Programowanie obiektowe i grafika komputerowa

Temat projektu: Gra zręcznościowa - Space Invaders

Osoby w zespole: Rozpoczęcie projektu: 28.04.2025r.

Kewin Kisiel 197866

Mateusz Kuczerowski 197900

Sprawozdanie z realizacji końcowej projektu – etap III

Repozytorium: https://github.com/KWN-KSL/Space-Invaders

Język programowania: Python

Biblioteki:

```
import pygame
```

Biblioteka pygame jest podstawowym narzędziem do tworzenia grafiki 2D w Pythonie. W projekcie odpowiada za rysowanie obiektów gry (statków, pocisków, tła), obsługę dźwięku (muzyka, efekty wystrzałów i eksplozji), wykrywanie kolizji pomiędzy obiektami oraz rejestrowanie interakcji z użytkownikiem za pomocą klawiatury i myszy.

Reszta to standardowe biblioteki pythona:

```
import random
import sys
import copy
import time
import math
import asyncio
import platform
import os
```

Biblioteka random służy do generowania wartości losowych. W grze jest wykorzystywana do losowego wyboru obrazków pocisków oraz do losowego momentu strzału przez przeciwników i bossa.

Moduł sys umożliwia interakcję z systemem operacyjnym. W projekcie służy m.in. do zamykania gry (sys.exit()) oraz do obsługi poprawnego ładowania zasobów (grafik i dźwięków) w wersji spakowanej jako plik .exe dzięki zmiennej sys._MEIPASS.

Moduł copy umożliwia kopiowanie obiektów. W grze jest używany do tworzenia kopii pozycji i stanów obiektów w celu zapisywania stanu gry (do cofania o kilka sekund wstecz) bez wpływu na aktualny stan rozgrywki.

Biblioteka time służy do operacji związanych z pomiarem czasu. W grze wykorzystywana jest do kontrolowania odstępów między zdarzeniami (np. co ile sekund zapisać stan gry, kiedy boss ma zmienić kierunek, kiedy można ponownie cofnąć rozgrywkę).

Moduł math zapewnia funkcje matematyczne, takie jak sinusy, cosinusy czy pierwiastki. W projekcie używany jest m.in. do nadawania losowych kierunków ruchu asteroid (na podstawie losowego kąta i funkcji trygonometrycznych).

Biblioteka asyncio umożliwia tworzenie aplikacji asynchronicznych. W grze wykorzystywany jest do uruchomienia głównej pętli gry jako funkcji asynchronicznej, co pozwala na bardziej płynną kontrolę czasu gry i renderowania.

Moduł platform pozwala sprawdzić system operacyjny, na którym uruchomiona jest gra. W projekcie służy do warunkowego uruchomienia gry w przeglądarce (Emscripten) lub jako aplikacji desktopowej (Windows, Linux, itp.).

Moduł os umożliwia operacje na systemie plików. W grze służy głównie do pobierania i poprawnego odczytywania ścieżek do plików graficznych i dźwiękowych, niezależnie od systemu operacyjnego, z uwzględnieniem pakowania gry do pliku wykonywalnego.

1. Opis gry:

W grze Space Invaders gracz steruje statkiem kosmicznym, który porusza się po dolnej części ekranu i eliminuje fale nadlatujących przeciwników. Z każdym poziomem trudności przeciwnicy stają się szybszi i bardziej agresywni. Celem gracza jest przetrwanie i zdobycie jak największej liczby punktów. W ostatniej fali pojawia się Boss, który wymaga wielu trafień i wypuszcza śmiercionośne asteroidy.

2. Cel projektu:

Celem było stworzenie kompletnej gry zręcznościowej 2D inspirowanej klasycznym "Space Invaders". Projekt miał wykorzystywać programowanie obiektowe, animacje, kolizje, grafikę, dźwięk oraz system odtwarzania rozgrywki.

3. Opis funkcjonalności:

Tryby gry:

Tryb klasyczny – 3 fale przeciwników (2 standardowe + boss)
Tryb Boss – bezpośrednia walka z bossem

Poziomy trudności:

🟢 Łatwy

3 życia
Najwolniejsi i najmniej agresywni przeciwnicy
Boss: 20 HP
Asteroidy: 2 HP

🟡 Średni

2 życia
Szybsi i częściej strzelający przeciwnicy
Boss: 30 HP
Asteroidy: 3 HP

🔴 Trudny

1 życie
Najszybsi i bardzo agresywni przeciwnicy
Boss: 50 HP
Asteroidy: 5 HP

👾 Boss

1 życie
Rozgrywka polega wyłącznie na walce z bossem
Przeciwnicy nie pojawiają się - tylko boss i asteroidy
Boss: 50 HP
Asteroidy: 5 HP

Sterowanie:

Klawisze: A/D lub strzałki (←/→) - odpowiadają za ruch

Spacja / PPM / LPM – odpowiadają za strzelanie

ESC – pauza

Klawisze 1–6 – obsługa menu pauzy

System kolizji:

W pełni zintegrowana detekcja kolizji pomiędzy:

- pociskami a przeciwnikami / bossami / asteroidami / przeszkodami
- przeciwnikami a przeszkodami
- pociskami przeciwników a graczem

Zapis gry i cofanie:

Automatyczny zapis co 1 sekundę

Możliwość cofnięcia do jednego z 5 poprzednich stanów

Odtworzenie dostępne tylko po utracie życia i zdobyciu ≥5 punktów

Efekty graficzne i dźwiękowe:

Animacje: eksplozje, odrodzenie gracza, strzały

Dźwięki: strzał, eksplozja, muzyka tła, muzyka zwycięstwa/przegranej

Różne ścieżki audio w zależności od etapu gry

Interfejs użytkownika:

Menu główne, wybór poziomu trudności, ustawienia głośności, testerzy

Ergonomiczne rozmieszczenie przycisków i intuicyjna nawigacja

4. Zasoby graficzne i dźwiękowe:

Wszystkie grafiki, efekty dźwiękowe oraz ścieżki muzyczne wykorzystane w grze pochodzą z serwisu https://opengameart.org – otwartej platformy z darmowymi zasobami do gier. Wiele z nich zostało dodatkowo przeskalowanych i przetworzonych graficznie w celu lepszego dopasowania do rozgrywki i spójności wizualnej.

5. Dlaczego stosujemy programowanie obiektowe, a nie strukturalne (liniowe):

Programowanie obiektowe daje większą kontrolę nad złożonością kodu oraz ułatwia jego rozwój i modyfikację. W kontekście gry Space Invaders:

- Modularność: Każdy element gry (statek gracza, przeciwnik, pocisk, itp.) może być reprezentowany jako osobna klasa z własną logiką i danymi.
- Czytelność i organizacja kodu: Zamiast jednej długiej funkcji zarządzającej całą logiką gry, mamy zorganizowaną strukturę z podziałem odpowiedzialności.
- Łatwiejsze testowanie i rozbudowa: Możemy łatwo modyfikować lub dodawać nowe typy wrogów, animacje czy logikę kolizji bez przekształcania całego kodu.
- Obsługa zdarzeń i stanu: Każdy obiekt może sam kontrolować swój stan (np. życie, pozycję, kolizje), co jest naturalne w grach.

6. Obiektowość w grze Space Invaders:

W naszym projekcie zaprojektowaliśmy strukturę klas w sposób typowy dla gier:
Klasa Player - odpowiada za statek gracza: ruch, wystrzeliwanie pocisków, rysowanie, detekcję kolizji.
Klasa Enemy - reprezentuje każdego przeciwnika, zawiera informacje o pozycji, kierunku ruchu, oraz metodę aktualizacji.
Klasa Bullet - obsługuje zarówno pociski gracza, jak i przeciwników.
Klasa Mothership - reprezentuje bossa gry. Posiada sztuczną inteligencję (zmienny kierunek ruchu, własne ataki), pasek życia oraz możliwość przyjmowania obrażeń.
Klasa Asteroid - reprezentuje poruszające się przeszkody losowo wypuszczane przez bossa. Mogą zniszczyć gracza i są odporne na jedno trafienie.
Klasa Obstacle - tworzy statyczne przeszkody pomiędzy graczem a przeciwnikami. Przeszkody mają dwa poziomy uszkodzenia i mogą chronić gracza przed pociskami.
Klasy Explosion, HitEffect, PlayerRespawnAnimation – odpowiadają za animacje specjalne: wybuchy przeciwników, trafienia gracza i efekt odradzania się po stracie życia.
Obsługę interfejsu użytkownika zapewniają:
Klasa Button – odpowiada za przyciski w menu gry (start, restart, cofnięcie gry, itp.).
Klasa VolumeSlider – obsługuje suwaki głośności muzyki i efektów dźwiękowych.
Klasa GameEndMenu – wyświetla menu końca gry z opcjami powrotu do menu, restartu, wyjścia oraz cofnięcia rozgrywki.

Każda z tych klas posiada metody takie jak init, update() i draw(), a niektóre zawierają również metody hit() (obsługa uszkodzeń) lub restore() (przywracanie stanu). Obiekty są zarządzane za pomocą grup pygame.sprite.Group, co umożliwia efektywne renderowanie, zarządzanie i wykrywanie kolizji między obiektami.
Modularna architektura:

Logika gry jest rozdzielona na:
- obsługę stanów (state, switch_state)
- tryby gry (create_game, create_boss_game)
- ustawienia (volume_menu)
- ekran końca gry (GameEndMenu)
- zarządzanie przyciskami (Button, VolumeSlider)

7. Podsumowanie:

Projekt Space Invaders to kompletna i dopracowana gra 2D, w pełni wykorzystująca możliwości biblioteki pygame. Wszystkie elementy gry, takie jak gracz, przeciwnicy, pociski, bossowie, przeszkody czy interfejs użytkownika, zostały umieszczone w osobnych klasach, co zapewnia czytelność, modularność i łatwość rozbudowy kodu.

Zasoby graficzne i dźwiękowe pochodzą z legalnego źródła (serwisu https://opengameart.org) - i zostały odpowiednio dostosowane do potrzeb gry.

Gra spełnia wszystkie wymagania projektowe: zawiera detekcję kolizji, cofania stanu gry, różne poziomy trudności, pełne sterowanie klawiaturą i myszą oraz rozbudowaną obsługę interfejsu. Duży nacisk położono również na stronę wizualną i dźwiękową - zastosowano animacje, efekty graficzne oraz dynamiczną muzykę zmieniającą się w zależności od etapu rozgrywki.

W projekcie zaimplementowano także funkcje wykraczające poza podstawowy zakres, takie jak tryb walki z bossem, pojawiające się losowo asteroidy, pasek życia bossa oraz opcja cofania gry o kilka sekund.