Politechnika Śląska

Wydział Matematyk Stosowanej

Kierunek Informatyka

Gliwice, 02.02.2022

Programowanie I

**projekt zaliczeniowy**

**"*Gra Pinball*"**

**Antoni Jaszcz gr. lab. 3/5**

**1. Opis projektu.**

Projekt ma na celu odtworzenie popularnej gry „Pinball” (znanej także jako „Flipper”), w dwuwymiarowej, komputerowej symulacji. Gra została stworzona przy użyciu biblioteki graficznej SFML oraz biblioteki fizycznej Box2D.

<https://pl.wikipedia.org/wiki/Flipper>

Gra została stworzona w tematyce serialu „Arcane”.

**2. Wymagania**

Do poprawnego działania programu niezbędne są:

* Biblioteka SFML,
* Biblioteka Box2D,
* Tekstury (folder „Textures” z plikami .png),
* Dźwięk (folder „Sound” z plikami .wav),
* Czcionka (folder „Text” z plikiem „Font.otf”).

**3. Przebieg realizacji**

1. Pobranie i dołączenie do projektu dwóch bibliotek:
   1. SMFL – biblioteka multimedialna; Biblioteka zaprojektowana do łatwej implementacji API oraz wykorzystywania multimediów (grafiki, dźwięku) w programie; Biblioteka obsługuje akcelerację sprzętową grafiki 2D przy użyciu OpenGL; Biblioteka jest ogólnodostępnym oprogramowaniem (open-source); Biblioteka jest wieloplatformowa.  
      <https://www.sfml-dev.org/>
   2. Box2D – ogólnodostępny, dwuwymiarowy silnik fizyczny, stworzony przez Erina Catto (pracującego w Blizzard Entertainment); został wykorzystany m. in. przy tworzeniu „Angry Birds”; Silnik ogranicza się do symulacji brył sztywnych; Silnik świetnie nadaje się, do tworzenia symulacji fizycznych.  
      <https://box2d.org/about/>
2. Utworzenie dwóch klas:
   1. Game – odpowiada za graficzną i dźwiękową reprezentację gry. Innymi słowy, łączy świat symulacji oraz interpretuje informacje podawane przez użytkownika (naciskanie odpowiednich klawiszy), aby następnie przedstawić ich graficzne i dźwiękowe odzwierciedlenie (renderować na ekranie grę).
   2. WorldClass – klasa kontrolująca silnik fizyczny (obiekt klasy b2World oraz obiekty, w niej się znajdujące). Dopowiada za inicjalizacje obiektów, zwracanie ich parametrów (np. pozycji) oraz detekcję kolizji.
3. Box2D, a SFML:

Obydwie biblioteki mają swoje zasady działania, które nie zawsze są zgodne/optymalne. Trzeba tu wspomnieć o następujących zależnościach:

* Obydwa silniki używają nieujemnego układu współrzędnych, których punkt (0,0) znajduje się w lewym, górnym rogu.
* Pozycja ciała w Box2D jest definiowana jako jego środek, natomiast w SFML, jest nią jego lewy górny róg. Jest to uciążliwe, szczególnie kiedy obiekt ulega obrotowi.
* W Box2D, ciała domyślnie są statyczne (nie działają na nie żadne siły zewnętrzne, np. grawitacja). Oczywiście możemy definiować ciała jako dynamiczne, a nawet (jest to uciążliwe), łączyć ciała różnych typów.
* W Box2D, detekcja kolizji odbywa się przez klasę „b2ContactListener”, którą można stosować na własne potrzeby, po jej odziedziczeniu.
* 1 piksel w świecie Box2D odpowiada 1m w symulacji fizycznej. Dlatego, aby uzyskać najlepsze efekty (i utrzymać wielkości fizyczne w rozsądnych przedziałach), zalecana jest konwersja jednostek używanych w SFML na te, używane w Box2D, w odpowiednich proporcjach (np. w skali 1:10 – 10 pikseli odpowiada 1 metrowi; tak też jest w projekcie).

