WormyReloaded

Modyfikacja gry Wormy

Projekt zaliczeniowy

Krzysztof Cieślik s27115

Cel gry

Celem gry jest maksymalizacja uzyskanego wyniku za punktowane akcje w grze przy jednoczesnym poruszaniu się tytułowym wormy wg ustalonych zasad.

Zasady poruszania się:
- Ruch jest nieprzerwany , robak może skręcać tylko w lewo i prawo względem swojej głowy.
- Nie wolno uderzyć przeszkode ,a w trybie dead walls w ścianę - krawędz planszy.

Punktowane akcje:
- Zebranie jabłka.
- Wykonywanie zakrętów w trybie przyspieszenia.

Zastosowane technologie

Python 3.13.3
Z bibliotekami:

- pygame
- random
- sys
- yaml

Muzyka w postaci plików .ogg

Dodane funkcje

1. Tablica wyników
2. Powerup
3. Poziomy trudności
4. 2 Tryby gry
5. Efekty dźwiękowe
6. Tryb turbo
7. Muzyka zależna od poziomu
8. System losowych przeszkód
9. Instrukcja i interfejs
10. Dodatkowa mechanika zbierania liter, rozbudowany system punktacji, pauza.

Napotkane problemy

1. Robak w wormy mógł wejść sam w siebie poprzez wykonywanie nielegalnego ruchu tj.
   W ciągu jednego ticka gry można było wywołać więcej niż jeden ruch przez co przed wygenerowaniem się obrazu dochodziło do kolizji.

   Rozwiązanie:
   Zapamiętanie ostatniego kliknięty legalnego wyboru kierunku i wykonanie go dopiero podczas renderowania.
2. Ruch ustalany wg ilości fps.
   Powodowało to dużą ziarnistość regulacji prędkości ruchu, dodatkowym problemem jest limit ilości fps jaką mogą uzyskać słabe komputery.

   Rozwiązanie:
   Ustawienie stałej ilość fps na 60 i wykonywanie aktualizacji pozycji co określoną w zależności od poziomu trudności liczby klatek.
3. Brak możliwości zmiany prędkości odtwarzanej muzyki w pygame
   Tryb turbo miał mieć przyspieszony dźwięk o 10 procent ,ale biblioteka pygame nie obsługuje takiej funkcjonalności.
   Można to wykonać przy pomocy innej biblioteki ,ale wymaga to formatu .wav co zajmuje dużo miejsca.
   Inne rozwiązania wymagają dodatkowych składników zainstalowanych w systemie operacyjnym.
   Rozwiązanie: Zmiana muzyki w trybie turbo na inną dużo szybszą, po wyjściu z turbo powrót do normalnego podkładu.

Napotkane problemy

4. Dobranie koloru powerup.
   Kolor powerup ,a w szczególności jego poziom przezroczystości może negatywnie wpłynąć na grywalność.

   Rozwiązanie: Użycie skalibrowanego monitora do dobrania odcienia i alpha.

5. Złożony system kolizji
   System kolizji musi sprawdzać bardzo dużo warunków np. Kolizja z przeszkodą, zebrania jabłka, teleportacja lub kolizja z ścianą, kolizja węża z samym wężem.
   W przypadku dalszej rozbudowy tego systemu bezpośrednio w kodzie odpowiedzialnym za wyświetlanie gry spowoduje to że kod będzie zbyt zawiły dodatkowo może nie zmieścić się w ramce czasowej 1/60 sekundy.
   Potencjalnie należy go wydzielić do osobnej funkcji dzieląc na mniejsze funkcje oraz skorzystać z możliwości pytona 3.13 i wykonywać go asynchronicznie poza GIL.

6. Obsługa różnych trybów gry powinna być zaimplementowana jako osobne klasy z wykorzystaniem wzorca projektowego strategia.

Q&A