The Legend of Python, Modular Interface Specification

Revison 1.8 11/9/2018

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

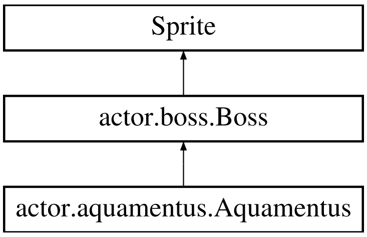
Chapter 4

Class Documentation

4.1 actor.aquamentus.Aquamentus Class Reference

This class represents the Aquamentus Boss.

Inheritance diagram for actor.aquamentus.Aquamentus:



Public Member Functions

- def __init__ (self, x, y)

  Constructor for Aquamentus.
- def checkState (self)

  Evaluates the state of the Aquamentus.
- def swapDirection (self)

  Swaps Aquamentus' direction.
- def attack (self)

  Allows for Aquamentus to attack.
- def bossLogic (self)

  Controls Aquamentus logic.

Public Attributes

- isAttacking

  Represents wether or not the boss is attacking.
- attackStartFrame

  Represents the frame that the boss starts an attack.
- fireballs

  Represents the boss' fireballs.
- xSpeed

  Aquamentus' set x speed.
- maxHP

  Aquamentus' max health.
- HP

  Aquamentus' current health.
- dmg

  Aquamentus' damage.
- image

  Sprite image.
- sprites

  Array of sprites.
- obj

  Collision list.
- spriteIndex

  Index for the array of sprites.
- hitCount

  Integer value representing the buffer for the number of hits for Aquamentus after being hit by player character.
- isHit

  Represents the current state if Aquamentus has collided with player character attack.
- oldx

  This represents the previous x-location of Aquamentus in movement/stationary state.
- oldy

  This represents the previous y-location of Aquamentus in movement/stationary state.

4.1.1 Detailed Description

This class represents the Aquamentus Boss.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
def actor.aquamentus.Aquamentus.__init__ (
              self,
              x,
              y )
```

Constructor for Aquamentus.

Constructor takes two parameters, the x and y coordinates

Parameters

x	X coordinate of the starting postion of the Aquamentus
y	Y coordinate of the starting postion of the Aquamentus

4.1.3 Member Function Documentation

4.1.3.1 attack()

```
def actor.aquamentus.Aquamentus.attack (
              self )
```

Allows for Aquamentus to attack.

Spawns fireballs at Aquamenuts' mouth and sets their move speed

4.1.3.2 bossLogic()

```
def actor.aquamentus.Aquamentus.bossLogic (
              self )
```

Controls Aquamentus logic.

Uses the states to control the Aquamentus

4.1.3.3 checkState()

```
def actor.aquamentus.Aquamentus.checkState (
              self )
```

Evaluates the state of the Aquamentus.

Evalautes if Aquamentus can stop, if it is in iframes, if it collides with something, and if it has died

4.1.3.4 swapDirection()

```
def actor.aquamentus.Aquamentus.swapDirection (
              self )
```

Swaps Aquamentus' direction.

Multiplies the speed in the x direction by -1

4.1.4 Member Data Documentation

4.1.4.1 hitCount

```
actor.aquamentus.Aquamentus.hitCount
```

Integer value representing the buffer for the number of hits for Aquamentus after being hit by player character.

4.1.4.2 isHit

```
actor.aquamentus.Aquamentus.isHit
```

Represents the current state if Aquamentus has collided with player character attack.

4.1.4.3 oldx

```
actor.aquamentus.Aquamentus.oldx
```

This represents the previous x-location of Aquamentus in movement/stationary state.

4.1.4.4 oldy

```
actor.aquamentus.Aquamentus.oldy
```

This represents the previous y-location of Aquamentus in movement/stationary state.

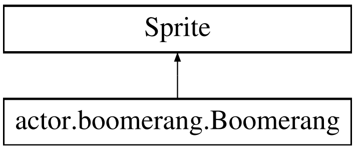
The documentation for this class was generated from the following file:

- src/actor/aquamentus.py

4.2 actor.boomerang.Boomerang Class Reference

Boomerang Weapon Class The class holding the creation, behaviour, and collision effects of the player's boomerang weapon.

Inheritance diagram for actor.boomerang.Boomerang:



Public Member Functions

- def __init__ (self, x, y, direction, obj, player)

  Boomerang constructor A sprite subclass constructor that takes an x and y position (the players), a direction for the boomerang's trajectory, a list of collidable objects, as well as the player object (to add to the collision list)
- def moveupdate (self)

  Boomerang position updating function, updating position based on changing trajectory speed.
- def collisionupdate (self)

  Boomerang collision updating function, constantly checking for collisions and acting accordingly.
- def spriteupdate (self)

  Updates the boomerang sprite every 10 frames.
- def update (self)

  Boomerang updating function, repeatedly running moveupdate and collisionupdate.
- def collision (self)

  Default collision function to satisfy other class collision calls to this object.

Public Attributes

- image

  Boomerang sprite image.
- rect

  Position of boomerang.
- dir

  Initial travel direction of boomerang.
- speed

  Boomerang initial update speed.
- obj

  List of objects the boomerang could collide with.
- killable

  Boolean to tell when the boomerang should be deleted.
- sprites

  List of sprites for boomerang.
- spriteIndex

  Index of the current sprite.
- frameCounter

  Total number of frames boomerang is onscreen.

4.2.1 Detailed Description

Boomerang Weapon Class The class holding the creation, behaviour, and collision effects of the player's boomerang weapon.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
def actor.boomerang.Boomerang.__init__ (
              self,
              x,
              y,
              direction,
              obj,
              player )
```

Boomerang constructor A sprite subclass constructor that takes an x and y position (the players), a direction for the boomerang's trajectory, a list of collidable objects, as well as the player object (to add to the collision list)

Parameters

x	X coordinate of boomerang spawn
y	Y coordinate of boomerang spawn
direction	Direction of boomerang path
obj	List of objects to check for collision with the boomerang
player	Player object to also check collision for

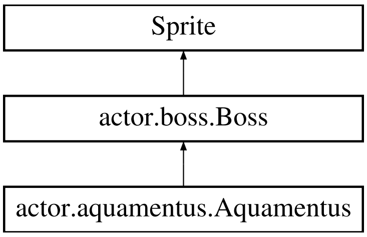
The documentation for this class was generated from the following file:

- src/actor/boomerang.py

4.3 actor.boss.Boss Class Reference

Superclass for representing a Boss.

Inheritance diagram for actor.boss.Boss:



Public Member Functions

- def __init__ (self, x, y)

  Constructor for Boss.
- def checkState (self)

  Empty function for evaluating Boss state.
- def move (self)

  Moves the Boss.
- def bossLogic (self)

  Empty function for logic of Boss.
- def update (self)

  Update loop for a Boss.
- def hit (self, dir)

  Hit dectetion for Boss.

Public Attributes

- image

  Boss Pygame surface.
- rect

  Rectanlge that represents boss.
- id

  X postion of the boss.
- isHit

  Wether the boss is hit or not.
- stuncount

  The amount of stun frames the boss has remaining.
- maxHP

  Boss' max health.
- HP

  Boss' current health.
- dmg

  Boss' damage.
- hitCount

  Boss' hit count.
- xSpeed

  Boss' speed in the x direction.
- ySpeed

  Boss' speed in the y direction.
- frameCounter

  Total number of frames the boss has been alive for.

4.3.1 Detailed Description

Superclass for representing a Boss.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 __init__()

```
def actor.boss.Boss.__init__ (
              self,
              x,
              y )
```

Constructor for Boss.

