Programowanie Komputerów 4

Projekt gry „Block Braker”

Autor	Michał Pokrzywa
Prowadzący	Dr Hab. Inż. Prof. PŚ Roman Starosolski
Rok akademicki	2021/2022
Kierunek	Informatyka
Rodzaj studiów	Stacjonarne
Semestr	4
Termin laboratorium	Wtorek 11:45 – 13:15
Grupa Dziekańska	2
Sekcja	1
Termin oddania sprawozdania	22.06.2022 r.

1. Temat projektu

Zadaniem projektu było gry „Block Braker”, w której znajdują się konkretne zagadnienia :

- Projekt jest napisany obiektowo
- Uwzględnienie części zagadnień z zajęć tematycznych
- Wykorzystanie Polimorfizmu
- Przynajmniej 6 istotnych własnych i wzajemnie powiązanych klas

2. Analiza Tematu

1. Doprecyzowanie tematu

Projekt zakłada zaimplementowanie gry „Block Braker” , która polega na odbijaniu piłki i zniszczeniu wszystkich klocków na planszy. ”Block Braker” jest własną interpretacją gier „Brake out” oraz „Arkanoid”. Jest to gra dla jednego gracza. W grze znajduje się kilka plansz , które gracz musi pokonać , żeby dostać się Bossa. Po pokonaniu Bossa gracz wygrywa grę i ma możliwość zapisania swojego wyniku w tabeli. Wynik jest liczony na podstawie punktów za zbicie bloków oraz za czas pokonania gry. Jeśli piłka spadnie pod platformę od której się odbija, gracz przegrywa i również dostaje możliwość zapisania swojego wyniku w tabeli. W Menu postanie dodatkowa opcja zobaczenia 10 najlepszych wyników w tabeli. Oprócz samego niszczenia bloków będą również wzmocnienia gracza tzw. Power-up . Przykładowo szybsza piłka będzie miała większą prędkość lub powiększenie platformy , aby łatwiej można było odbić piłkę. Na planszy również znajdą się różnego zniszczenia bloki ,które będzie trzeba uderzyć więcej niż jeden raz, które posiadają inny kolor zależności od posiadanego zdrowia. Po uruchomieniu aplikacji gracz będzie mógł zacząć grę , zmienić wygląd platformy, zobaczenie tabeli wyników oraz opcja wyjścia z gry. Gra jest możliwa tylko przy pomocy kontrolera (np. Xbox One Controler).

2. Wykorzystane biblioteki zewnętrzne

Do stworzenia gry „Block Braker” została wykorzystana biblioteka graficzna SFML 2.5 , dzięki której jest możliwe tworzenie obiektów graficznych , tworzenie okien oraz dodawanie plików dźwiękowych.

3. Wykorzystane Algorytmy

Do działania gry zostały stworzone algorytmy

- Fizyka i odbijanie piłki
- Obliczania wyniku gry
- Wczytywania i zapisywania danych

Każdy z tych algorytmów zostanie rozwinięty w specyfikacji wewnętrznej.

3. Specyfikacja zewnętrzna

1. Instrukcja dla użytkownika

Program jest uruchamiany przy pomocy pliku .exe. Po włączeniu gry pojawia się menu, gracz porusza się przy pomocy strzałek na kontrolerze (żółte kółko zaznaczone na Rysunku nr 1), aby wybrać jedną z wielu opcji. Aby zatwierdzić jedną z wielu opcji trzeba wcisnąć przycisk o kodzie 0 (w przypadku kontrolera xbox Series X jest to przycisk A, widoczny na Rysunku nr1). Po wybraniu menu ustawień gracz ma możliwość wybrania koloru platformy. Kolor platformy jest zapisywany i zapamiętany przez grę po jej wyłączeniu. W menu rankingu znajduje się możliwość zobaczenia 10 najlepszych wyników z poprzednich rozgrywek. Po wybraniu opcji gry , gracz ma do wyboru poziom trudności, który będzie definiował trudność rozgrywki. Po wybraniu poziomu trudności, gracz może zacząć poruszać platformę przy pomocy lewej gałki (niebieskie kółko zaznaczone na Rysunku nr 1). Aby zacząć poziom gracz musi nacisnąć przycisk A , aby piłka zaczęła się poruszać. Gracz musi zbić wszystkie klocki w danym poziomie, aby przejść do następnego poziomu. Jeśli piłka spadnie pod platformę to gra się kończy i gracz przechodzi do zapisania wyniku swojej rozgrywki. Ostatni poziom gry jest poziomem z Bossem. Aby pokonać Bossa trzeba trafić w niego piłką 10 razy. Boss również

