Politechnika Śląska

Wydział Matematyki Stosowanej

Kierunek Informatyka

Gliwice, 02.02.2022

Programowanie I

**projekt zaliczeniowy**

**"*Klon „Flappy Bird” używając biblioteki SFML*"**

**Radosław Olesiński gr. lab. 08**

**1. Opis projektu.**

Projekt jest napisany w C++ z użyciem biblioteki SFML. Ma odwzorowywać popularną grę na telefony z 2013 roku. Sama gra ma w sobie system punktów, medali za określoną liczbę punktów. Polega na „przelatywaniu” przez rury – czyli przeszkody. W samym kodzie używane są funkcjonalności biblioteki SFML jak np. Sprite czy obsługa okna lub myszy.

**2. Wymagania**

* Wczytywanie tekstur i poprawne użycie ich.
* Funkcjonalny „Silnik” gry, który podzielony jest na różne fragmenty jak np. InputManager czy AssetManager.
* Proste menu służące do rozpoczęcia gry.
* Tworzenie i dynamiczne usuwanie elementów z pola gry.
* Reakcja na kliknięcia myszą.
* Wykrywanie kolizji z innymi obiektami.
* System punktów i medali.
* Dodanie dźwięku.
* Zapisywanie najlepszego wyniku.

**3. Przebieg realizacji**

**DEFINITIONS.hpp**

Plik nagłówkowy, który zawiera wszystkie definicje, przypisania takie jak np. ścieżki do plików czy wartości odnoszące się do funkcjonalności typu częstotliwość ustawiania się rur w grze.

**AssetManager.cpp + AssetManager.hpp**

Plik .cpp i plik nagłówkowy. Jest to część silnika gry. Jest odpowiedzialny za wczytywanie tekstur i czcionek i ich obsługę.

**InputManager.cpp + InputManager.hpp**

Kolejna część silnika gry. Odpowiada za obsługę myszy. Posiada dwie funkcje: sprawdzenie czy obiekt jest kliknięty i pokazanie pozycji myszy.

**StateMachine.cpp + StateMachine.hpp**

Pliki odpowiadające za obsługę stanów gry. Dodaje stan na stos, usuwa stan z niego i procesuje zmiany stanów. Również zwraca aktualnie aktywny stan ze stosu.

**state.hpp**

Plik nagłówkowy posiadający klasę State.

**Game.cpp + Game.hpp**

Plik odpowiadający za utworzenie okna z grą, zaimplementowanie pierwszego stanu gry i same uruchomienie gry.

**MainMenuState.cpp + MainMenuState.hpp**

Pierwszy stan gry. Jego funkcja to wyświetlanie Menu i przejście do stanu gry.

Obraz zawierający tekst, niebo, trawa, znak

Opis wygenerowany automatycznie

**GameState.cpp + GameState.hpp**

Najważniejszy plik w projekcie. Jest to stan gry, gdzie wczytuje się wszystkie potrzebne tekstury (tło, ptak, czcionka itp.). Funkcja Update aktualizuje cały czas dane. W niej sprawdzana jest kolizja, tworzone są kolejne rury i usuwane dynamicznie, liczone są punkty. W GameState również obsługiwane jest przechodzenie do następnego stanu gry, czyli GameOverState.

Obraz zawierający tekst, zielony, znak

Opis wygenerowany automatycznie

**GameOverState.cpp + GameOverState.hpp**

Pliki te obsługują ostatni stan gry, czyli sam jej koniec. Tutaj wczytywany jest wynik gracza, porównywany z najwyższym wynikiem zapisanym w pliku HighScore.txt. Wyświetla końcowy wynik, zdobyty medal i umożliwia ponowne rozpoczęcie gry.

Obraz zawierający tekst, trawa, znak, pomarańczowy

Opis wygenerowany automatycznie

**Bird.cpp + Bird.hpp**

Pliki odpowiadające za klasę Bird, czyli ptaka, którym kontroluje gracz. Wdrożona jest animacja ptaka, który zmienia swoje klatki powodując animację latania. Ptak po kliknięciu rotuje w górę, przy opadaniu rotuje w dół. Funkcja Tap odpowiada za uniesienie ptaka za każdym kliknięciem.

**Collision.cpp + Collision.hpp**

Prosta obsługa kolizji. Jeśli dany element znajduje się w drugim, to poprzez funkcję CheckSpriteCollision możemy się o tym dowiedzieć, co pozwala na zatrzymanie gry w momencie zderzenia się ptaka z rurą lub z ziemią.

**Flash.cpp + Flash.hpp**

Dzięki temu, po wykryciu kolizji ekran zabłyśnie na biało przez krótką chwilę.

**HUD.cpp + HUD.hpp**

Pliki odpowiadające za wyświetlanie wyniku, medalu i najwyższego wyniku na końcowym ekranie.

**Land.cpp + Land.hpp**

Wyświetlanie i przesuwanie się terenu. Dzięki osobnej klasie na teren łatwo można wykryć kolizję.

**Pipe.cpp + Pipe.hpp**

Pliki odpowiadające za wyświetlanie się rur, które trzeba omijać. Poza wyświetlaniem widocznych rur, wyświetlane są również niewidzialne rury, przez które ptak przelatuje i nalicza się punkt. Rury tworzą się losowo i usuwają się dynamicznie z pamięci.

**main.cpp**

Plik main. W nim zawarte jest tylko połączenie z plikami nagłówkowymi i utworzenie samej klasy Game.

**4. Instrukcja użytkownika**

W grze poruszamy się ptakiem w górę za pomocą kliknięcia lewym przyciskiem myszy. Ptak opada sam. Cel gry to uzbieranie jak największej ilości punktów. Punkty zdobywa się poprzez przelatywanie przez rury. Gra kończy się, gdy zderzymy się z jakąś rurą bądź z ziemią. Po skończeniu gry wyświetla się ekran, gdzie pokazane są punkty i możliwość rozpoczęcia nowej. W grze występuje system medali:

* Brązowy – automatycznie od 0pkt
* Srebrny – od 10pkt
* Złoty – od 30pkt
* Platynowy – od 50pkt

**5. Podsumowanie i wnioski.**

Prawie wszystkie założenia projektu zostały osiągnięte. Niestety, nie udało się zrealizować dźwięku w grze. Największym problemem była obsługa silnika gry, czyli wczytywanie stanów i obsługa kolizji.