Constructor takes two parameters, the x and y coordinates

Parameters

x	X coordinate of the starting postion of the Boss
y	Y coordinate of the starting postion of the Boss

4.3.3 Member Function Documentation

4.3.3.1 hit()

```
def actor.boss.Boss.hit (
              self,
              dir )
```

Hit dectetion for Boss.

Handles health and iframes

4.3.3.2 move()

```
def actor.boss.Boss.move (
              self )
```

Moves the Boss.

Adds the x speed and y speed to the x and y postion of the Boss

4.3.3.3 update()

```
def actor.boss.Boss.update (
              self )
```

Update loop for a Boss.

Checks state, does the logic, and then moves Boss

4.3.4 Member Data Documentation

4.3.4.1 id

```
actor.boss.Boss.id
```

X postion of the boss.

Y postion of the boss Boss ID

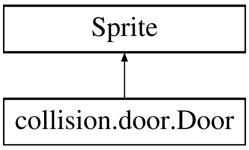
The documentation for this class was generated from the following file:

- src/actor/boss.py

4.4 collision.door.Door Class Reference

Dungeon Door Class This class is used to place a door on one of four places of a dungeon room, to allow the player to walk through and traverse to other rooms within the dungeon.

Inheritance diagram for collision.door.Door:



Public Member Functions

- def __init__ (self, direction, state)

  Door constructor, making a door on the specified side of the dungeon.
- def collision (self, i)

  This method checks for collision with the Door object and other sprite object.
- def openDoor (self)

  Function to set a door state and sprite to open Changes a blocked door (state 1/2) to an open door (state 0), changing collision attributes and sprites respectively.

Public Attributes

- **dungeon_sheet**
- state

  DOOR STATE 0 = open 1 = blocked (objective door) 2 = locked (key door) State of the door, integer value on whether the door is open(0), blocked by an objective(1), or locked(2) (blocked = eg.
- **dir**
- image

  Sprite image of the door.
- rect

  X and y coordinates of the image.
- id

  Collision ID (to help other objects identify what they are colliding with and how to react)

4.4.1 Detailed Description

Dungeon Door Class This class is used to place a door on one of four places of a dungeon room, to allow the player to walk through and traverse to other rooms within the dungeon.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 __init__()

```
def collision.door.Door.__init__ (
              self,
              direction,
              state )
```

Door constructor, making a door on the specified side of the dungeon.

Parameters

direction	Integer value of the wall the door will be on (direction from centre of room)

4.4.3 Member Function Documentation

4.4.3.1 collision()

```
def collision.door.Door.collision (
              self,
              i )
```

This method checks for collision with the Door object and other sprite object.

The collision will be checked with the Door and other sprite object as the object collides with the wall.

Parameters

i	This is the sprite object that is passed into the method, and checks if the object is colliding with the Door object, reseting the sprite objects location accordingly.

4.4.4 Member Data Documentation

4.4.4.1 state

```
collision.door.Door.state
```

DOOR STATE 0 = open 1 = blocked (objective door) 2 = locked (key door) State of the door, integer value on whether the door is open(0), blocked by an objective(1), or locked(2) (blocked = eg.

kill all enemies in room to open)

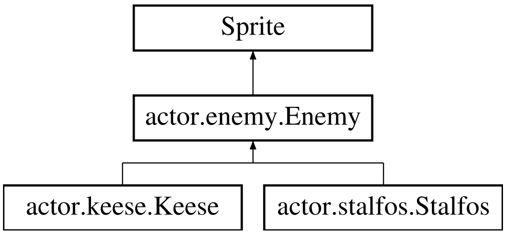
The documentation for this class was generated from the following file:

- src/collision/door.py

4.5 actor.enemy.Enemy Class Reference

Superclass for representing an Enemy.

Inheritance diagram for actor.enemy.Enemy:



Public Member Functions

- def __init__ (self, x, y)

  Constructor for Enemy.
- def checkState (self)

  Empty function for evaluating Enemy state.
- def move (self)

  Moves the Enemy.
- def enemyLogic (self)

  Empty function for logic of Enemy.
- def update (self)

  Update loop for a Enemy.
- def hit (self, direc)

  Hit dectetion for Enemy.

Public Attributes

- image

  Enemy Pygame surface.
- rect

  Rectanlge that represents enemy.
- id

  Enemy x postion.
- isHit

  Wether the enemy is hit.
- stuncount

  Remaining stun frames.
- maxHP

  Enemy max health.
- HP

  Enemy current health.
- dmg

  Enemy damage.
- hitCount

  This represents the buffer for the number of hits for the Enemy after being hit by player character.

- hitdir

  Represents the direction Enemy is hit in by player character attack.
- xSpeed

  Enemy speed in the x direction.
- ySpeed

  Enemy speed in the y direction.
- frameCounter

  Total number of frames the enemy has been alive.

4.5.1 Detailed Description

Superclass for representing an Enemy.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 __init__()

```
def actor.enemy.Enemy.__init__ (
              self,
              x,
              y )
```

Constructor for Enemy.

Constructor takes two parameters, the x and y coordinates

Parameters

x	X coordinate of the starting postion of the Enemy
y	Y coordinate of the starting postion of the Enemy

4.5.3 Member Function Documentation

4.5.3.1 hit()

```
def actor.enemy.Enemy.hit (
              self,
              direc )
```

Hit dectetion for Enemy.

Handles health, iframes and knockback direction

Parameters

direc	Direction of the knockback

4.5.3.2 move()

```
def actor.enemy.Enemy.move (
              self )
```

Moves the Enemy.

Adds the x speed and y speed to the x and y postion of the Enemy

4.5.3.3 update()

```
def actor.enemy.Enemy.update (
              self )
```

Update loop for a Enemy.

Checks state, does the logic, and then moves Enemy

4.5.4 Member Data Documentation

4.5.4.1 hitCount

```
actor.enemy.Enemy.hitCount
```

This represents the buffer for the number of hits for the Enemy after being hit by player character.

4.5.4.2 hitdir

```
actor.enemy.Enemy.hitdir
```

Represents the direction Enemy is hit in by player character attack.

4.5.4.3 id

```
actor.enemy.Enemy.id
```

Enemy x postion.

Enemy y postion Enemy Id type

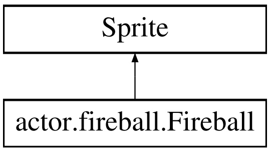
The documentation for this class was generated from the following file:

- src/actor/enemy.py

4.6 actor.fireball.Fireball Class Reference

This class represents the Fireball object.

Inheritance diagram for actor.fireball.Fireball:



Public Member Functions

- def __init__ (self, x, y, xSpeed, ySpeed)

  Constructor for Fireball.
- def start (self, x, y, xs, ys)

  Starts the Fireball's movement.
- def end (self)

  Ends the Fireball's movement.
- def hit (self, dir)

  Empty function for being hit by player.
- def checkState (self)

  Evaluates the state of the Fireball.
- def move (self)

  Moves the Fireball.
- def logic (self)

  Updates the Fireball sprite.
- def update (self)

  Updates the Fireball every frame

Public Attributes

- image

  Fireball surface.
- rect

  Rectangle that represents the fireball.
- id

  Fireball x postion.
- isHit

  Wether the fireball is hit.
- dmg

  Fireball damage.
- hitCount

  Fireball hit frames.
- xSpeed

  Fireball speed in the x direction.
- ySpeed

  Fireball speed in the y direction.
- frameCounter

  Total number of frames fireball is alive.
- **sprites**
- obj

  Fireball collision list.
- spriteIndex

  Sprite list index.

4.6.1 Detailed Description

This class represents the Fireball object.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 __init__()

```
def actor.fireball.Fireball.__init__ (
              self,
              x,
              y,
              xSpeed,
              ySpeed )
```

Constructor for Fireball.

