Michał Góral
Gr2. INF sem 4.

10.06.2020r. Katowice

Sprawozdanie z laboratorium
Programowania komputerów 4

Projekt semestralny

Super Mario Bros

Prowadzący:
dr inż. Piotr Pecka

1. Temat

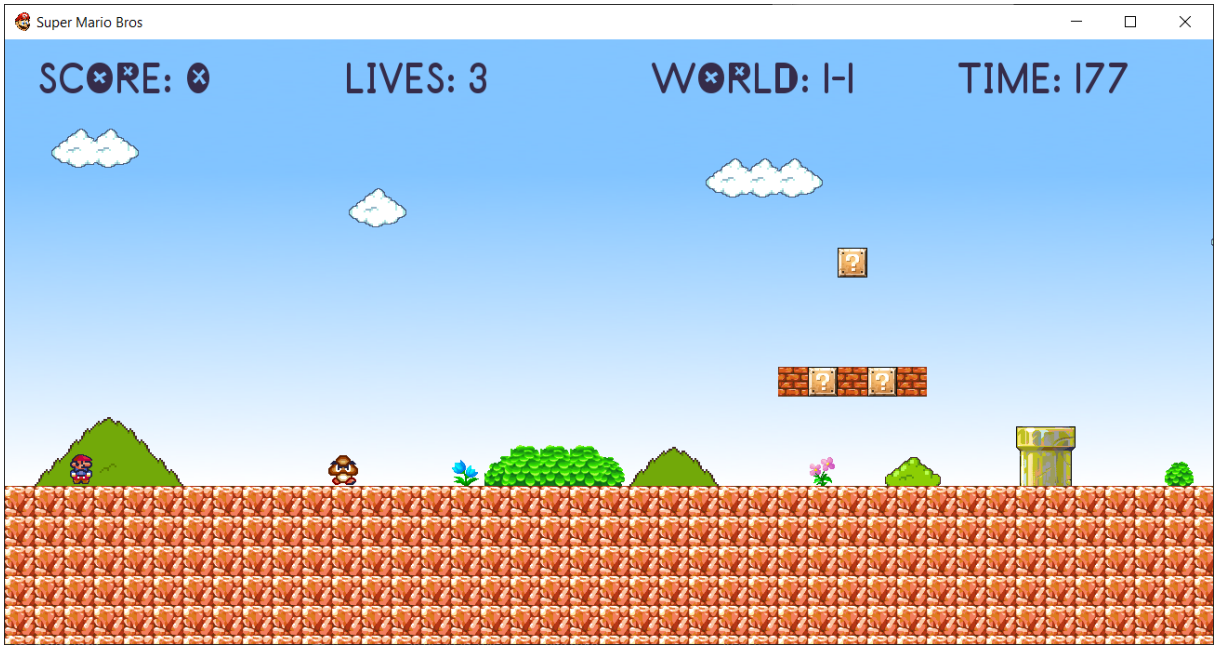
Tematem mojego projektu zaliczeniowego jest gra platformowa stylizowana na klasyczną grę firmy Nintendo Super Mario Bros.

2. Analiza tematu

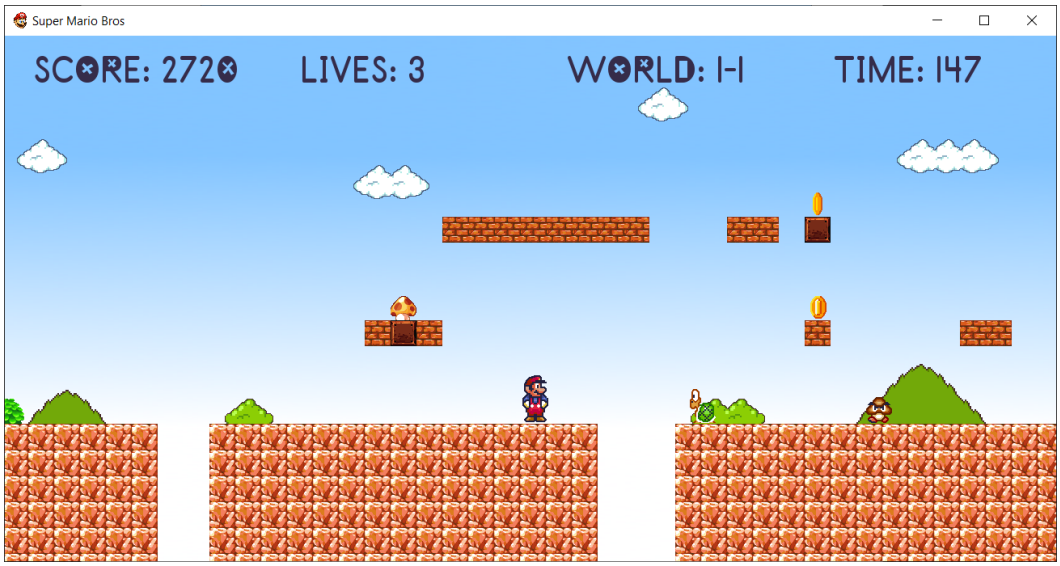
W grze naszym głównym celem jest dotarcie do zamku i uwolnienie księżniczki .Jako bibliotekę graficzną wybrałem *SFML 2.5.1*, ponieważ poznałem już ją w zeszłym semestrze i uznałem, że w zupełności ona wystarczy do prostej gry platformowej.

