DOKUMENTACJA TECHNICZNA PROJEKTU PUYO PUYO SURUYO

1. Krótki opis projektu

Celem projektu było napisanie wersję japońskiej gry automatowej „Puyo Puyo" we framework-u PyGame

2. Architektura systemu

Kod projektu jest podzielony na 3 moduły:

- moduł główny, w którym są klasy związane z głównym działaniem gry oraz obsługa klawiatury
- Moduł Graphics, który zawiera klasy odpowiedzialne za wyświetlanie gry na ekranie
- Moduł System, który zawiera model planszy i bloków oraz przelicza wydarzenia na niej się dziejące

Współrzędne są brane jako para (x, y) gdzie x to numer wiersza od góry, a y to pozycja w wierszu od lewej, numerowane od 0

3. Moduł Główny

Pliki które składają się na ten moduł(oprócz __init__.py) to:

1. GameUI.py – zawiera klasę GameUI która jest mediatorem dla całej gry. Jej metody to:
   - setPlayers(self, player_1, player_2) – ustawia kontrolę input-u graczy
   - setGameBoards(self, board_1, board_2) – ustawia moduły graficzne do wyświetlania plansz
   - setScoreboard(self, scoreboard_1, scoreboard_2) - ustawia moduły graficzne do wyświetlania punktów
   - start(self) – główna funkcja wykonywania gry
   - get_score(self, gameMap, combo, cell_count) – metoda pozwalająca systemowi mapy(gameMap) na informowanie o zbiciu w sumie cell_count elementów w combo różnych grupach
   - drawNewGame(self) – wyświetlanie przed rozpoczęciem gry
   - drawGame(self) – wyświetlanie plansz
   - drawTips(self) – wyświetlanie dodatkowych komunikatów
   - drawText(self, text, size, txt_color, bg_color, start_x, start_y) – wyświetlanie tektu text w miejscu (start_x, start_y). txt_color i bg_color określają odpowiednio kolor tektu i tła ramki
   - drawStopper(self, text, txt_color, bg_color) – wyspecjalizowanie funkcji drawText do wypisywania na środku ekranu
   - terminate(self) – wyjście z gry
   - restartGame(self) – resetuje grę, czyszcząc plansze i zerując punktację
   - eventHandler(self) – obsługa naciśniętych klawiszy
2. PlayerController.py – zawiera klasę PlayerController która przechowuje informację o ustawieniach klawiatury gracza

3. main.py – tworzy wszystkie potrzebne obiekty do uruchomienia gry oraz ją uruchamia. Istotnym parametrem jest FPS, którym można przyspieszać działanie gry(chociaż przy wyższych FPS-ach może trochę „mrugać”).

4. Moduł Graphics

Pliki które składają się na ten moduł(oprócz __init__.py) to:

1. Frame.py – zawiera klasę Frame która jest klasą bazową do rysowania wszystkich poniższych ramek. Jej metody to:
   - draw(self) – rysuje ramke
   - drawBorder(self) i drawBoard(self) – rysują odpowiednio obramowanie i wnętrze ramki
2. BoardFrame.py – zawiera klasę BoardFrame służącą do wyświetlania planszy jednego z graczy. Jej metody to:
   - setSize(self, size) – ustawia rozmiar planszy
   - setBorder(self, size, color) – ustawia rozmiar i kolor obramowania planszy
   - drawElement(self, color, y, x) – rysuje pojedynczy element koloru color na pozycji (x, y)
   - draw(self, gameMap) – rysuje całą mapę gameMap, gdzie gameMap to lista zwracana przez getMap klasy GameMap
3. ScoreBoardFrame.py - zawiera klasę BoardFrame służącą do wyświetlania punktacji jednego z graczy. Jej metody to:
   - setFont(self, name) – ustawia czcionkę napisu
   - draw(self) – wysuje punktacje
   - drawName(self) i drawScore(self) – rysują odpowiednio nazwę gracza i jego punktację
   - setScore(self, score) – ustawia wartość punktów
   - addScore(self, points) – dodaje punkty
4. TextFrame.py - zawiera klasę BoardFrame służącą do wyświetlania tekstu. Jedyna jej metoda to to makeText, która przygotowuje elementy ramki
5. Colors.py – zawiera strukturę Colors zawierającą stałe kolory

5. Moduł System

Pliki które składają się na ten moduł(oprócz __init__.py) to:

1. GameMap.py – zawiera, poza kilkoma pomocniczymi funkcjami, klasę GameMap która stanowi model do obsługi planszy pojedynczego gracza. Klasa operuje na grupach elementów, zamiast na pojedynczych elementach. W pamięci przechowuje, oprócz kwadratowej tablicy planszy, m.in. grupy ruchome(zawsze jednoelementowe) i grupy przeznaczone do usunięcia. Jej metody to:
   - startSpeed(self) i stopSpeed(self), odpowiednio włączające i wyłączające akceleracje obiektu ruchomego
   - addColorless(self, count) – zwiększa ilość bezbarwnych elementów do dodania, wywoływana z zewnątrz
   - placeAt(self, color, pos) – ustawia trwale element koloru color na

miejscu pos
    - getMap(self) – zwraca tablicę kwadratową kolorów elementów na planszy, wstawiając None gdzie nie ma żadnego elementu
    - keyMove(self, direction) – wykonuje boczne przesunięcie ruchomych elementów o direction
    - rotate(self) – obraca elementy ruchome(zakłada że są one kształtu 1x2)
    - deleteStored(self) – usuwa grupy oznaczone jako do usunięcia
    - makeMoving(self, color, pos) – tworzy ruchomy element o kolorze color w miejscu pos
    - makeBlock(self) – tworzy nowy ruchomy element i przywraca obsługę graczowi. W przypadku gdy jest co najmniej pełny rząd bezbarwnych elementów do dodania, dodaje odpowiednią ilość rzędów zamiast tego
    - draw(self) – wypisuje planszę w konsoli, używane w celach testowych i debug-owych
    - move(self) – porusz elementy ruchome
    - tick(self) – wykonaj pojedynczą jednostkę czasu na planszy
    - deleteGroup(self, group) – usuń grupę group z planszy i zmień odpowiednie elementy na ruchome grupy
2. Group.py – zawiera klasę Group przechowującą informacje o pojedynczej grupie na planszy. Jej metody to:
    - cells(self) – zwraca listę pozycji elementów grupy
    - size(self) – zwraca ilość elementów w grupie
    - remove(self, removed) – usuwa element o pozycji removed z grupy
    - addColorless(self, positions) – dodaje positions do listy pozycji elementów bezbarwnych sąsiadujących z grupą
    - updateMap(self) – ustaw siebie jako grupę na wszystkich pozycjach elementów z grupy na rodzimej planszy
3. MapCell.py – zawiera klasę MapCell odpowiadającą pojedynczej komórce w planszy. Jako że jest to bardziej struktura, jedyną jej metodą jest neighbours(self) zwracającą pozycje sąsiadujących komórek

6. Rzeczy do dodania do projektu:

- poprawa jakości kodu – wlicza się w to m.in. dodanie docstrings-ów oraz testów
- dodanie obsługi dźwięków
- eliminacja błędów
- obsługa parametrów gry przez linię komend