Constructor takes four parameters, the x and y coordinates and the x and y speeds

Parameters

x	X coordinate of the starting postion of the Fireball
y	Y coordinate of the starting postion of the Fireball
xSpeed	The speed in the x direction of the Fireball
ySpeed	The speed in the y direction of the Fireball

4.6.3 Member Function Documentation

4.6.3.1 checkState()

```
def actor.fireball.Fireball.checkState (
              self )
```

Evaluates the state of the Fireball.

Checks for collision with walls, doors, and players

4.6.3.2 end()

```
def actor.fireball.Fireball.end (
              self )
```

Ends the Fireball's movement.

Sets the x and y speed to 0 and places the Fireball of screen

4.6.3.3 hit()

```
def actor.fireball.Fireball.hit (
              self,
              dir )
```

Empty function for being hit by player.

Needs to exist for when player sword collides with Fireball

4.6.3.4 logic()

```
def actor.fireball.Fireball.logic (
              self )
```

Updates the Fireball sprite.

Swaps between the 2 sprites every 15 frames

4.6.3.5 move()

```
def actor.fireball.Fireball.move (
              self )
```

Moves the Fireball.

Adds the x speed and y speed to the x and y postion of the Fireball

4.6.3.6 start()

```
def actor.fireball.Fireball.start (
              self,
              x,
              y,
              xs,
              ys )
```

Starts the Fireball's movement.

Sets the x and y postions and x and y speeds for Fireball

Parameters

x	X coordinate of where the Fireball is placed
y	Y coordinate of where the Fireball is placed
xs	The speed in the x direction of the Fireball
ys	The speed in the y direction of the Fireball

4.6.3.7 update()

```
def actor.fireball.Fireball.update (
              self )
```

Updates the Fireball every frame

Checks the state, applies the logic, and then moves the Fireball

4.6.4 Member Data Documentation

4.6.4.1 id

```
actor.fireball.Fireball.id
```

Fireball x postion.

Fireball y positon Fireball ID type

4.6.4.2 image

```
actor.fireball.Fireball.image
```

Fireball surface.

Fireball current sprite.

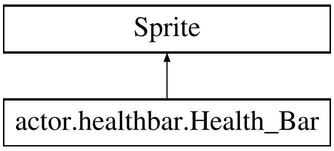
The documentation for this class was generated from the following file:

- src/actor/fireball.py

4.7 actor.healthbar.Health_Bar Class Reference

This class represents the HealthBar for the user controlled player character.

Inheritance diagram for actor.healthbar.Health_Bar:



Public Member Functions

- def __init__ (self, x, y)

    Constructor HealthBar class.
- def health (self, i)

    Method to update the current sprite image for the healthbar.

Public Attributes

- image

    This represents the sprite image for the Health_Bar object.
- h_sprite_sheet

    This represents the spritesheet for the image for the Health_Bar object, containing all sprites associated with the health bar.
- rect

    This represents rectangle for position for the sprite image of the Health_Bar object.

4.7.1 Detailed Description

This class represents the HealthBar for the user controlled player character.

The HealthBar class uses the base class for visible game objects from Pygame library.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 __init__()

```
def actor.healthbar.Health_Bar.__init__ (
              self,
              x,
              y )
```

Constructor HealthBar class.

Constructor for class initializes the x and y location of the HealthBar object.

Parameters

x	this represents the x-coordinate at which the HealthBar object will be drawn.
y	this represents the y-coordinate at which the HealthBar object will be drawn.

4.7.3 Member Function Documentation

4.7.3.1 health()

```
def actor.healthbar.Health_Bar.health (
              self,
              i )
```

Method to update the current sprite image for the healthbar.

This method will allow the HealthBar to update as soon as the player character is damaged/receives health.

Parameters

i	value represening the number of heart sprites to render to screen.

4.7.4 Member Data Documentation

4.7.4.1 h_sprite_sheet

```
actor.healthbar.Health_Bar.h_sprite_sheet
```

This represents the spritesheet for the image for the Health_Bar object, containing all sprites associated with the health bar.

4.7.4.2 image

```
actor.healthbar.Health_Bar.image
```

This represents the sprite image for the Health_Bar object.

4.7.4.3 rect

```
actor.healthbar.Health_Bar.rect
```

This represents rectangle for position for the sprite image of the Health_Bar object.

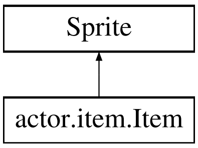
The documentation for this class was generated from the following file:

- src/actor/healthbar.py

4.8 actor.item.Item Class Reference

Consumable Item Class This class is used for the creation of a random consumable item, spawned once an enemy has been defeated on their last position before despawn.

Inheritance diagram for actor.item.Item:



Public Member Functions

- def __init__ (self, x, y, typ)

  Item constructor A sprite subclass constructor which takes a pair of x-y coordinates, commonly those of the killed enemy, and an item type, in the form of a randomly generated integer from 0 to the # of possible items.
- def collision (self, p)

  Collision handler for the player and the consumable item, depending on the item's type.

Public Attributes

- image

  Item sprite image.
- itemsprite

  List of possible item sprite images.
- rect

  Item x and y position.
- id

  Collision ID (how other items tell what this item is)
- type

  Integer value to determine what kind of item this is (rupee, heart, etc)

4.8.1 Detailed Description

Consumable Item Class This class is used for the creation of a random consumable item, spawned once an enemy has been defeated on their last position before despawn.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 __init__()

```
def actor.item.Item.__init__ (
              self,
              x,
              y,
              typ )
```

Item constructor A sprite subclass constructor which takes a pair of x-y coordinates, commonly those of the killed enemy, and an item type, in the form of a randomly generated integer from 0 to the # of possible items.

Parameters

x	X coordinate of the spawned item
y	Y coordinate of the spawned item
typ	Integer value to specify the type of consumable item

4.8.3 Member Function Documentation

4.8.3.1 collision()

```
def actor.item.Item.collision (
              self,
p )
```

Collision handler for the player and the consumable item, depending on the item's type.

Parameters

p	Player object

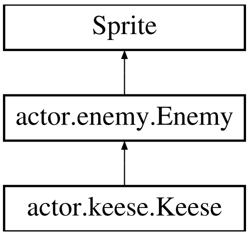
The documentation for this class was generated from the following file:

- src/actor/item.py

4.9 actor.keese.Keese Class Reference

This class represents the Keese enemy.

Inheritance diagram for actor.keese.Keese:



Public Member Functions

- def __init__ (self, x, y)

  Constructor for Keese.
- def genRestLength (self)

  Sets the Keese rest length.
- def genTravelPoint (self)

  Creates a new travel point for Keese.
- def switchSprite (self)

  Iterates through the sprite list.
- def stop (self)

  Stops the Keese.
- def setMoveSpeed (self)

  Sets the Keese x and y movement speed.
- def checkState (self)

  Evaluates the state of the Keese.
- def enemyLogic (self)

  Controls Keese logic.

Public Attributes

- isMoving

  Whether the Keese is moving.
- canStop

  Whether the Keese can stop.
- isResting

  Whether the Keees is resting.
- travelPoint

  Point the Keese is traveling to.
- restTime

  The amount of frames the Keese will rest for.
- restStartFrame

  The frame the Keese starts resting.
- flyStartFrame

  The frame the Keese starts flying on.
- spriteIndex

  The sprite list index.
- maxHP

  Keese max health.
- HP

  Keese current health.
- dmg

  Keese damage.
- image

  Keese current sprite.
- sprites

  Keese sprite list.
- xSpeed

  The current x-directional speed for Keese in movement/stationary state.
- ySpeed

  The current y-directional speed for Keese in movement/stationary state.
- hitCount

  Integer value representing the buffer for the number of hits for Keese after being hit by player character.
- isHit

  Represents the current state if Keese has collided with player character attack.