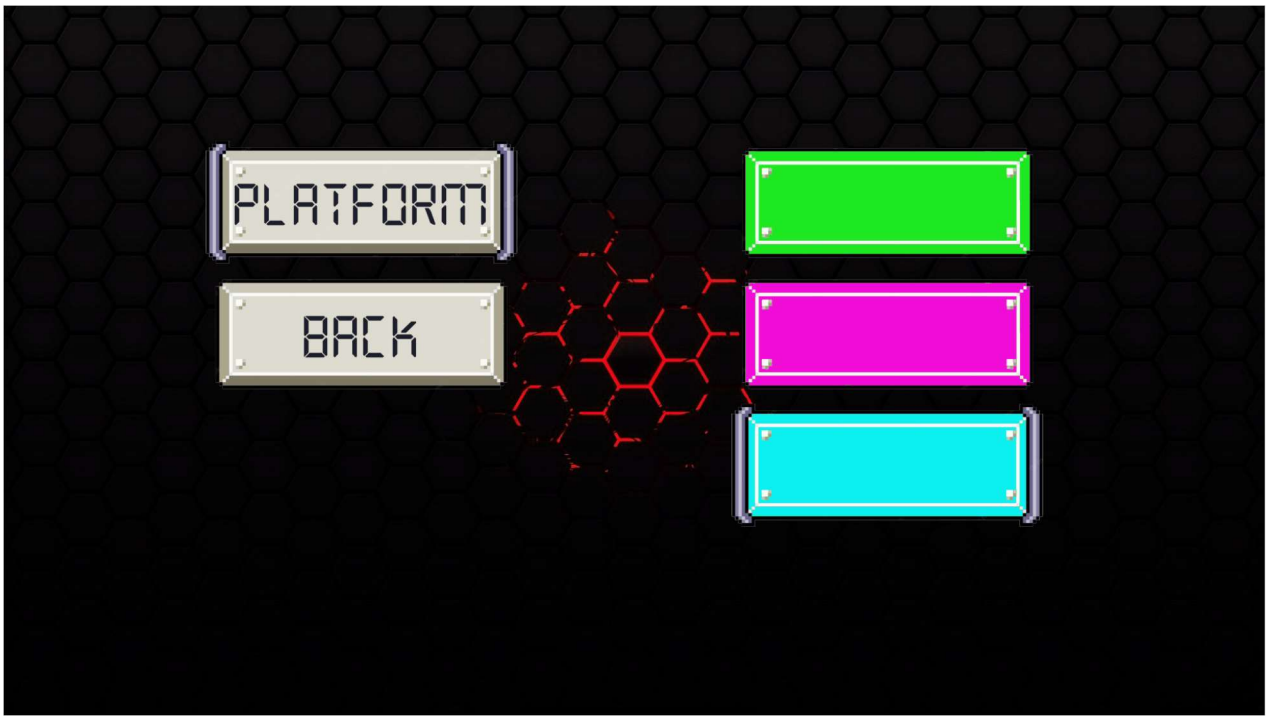
strzela do nas pociskami przed którymi musimy unikać oraz boss porusza się po planszy.