3. Specyfikacja zewnętrzna

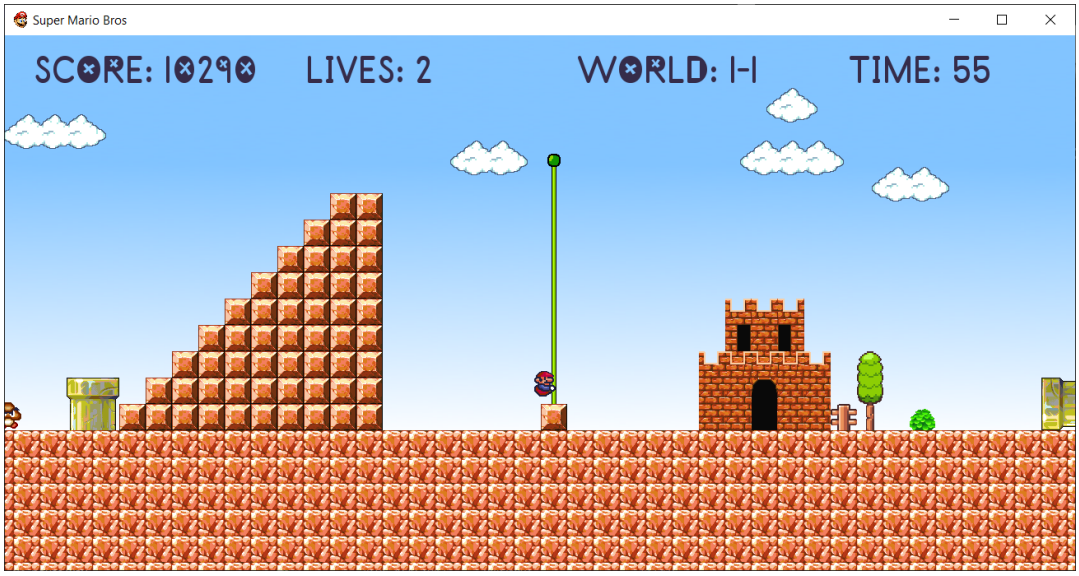
Mario poruszamy używając klawiszy strzałek lub klawiszy WSAD. Podczas gry będziemy napotykać wrogów. Możemy ich unieszkodliwić skacząc na ich głowy. Na mapie w blokach ze znakiem zapytania poukrywane są *power up'y* lub monety, jednakże możemy je również znaleźć w innych miejscach, z reguły dobrze ukrytych. Punkty można zdobywać zbierając monety, grzybki, powodujące zwiększenie rozmiaru Mario i dodanie życia, i gwiazdeczki, dające chwilową nieśmiertelność. Możemy niszczyć wybrane bloki po dotknięciu ich głową. Po dotarciu do zielonego masztu przechodzimy do kolejnego świata. Celem gry jest przejście światów, pokonanie po drodze wrogów, zebranie jak największej ilości punktów i dotarcie do zamku, w którym musimy pokonać Bowser'a, który rzuca kulami ognia mogące nas zabić.



