Dokumentacja wstępna

Temat:

Gra Python - pygame floppybird

Skład grupy:

Aleksander Hybś, Szymon Romanowski, Marcin Matysiak

Opis projektu

Gra komputerowa wykonana w języku programowania Python z wykorzystaniem biblioteki pygame. Celem naszej gry jest uniknięcie jak największej liczby przeciwników. Za każdego przeciwnika którego ominiemy są przyznawane punkty a najwyższy wynik będzie zapisywany. Jest to gra w stylu „endless runner”.

Spis klas

```
class Game:
    def __init__(self):
        self._running
        self._display_surf
        self._size
        self._background_image
        self._enemies
        self._enemy_id
        self._player
        self._laderboard
        self._menu
        self._ui

      def on_init(self):

      def on_event(self, event):

      def on_loop(self):

      def on_render(self):

      def on_reset(self):

      def on_cleanup(self):

      def on_execute(self):
```

```
class Menu:
    def __init__(self, screen):
        self.screen
        self.font
        self.text
        self.ismenuon
        self.surface

      def render(self):

      def onloop(self, isplayerdead):

      def click(self):

      def ismenuon(self):
```

```
class Score_system:
    def __init__(self):
        self._score
        self._hiscore

      def load_hiscore(self):

      def save_hiscore(self):

      def set_hiscore(self):

      def score_update(self, points):

      def score_reset(self):
```

```
class Ui:
    def __init__(self, screen):
        self.screen
        self.font
        self.score
        self.hiscore

      def render(self, score, hiscore):
```

```
class Player:

    def __init__(self):
        self.pos
        self.vel
        self.acc
        self.isdead

      def force(self,Force):

      def addforce(self):

      def tick(self):

      def draw(self,screen):

      def collision(self,enemy):

      def isDead(self):

      def click(self):
```

```
class Enemy:
    def __init__(self):
        self._enemy_image
        self._speed
        self._points
        self._x_pos
        self._y_pos
        self._starting_pos

      def renderEnemy(self, display_surf):

      def enemyMovment(self):

      def resetPosition(self):

      def isOutOfBoundry(self):

      def returnPoints(self):
```

```
class Bird(Enemy):
    def __init__(self):
        super().__init__()
        self._speed
        self._y_speed
        self._points
        self._enemy_image

      def enemyMovment(self):
```

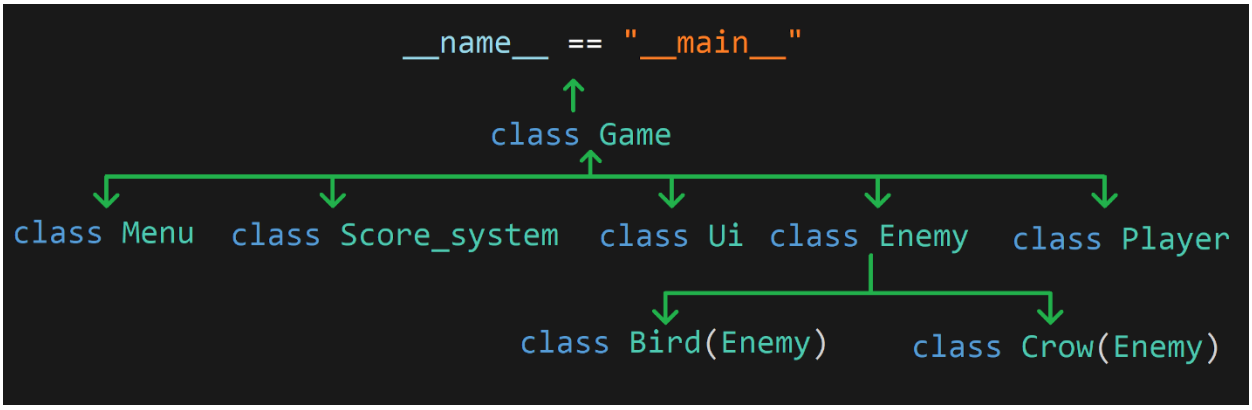
```
class Crow(Enemy):
    def __init__(self):
        super().__init__()
        self._enemy_image
```

Hierarchia

Plików:

```
ProgramowanieObiektowe /
│
├── Spritesheets/
├──.gitignore
├── highscore.txt
├── README.md
├── dokumentacjaWstpepna.pdf
├── main.py
├── menu.py
├── ui.py
├── score_system.py
├── player.py
├── enemy.py
```