Rysunek 1 Zdjęcie wyglądu kontrolera Xbox Series X

2. Zrzuty Ekranu

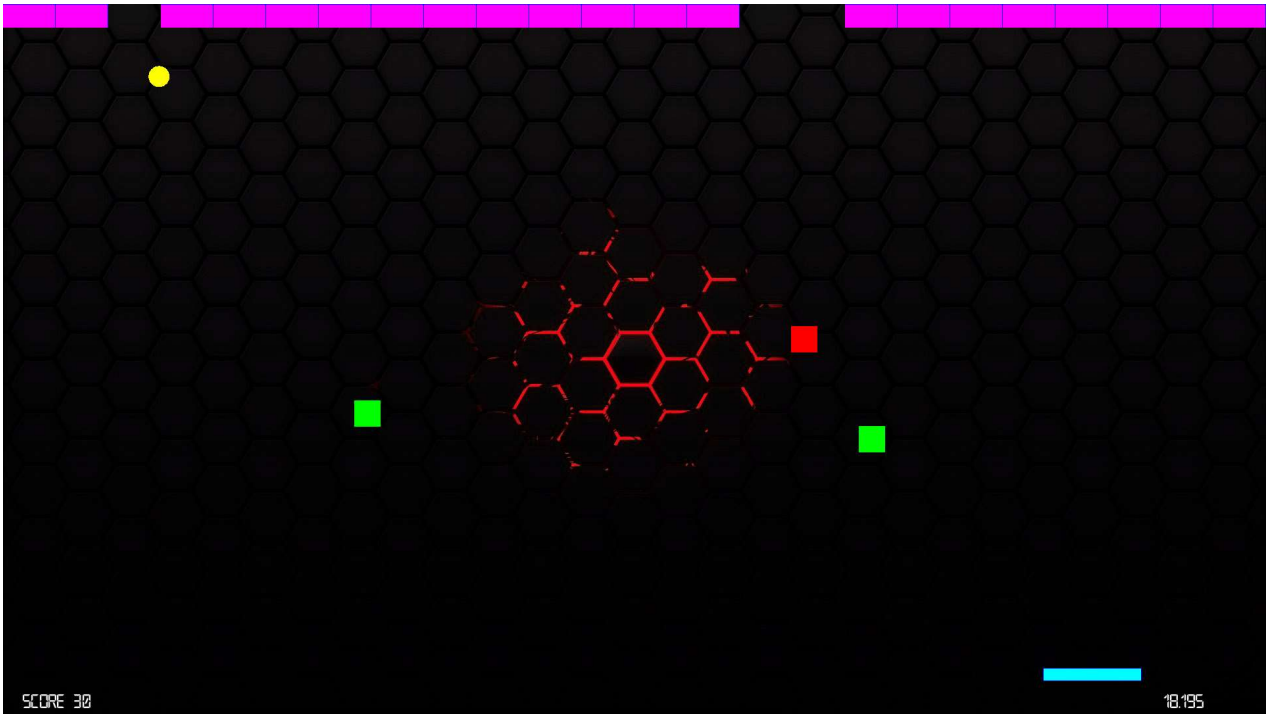
Rysunek 2 Wygląd Menu głównego

Rysunek 3 Wygląd tabeli wyników

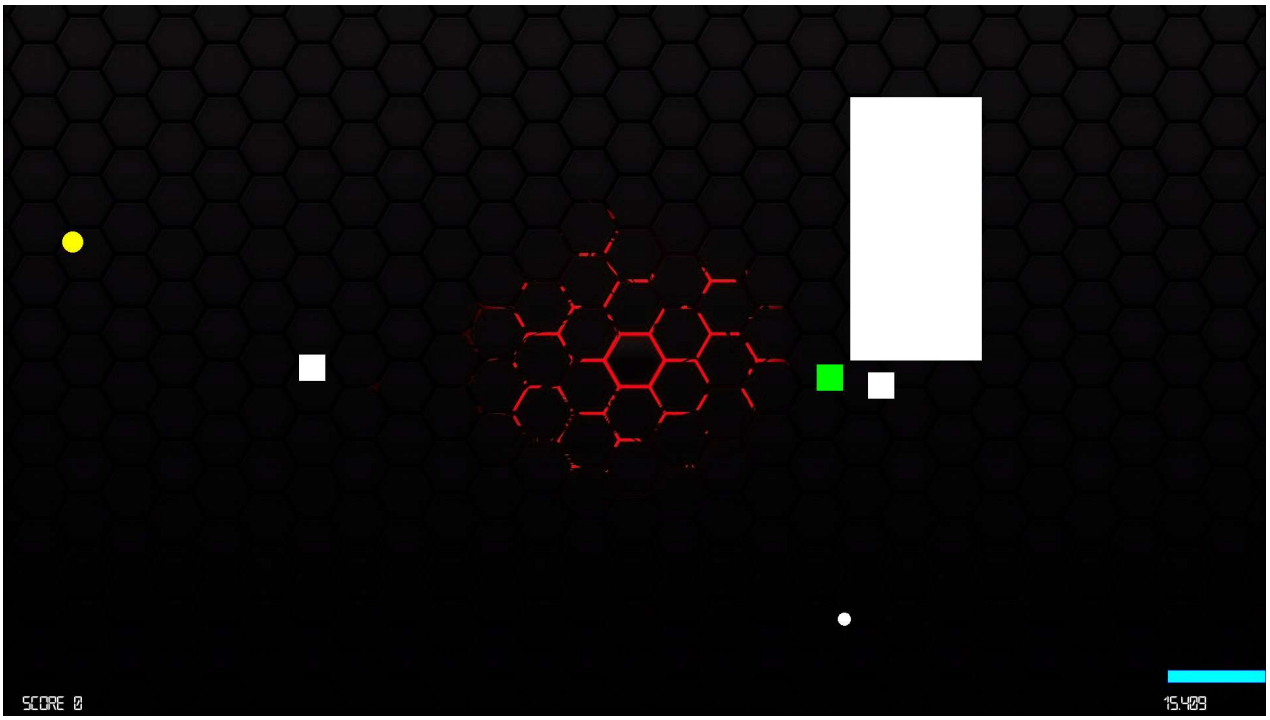


Rysunek 4 Wygląd Ustawień

Rysunek 5 Wygląd poziomu trudności



Rysunek 6 Wygląd gry



Rysunek 7 Wygląd walki z Bossem

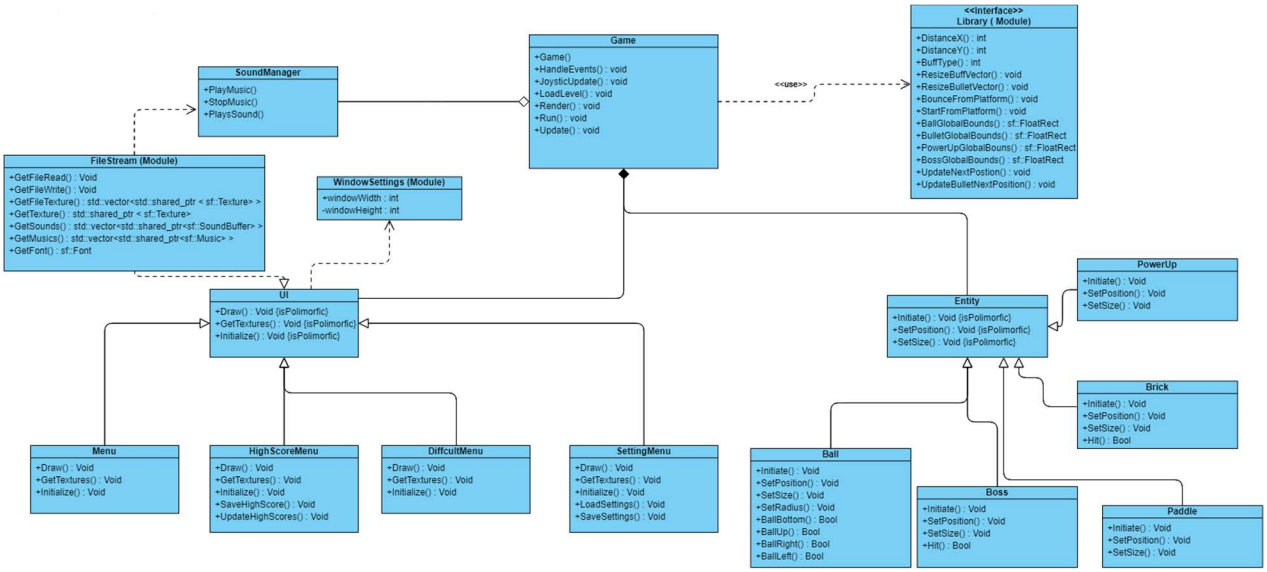
Rysunek 8 Wygląd ekranu porażki

Rysunek 9 Wygląd ekranu zwycięstwa

Rysunek 10 Wygląd Wpisywania wyniku do tabeli

4. Specyfikacja wewnętrzna

1. Diagram hierarchii klas



2. Istotnie struktury danych

W głównej klasie Game od Klasy Entity znajdują się vectory shared pointerów dla Brick , PowerUp oraz pocisków bossa. Dla Ball , Paddle i Boss znajduje się zmienna. Każda klasa pochodna Ui posiada zmienną w klasie Game. Wewnątrz modułu Library znajdują się pojedyncze funkcję wykorzystywane w klasie Game, które głównie służyły czytelności kodu. Filesystem jest modułem który służy wczytywaniu i zapisywaniu różnych typów plików. Sound Manager posiada wszystkie dźwięki ,które są w grze ,a Game wywołuje. Każda z klas posiada dziedziczących wywołuje swoje odpowiednie funkcję w Game.