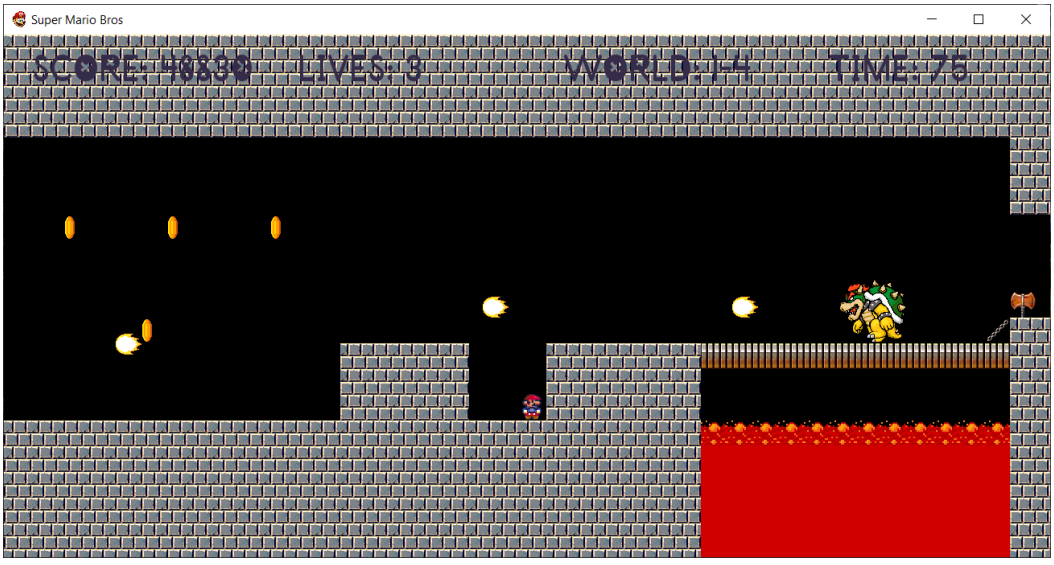
Rys. 1. Stan początkowy



Rys. 2. Mario po zebraniu grzybka, z widocznymi monetami i grzybkiem do zebrania



Rys. 3. Mario po ukończeniu poziomu



Rys. 4. Bowser miotający kulami ognia na ostatnim poziomie

Rys. 5. Ekran końcowy

4. Specyfikacja wewnętrzna

- `TileMap`

  Klasa ta zajmuje się generowaniem i wyświetlaniem mapy. Jest ona wyświetlana w formie kafelek. Każdy kwadracik jest wyświetlany jako osobny `Sprite`. Informację o kafelkach przechowuję w wektorze: `vector<vector<int>> Map;` i pobieram je z pliku. Klasa zajmuje się również rysowaniem *power up'ów* na ekranie oraz obsługą ruszających się platform. Posiada ona również atrybuty typu `Sprite` i `Texture` z biblioteki *SFML* dzięki którym możemy podziwiać ją na ekranie.

- `Mario`

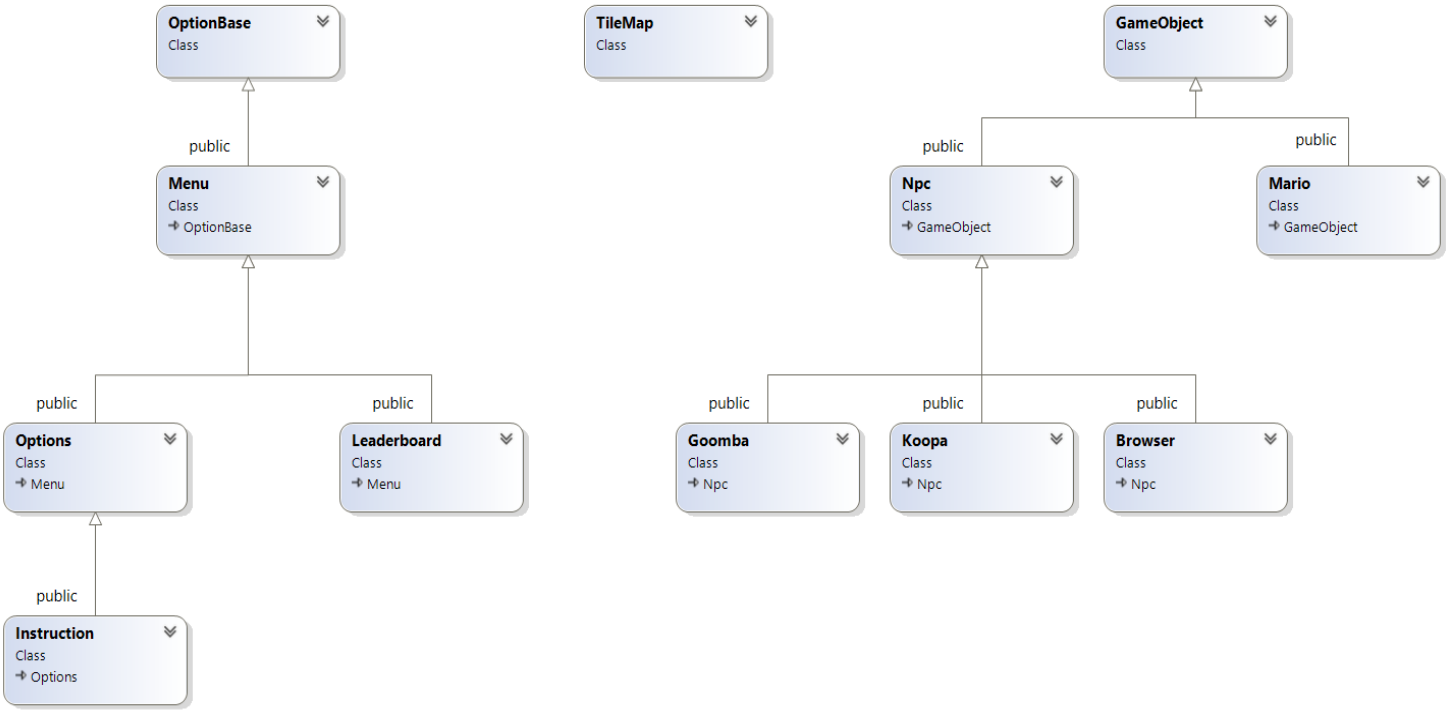
  Jest to najbardziej obszerna klasa. Dziedziczy ona po klasie `GameObject.` Zajmuje się ona głównie obsługą naszego bohatera. Dwie najważniejsze metody to: `void update(float deltaTime) override;` oraz `void intersects(TileMap& Map, float deltaTime) override;`. Zajmują się one odpowiednio poruszaniem po mapie i interakcje z mapą. Obiekt tej klasy przechowuje wszelkie informacje o stanie Mario. Na przykład atrybut `unsigned int lives;` zawiera informację o ilości żyć jaką Mario dysponuje. Posiada ona również atrybuty typu `Sprite` i `Texture` z biblioteki *SFML* dzięki którym możemy podziwiać ją na ekranie. Znajdują się tam także atrybuty i metody odpowiedzialne za dźwięki, które Mario wydaje.

