Politechnika Śląska w Gliwicach

Wydział Automatyki, Elektroniki i Informatyki

Programowanie Komputerów 4

Gra – Jynx Jump

Autor	Tomasz Sojka
Prowadzący	dr inż. Anna Gorawska
Rok akademicki	2018/2019
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	4
Grupa	2

1. Specyfikacja zewnętrzna

1.1. Działanie gry

1.1.1. Stan początkowy

Gra jest w stanie początkowym zaraz po włączeniu programu lub po przegranej grze, gdy użytkownik chciał zagrać jeszcze raz. W stanie tym pojawiają się już ustawione platformy, oraz menu z nazwą gry z najlepszym wynikiem oraz z informacjami dla użytkownika, co powinien zrobić, aby rozpocząć grę.

Użytkownik ma możliwość rozpoczęcia gry(przejścia do stanu gry), przez wciśnięcie klawisza spacji, przesunięcia Junx(naszej postaci), za pomocą prawej i lewej, oraz wyjścia z gry, przez kliknięcie krzyżyka.

1.1.2. Stan rozgrywki

Gra jest w stanie rozgrywki, jeśli została rozpoczęta w stanie początkowym. Z ekranu znikają wszystkie napisy, a Jynx zaczyna nieustannie skakać (jeśli tylko ma się od czego odbić).

Użytkownik może kierować Jynx za pomocą prawej i lewej strzałki. Po osiągnięciu pewnej wysokości, wyświetlone platformy zaczynają przesuwać się w dół, a w lewym górnym rogu pojawia się aktualny wynik, który będzie rósł wraz z przebytą drogą w górę. Co jakiś czas(zmienny) pojawiają się platformy ruszające się na boki, platformy kruszące się po jednym obiciu, oraz Farfetch'd(nasz przeciwnik),który leci się od lewej do prawej, a następnie pojawia się z powrotem po lewej.

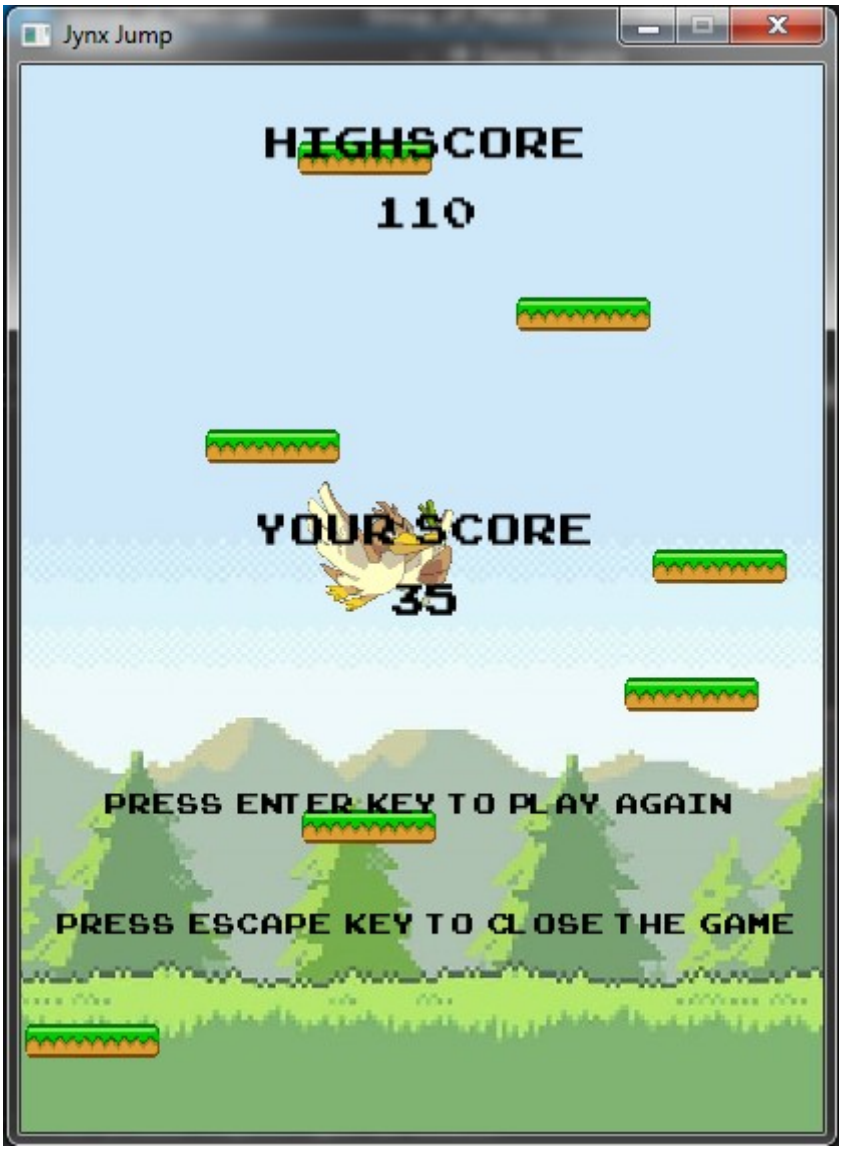
Jynx nie może „przechodzić przez ściany", a kontakt z nimi zatrzymuje ją na krawędzi ekranu. Kontakt z Farrfetch'dem lub spadek poniżej dolnej krawędzi oznacza koniec stanu rozgrywki(i przejście do stanu końcowego).