3. Wykorzystane techniki obiektowe

W projekcie zostały wykorzystane Moduły, Biblioteka Regex , Biblioteka Filesystem oraz Biblioteka Ranges. Regex służy do sprawdzania poprawnego wpisanego nazwy gracz po skończonej grze oraz przy pobieraniu tekstur czy posiadają odpowiednie rozszerzenie. Filesystem został wykorzystany przy pobieraniu oraz przeszukiwaniu

odpowiedniego folderu w którym znajdowały się odpowiednie pliki i tekstury. Ranges został użyty do sprawdzania vectorów , sortowania oraz wyświetlania konkretnych elementów. Moduły posłuży w programie jako interfejsy dla konkretnych klas.

4. Najważniejsze Klasy

- Game – jest to główna klasa w której dzieje się gra . W klasie Game metoda Run odpala główną pętle działania gry która wywołuje 4 różne aktualizacje. Główna aktualizacja to metoda Update która pozwala na aktualizacje obiektów gry i statystyk. HandleEvents wywołuje się kiedy gracz wykona konkretne akcje z klawiaturą , myszką lub kontrolerem. JoysticUpdate jest rozszerzeniem metody HandleEvents głównie dla kontrolera. Metoda Render co każdą klatkę Rysuje wszystkie obiekty widoczne w oknie gry.
- Ui – jest to klasa wirtualna która pozwala na wyświetlanie konkretnych menu w czasie gry
- Entity – jest to klasa wirtualna obiektów gry które zmieniają się w trakcie trwania rozgrywki
- FileStream – Jest to moduł poświęcony głównie wczytywaniu i zapisywaniu różnych typów danych
- SoundManager – Jest to klasa która zarządza muzyką i dźwiękami w trakcie grania rozgrywki

5. Istotne Algorytmy

- Fizyka i odbijanie piłki – Ten algorytm posiada obliczenia matematyczne aby piłka leciała pod danym kątem. Aby obliczyć dany kąt wystrzelenia pocisku, jest stosowana twierdzenie pitagorasa. Odbicie od bloku albo od krawędzi okna ustawia daną prędkość na danej osi na przeciwną (zmienia kąt lotu o 90 stopni w danym kierunku odbicia). Odbicie od platformy jest zależne w którym miejscu piłka uderzy platformę. Im bliżej krawędzi platformy tym pod większym kątem się odbije.

