Flanki 2D

Karol Hoerner de Roithberg
Kacper Syrnik
Jakub Urbański

14 czerwca 2022

1 Opis projektu

Celem projektu jest napisanie gry studenckiej o nazwie "flanki". Jest to gra w której dwie drużyny muszą jak najszybciej wypić swoje piwo po trafieniu w stojącą po środku między nimi puszkę. Piwo można pić jedynie kiedy puszka na środku leży przewrócona po rzucie zawodnika swojej drużyny. Gra kończy się gdy jednej z drużyn skończy się piwo.

Każdy gracz posiada swoją drużyne oraz za pomocą suwaka może trafić lub podnieść puszkę. Nad każdą z drużyn wyświetla się ilość pozostałego piwa.

2 Założenia wstępne przyjęte w realizacji projektu

W grze każda drużyna składa się z 3 studentów którzy na zmiane rzucają oraz biegają po puszkę. Każda drużyna ma nad sobą pokazaną ilość piwa pozostałą do wypicia.

Każdy z graczy może zrestartować grę klikając przycisk again, a także trafić lub podnieść puszkę za pomocą kliknięcia *Spacji* lub *Enter* - odpowiednio dla lewego lub prawego gracza.

3 Analiza projektu

3.1 Zdefiniowanie struktur danych

Wszystkie dane są przechowywane w klasach odpowiadającym odpowienim obiektom, np. za studenta odpowiada klasa *Student* lub za puszkę na środku odpowiada klasa *Beer*.

3.2 Specyfikacja interfejsu użytkownika

Użytkownik w dowolnym momencie może zrestartować grę klikając myszką w przycisk *Again*.
Dodatkowo użytkownik może wcielić się w jednego z graczy - rzucić puszką lub podnieść puszkę klikając w odpowienim momencie *Spację* lub *Enter* - odpowiednio dla lewego lub prawego gracza.

3.3 Wyodrębnienie i zdefiniowanie zadań

- Ustawienie tła, przycisku, suwaków oraz reszty elementuch widocznych na ekranie.
- Ustawienie animacji naszych studentów oraz puszek.
- Dodanie logiki suwaków oraz ilości piwa.
- Dodanie napisów oraz ostateczne poprawki.

3.4 Decyzja o wyborze narzędzi programistycznych

Do naszego projektu zdecydowaliśmy się użyć biblioteki SFML, ponieważ jest to dosyć szybka biblioteka dostarczająca przyjazny dla programisty interfejs. W zadaniu używaliśmy standardu C++17. Sam projekt robiliśmy przy użyciu programu *VisualStudio*2022, używając jego domyślnego kompilatora.

4 Podział pracy i analiza czasowa

Większość projektu zrobiliśmy wspólnie. Na początku ustawiliśmy tło. Dodaliśmy wszystkie potrzebne przyciski i suwaki. Następnie zajęliśmy się implementacją logiki związanej z naszymi postaciami oraz puszkami widocznymi w grze (rotacja, poruszanie się, rzut). W następnym kroku dodaliśmy logike suwaków oraz przycisku, zaimplementowaliśmy również logikę picia piwa (procenty spadające na kuflu piwa). Na końcu dodaliśmy pojawiające się podczas gry napisy oraz wprowadziliśmy ostatnie poprawki wizualne.

5 Opracowanie i opis niezbędnych algorytmów

W naszym projekcie nie wykorzystywaliśmy złożonych algorytmów. Program opiera się o różne flagi typu *bool*, dzięki czemu wie które operacje w danym momencie może wykonywać co przekłada się na to co użytkownik widzi na ekranie.

6 Kodowanie

6.1 Klasy

- **Beer** - klasa odpowiadająca za rysowanie puszek
- **Button** - klasa odpowiadająca za przycisk restartujący grę
- **Player** - klasa odpowiadająca za stan, w którym znajduje się aktualnie gracz
- **Rect** - klasa odpowiadająca za rysowanie prostokątów (nakładka na sf::RectangleShape)
- **SliderBeer** - klasa dziedzicząca po Beer, odpowiada za puszki na sliderach (logika jak się przemieszczają)
- **Student** - klasa odpowiadająca za animacje studentów
- **Text** - klasa odpowiadająca za pojawianie się napisów (nakładka na sf::Text).