Użytkownik w każdej chwili może wyjść z gry, poprzez kliknięcie krzyżyka.

Wyjście z gry powoduje zapis aktualnego wyniku, jeśli tylko jest większy od najlepszego.

1.1.3. Stan końcowy

Gra jest w stanie końcowym, jeśli użytkownik przegrał w stanie rozgrywki (lecz nie wyszedł za pomocą krzyżyka). W stanie tym pojawiają się już menu z najlepszym wynikiem, wynikiem zdobytym w ostatniej rozgrywce, oraz informacjami dla użytkownika, co może zrobić.

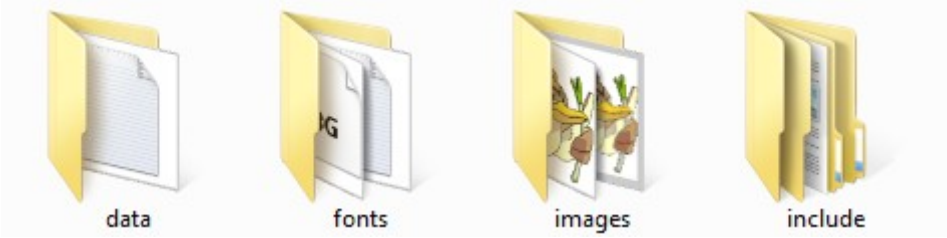


Użytkownik może rozpocząć kolejną rozgrywkę(przejść do stanu początkowego), przez wciśnięcie klawisza Enter lub może zakończyć program, przez naciśnięcie klawisza Escape lub kliknięcie krzyżyka.

Wyjście z gry lub przejście do stanu początkowego powoduje zapis aktualnego wyniku, jeśli tylko jest większy od najlepszego.

1.2. Możliwe błędy

Jeśli w folderze docelowym z Aplikacją zabraknie któregoś z następujących plików:



Program wyświetli komunikat w konsoli, o braku pliku (z opisem czego brakuje). Program nie będzie działał poprawnie.

2. Specyfikacja wewnętrzna

2.1. Użyte klasy

Klasy użyte w programie, oraz ich opis	
Collision	**Kolizje skoczka. Klasa wirtualna.**
Collision_with_Opponent	**Kolizja skoczka z przeciwnikiem. Dziedziczy po klasie *Collision.***
Collision_with_Plat	**Kolizja skoczka z platformą. Dziedziczy po klasie *Collision.***
Collision_with_Window_edges	**Kolizja z bocznymi i dolną krawędzią ekranu, oraz z czujnikiem wysokości. Dziedziczy po klasie *Collision.***
Draw	**Losowanie współrzędnych(dwóch liczb typu float) i indeksu platformy (liczba typu int)**
Game_Engine	**Silnik gry. W nim znajduje się główna funkcja play(). Zarządzanie stanami gry oraz pozwoleniem na ruch platform i przeciwnika w dół.**
Group_of_Plats	**Reprezentacja grupy platform (bazowych, oraz pochodnych). Tworzenie grupy, zamiana platform na pochodne, ruch grupy platform.**
Interface	**Interfejs. Wszystko co jest wyświetlane na ekranie za pomocą bibliotek SFML'a. Tworzenie i zamykanie okna.**
Jumper	**Implementacja skoczka. Rozmiary skoczka, oraz współrzędne na ekranie.**
MovingPlatform	**Implementacja platform ruszającej się na boki. Dziedziczy po klasie Platform.**
OneJumpPlatform	**Implementacja platform kruszącej się po jednym skoku. Dziedziczy po klasie Platform.**
Opponent	**Implementacja przeciwnika.**
Platform	**Implementacja bazowej platform.**
Score_Counter	**Licznik wyniku. Odczyt i zapis najlepszego wyniku z pliku.**

Window	**Implementacja okna. Rozmiary okna, oraz pozycja czujnika wysokości.**

2.2. Powiązanie klas

Diagram klas UML znajduje się w pliku Diagram.jpg

2.3. Elementy zaawansowanego C++

2.3.1. Mechanizm wyjątków

Mechanizm wyjątków został użyty przy operacjach wczytania z pliku. Wyjątek wyrzucany jest w postaci tekstu w konsoli, w przypadku braku któregoś pliku. Różny tekst jest wyświetlany w zależności od brakującego pliku. Może to być plik z którąś teksturą, plik z czcionką lub plik z najlepszym wynikiem gry.

2.3.2. Kontenery STL

W programie został użyty std::vector do przechowania grupy platform (obiektów klay Platform, oraz jej klas pochodnych). Ten typ kontenera STL został wybrany z powodu jednorazowej alokacji, lecz częstego dostępu do elementów kontenera.

2.3.3. Algorytmy i Iteratory STL

Iteratory STL ułatwiły przemieszczanie się po kontenerze, jak i pozwoliły na wywołanie metod ze wskazywanego przez iterator obiektu.

2.3.4. RTTI

Technika RTTI została wykorzystana do operacji na grupie platform w celu rozpoznania obiektów klas pochodnych(klasy MovingPlatform i OneJumpPlatform). Platformy były zamieniane na pochodne w sposób pseudolosowy, więc metody RTTI idealnie się nadały do ich rozróżnienia i ograniczyły ilość zmiennych pomocniczych.