Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów

Space Invaders

autor	Michał Pawłowski
prowadzący	dr hab. inż., prof. Roman Starosolski
rok akademicki	2020/2021
kierunek	informatyka
rodzaj studiów	SSI
semestr	4
termin laboratorium	wtorek 13:30
sekcja	1
termin oddania sprawozdania	2021-05-13

1 Temat

Gra Space Invaders wydana w 1978 to jedna z pierwszych gier komputerowych. Pierwotnie stworzona na dedykowane automaty do gry a następnie przeniesiona na urządzenia Atari 2600. Realizowany projekt polega na implementacji tej klasycznej gry 2D w języku C++.

2 Analiza tematu

Założenia gry są następujące. Kosmici poruszają się poziomo z jednakową prędkością, gdy znajdą się przy krawędzi planszy przenoszą się poziom niżej. Istnieje jednak pewna szansa, iż pojedynczy kosmita odłączy się od grupy i zaatakuje sam. Dodatkowo kosmici w sposób losowy oddają strzały w kierunku gracza. Nad grupą atakujących kosmitów od czasu do czasu przelatuje UFO, którego zestrzelenie gwarantuje dodatkowe punkty. Gracz przesuwa poziomo działo, którym stara się zestrzelić wszystkich kosmitów. Może się on schronić się za osłony, te jednak niszczą się wraz z zablokowanymi strzałami. Trafienie kosmity oznacza jego eliminację (usunięcie z planszy). Gracz otrzymuje 10, 20 lub 30 punktów za eliminację kosmity w zależności od jego rodzaju oraz 150 punktów i dodatkowe życie za eliminację UFO. Każdy poziom oznacza nową falę przeciwników, która różni się liczbą kosmitów. Gra kończy się gdy poziom życia gracza spadnie do 0 lub gdy kosmici dotrą na wysokość gracza, czyli do dołu planszy.

Program ma być uruchamiany w trybie okienkowym oraz wykorzystywać klawiaturę jako urządzenie wejścia. Dlatego wykorzystana została biblioteka SFML, która pozwala w prosty sposób zarządzać urządzeniami I/O oraz renderowaniem obrazu.

3 Specyfikacja zewnętrzna

3.1 Instrukcja dla użytkownika

Skompilowany program został umieszczony w folderze razem z plikami niezbędnymi do poprawnego działania. Aby uruchomić program należy wykonać plik spaceinvaders.exe

Po uruchomieniu programu pojawia się plansza startowa. Gra rozpoczyna się po naciśnięciu przycisku `enter`. Podczas gry użytkownik przesuwa działo w poziomie za pomocą klawiszy `left` i `right` oraz oddaje strzały za pomocą klawisza `spacja`. Informacje o bieżącym wyniku, poziomie oraz zdrowiu gracza wyświetlane są w górze ekranu. Po zakończeniu gry na ekranie pojawia

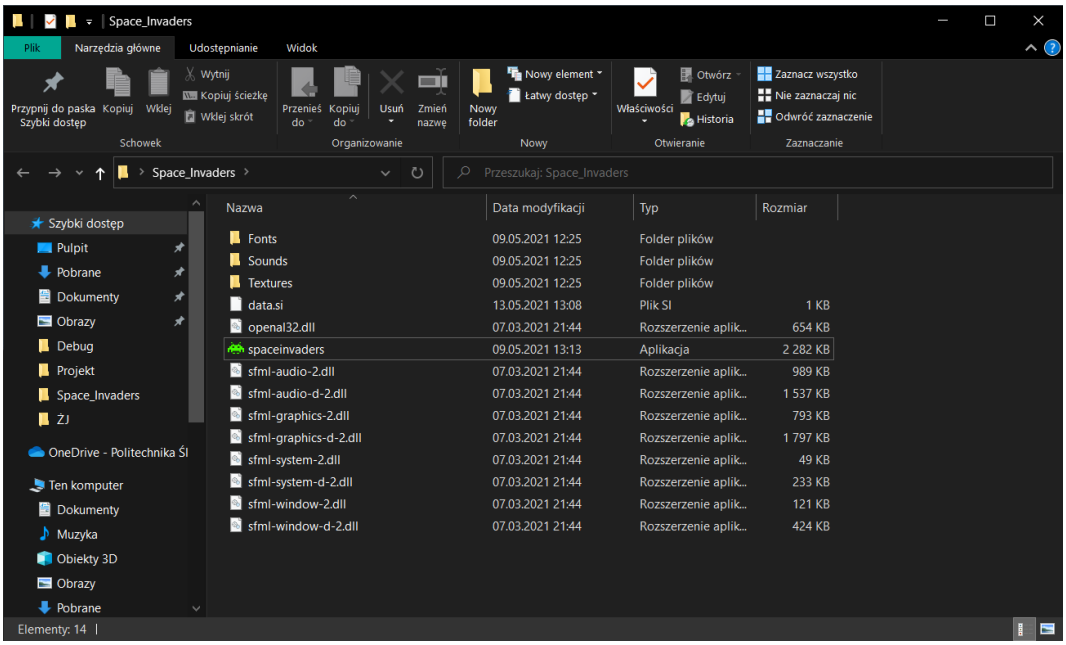
się informacyjna plansza z wynikiem uzyskanym oraz najlepszym w historii. Użytkownik może rozpocząć nową grę klawiszem `enter` lub wyjść z programu za pomocą `esc`.