4.9.1 Detailed Description

This class represents the Keese enemy.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 __init__()

```
def actor.keese.Keese.__init__ (
              self,
              x,
              y )
```

Constructor for Keese.

Constructor takes two parameters, the x and y coordinates

Parameters

x	X coordinate of the starting postion of the Keese
y	Y coordinate of the starting postion of the Keese

4.9.3 Member Function Documentation

4.9.3.1 checkState()

```
def actor.keese.Keese.checkState (
              self )
```

Evaluates the state of the Keese.

Evalautes if the Keese if moving, if it can stop, if it is in iframes, and if it has died

4.9.3.2 enemyLogic()

```
def actor.keese.Keese.enemyLogic (
              self )
```

Controls Keese logic.

Uses the states to control the Keese

4.9.3.3 genRestLength()

```
def actor.keese.Keese.genRestLength (
              self )
```

Sets the Keese rest length.

Generates a random number between [1,2] inclusive as the rest time between movements and sets the rest start frame to the current frame

4.9.3.4 genTravelPoint()

```
def actor.keese.Keese.genTravelPoint (
              self )
```

Creates a new travel point for Keese.

Generate a random point to move to between 0 an the width - 30 of the screen for the x coordinate and between 0 and the height - 30 of the screen for the y coordinate

4.9.3.5 setMoveSpeed()

```
def actor.keese.Keese.setMoveSpeed (
              self )
```

Sets the Keese x and y movement speed.

Compares the two lengths of travel (x and y) and sets the speed in which ever direction is longer to the max speed and the other to a scalar multiple of the max speed based on the ratio of the two lengths

4.9.3.6 stop()

```
def actor.keese.Keese.stop (
              self )
```

Stops the Keese.

Sets the Keese speed in the x direction and the y direction to zero

4.9.3.7 switchSprite()

```
def actor.keese.Keese.switchSprite (
              self )
```

Iterates through the sprite list.

Swaps between the two sprites that are avalible for the keese

4.9.4 Member Data Documentation

4.9.4.1 hitCount

```
actor.keese.Keese.hitCount
```

Integer value representing the buffer for the number of hits for Keese after being hit by player character.

4.9.4.2 image

```
actor.keese.Keese.image
```

Keese current sprite.

Set Keese starting sprite.

4.9.4.3 isHit

```
actor.keese.Keese.isHit
```

Represents the current state if Keese has collided with player character attack.

4.9.4.4 xSpeed

```
actor.keese.Keese.xSpeed
```

The current x-directional speed for Keese in movement/stationary state.

4.9.4.5 ySpeed

```
actor.keese.Keese.ySpeed
```

The current y-directional speed for Keese in movement/stationary state.

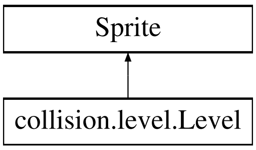
The documentation for this class was generated from the following file:

- src/actor/keese.py

4.10 collision.level.Level Class Reference

Dungeon level background class Creates the background of a dungeon, can be modified in the future to change colour based on sprite sheet and new arg.

Inheritance diagram for collision.level.Level:



Public Member Functions

- def __init__ (self)

  Background initializer Initialize a background to constantly be printed as game background.

Public Attributes

- image

    Level background sprite image.
- rect

    Background x and y coordinates (adjusted for the HUD display)

4.10.1 Detailed Description

Dungeon level background class Creates the background of a dungeon, can be modified in the future to change colour based on sprite sheet and new arg.

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

- src/collision/level.py

4.11 collision.levelmanager.LevelManager Class Reference

Dungeon Level Creation Class Class used as a master constructor for every dungeon levels, getting pre-written data and loading it when the game starts and when a transition occurs.

Public Member Functions

- def __init__ (self, spritelist, collidlist, updatelist)

    Dungeon Level Master Initializer The class initializer, taking empty lists, each for letting the make() function define what objects need to be drawn/collidable/updatable.
- def make (self, x, y)

    Dungeon Level Constructor The main function to load a given index of data from leveldata.py, and load it into the game environment.
- def transition (self, xchange, ychange)

    Dungeon level transition function, used to go to an adjacent level, clear all current data, and load new room data.
- def open (self)

    Function to open all blocked doors in the current dungeon room (usually used when all enemies defeated, locked doors stay locked)
- def endroom (self)

    Function to store current data, open all blocked doors, and possibly spawn a key/prize when all enemies defeated When all enemies in a room are killed, this function is called, changing the enemy array in leveldata.py for the current level to an empty array (so enemies don't reload once you enter again), running the self.open() function to open all blocked doors, and if the roomID is a predetermined key/rupy room, spawn a key/rupy in the middle of the room.
- def clear (self)

    Clears the levelmanager's update lists Empty all sprite lists for the game (spritelist, collisionlist, and updatelist), usually used in transition from one room to another.

Public Attributes

- sl

  List of objects to be printed (ie player, not invisible walls)
- cl

  List of objects that can be collided with (ie keese, not level background)
- ul

  List of objects that need to be constantly updated (ie player, not walls/static objects)
- x

  X-Coordinate of the level in leveldata's Room ID array (RID[y][x])
- y

  Y-Coordinate of the level in leveldata's Room ID array (RID[y][x])
- level

  Level background.
- RID

  Misc room spawnings (ie shop room)
- **enarray**
- **boss**
- **doors**
- **killed**

Static Public Attributes

- list **keyset** = [3, 7, 17]
- list **rupeset** = [1, 16, 12]
- LD = None

  List of all rooms to load (to be altered)

4.11.1 Detailed Description

Dungeon Level Creation Class Class used as a master constructor for every dungeon levels, getting pre-written data and loading it when the game starts and when a transition occurs.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 __init__()

```
def collision.levelmanager.LevelManager.__init__ (
                self,
                spritelist,
                collidlist,
                updatelist )
```

Dungeon Level Master Initializer The class initializer, taking empty lists, each for letting the make() function define what objects need to be drawn/collidable/updatable.

Parameters

spritelist	List of objects for the game to print (usually empty pygame.sprite.Group)
collidlist	List of objects which have a collision interaction with the player (usually empty pygame.sprite.Group)
updatelist	List of objects which need to be regularly updated (usually empty pygame.sprite.Group)

4.11.3 Member Function Documentation

4.11.3.1 make()

```
def collision.levelmanager.LevelManager.make (
              self,
              x,
              y )
```

Dungeon Level Constructor The main function to load a given index of data from leveldata.py, and load it into the game environment.

Parameters

i	Index of the room to be created

4.11.3.2 transition()

```
def collision.levelmanager.LevelManager.transition (
              self,
              xchange,
              ychange )
```

Dungeon level transition function, used to go to an adjacent level, clear all current data, and load new room data.

Parameters

j	Value to add to the current index to get to new room

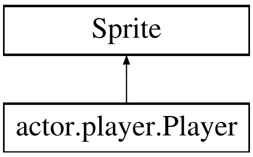
The documentation for this class was generated from the following file:

- src/collision/levelmanager.py

4.12 actor.player.Player Class Reference

Player Class A pygame sprite subclass for defining the creation of the game's playable character, as well as its interactions with both the user and other entities within the game.