- `Goomba, Koopa i Bowser`

  Obiekty tych klas odpowiadają za wyświetlanie i obsługę naszych wrogów. Klasa ta dziedziczy po `Npc`, która dziedziczy po `GameObject.` Tak jak w przypadku klasy `Mario` dwoma najważniejszymi metodami są:
  `void update(float deltaTime) override;` oraz
  `void intersects(TileMap& Map, float deltaTime) override;`.
  Dodatkowo zawierają one metodę
  `void isTouchingWithMario(Mario& mario, float deltaTime);`.
  Która z kolei jest odpowiedzialna za interakcje z naszą postacią. Atrybuty takie jak `vector<Vector2f> goombaPlaces;` przechowują niezbędne informacje o tych obiektach. Posiadają one również atrybuty typu `Sprite` i `Texture` z biblioteki *SFML* dzięki którym możemy podziwiać je na ekranie. Znajdują się tam także atrybuty i metody odpowiedzialne za dźwięki, które nasi wrogowie wydają.

- `Instruction, Leaderboard, Menu i Options`

  Obiekty tych klas są odpowiedzialne za obsługę gry podczas pauzy czy po jej uruchomieniu. Dzięki nim możemy sprawdzić krótką instrukcję gry lub dostosować ustawienia do swoich preferencji. Metody takie jak:
  `void moveUp() override;, void moveDown() override;,`
  `int getPressedItem() override { return selectedItemIndex;},` są odpowiedzialne za poruszanie się po menu.
  Obiekty tych klas przechowują informacje o tekstach do wyświetlenia np. w atrybutach takich jak:`Font font; Text menu[MAX_NUMBER_OF_ITEMS];`



Rys. 6 Diagram klas

5. Testowanie i uruchamianie

Większość błędów wykrytych przeze mnie dotyczyły biblioteki graficznej. Błędy natury algorytmicznej praktycznie wcale nie występowały, ponieważ poświęciłem sporo czasu na przemyślenie problemu. Duży odsetek błędów zdołałem wyeliminować. Niekiedy podczas zamiany w dużego Mario blokuje się on w bloku. Pomaga wtedy wykonanie skoku. Nie wystąpuje całkowita blokada, z której nie jesteśmy w stanie wyjść.

6. Wnioski

Projekt okazał się bardziej czasochłonny niż na początku założyłem. Większość problemów napotkałem podczas korzystania z biblioteki graficznej. Zachowywała się ona niekiedy w sposób losowy i niestety nie doszedłem do tego dlaczego tak się działo. Na przykład po wprowadzeniu tabulatora przed jedną metodą z klasy z biblioteki *SFML* program kompilował się jednak nic nie wyświetlało się. Po usunięciu tego tabulatora wszystko wracało do normy. Mimo to jestem usatysfakcjonowany swoją pracą. Napisanie gry z dzieciństwa było ciekawym doświadczeniem i końcowy efekt mnie zadowala. Prawdopodobnie ze względu na niewyjaśnione błędy biblioteki z której korzystałem nie będę chciał do niej wracać, a na pewno nie w najbliższej przyszłości.