Jeżeli podczas otwierania programu lub w czasie jego działania wykryty zostanie błąd, program automatycznie zakończy swoje działanie a informacja o nim zostanie wypisana na strumień błędów np.

```
ERROR[2]: Unable to open texture.
```

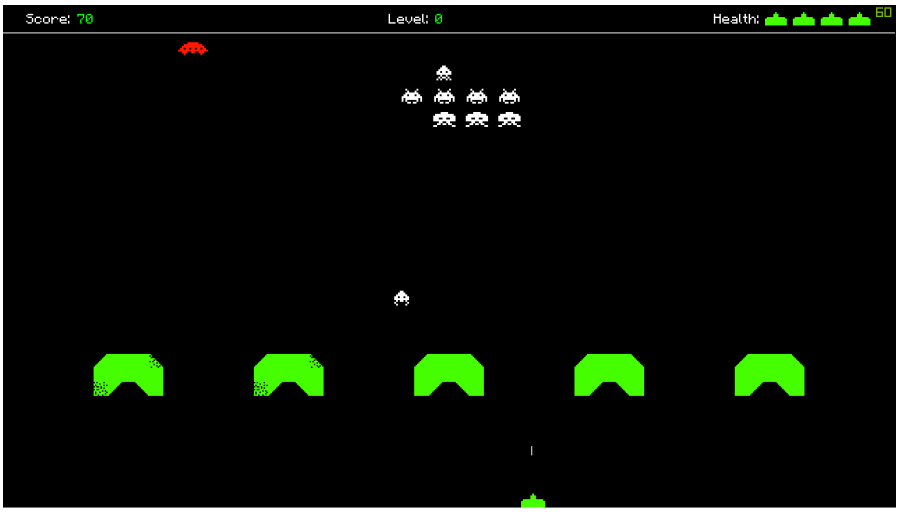
Uwaga! Do poprawnego działania programu niezbędny jest pakiet Microsoft Visual C++ 2015-2019 Redistributable (x64) w wersji 14.23.27820

3.2 Zrzuty ekranu

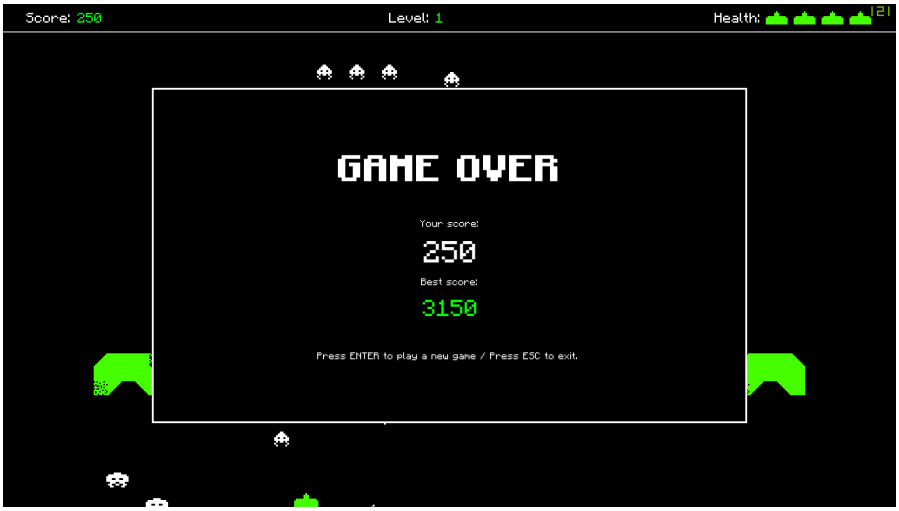


Rysunek 1: Katalog zawierający pliki gry.

Rysunek 2: Plansza startowa wyświetlana po uruchomieniu programu.



Rysunek 3: Interfejs gry.



Rysunek 4: Plansza końcowa wyświetlana po zakończeniu gry.

4 Specyfikacja wewnętrzna

4.1 Omówienie najważniejszych klas

W programie możemy wyróżnić co najmniej kilka najważniejszych klas. Silnik gry został zadeklarowany w klasie Game. Zawiera ona w sobie wszystkie obiekty, tworzone na potrzeby działania programu. Znajdziemy w niej zatem zarówno obiekty definiujące konkretne elementy gry jak i obiekty klas pochodzących z biblioteki SFML, odpowiedzialnych za otwieranie programu w oknie, renderowania itp. Metody zadeklarowane w tej klasie to zbiór metod stanowiących główną mechanikę gry.

