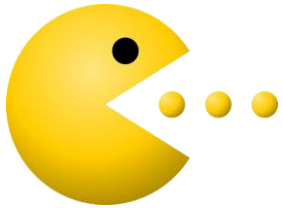
Analysis

Methods

```
import random

class Player:
    max_x = 450
    max_y = 450

    def __init__(self, x, y, character):
        self._x = x
        self._y = y
        self._num_lives = 10 # Initial value
        self._character = character

class Candy:
    speed = 40

    def __init__(self, x, y, type_of_movement="horizontal"):
        self._x = x
        self._y = y
        self._type_of_movement = type_of_movement

class Enemy:
    max_x = 450
    max_y = 450

    def __init__(self, x, y, speed, type_of_movement="vertical",):
        self._x = random.randint(0, max_x)
        self._y = random.randint(0, max_y)
        self._type_of_movement = type_of_movement
        self._num_lives = 15 # Initial value
        self._speed = speed
```

Methods

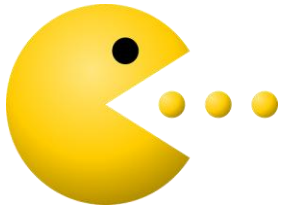
Player



- The player has to be able to move up, move down, move left, and move right as long as it is not beyond the boundaries of the window (0, 450) both horizontally and vertically.
- The player initially has 10 lives and it displays a welcome message when the game starts.
- The player has a specific character assigned.
- The player is able to shoot candy. This candy acts like a bullet.v

Methods

Player



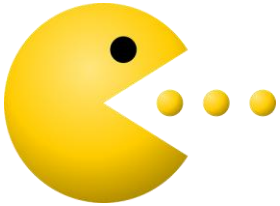
- The player has to be able to move up, move down, move left, and move right as long as it is not beyond the boundaries of the window (0, 450) both horizontally and vertically.
- The player initially has 10 lives and it displays a welcome message when the game starts.
- The player has a specific character assigned.
- The player is able to shoot candy. This candy acts like a bullet.

Methods

Other

- When a player collides with an enemy, the player loses one life.
- When the candy shot by the player hits an enemy, the enemy loses one life.

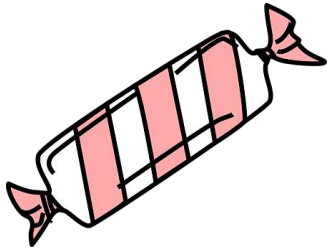
Methods



Player

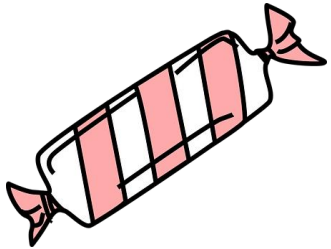
- move_up
- move_down
- move_left
- move_right
- display_welcome
- shoot_candy
- lose_life

Candy



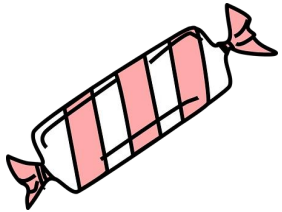
- The player is able to shoot candy.
- Candy acts like a bullet in the game. It moves either horizontally or vertically. This is determined when the instance is created.
- The speed of the candy has to be a specific number within a range from 5 to 45.

Candy



- The player is able to shoot candy.
- Candy acts like a bullet in the game. It moves either horizontally or vertically. This is determined when the instance is created.
- The speed of the candy has to be a specific number within a range from 5 to 45.

Candy



- **move** - update x or y coordinates based on the current type of movement and speed.
- **is_beyond_boundaries**

Methods

Enemy

- Enemies are created at random locations, so their initial x coordinates and y coordinates are randomly generated integers in a range from 0 to 450.
- Enemies have a fixed direction of movement (vertical or horizontal).
- When an enemy reaches the end of the screen (0 or 450 vertically or horizontally), it changes direction (if the previous direction was vertical, it will now move horizontally and vice versa).
- The speed of the enemies depends on the difficulty of the game selected by the human player, and it is determined when the instances are created.
- Enemies initially have 15 lives.

Methods

Enemy

- Enemies are created at random locations, so their initial x coordinates and y coordinates are randomly generated integers in a range from 0 to 450.
- Enemies have a fixed direction of movement (vertical or horizontal).
- When an enemy reaches the end of the screen (0 or 450 vertically or horizontally), it changes direction (if the previous direction was vertical, it will now move horizontally and vice versa).
- The speed of the enemies depends on the difficulty of the game selected by the human player, and it is determined when the instances are created.
- Enemies initially have 15 lives.

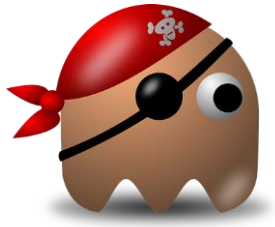
Methods

Other

- When a player collides with an enemy, the player loses one life.
- When the candy shot by the player hits an enemy, the enemy loses one life.

Methods

Enemy



- **move** – according to the current direction
- **change_type_of_movement**
- **lose_life**

```python
class Player:
    max_x = 450
    max_y = 450

    # Added to move the Player
    speed = 20

    def __init__(self, x, y, character):
        self._x = x
        self._y = y
        self._num_lives = 10 # Initial value
        self._character = character

    def move_up(self):
        self._y += Player.speed

    def move_down(self):
        self._y -= Player.speed

    def move_left(self):
        self._x -= Player.speed

    def move_right(self):
        self._x += Player.speed

    def display_welcome(self, message="Welcome to the game"):
        print(message)

    def shoot_candy(self):
        return Candy(self._x, self._y)

    def lose_life(self):
        self._num_lives -= 1
```

```python
class Candy:
    speed = 40

    def __init__(self, x, y, type_of_movement="horizontal"):
        self._x = x
        self._y = y
        self._type_of_movement = type_of_movement

    def move(self):
        if self._type_of_movement == "horizontal":
            self._x += Candy.speed
        else:
            self._y += Candy.speed

    def is_beyond_boundaries(self):
        return  (self._x > 450 or self._x < 0) or (self._y > 450 or self._y < 0)
```

```python
class Enemy:
    max_x = 450
    max_y = 450

    def __init__(self, x, y, speed, type_of_movement="vertical",):
        self._x = random.randint(0, max_x)
        self._y = random.randint(0, max_y)
        self._type_of_movement = type_of_movement
        self._num_lives = 15 # Initial value
        self._speed = speed

    def move(self):
        if self._type_of_movement == "vertical":
            self._y += self._speed
        else:
            self._x += self._speed

    def change_type_of_movement(self):
        if self._type_of_movement == "vertical":
            self._type_of_movement = "horizontal"
        else:
            self._type_of_movement = "vertical"

    def lose_life(self):
        if self._num_lives > 0:
            self._num_lives -= 1
```

Result

Methods