Inheritance diagram for actor.player.Player:



Public Member Functions

- def __init__ (self, x, y, hud)

  Player constructor The constructor for the player object, used within the initialization of the game.
- def movechar (self)

  Function to move the player in a specified direction.
- def move (self, d, b)

  Function to store which movement buttons are being pressed and let go of by the user.
- def attack (self)

  Function to react to the user pressing the attack button, putting the player into an attack animation and spawning a sword for the duration.
- def useitem (self)

  Function to react to the user pressing the use item button, putting the player into an attack animation and spawning a boomerang for the duration.
- def moveupdate (self)

  UPDATE LOOP FUNCTIONS.
- def attackupdate (self)

  Update function for attack and boomerang conditions, as well as handing user collision with enemies (damage taken) Checks sword collision on enemies, and checks the timeout on both the sword and boomerang attack animations, telling the player when they can move again.
- def collisionupdate (self)

  Update function to check collisions between the player and any specified collidable objects, and call their respective collision events.
- def update (self)

  Player master update function, running all update functions, repeated in main file loop.

Public Attributes

- image

  Player sprite image.
- id

  Collision ID (To tell other objects what this object is)
- leveltrans

  Boolean value to tell the game when the player has left the room, and a room transition is needed.
- rect

  Player x and y position.
- walkLeft

  Animation sprites for player walking left.
- walkRight

  Animation sprites for player walking right.
- walkUp

  Animation sprites for player walking up.

- walkDown

  Animation sprites for player walking down.
- attacksprite

  Animation sprites for player attacking.
- obj

  List of objects that the player can collide with.
- hbar

  Health bar corresponding to the player's health.
- **rbar**
- **kbar**
- totalhp

  Player's maximum HP amount.
- hp

  Player's current HP amount (starts at max)
- rupes

  Player's rupee counter.
- keys

  Player's key counter.
- **collision**
- **collidbuy**
- **doorcount**
- **spawning**
- **spawncount**
- moveable

  Boolean allowing and stoping the player from moving.
- attackbool

  Boolean to tell whether the player is attacking or not.
- attackcount

  Attack counter, to control how long the attack lasts.
- attacksword

  Holder for a sword object when attacking.
- item

  Holder for a boomerang object when using item.
- itembool

  Boolean to tell whether the player is using their boomerang or not.
- oldx

  Player's x value one frame ago.
- oldy

  Player's y value one frame ago.
- dirbool

  Boolean array to tell which directional buttons are pressed or not (starting left going clockwise, dirbool[0] is left, dirbool[1] is up, etc)
- dir

  Current direction the player is facing.
- hit

  Boolean on whether the player was recently hit.
- hitcount

  How long a hit counts for on the player (how long until self.hit turns off)
- **speed**
- **debug**
- hasWon

  Has the player won or not.

4.12.1 Detailed Description

Player Class A pygame sprite subclass for defining the creation of the game's playable character, as well as its interactions with both the user and other entities within the game.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 __init__()

```
def actor.player.Player.__init__ (
              self,
              x,
              y,
              hud )
```

Player constructor The constructor for the player object, used within the initialization of the game.

The only arguments passed are an initial x and y position, as well as heads-up-display (HUD) elements.

Parameters

x	Initial player x coordinate
y	Initial player y coordinate
hud	Array of HUD elements which act directly with the user's tracked values (health bar, rupy/key count)

4.12.3 Member Function Documentation

4.12.3.1 attackupdate()

```
def actor.player.Player.attackupdate (
              self )
```

Update function for attack and boomerang conditions, as well as handing user collision with enemies (damage taken) Checks sword collision on enemies, and checks the timeout on both the sword and boomerang attack animations, telling the player when they can move again.

Also updates the player and reacts accordingly to recieving damage.

4.12.3.2 move()

```
def actor.player.Player.move (
              self,
              d,
              b )
```

Function to store which movement buttons are being pressed and let go of by the user.

Parameters

d	Direction related to the button pressed (integer value from 0 to 3, starting from the left and going clockwise)
b	Boolean value on whether the button is pressed or not

4.12.3.3 moveupdate()

```
def actor.player.Player.moveupdate (
              self )
```

UPDATE LOOP FUNCTIONS.

Update function for player movement and according sprite animation

4.12.4 Member Data Documentation

4.12.4.1 attacksword

```
actor.player.Player.attacksword
```

Holder for a sword object when attacking.

Sprite function to delete a sprite object and all relations of the object in the project.

4.12.4.2 dir

```
actor.player.Player.dir
```

Current direction the player is facing.

Render Appropriate Sprites According to movement.

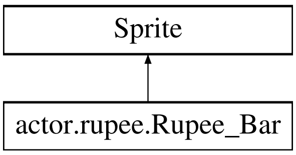
The documentation for this class was generated from the following file:

- src/actor/player.py

4.13 actor.rupee.Rupee_Bar Class Reference

This class represents the RupeeBar object.

Inheritance diagram for actor.rupee.Rupee_Bar:



Public Member Functions

- def __init__ (self, x, y)

  Constructor for the RupeeBar object.

Public Attributes

- image

  This represents the sprite image of the Rupee_Bar object.
- rect

  This represents rectangle for collision and position for the sprite image of the Rupee_Bar object.

4.13.1 Detailed Description

This class represents the RupeeBar object.

The Rupee class uses the Pygame library and SpriteSheet module to create an image for the

4.13.2 Constructor & Destructor Documentation

4.13.2.1 __init__()

```
def actor.rupee.Rupee_Bar.__init__ (
              self,
              x,
              y )
```

Constructor for the RupeeBar object.

Constructor for class initializes the x and y location of the RupeeBar object.

Parameters

x	this represents the x-coordinate at which the RupeeBar object will be drawn.
y	this represents the y-coordinate at which the RupeehBar object will be drawn.

4.13.3 Member Data Documentation

4.13.3.1 image

```
actor.rupee.Rupee_Bar.image
```

This represents the sprite image of the Rupee_Bar object.

4.13.3.2 rect

```
actor.rupee.Rupee_Bar.rect
```

This represents rectangle for collision and position for the sprite image of the Rupee_Bar object.

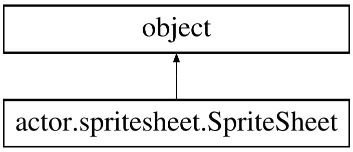
The documentation for this class was generated from the following file:

- src/actor/rupee.py

4.14 actor.spritesheet.SpriteSheet Class Reference

This class represents the SpriteSheet object, allowing sprites to be loaded and processed.

Inheritance diagram for actor.spritesheet.SpriteSheet:



Public Member Functions

- def __init__ (self, file_name)

  Constructor for the SpriteSheet class.
- def get_image (self, x, y, width, height)

  Accessor to return an image splice from the loaded SpriteSheet.
- def get_imageNT (self, x, y, width, height)

  Accessor to return an image splice from the loaded SpriteSheet, WITHOUT removing the image background.

Public Attributes

- sprite_sheet

  This represents the current spritesheet loaded from the specific file_name path.

4.14.1 Detailed Description

This class represents the SpriteSheet object, allowing sprites to be loaded and processed.

The SpriteSheet class uses the Pygame library to load images and change the transperancy on those images.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 __init__()

```
def actor.spritesheet.SpriteSheet.__init__ (
              self,
              file_name )
```

Constructor for the SpriteSheet class.

This constructor initializes an image file using the Pygame library.

Parameters

file_name	This is the string representing the path to the image file.

4.14.3 Member Function Documentation

4.14.3.1 get_image()

```
def actor.spritesheet.SpriteSheet.get_image (
              self,
              x,
              y,
              width,
              height )
```

Accessor to return an image splice from the loaded SpriteSheet.

This accessor returns an image splice based on the x,y location and the height and width of the image.