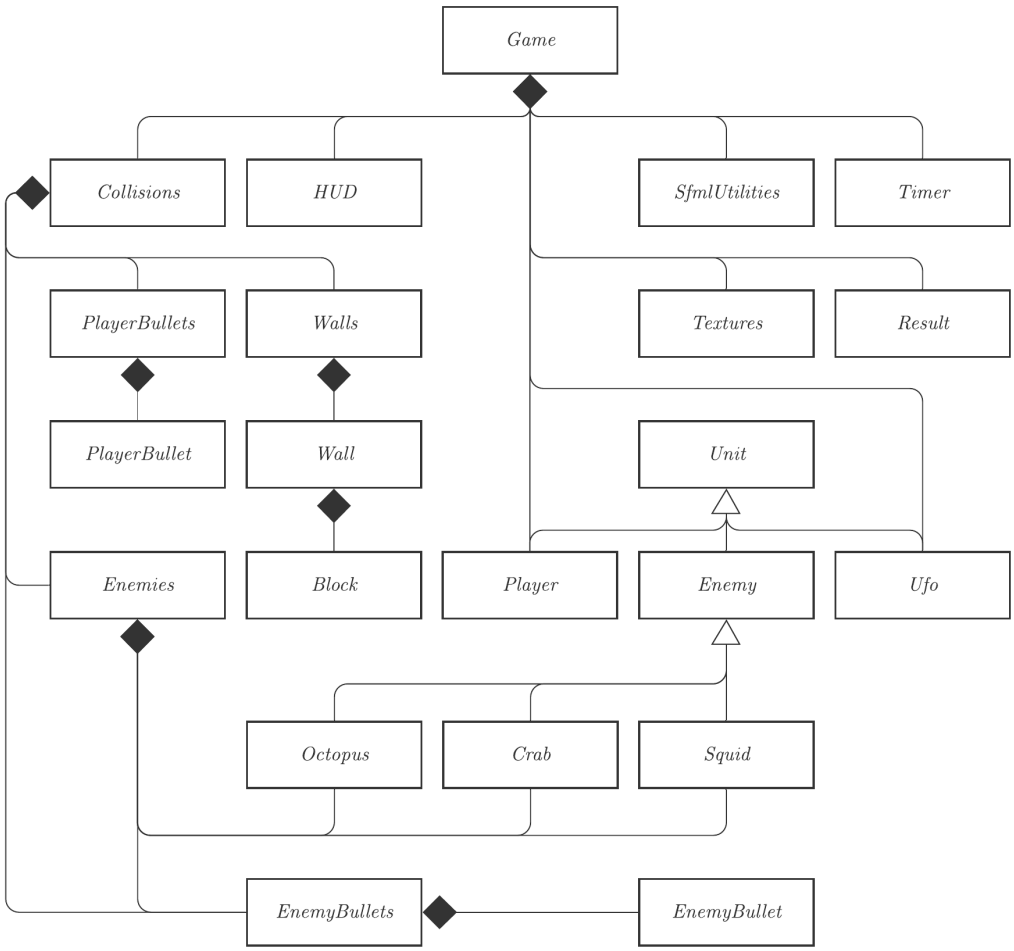
Zadaniem klasy Wall jest poprawne skonstruowanie i przechowywanie tablicy obiektów klasy Block, która definiuje pojedynczy blok osłony za którą może się chować gracz. Klasa Walls przechowuje w wektorze obiekty Wall oraz zarządza nimi.

Unit to abstrakcyjna klasa bazowa po której dziedziczą wszystkie klasy definiujące jednostki (postacie) z gry. Zawiera ona głównie pola i metody wywodzące się z biblioteki SFML. Bezpośrednio po tej klasie dziedziczy Player oraz Ufo. W pierwszej klasie zaimplementowane są metody umożliwiające interakcję z użytkownikiem, czyli sterowanie i strzelanie. Druga klasa definiuje obiekt UFO, który od czasu do czasu pojawia się na ekranie. Trzecią klasą dziedziczącą po Unit jest klasa abstrakcyjna Enemy. Stanowi ona klasę bazową dla klas Squid, Crab i Octopus. Klasy pochodne w różny sposób deklarują

metody wirtualne klasy bazowej, tak by zachowanie na planszy przeciwników innych klasy było zróżnicowane. Obiekty tych klas znajdują się w strukturze danych zawartej w klasie Enemies.

Klasa Collisions implementuje metody niezbędne do wykrywania kolizji pomiędzy wszelkimi obiektami.