Klas:



Spis funkcji i opis różnych funkcjonalności

```
class Game

    def on_init(self):
```

- Metoda wykonujaca się przy inicjalizacji gry (Game setup). Wykonuje się tylko raz.

```
    def on_event(self, event):
```

- Metoda odpowiedzialna za zbieranie eventow z gry, jak wciśnięcie klawisza czy myszy.

```
    def on_loop(self):
```

- Metoda w której wykonuje się pętla gry x razy na sekundę tzw. Game loop.

```
    def on_render(self):
```

- Metoda odpowiedzialna za renderowanie grafik na ekranie. Należy zwracać na kolejność linii kodu renderującego. Im dany render jest wyżej w hierarchii, tym zostanie on wyrenderowany szybciej. (przykład: najpierw renderujemy tło a potem na nim obiekt gracza).

```
    def on_reset(self):
```

- Metoda odpowiedzialna za resetowanie, wykonuje się po wykryciu kolizji.

```
    def on_cleanup(self):
```

- Metoda odpowiedzialna za “sprzatanie” po kliknieciu przycisku wyjścia z gry jak i zapisaniu postępu.

```
    def on_execute(self):
```

- Metoda odpowiedzialna za ułożenie wszystkich wyżej wymienionych metod w sposób logiczny (odpowiedniej kolejności). Wywołujemy ją w main, w celu uruchomienia kodu rozgrywki.

```
class Menu:

    def render(self):
```

- metoda odpowiedzialna za render menu.

```
    def onloop(self, isplayerdead):
```

- metoda odpowiedzialna za kod który ma się wykonywać w game loop.

```
    def click(self):
```

- metoda ktore ma wykonac się po kliknięciu.

```
    def ismenuon(self):
```

- metoda zwracająca wartość logiczną.

```
class Score_system

def load_hiscore(self)
```

- metoda odpowiedzialna za wczytanie danych z pliku.

```
def save_hiscore(self)
```

- metoda odpowiedzialna za zapisanie danych do pliku.

```
def set_hiscore(self)
```

- metoda odpowiedzialna za ustawienie nowego rekordu, jeśli aktualna punktacja jest większa od aktualnego rekordu.

```
def score_update(self, points)
```

- metoda odpowiedzialna aktualizowanie aktualnej punktacji.

```
def score_reset(self)
```

- metoda odpowiedzialna za resetowanie aktualnej punktacji.

```
class Ui

      def render(self, score, hiscore)
```

- metoda odpowiedzialna za renderowanie interfejsu użytkownika .

```
class Player

def force(self,Force)
```

- metoda odpowiedzialna za dodanie mocy do akceleracji do obiektu gracza.

```
def addforce(self)
```

- metoda odpowiedzialna za zwiekszenie mocy.

```
def tick(self)
```

- moetda odpowiedzialna za wykonywanie kodu w game loop.

```
def draw(self,screen)
```

- metoda odpowiedzialna za renderowanie.

```
def collision(self,enemy)
```

- metoda odpowiedzialna za wykrywanie kolizji z przeciwnikiem.

```
def isDead(self)
```

- metoda zwracająca logiczna wartość czy gracz nie żyje.

```
def click(self)
```

- metoda wykonujaca sie po kliknieciu.

```
class Enemy

def renderEnemy(self, display_surf)
```

- metoda odpowiedzialna za renderowanie

```
def enemyMovment(self)
```

- metoda odpowiedzialna za poruszanie się przeciwnika

```
def resetPosition(self)
```

- metoda odpowiedzialna za resetowanie przeciwnika. Służy jako generowanie nowego przeciwnika.

```
def isOutOfBoundry(self)
```

- metoda sprawdzająca czy przeciwnik wyszedł za ekran widziany przez użytkownika

```
def returnPoints(self)
```

- metoda odpowiedzialna za zwracanie punktów gdy przeciwnik wyszedł za ekran widziany przez użytkownika.

```
class Bird(Enemy)
```

- klasa dziedzicząca po klasie przeciwnik, dziedziczy wszystkie metody i atrybuty z nadpisaniem metod.

```
def enemyMovment(self)
```

- metoda odpowiedzialna za poruszanie się przeciwnika. Nadpisuje metodę z klasy enemy.

```
class Bird(Enemy)
```

- klasa dziedzicząca po klasie przeciwnik, dziedziczy wszystkie metody i atrybuty bez nadpisywania metod.