- Obliczania wyniku gry – Wynik danego poziomu jest obliczaniu w zależności od ilości zbitych bloków na danym poziomie oraz

od czasu ukończenia (im szybciej skończymy poziom , tym więcej punktów dostanie się do wyniku).

- Wczytywania i zapisywania danych – Odpowiada za to moduł FileSystem który pozwala na pobieranie i wczytywanie danych dla odpowiednich zmiennych

5. Testowanie

Podczas testowania program nie pokazał żadnych rażących błędów. Program był testowany pod kątem wycieków pamięci za pomocą systemowej komendy:

```
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF)
```

Komenda ta wyświetla informacje na temat wycieków pamięci, jedynie kiedy się pojawiają. Przykład:

```
Detected memory leaks!
Dumping objects ->
{263} normal block at 0x000001FD55A42210, 16 bytes long.
 Data: <   a            > B8 F4 D0 61 85 00 00 00 00 00 00 00 00 00 00 00
{262} normal block at 0x000001FD55A42940, 16 bytes long.
 Data: <x  a            > 78 F4 D0 61 85 00 00 00 00 00 00 00 00 00 00 00
{250} normal block at 0x000001FD55A45A20, 96 bytes long.
 Data: < C U       U    > F0 43 A3 55 FD 01 00 00 C0 B5 A3 55 FD 01 00 00
{249} normal block at 0x000001FD55A45570, 24 bytes long.
 Data: <HDQ^            > 48 44 51 5E F6 7F 00 00 01 00 00 00 01 00 00 00
{248} normal block at 0x000001FD55A42C10, 16 bytes long.
 Data: < Y U            > B8 59 A4 55 FD 01 00 00 00 00 00 00 00 00 00 00
{247} normal block at 0x000001FD55A423A0, 16 bytes long.
 Data: < Y U            > 90 59 A4 55 FD 01 00 00 00 00 00 00 00 00 00 00
{244} normal block at 0x000001FD55A45990, 80 bytes long.
 Data: < # U    spoko   > A0 23 A4 55 FD 01 00 00 73 70 6F 6B 6F 00 CD CD
{240} normal block at 0x000001FD55A458D0, 24 bytes long.
 Data: <HDQ^            > 48 44 51 5E F6 7F 00 00 01 00 00 00 01 00 00 00
{239} normal block at 0x000001FD55A42BC0, 16 bytes long.
 Data: < 6 U            > 18 36 A4 55 FD 01 00 00 00 00 00 00 00 00 00 00
{238} normal block at 0x000001FD55A42440, 16 bytes long.
 Data: < 5 U            > F0 35 A4 55 FD 01 00 00 00 00 00 00 00 00 00 00
{235} normal block at 0x000001FD55A435F0, 80 bytes long.
 Data: <@$ U    ok      > 40 24 A4 55 FD 01 00 00 6F 6B 00 CD CD CD CD CD
{231} normal block at 0x000001FD55A44F10, 24 bytes long.
```

Program nie wyświetlił żadnego wycieku pamięci.

6. Podsumowanie

Jestem bardzo zadowolony z stworzenia gry na podstawie Arkanoid oraz Break out. Jestem zadowolony z funkcjonowania gry i działania gry , jednak chciałbym w późniejszym czasie zastosować refaktoryzację oraz rozbudować grę. Podejście do zajęć w postaci zajęć przygotowanych przez innych studentów było bardzo pozytywnym doświadczeniem. Stworzenie kolejnej gry coraz bardziej przybliża mnie do dołączenia branży growej jako deweloper , programista lub jako designer.