obstacles](#).

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

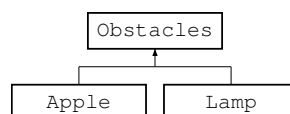
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/lamp.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/lamp.cpp

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

<code>SDL_Texture*</code>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

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 Obstacles object onto the screen when appropriate. 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

<i>flip</i>	provides the class with the orientation of the player which is of type <code>SDL_RendererFlip</code> .
<i>flag3</i>	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 coordinate of the Obstacle when loading from save file.

Reimplemented in [Lamp](#), and [Apple](#).

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

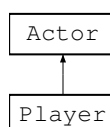
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/obstacles.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/obstacles.cpp

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 to 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 to 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

<i>SDL_Texture*</i>	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

<i>SDL_Renderer*</i>	is the window renderer.
<i>flip</i>	gives the Player 's orientation which is of type <code>SDL_RendererFlip</code> .

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

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

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:

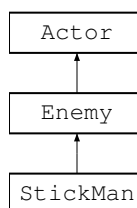
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Player.hpp
- C:/Users/akeel/OneDrive/Desktop/finalfinalfinalfinalfinal/Player.cpp

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

<i>SDL_Texture*</i>	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

<i>bgSRC</i>	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

<i>myfile</i>	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

<i>SDL_Renderer*</i>	
----------------------	--

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

<i>move</i>	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 appropriately plays the respective move animation called through emove.

Parameters

<i>mx</i>	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.
<i>flag</i>	boolean value which is set true when this enemy is created.
<i>flag2</i>	is a boolean variable that toggles true when left arrow key is down and false when it is up.
<i>flip</i>	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

<i>xmov</i>	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

~FactoryEnemy
 FactoryEnemy, 21
~FactoryObstacles
 FactoryObstacles, 22

Actor, 5
AdvancedSword
 Player, 38
Alien, 6
 Alien, 7
 attack, 7
 createOutFile, 7
 draw, 7
 emove, 8
 getType, 8
 hurt, 8
 move, 8
 setHealthnMover, 9

Apple, 9
 Apple, 10
 createOutFile, 10
 draw, 11
 move, 11
 name, 11
 setMove, 11

attack
 Alien, 7
 BigGuard, 13
 Enemy, 17
 StickMan, 41

BasicSword
 Player, 38

BigGuard, 12
 attack, 13
 BigGuard, 13
 createOutFile, 13
 draw, 14
 emove, 14
 getType, 14
 hurt, 14
 move, 15
 setHealthnMover, 15

celebration
 Player, 38

checkCollison
 Enemy, 17

createOutFile
 Alien, 7

Apple, 10
BigGuard, 13
Enemy, 17
Lamp, 31
Obstacles, 34
StickMan, 41

deleteAllObjects
 Game, 23

draw
 Alien, 7
 Apple, 11
 BigGuard, 14
 Enemy, 18
 Jasmine, 30
 Lamp, 32
 Obstacles, 35
 Player, 38
 StickMan, 42

drawAllItems
 Game, 24

drawAllObjects
 Game, 24

drawJas
 Game, 24

emove
 Alien, 8
 BigGuard, 14
 StickMan, 42

Enemy, 16
 attack, 17
 checkCollison, 17
 createOutFile, 17
 draw, 18
 Enemy, 16, 17
 getType, 18
 hurt, 18
 move, 18
 operator<=, 19
 reduceHealth, 19
 setHealthnMover, 19

EnemyAttack
 Game, 25

FactoryEnemy, 20
 ~FactoryEnemy, 21
 FactoryEnemy, 20
 getEnemy, 21
FactoryObstacles, 21

- ~FactoryObstacles, 22
 - FactoryObstacles, 22
 - getObstacles, 22
- Game, 22
 - deleteAllObjects, 23
 - drawAllItems, 24
 - drawAllObjects, 24
 - drawJas, 24
 - EnemyAttack, 25
 - gameReset, 25
 - JasmineCollision, 25
 - mouseOnBackgroundButton, 26
 - mouseOnQuitButton, 26, 27
 - objectCollision, 27
 - playerAttack, 27
 - readBGFromFile, 28
 - run, 28
 - status, 28
- gameReset
 - Game, 25
- getEnemy
 - FactoryEnemy, 21
- getMove
 - Obstacles, 35
- getObstacles
 - FactoryObstacles, 22
- getType
 - Alien, 8
 - BigGuard, 14
 - Enemy, 18
 - StickMan, 42
- HorizontalMove
 - Player, 38
- hurt
 - Alien, 8
 - BigGuard, 14
 - Enemy, 18
 - Player, 39
 - StickMan, 43
- Jasmine, 29
 - draw, 30
 - Jasmine, 29
 - Move, 30
- JasmineCollision
 - Game, 25
- Lamp, 30
 - createOutFile, 31
 - draw, 32
 - Lamp, 31
 - move, 32
 - name, 32
 - setMove, 33
- mouseOnBackgroundButton
 - Game, 26
- mouseOnQuitButton
 - Game, 26, 27
- Move
 - Jasmine, 30
- move
 - Alien, 8
 - Apple, 11
 - BigGuard, 15
 - Enemy, 18
 - Lamp, 32
 - Obstacles, 35
 - StickMan, 43
- name
 - Apple, 11
 - Lamp, 32
 - Obstacles, 35
- objectCollision
 - Game, 27
- Obstacles, 33
 - createOutFile, 34
 - draw, 35
 - getMove, 35
 - move, 35
 - name, 35
 - Obstacles, 34
 - setMove, 36
- operator<=
 - Enemy, 19
- operator++
 - Player, 39
- operator--
 - Player, 39
- operator=
 - Player, 39
- Player, 36
 - AdvancedSword, 38
 - BasicSword, 38
 - celebration, 38
 - draw, 38
 - HorizontalMove, 38
 - hurt, 39
 - operator++, 39
 - operator--, 39
 - operator=, 39
 - Player, 37
 - reduceHealth, 39
 - stateLoad, 39
 - stateReset, 39
 - stateWrite, 40
- playerAttack
 - Game, 27
- readBGFromFile
 - Game, 28
- reduceHealth
 - Enemy, 19

- Player, [39](#)
- run
 - Game, [28](#)
- setHealthnMover
 - Alien, [9](#)
 - BigGuard, [15](#)
 - Enemy, [19](#)
 - StickMan, [43](#)
- setMove
 - Apple, [11](#)
 - Lamp, [33](#)
 - Obstacles, [36](#)
- stateLoad
 - Player, [39](#)
- stateReset
 - Player, [39](#)
- stateWrite
 - Player, [40](#)
- status
 - Game, [28](#)
- StickMan, [40](#)
 - attack, [41](#)
 - createOutFile, [41](#)
 - draw, [42](#)
 - emove, [42](#)
 - getType, [42](#)
 - hurt, [43](#)
 - move, [43](#)
 - setHealthnMover, [43](#)
 - StickMan, [41](#)