4.2 Hierarchia klas



Rysunek 5: Diagram klas ULM.

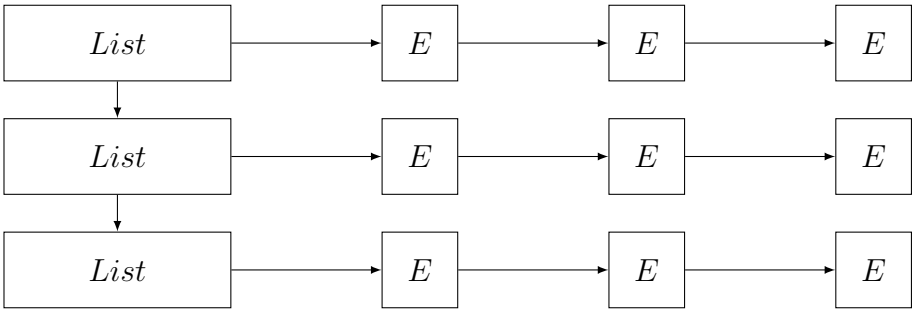
4.3 Szczegółowy opis klas i metod

Szczegółowy opis klas i metod zostanie wygenerowany przy pomocy Doxygen'a i załączony do finalnego sprawozdania.

4.4 Struktury danych

Ilość przeciwników jest zmienna w zależności od danego poziomu. Dlatego też obiekty klas pochodnych wirtualnej klasy Enemy są przechowywane w liście list znajdującej się w obiekcie klasy Enemies. Jest to największa struktura danych w programie. Oprócz niej wykorzystywane są pomniejsze struktury danych takie jak: mapy (do przechowywania wskaźników na tekstury) oraz wektory (do przechowywania obiektów klas Wall, Block, PlayerBullet , EnemyBullet.

Wszystkie struktury danych zostały stworzone w oparciu o kontenery STL typu list , vector czy map.



Rysunek 6: Przykładowy schemat listy list.

4.5 Ogólny schemat działania programu

Program rozpoczyna się od utworzenia obiektu klasy Game. W trakcie konstrukcji tworzone są również wszystkie zadeklarowane w niej obiekty. W obiekcie klasy Enemies zainicjowana zostaje struktura danych zawierająca obiekty klasy Enemy. Jej elementy stopniowo są usuwane wraz z eliminacjami dokonanymi przez gracza. Po przejściu do kolejnego poziomu struktura zostaje zainicjowana na nowo, tym razem z inną liczbą elementów. Program działa w nieskończonej pętli, dopóki okno aplikacji pozostaje otwarte, wywołując metody update() oraz render(). Pierwsza z nich odpowiedzialna jest za sczytanie sygnałów urządzeń wejściowych oraz wprowadzanie zmian w obiektach, tak by symulować postęp gry. Druga natomiast odpowiada za wy-

świetlanie zmian w oknie aplikacji. Przed zakończeniem działania programu zwolniona zostaje zaalokowana pamięć.

4.6 Biblioteki zewnętrzne i techniki obiektowe

Program opiera się o technikę programowania obiektowego. Wykorzystane zostały m.in.: klasy, klasy abstrakcyjne z metodami wirtualnymi, dziedziczenie, hermetyzacja, polimorfizm. W programie zawarte zostały również elementy języka C++ przedstawione na laboratoriach: mechanizm wyjątków, kontenery STL, algorytmy i iteratory STL oraz szablony funkcji. Program wykorzystuje bibliotekę Simple and Fast Multimedia Library (SFML).

5 Testowanie

Program był testowany zarówno na bieżąco podczas programowania jak i w finalnej wersji przez niezależne osoby. Podczas programowania wykryto typowe błędy programistyczne takie jak m.in.: wyjście po za zakres, odwołanie do usuniętego obiektu. Wykryte błędy skorygowano. Przetestowano wszystkie możliwe kolizje. Wynikające z nich usuwanie obiektów ze struktur danych, nie generuje problemów ani wycieków pamięci. Finalny program został przetestowany przez 3 niezależne osoby. Podczas testów nie wykryły one żadnych błędów.

6 Wnioski

Stworzenie prostej gry 2D daje możliwość przećwiczenia oraz wykazania się umiejętnościami programowania obiektowego. W porównaniu do projektów z poprzednich semestrów, wykorzystanie kontenerów STL zamiast struktur samodzielnie zaimplementowanych znacznie upraszcza i przyspiesza proces pisania programu. Wykorzystanie zewnętrznej biblioteki SFML pozwoliło na stworzenie programu w trybie okienkowym skupiając się na paradygmatach programowania obiektowego bez zagłębiania się w dokumentację OpenGL.

7 Link do repozytorium

https://github.com/polsl-aei-pk4/0fa12ac6-gr21-repo/tree/main/Projekt

Dodatek
Szczegółowy opis typów i funkcji