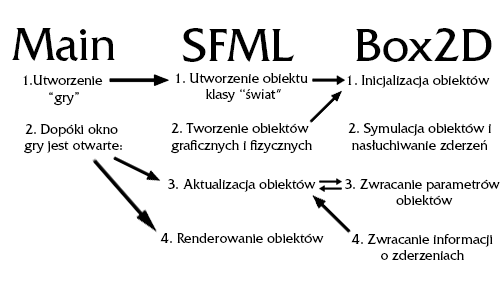
1. Stworzenie obiektów gry (zarówno fizycznych, jak i graficznych):
   1. Piłki,
   2. Flipperów (łopatek),
   3. Band,
   4. Bumperów,
   5. Okno.
2. Stworzenie odpowiednich funkcji i części kodu, odpowiadających za:
   1. Inicjalizację obiektów,
   2. Aktualizację obiektów (ich pozycji, rozmiarów, etc.),
   3. Wyświetlanie obiektów na ekranie,
   4. Pobieranie danych od użytkownika (naciskanie klawiszy),
   5. Obsługę wydarzeń:
      1. Game Over,
      2. Restart gry,
      3. Naciąganie sprężyny,
      4. Zderzenia piłki z Bumperami (liczenie punktów, dźwięk),
      5. Pauza,
      6. Zamknięcie gry,
      7. Zapis/odczyt/aktualizacja High Score.
   6. Kontrola muzyki w tle oraz dźwięku.
3. Płacz.
4. Dostosowanie odpowiednich parametrów.

Znaczna część odległości i wielkości obiektów została zdefiniowana przez 3 zmienne globalne:

* Wysokość okna,
* Szerokość okna,
* Promień piłki.

Dzięki temu, w łatwiejszy sposób można przeskalować obiekty na planszy, w razie zmiany koncepcji.

Relacja poszczególnych części programu jest następująca:



**4. Instrukcja użytkownika**

Gracz:

* Posiada 3 życia,
* Zdobywa punkty uderzając w Bumpery,
* Traci życie, gdy piłka spadnie poniżej ekranu,
* Może poruszać Fliperami za pomocą strzałek (←,→),
* Może zapauzować grę (P).
* Może wyjść z gry wciskając (ESC) lub zamykając okno.
* Gdy piłka znajduje się na dole „tunelu” (miejsca startowego), gracz może ją wystrzelić, przytrzymując, a następnie puszczając spację (SPACE). Tym dłużej ją przytrzyma, tym szybciej poleci (istnieje jednak ograniczenie górne dla siły, jaką można jej nadać).
* Po przegranej, gracz może zrestartować rozgrywkę wciskając (F3).
* Po prawej stronie ekranu wyświetlane są:
  + Aktualny wynik,
  + Aktualne życie,
  + High score.

**5. Podsumowanie i wnioski.**

Za projekt zabrałem się nie mając żadnego doświadczenia z SFML-em ani Box2D. Zawsze chciałem jednak napisać własną grę, a jako że byłem wielkim fanem „Space Cadet’a” i nadarzyła się okazja, powstał mój pomysł na projekt.

Sam projekt zajął mi znacznie więcej czasu niż się spodziewałem. Pochłonęło go nie tylko samo pisanie kodu, co uczenie się, jak działają użyte przeze mnie silniki, planowanie, rozwiązywanie problemów oraz tworzenie detali (np. tekstur).

Więcej problemów sprawił dla mnie Box2D, jest zdecydowanie napisany „mniej przyjaźnie” dla użytkownika niż SFML, który jest wysokopoziomową biblioteką. Obydwie jednak zapewniają masę możliwości, dla użytkownika.

Ze względu na czas i trudność z analizą zderzeń (prawo odbicia i detekcja kolizji wobec obiektów przekrzywionych), postanowiłem sięgnąć po silnik graficzny. Praca z nim nie była łatwa, szczególnie implementacja flipperów (zaczepianie dwóch ciał o innym typie, działanie na nie siłą i reprezentacja graficzna, przy obracających się ciałach, zaczepionych w jednym punkcie i o różnych przemieszeniach, względem wersji fizycznej i graficznej…).

Ostatecznie jednak, udało się. Jestem bardzo zadowolony z tego, jak program wyszedł. Sam w sobie stanowi dobrą podstawę do dalszego rozwoju i implementacji dodatkowych funkcji, na które teraz (sesja), zabrakło mi czasu, a które na pewno w przyszłości będę dodawał.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated