Gliwice, 14.06.2018


# Sprawozdanie z projektu PK4

## Temat: Tower Defence

Prowadzący: dr inż Piotr Pecka

Bartłomiej Caban
Informatyka SSI, sem 4
grupa 5

# 1. Analiza tematu

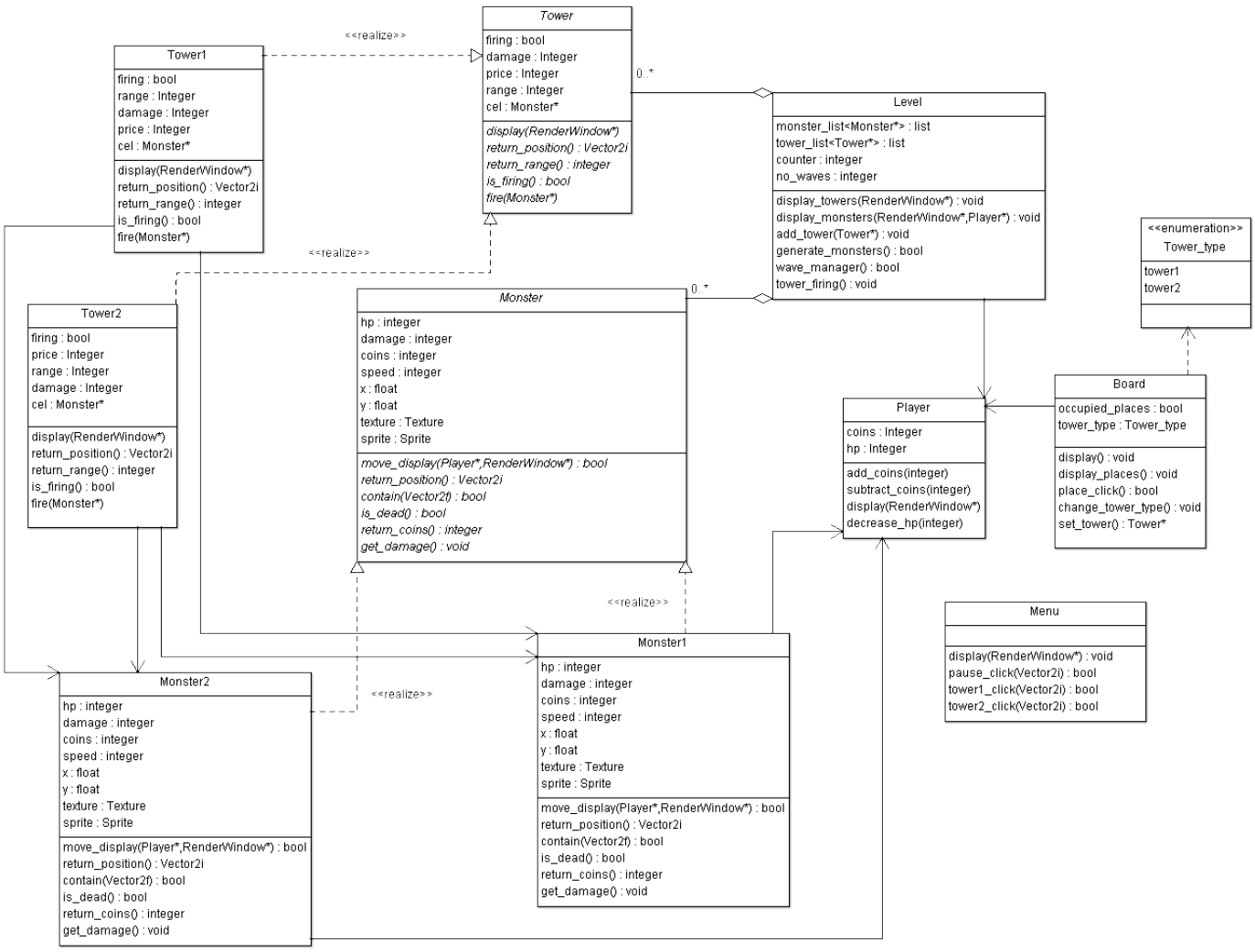
Zadanie polegało na stworzeniu gry typu tower defence. Polega ona na obronie swojej bazy przed falami wrogów za pomocą wież, które są ustawiane na ich drodze.
Program jest napisany w języku C++ za pomocą środowiska Microsoft Visual Studio 2015.
W celu dobrego przedstawienia gry użyto biblioteki graficznej SFML.
W programie zaimplementowano jedną dostępną planszę do gry. Są dwa rodzaje wież i wrogów, lecz ich ilość można bezprobleowo zwiększać.

# 2. Specyfikacja zewnętrzna

Aplikacja działa w trybie okienkowym. Do komunikacji z programem użytkownikowi jest potrzebna mysz. Po uruchomieniu gra zaczyna się od razu. Jedyna interakcja z użytkownikiem jest zredukowana do stawiania wież i ewentualnego włączenia/wyłączenia pauzy.

# 3. Specyfikacja wewnętrzna

**Diagram klas:**



## Opis klas i ważniejszych metod:

`class Board` – klasa jest wykorzystywana do przedstawienia planszy – zarówno grafiki, jak i interakcji pomiędzy nią, a użytkownikiem

| |
|---|
| `void display(RenderWindow * window);` |
| Wyświetla planszę w oknie. |
| `void display_places(RenderWindow * window);` |
| Wyświetla miejsca, w których można postawić wieże, jeśli gracz wybierze, którą strukturę chce zbudować. |
| `bool place_click(Vector2i pos);` |
| Sprawdza, czy gracz kliknął w odpowiednie miejsce na wieżę. |
| `tower * set_tower(Vector2i pos,Player * player);` |
| Metoda tworzy wieżę, ustala jej typ i pozycję, a następnie zwraca ją, aby mogła zostać dodana do listy. |

`class Menu` – klasa, która jest odpowiedzialna za wyświetlanie i obsługę menu, za pomocą którego można włączać pauzę i wybierać wieże

| |
|---|
| `void display(RenderWindow * window);` |
| Wyświetla menu – grafiki atrybutów gracza (zdrowia i ilości monet), wybór wież i klawisz pauzy. |
| `bool pause_click(Vector2i pos);` |
| Sprawdza, czy kliknięto pauzę. |
| `bool tower1_click(Vector2i pos); bool tower2_click(Vector2i pos);` |
| Sprawdza, czy kliknięto którąś z wież. |

`class Player` – klasa przechowująca atrybuty gracza.

| |
|---|
| `bool display(sf::RenderWindow * window);` |
| Wyświetla ilość życia i monet. |

`class Level` – klasa, która przechowuje listę wież i potworów za pomocą kontenera biblioteki stl - list. Jest odpowiedzialna za przebieg całej gry.

| |
|---|
| `void add_tower(Tower * new_tower);` |
| Dodaje wieżę przekazaną w parametrze do listy. |
| `void display_towers(RenderWindow * window);` |
| Wyświetla wieże (wywołuje u każdej wieży z listy metodę display()). |
| `void display_monsters(RenderWindow * window,Player * player);` |
| Wyświetla wrogów (wywołuje u każdego potwora z listy metodę move_display()). |
| `bool generate_monsters();` |
| Generuje potwory współpracując z metodą wave_manager(). |
| `void tower_firing();` |
| Dla każdej wieży z listy, która nie ma obecnie ustawionego celu wybiera najbliższego potwora. Wieża następnie bierze go sobie na cel i strzela do niego (wywołuje dla wieży metodę fire()). |
| `bool wave_manager();` |
| Generuje potwory współpracując z metodą generate_monsters(). |

`class Tower` – klasa abstrakcyjna reprezentująca wieżę
`class Tower1 :public Tower, class Tower2 :public Tower` – klasy reprezentujące konkretne typy wież, dziedziczą po `Tower`

| |
|---|
| `void display(RenderWindow * window);` |
| Wyświetla wieżę. |
| `bool is_firing();` |
| Metoda informuje o tym, czy wieża ma wybrany cel do ostrzału. |
| `void fire(Monster * monster);` |
| Ustawia cel do strzelania na potwora z parametru. |

`class Monster` – klasa abstrakcyjna reprezentująca potwora
`class Monster1 :public Monster, class Monster2 :public Monster` – klasy reprezentujące konkretne typy potwora

| |
|---|
| `bool move_display(RenderWindow * window, Player * player);` |
| Wyświetla potwora i realizuje jego przemieszczenie. |
| `bool contain(Vector2f arrow_pos);` |
| Sprawdza, czy sprajt potwora zawiera pocisk od wieży. |
| `void get_damage(int dmg);` |
| Odejmuje punkty życia. |
| `bool is_dead();` |
| Sprawdza, czy potwór powienien jeszcze żyć (hp<0). |

# 4. Testowanie i uruchamianie

Ze względu na bardzo zredukowane możliwości użytkownika jest niewielka szansa na wystąpienie błędów. Mimo wszystko program był wielokrotnie testowany i żadne błędy nie zostały wykryte.