Parameters

x	The x-coordinate for starting point of the splice on the SpriteSheet.
y	The y-coordinate for starting point of the splice on the SpriteSheet.
width	The width of the splice on the SpriteSheet.
height	The height of the splice on the SpriteSheet.

Returns

image This returns the newly spliced image after removing the image background transparency.

4.14.3.2 get_imageNT()

```
def actor.spritesheet.SpriteSheet.get_imageNT (
              self,
              x,
              y,
              width,
              height )
```

Accessor to return an image splice from the loaded SpriteSheet, WITHOUT removing the image background.

This accessor returns an image splice based on the x,y location and the height and width of the image.

Parameters

x	The x-coordinate for starting point of the splice on the SpriteSheet.
y	The y-coordinate for starting point of the splice on the SpriteSheet.
width	The width of the splice on the SpriteSheet.
height	The height of the splice on the SpriteSheet.

Returns

image This returns the newly spliced image WITHOUT removing the image background transparency.

4.14.4 Member Data Documentation

4.14.4.1 sprite_sheet

```
actor.spritesheet.SpriteSheet.sprite_sheet
```

This represents the current spritesheet loaded from the specific file_name path.

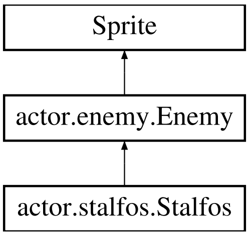
The documentation for this class was generated from the following file:

- src/actor/spritesheet.py

4.15 actor.stalfos.Stalfos Class Reference

This class represents the Stalfos enemy.

Inheritance diagram for actor.stalfos.Stalfos:



Public Member Functions

- def __init__ (self, x, y)

  Constructor for Stalfos.
- def checkState (self)

  Evaluates the state of the Stalfos.
- def enemyLogic (self)

  Controls Stalfos logic.
- def genTravelPath (self)

  Creates a new travel point for Stalfos.
- def setWalkSpeed (self)

  Sets the walk speed of the Stalfos.
- def stop (self)

  Stops the Stalfos.

Public Attributes

- isMoving

  Call superclass constructor.
- previousDirection

  Movement integer value for previous direction of movement for Stalfos state.
- direction

  Movement integer value for current direction of movement for Stalfos state.
- walkFrames

  Integer representing the frames walked by Staflos in movement state.
- walkStartFrame

  Integer value for movement frame for Stalfos during movement state.
- oldx

  This represents the previous x-location of Stalfos in movement/stationary state.
- oldy

  This represents the previous y-location of Stalfos in movement/stationary state.
- obj

  This represents the list of objects Stalfos can collide with, in movement/stationary state.
- maxHP

  The integer value setting the Stalfos health to maximum health.
- HP

  The integer value representing Current Stalfos health.
- dmg

  The integer value representing the current damage value that the Stalfos object has received from player character.
- image

  Spritesheet for the acessing of the Stalfos sprite image.
- sprites

  Creates a list of sprites list for Stalfos.
- spriteIndex

  Represents the integer value for the current index in the sprite list for Stalfos.
- hitdir

  Represents the integer value (0,1,2,3) direction Stalfos is hit in by player character attack.
- hitCount

  Integer value representing the buffer for the number of hits for the Enemy after being hit by player character.
- isHit

Represents the current state if Stalfos has collided with player character attack.

- xSpeed

  The current x-directional speed for Stalfos in movement/stationary state.

- ySpeed

  The current y-directional speed for Stalfos in movement/stationary state.

4.15.1 Detailed Description

This class represents the Stalfos enemy.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 __init__()

```
def actor.stalfos.Stalfos.__init__ (
              self,
              x,
              y )
```

Constructor for Stalfos.

Constructor takes two parameters, the x and y coordinates

Parameters

x	X coordinate of the starting postion of the Keese
y	Y coordinate of the starting postion of the Keese

4.15.3 Member Function Documentation

4.15.3.1 checkState()

```
def actor.stalfos.Stalfos.checkState (
              self )
```

Evaluates the state of the Stalfos.

Evalautes if the Stalfos if moving, if it can stop, if it is in iframes, if it collides with something, and if it has died

4.15.3.2 enemyLogic()

```
def actor.stalfos.Stalfos.enemyLogic (
              self )
```

Controls Stalfos logic.

Uses the states to control the Stalfos

4.15.3.3 genTravelPath()

```
def actor.stalfos.Stalfos.genTravelPath (
              self )
```

Creates a new travel point for Stalfos.

Generates a direction to walk and the distance to move

4.15.3.4 setWalkSpeed()

```
def actor.stalfos.Stalfos.setWalkSpeed (
              self )
```

Sets the walk speed of the Stalfos.

Sets speed based on the direction

4.15.3.5 stop()

```
def actor.stalfos.Stalfos.stop (
              self )
```

Stops the Stalfos.

Sets the Stalfos speed in the x direction and the y direction to zero

4.15.4 Member Data Documentation

4.15.4.1 direction

```
actor.stalfos.Stalfos.direction
```

Movement integer value for current direction of movement for Stalfos state.

4.15.4.2 dmg

```
actor.stalfos.Stalfos.dmg
```

The integer value representing the current damage value that the Stalfos object has received from player character.

4.15.4.3 hitCount

```
actor.stalfos.Stalfos.hitCount
```

Integer value representing the buffer for the number of hits for the Enemy after being hit by player character.

4.15.4.4 hitdir

```
actor.stalfos.Stalfos.hitdir
```

Represents the integer value (0,1,2,3) direction Stalfos is hit in by player character attack.

4.15.4.5 HP

```
actor.stalfos.Stalfos.HP
```

The integer value representing Current Stalfos health.

4.15.4.6 image

```
actor.stalfos.Stalfos.image
```

Spritesheet for the acessing of the Stalfos sprite image.

Set Stalfos starting sprite.

Sprite image for the Stalfos object.

4.15.4.7 isHit

```
actor.stalfos.Stalfos.isHit
```

Represents the current state if Stalfos has collided with player character attack.

4.15.4.8 isMoving

```
actor.stalfos.Stalfos.isMoving
```

Call superclass constructor.

Check for movement.

Boolean values for keeping track of the Stalfos state.

4.15.4.9 maxHP

```
actor.stalfos.Stalfos.maxHP
```

The integer value setting the Stalfos health to maximum health.

4.15.4.10 obj

```
actor.stalfos.Stalfos.obj
```

This represents the list of objects Stalfos can collide with, in movement/stationary state.

4.15.4.11 oldx

```
actor.stalfos.Stalfos.oldx
```

This represents the previous x-location of Stalfos in movement/stationary state.

4.15.4.12 oldy

```
actor.stalfos.Stalfos.oldy
```

This represents the previous y-location of Stalfos in movement/stationary state.

4.15.4.13 previousDirection

```
actor.stalfos.Stalfos.previousDirection
```

Movement integer value for previous direction of movement for Stalfos state.

4.15.4.14 spriteIndex

```
actor.stalfos.Stalfos.spriteIndex
```

Represents the integer value for the current index in the sprite list for Stalfos.

4.15.4.15 sprites

```
actor.stalfos.Stalfos.sprites
```

Creates a list of sprites list for Stalfos.

4.15.4.16 walkFrames

```
actor.stalfos.Stalfos.walkFrames
```

Integer representing the frames walked by Staflos in movement state.

4.15.4.17 walkStartFrame

```
actor.stalfos.Stalfos.walkStartFrame
```

Integer value for movement frame for Stalfos during movement state.

4.15.4.18 xSpeed

```
actor.stalfos.Stalfos.xSpeed
```

The current x-directional speed for Stalfos in movement/stationary state.