6.2 Zmienne

Klasa **Beer**

- sf::Sprite sprite - zmienna przechowująca wygląd naszej puszki

Klasa **Player**

- double beerStatus - zmienna przechowująca ilość piwa danego gracza.
- bool turn - zmienna przechowująca informację czy jest kolej danego gracza.
- bool throwTurn - zmienna przechowująca informację czy jest kolej rzucania danego gracza.
- bool isDrinking - zmienna przechowująca informację czy aktualnie druzyna gracza pije piwo.
- Student* actualStudent - wskaźnik przechowujący adres studenta którego będzie rzucal lub podnosił piwo (animacja).
- Student* students[3] - tablica przechowująca wskaźniki do wszystkich studentów z jednej drużyny.

Klasa **Rect**

- sf::RectangleShape rectangle - zmienna przechowująca prostokąt do narysowania.

Klasa **SliderBeer**

- bool animationFlag - zmienna pomocnicza przechowująca wartość $True$, jeśli zgnieciona puszka na sliderze przemieszcza się w lewo, a gdy w prawo ma wartość $False$

Klasa **Student**

- int id - zmienna przechowująca id studenta.
- static const int width, height - zmienne przechowujące rozmiary obrazka studenta.
- sf::Sprite sprite - zmienna przechowująca wygląd naszego studenta.
- bool isFlipped - zmienna przechowująca informację, w którą strone zwrócony jest student.
- sf::Vector2f crushedCanPosition - zmienna przechowująca początkowe współrzędne rzucanej puszki (gdy znajduje się w ręce studenta).
- sf::Vector2f initialPosition - zmienna przechowująca współrzędne początkowe studenta.

Klasa **Text**

- sf::Text text - zmienna przechowująca tekst wypisywany na ekranie.

6.3 Funkcje

Każda klasa posiada *gettery* oraz *settery* które odpowienio zwracają lub ustalają wartości poszczególnych zmiennych (nazwa bezpośrednio mówi za co odpowiada) oraz *konstruktory* tworzące obiekty. Oprócz nich klasy mają również metody opisane poniżej.

- void draw - metoda rysująca nasze obiekty na ekranie

Klasa **Button**

- bool isMouseOver - metoda sprawdzająca czy mysz jest na przycisku.

Klasa **Player**

- void nextStudent - metoda zmieniająca wskaźnik aktualnego studenta po turze.

Klasa **SliderBeer**

- void animationSlider - metoda odpowiadająca za poruszanie się suwaka.
- void animationSliderReset - metoda odpowiadająca za reset suwaka po trafieniu lub podniesieniu puszki.

Klasa **Student**

- void nextSprite - metoda odpowiadająca za zmienianie 'klatki' w animacji studenta
- bool throwingAnimation - metoda odpowiadająca za animację rzucania puszką przez studentów
- bool raisingAnimation - metoda odpowiadająca za animację podnoszenia puszki przez studentów

7 Testowanie

Gra była testowana na bieżąco przy wykonywaniu każdego elementu. Po ustawieniu tła oraz obrazków sprawdzaliśmy wygląd naszej gry oraz poprawialiśmy go kilkukrotnie, póki nie był zadowalający.

Po dodaniu do naszej aplikacji logiki suwaków oraz przycisku także sprawdzaliśmy jej działanie uruchamiając grę. Jednocześnie sprawdzaliśmy działanie animacji ruchu postaci oraz rzucania puszką.

Ostatecznie dodaliśmy napisy oraz dokonaliśmy końcowych poprawek w przejrzystości kodu. Finalny test okazał się sukcesem - gra działa jak powinna.

8 Wdrożenie, raport i wnioski

W programie udało się wykonać wszystkie wymagania podstawowe oraz dodatkowe - obaj gracze mają możliwość rzutu w puszkę lub podniesienia jej, dzięki animacjom i napisom przejrzyste są następne kroki w grze.

Stan "upojenia" - czyli u nas zarazem ilość piwa (im mniej piwa tym większe "upojenie") - wprowadziliśmy poprzez zwiększenie prędkości przesuwającej się puszki na suwaku, przez co ciężej jest trafić w odpowiedni punkt. Bieg studentów po puszkę jest bardzo płynny.

Wszystkie wymagania zostały spełnione. Jedynymi rzeczami, które można by poprawić są przejrzystość oraz optymalizacja kodu.