4.15.4.19 ySpeed

```
actor.stalfos.Stalfos.ySpeed
```

The current y-directional speed for Stalfos in movement/stationary state.

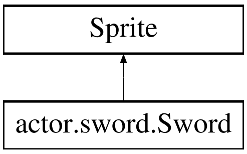
The documentation for this class was generated from the following file:

- src/actor/stalfos.py

4.16 actor.sword.Sword Class Reference

Player Sword Class Class for the creation and deletion of the sword sprite object, made when the player attacks.

Inheritance diagram for actor.sword.Sword:



Public Member Functions

- def __init__ (self, x, y, direction)

  Sword constructor, taking an x and y coordinate, and the direction for the sword to be pointing (player direction)

Public Attributes

- sprite

  Array of all possible sword directions (following usual directional standards, 0-3, starting at left going clockwise)
- image

  Sprite image of the sword.
- rect

  X and Y position of the sword.

4.16.1 Detailed Description

Player Sword Class Class for the creation and deletion of the sword sprite object, made when the player attacks.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 __init__()

```
def actor.sword.Sword.__init__ (
              self,
              x,
              y,
              direction )
```

Sword constructor, taking an x and y coordinate, and the direction for the sword to be pointing (player direction)

Parameters

x	Sword's x coordinate
y	Sword's y coordinate
direction	Sword's direction (integer from 0 to 3, starting left, going clockwise)

4.16.3 Member Data Documentation

4.16.3.1 image

```
actor.sword.Sword.image
```

Sprite image of the sword.

Below is for testing self.image = pygame.Surface([26, 14]) self.image.fill((200, 0, 0))

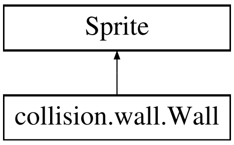
The documentation for this class was generated from the following file:

- src/actor/sword.py

4.17 collision.wall.Wall Class Reference

This class represents the Wall class for collision for objects in the environment.

Inheritance diagram for collision.wall.Wall:



Public Member Functions

- def __init__ (self, x, y, w, h, sprite)

  Constructor for the Wall class.
- def collision (self, i)

  This method checks for collision with the Wall object and other sprite object.

Public Attributes

- image

  This represents the sprite image for Wall object.
- rect

  This represents rectangle for position for the sprite image of the Wall object.
- id

  *This represents the ID for the Wall object, ised for collision in **main.py**.*

4.17.1 Detailed Description

This class represents the Wall class for collision for objects in the environment.

The Wall class uses the base class for visible game objects from Pygame library.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 __init__()

```
def collision.wall.Wall.__init__ (
              self,
              x,
              y,
              w,
              h,
              sprite )
```

Constructor for the Wall class.

Constructor for Wall class initializes a Wall object based on the x/y location and the width/height of the wall, and the sprite the collision for the wall will be created upon.

Parameters

x	This represents the integer value for the x-location for the Wall object to be created.
y	This represents the integer value for the y-location for the Wall object to be created.
w	This represents the integer value for the width of the Wall object when created.
h	This represents the integer value for the height of the Wall object when created.
sprite	This represents the sprite that the collison for a Wall will be present upon at all times.

4.17.3 Member Function Documentation

4.17.3.1 collision()

```
def collision.wall.Wall.collision (
              self,
              i )
```

This method checks for collision with the Wall object and other sprite object.

The collision will be checked with the Wall and other sprite object as the object collides with the wall.

Parameters

i	This is the sprite object that is passed into the method, and checks if the object is colliding with the Wall object, reseting the sprite objects location accordingly.

4.17.4 Member Data Documentation

4.17.4.1 id

```
collision.wall.Wall.id
```

This represents the ID for the Wall object, ised for collision in **main.py**.

4.17.4.2 image

```
collision.wall.Wall.image
```

This represents the sprite image for Wall object.

4.17.4.3 rect

```
collision.wall.Wall.rect
```

This represents rectangle for position for the sprite image of the Wall object.

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

- src/collision/wall.py

Chapter 5

File Documentation

5.1 src/actor/aquamentus.py File Reference

Aquamentus Boss

Classes

- class actor.aquamentus.Aquamentus

  This class represents the Aquamentus Boss.

Functions

- def actor.aquamentus.populateSprites ()

  Creates the sprite list for Aquamentus.

Variables

- string **actor.aquamentus.SPRITE_MAP** = 'src/actor/sprites/aquamentus.png'

5.1.1 Detailed Description

Aquamentus Boss

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 8 2018

5.1.2 Function Documentation

5.1.2.1 populateSprites()

```
def actor.aquamentus.populateSprites ( )
```

Creates the sprite list for Aquamentus.

Iterates through a sprite sheet to pull images for the sprite array

Returns

sprites Array if the sprites that represent the Aquamentus

5.2 src/actor/boomerang.py File Reference

Boomerang Weapon

Classes

- class actor.boomerang.Boomerang

  Boomerang Weapon Class The class holding the creation, behaviour, and collision effects of the player's boomerang weapon.

Functions

- def actor.boomerang.populateSprites ()

  Creates the sprite list for Boomerang.
- def actor.boomerang.stuncoeffient (x)

  Finds the coeffient for the stun value Uses a quadradic equation with the frame counter to find the coefficent.

Variables

- string **actor.boomerang.SPRITE_MAP** = 'src/actor/sprites/boomerang.png'

5.2.1 Detailed Description

Boomerang Weapon

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 6 2018

5.2.2 Function Documentation

5.2.2.1 populateSprites()

```
def actor.boomerang.populateSprites ( )
```

Creates the sprite list for Boomerang.

Iterates through a sprite sheet to pull images for the sprite array

Returns

sprites Array if the sprites that represent the Boomerang

5.2.2.2 stuncoeffient()

```
def actor.boomerang.stuncoeffient (
              x )
```

Finds the coeffient for the stun value Uses a quadradic equation with the frame counter to find the coefficent.

Parameters

x	The current frame of the boomerang

Returns

c The coefficent for the boomerang stun

5.3 src/actor/boss.py File Reference

Boss Template

Classes

- class actor.boss.Boss

  Superclass for representing a Boss.

5.3.1 Detailed Description

Boss Template

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.4 src/actor/constants.py File Reference

Actor Constants

Variables

- tuple actor.constants.BLACK = (0,0,0)

  Defines the colour black.
- tuple actor.constants.WHITE = (255, 255, 255)

  Defines the colour white.
- tuple actor.constants.PURPLE = (255, 50, 255)

  Defines the colour purple.
- tuple actor.constants.RED = (255, 50, 50)

  Defines the colour red.
- tuple actor.constants.BLUE = (0,0,255)

  Defines the color blue.
- int actor.constants.PLAYER_WIDTH = 15

  Defines the player's width.
- int actor.constants.PLAYER_HEIGHT = 15

  Defines the player's height.
- int actor.constants.PLAYER_SPEED = 2

  Defines the player's speed.
- int actor.constants.PLAYER_MAX_HP = 3

  Defines the player's max hp.
- int actor.constants.PLAYER_DOORCOUNT = 20

  Defines how many frames the player collides with a locked door before a key is used.
- int actor.constants.PLAYER_SPAWNCOUNT = 8

  Defines how long the player is uncontrollable for when entering a new room (movement to walk out of door)
- int actor.constants.HIT_SPEED = 12

  Defines the speed of the knock back applied per frame.
- int actor.constants.HIT_TIME = 5

  Defines the number of frames hit speed is applied for.
- int actor.constants.HIT_IFRAME = 30

  Defines the number of iframes.
- int actor.constants.ATK_WIDTH = 15

  Defines the width of the attack hitbox.

- int actor.constants.ATK_HEIGHT = 15

  Defines the height of the attack hitbox.
- int actor.constants.ATK_LENGTH = 10

  Defines the number of frames the hit stays out for.
- int actor.constants.ATK_BUFFER = 20

  Defines the number of frames before the next attack can start.
- int actor.constants.BOOM_SPEED = 9

  Defines the bommerang speed.
- float actor.constants.BOOM_RETURN = 0.3

  Defines the acceleration in the opposite direction of the bommerang's travel path.
- int actor.constants.GLOBAL_FRAME_BUFFER = 12

  Coordinate Divisor that determines range in which next sprite should render.
- int actor.constants.KEESE_MAX_HP = 1

  Defines the keese's max hp.
- float actor.constants.KEESE_DMG = 0.5

  Defines the keese's damage.
- int actor.constants.KEESE_MAX_SPEED = 2

  Defines the keese's max speed.
- int actor.constants.KEESE_MIN_SPEED = 1

  Defines the keese's min speed.
- int actor.constants.ACCEPTABLE_RADIUS = 20

  Defines the keese's minimum magnitude from its determined travel point before stopping.
- int actor.constants.KEESE_MAGNITUDE_MIN = 200

  Defines the keese's minimum magnitude for selecting its next travel point.
- int actor.constants.STALFOS_MAX_HP = 2

  Defines the stalfos' max hp.
- int actor.constants.STALFOS_DMG = 1

  Defines the stalfos' damage.
- int actor.constants.STALFOS_SPEED = 1

  Defines the stalfos' speed.
- int actor.constants.STALFOS_HIT_SPEED = 10

  Defines the stalfos' speed during knockback.
- int actor.constants.ENEMY_STUNCOUNT = 35

  Length of frames an enemy will be stunned for when hit by a boomerang.
- int actor.constants.AQUAMENTUS_MAX_HP = 5

  Defines the aquamentus' max hp.
- int actor.constants.AQUAMENTUS_DMG = 2

  Defines the aquamentus' damage.
- int actor.constants.AQUAMENTUS_SPEED = 1

  Defines the aquamentus' speed.
- int actor.constants.FIREBALL_DMG = 1

  Defines the fireballs damage.

5.4.1 Detailed Description

Actor Constants

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 9 2018

5.5 src/actor/enemy.py File Reference

Enemy Template

Classes

- class actor.enemy.Enemy

  Superclass for representing an Enemy.

5.5.1 Detailed Description

Enemy Template

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.6 src/actor/fireball.py File Reference

Fireball object

Classes

- class actor.fireball.Fireball

  This class represents the Fireball object.

Functions

- def actor.fireball.populateSprites ()

  Creates the sprite list for Fireball.

Variables

- string **actor.fireball.SPRITE_MAP** = 'src/actor/sprites/fireball.png'

5.6.1 Detailed Description

Fireball object

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.6.2 Function Documentation

5.6.2.1 populateSprites()

```
def actor.fireball.populateSprites ( )
```

Creates the sprite list for Fireball.

Iterates through a sprite sheet to pull images for the sprite array

Returns

sprites Array if the sprites that represent the Fireball

5.7 src/actor/healthbar.py File Reference

HealthBar Class

Classes

- class actor.healthbar.Health_Bar

  This class represents the HealthBar for the user controlled player character.

5.7.1 Detailed Description

HealthBar Class

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.8 src/actor/item.py File Reference

Consumable Items

Classes

- class actor.item.Item

  Consumable Item Class This class is used for the creation of a random consumable item, spawned once an enemy has been defeated on their last position before despawn.

5.8.1 Detailed Description

Consumable Items

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 6 2018

5.9 src/actor/keese.py File Reference

Keese Enemy

Classes

- class actor.keese.Keese

  This class represents the Keese enemy.

Functions

- def actor.keese.populateSprites ()

  Creates the sprite list for Keese.

Variables

- string **actor.keese.SPRITE_MAP** = 'src/actor/sprites/keese.png'

5.9.1 Detailed Description

Keese Enemy

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.9.2 Function Documentation

5.9.2.1 populateSprites()

```
def actor.keese.populateSprites ( )
```

Creates the sprite list for Keese.

Iterates through a sprite sheet to pull images for the sprite array

Returns

sprites Array if the sprites that represent the Keese

5.10 src/actor/player.py File Reference

Playable Characer

Classes

- class actor.player.Player

  Player Class A pygame sprite subclass for defining the creation of the game's playable character, as well as its interactions with both the user and other entities within the game.

5.10.1 Detailed Description

Playable Characer

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 6 2018

5.11 src/actor/rupee.py File Reference

Rupee_Bar Class

Classes

- class actor.rupee.Rupee_Bar

  This class represents the RupeeBar object.

5.11.1 Detailed Description

Rupee_Bar Class

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.12 src/actor/spritesheet.py File Reference

SpriteSheet Class

Classes

- class actor.spritesheet.SpriteSheet

  This class represents the SpriteSheet object, allowing sprites to be loaded and processed.

5.12.1 Detailed Description

SpriteSheet Class

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.13 src/actor/stalfos.py File Reference

Stalfos Enemy

Classes

- class actor.stalfos.Stalfos

  This class represents the Stalfos enemy.

Functions

- def actor.stalfos.populateSprites ()

  Creates the sprite list for Stalfos.

Variables

- string **actor.stalfos.SPRITE_MAP** = 'src/actor/sprites/stalfos.png'

5.13.1 Detailed Description

Stalfos Enemy

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.13.2 Function Documentation

5.13.2.1 populateSprites()

```
def actor.stalfos.populateSprites ( )
```

Creates the sprite list for Stalfos.

Iterates through a sprite sheet to pull images for the sprite array

Returns

sprites Array if the sprites that represent the Stalfos

5.14 src/actor/sword.py File Reference

Player Sword

Classes

- class actor.sword.Sword

  Player Sword Class Class for the creation and deletion of the sword sprite object, made when the player attacks.

5.14.1 Detailed Description

Player Sword

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 8 2018

5.15 src/collision/door.py File Reference

Dungeon Door

Classes

- class collision.door.Door

  Dungeon Door Class This class is used to place a door on one of four places of a dungeon room, to allow the player to walk through and traverse to other rooms within the dungeon.

5.15.1 Detailed Description

Dungeon Door

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.16 src/collision/level.py File Reference

Dungeon Background

Classes

- class collision.level.Level

  Dungeon level background class Creates the background of a dungeon, can be modified in the future to change colour based on sprite sheet and new arg.

5.16.1 Detailed Description

Dungeon Background

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.17 src/collision/levelmanager.py File Reference

Dungeon Level Master Creator

Classes

- class collision.levelmanager.LevelManager

  Dungeon Level Creation Class Class used as a master constructor for every dungeon levels, getting pre-written data and loading it when the game starts and when a transition occurs.

5.17.1 Detailed Description

Dungeon Level Master Creator

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.18 src/collision/wall.py File Reference

Wall Class

Classes

- class collision.wall.Wall

  This class represents the Wall class for collision for objects in the environment.

5.18.1 Detailed Description

Wall Class

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.19 src/config/colour.py File Reference

Colour Constants

Variables

- tuple config.colour.WHITE = (255, 255, 255)

  Defines the colour white.

5.19.1 Detailed Description

Colour Constants

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

5.20 src/config/window.py File Reference

Window Constants

Variables

- int config.window.Y_OFFSET = 56

  Y offset for HUD.
- int config.window.Wwidth = 480

  Width of the window.
- int config.window.Wheight = 320 + Y_OFFSET

  Height of the window.

5.20.1 Detailed Description

Window Constants

Author

Giacomo Loparco, Bilal Jaffry, Lucas Zacharewicz

Date

November 7 2018

Index