Podstawy grafiki komputerowej

Projekt nr. 31

LABIRYNT 3D

1. Tytuł projektu i autorzy projektu

Celem projektu „Labirynt 3D” było stworzenie programu, który pozwala zmierzyć czas jaki potrzebuje użytkownik na pokonanie trójwymiarowego labiryntu.

Autorzy:

- Dominika Sarkowicz
- Weronika Węglińska
- Alicja Frankowicz

2. Opis projektu

Projekt zakłada stworzenie programu, który pozwala użytkownikowi na wczytanie labiryntu z pliku tekstowego stworzonego w dowolnym edytorze tekstowym. W labiryncie wszystkie ściany są do siebie prostopadłe, każdy segment ściany ma taką samą długość Ściany nie są w żaden sposób teksturowane. Głównym celem projektu jest mierzenie czasu, jaki jest potrzebny użytkownikowi do przejścia od punktu startowego do puntu końcowego. Czas jaki dotąd upłynął wyświetlany jest na bieżąco na ekranie, a po dotarciu do mety wyświetlany jest całkowity czas jaki użytkownik potrzebował na pokonanie labiryntu. W celu ułatwienia użytkownik może skorzystać z opcji wyświetlenia minimapy.

3. Założenia przyjęte w realizacji projektu

- Program wyświetla ściany labiryntu z perspektywy FPP – perspektywy pierwszej osoby.
- Ściany labiryntu są do siebie prostopadłe,
- Cały obszar labiryntu otoczony jest ścianami,
- Poruszanie odbywa się przy pomocy strzałek na klawiaturze (strzałka w górę – krok do przodu, strzałka w dół – krok do tyłu, strzałka w lewo – obrót w lewo, strzałka w prawo – obrót w prawo),
- Przed rozpoczęciem gry należy wybrać planszę,
- Możliwość wyświetlenia minimapy – jest to widok „z lotu ptaka” na labirynt (można ją włączyć tylko podczas gry)
- Możliwość zmiany field of view – pola widzenia gracza, od 45 do 120 stopni (można je zmieniać tylko przed rozpoczęciem rozgrywki)
- Możliwość przyspieszenia poruszania się poprzez wciśnięcie klawisza SHIFT (jednorazowe wciśnięcie klawisza powoduje zwiększenie prędkości gracza, ponowne wciśnięcie klawisza powoduje powrót do normalnej prędkości)
- Gra kończy się w momencie podejścia przez gracza do ściany symbolizującej wyjście (zaznaczonej na czerwono)

4. Analiza projektu

4.1 Specyfikacja danych wejściowych

W treści zadania został narzucony format danych wejściowych. Jest to plik tekstowy, który w pierwszej linii zawiera wymiary labiryntu: dwie liczby oddzielone przecinkiem. Kolejne linie przedstawiają układ ścian w labiryncie. Litera s oznacza miejsce w którym rozpoczyna się gra. Litera E oznacza wyjście z labiryntu. Litera X oznacza segment ściany a spacje reprezentują korytarze w labiryncie. Przykład:

```
5 , 5
X X X X X
X s       X
X X    X X
X           E
X X X X X
```

Każdy schemat labiryntu powinien być „obramowany" literami X, tak aby ściany uniemożliwiały wydostanie się gracza poza obszar labiryntu. W tym „obramowaniu" powinna się znaleźć jedna litera E symbolizująca wyjście. W schemacie powinna się znaleźć tylko jedna litera E i jedna litera s.
Należy zadbać o to aby podany rozmiar mapy zgadzał się z kolejnymi liniami, oraz aby rozwiązanie labiryntu było faktycznie możliwe (wyjście nie było zablokowane przez ścianę).

4.2 Opis oczekiwanych danych wyjściowych

W projekcie wyjściem jest graficzny interfejs udostępniający użytkownikowi funkcjonalności programu. Wynik działania programu wyświetlany jest na ekranie w postaci animacji przedstawiającej poruszanie się użytkownika po labiryncie. Dodatkowo wyświetlany jest czas jaki upłynął od rozpoczęcia rozgrywki oraz opcjonalnie minimapa. Po pokonaniu labiryntu wyświetlany jest całkowity czas gry w osobnym oknie dialogowym. Okna dialogowe wyświetlają się także przy nieodpowiednim korzystaniu z programu – np. przy próbie rozpoczęcia gry bez wyboru mapy labiryntu.

4.3 Zdefiniowanie struktur danych

Dane niezbędne do działania programu przechowywane są w instancji obiektu MapReader. Instancja tworzona jest w momencie rozpoczęcia rozgrywki. MapReader przechowuje mapę labiryntu w postaci macierzy integerów - Maze. Oprócz tego, klasa zawiera również pola typu double: ppos_x i ppos_y, które odpowiadają początkowej pozycji gracza na mapie, oraz pola typu integer: x i y, które przechowują informacje o wymiarach labiryntu. Podczas tworzenia instancji, z pliku wczytywana jest szerokość i wysokość labiryntu do zmiennych x i y, na podstawie których tworzona jest macierz o tych wymiarach. Następnie do tej macierzy zapisywane są wartości, na podstawie tych znajdujących się w pliku. Dla X: 1, a dla spacji i s 0. Współrzędne, w których znaleziono literę s, zapisywane są do zmiennych ppos_x i ppos_y.

MapReader umożliwia łatwy dostęp do informacji potrzebnych zarówno dla tworzenia widoku pierwszej osoby, jak i minimapy, dlatego nie ma potrzeby przechowywania w programie całego pliku.
Oprócz tego w programie przechowywane są zmienne typu bool – UP, DOWN, LEFT, RIGHT, które przechowują informację, który z klawiszów został wciśnięty, SHIFT, który odpowiada za określenie czy włączone jest przyspieszenie oraz canPlay, które określa czy znajdujemy się w trybie gry (czyli czy gracz może poruszać się po mapie – taka możliwość istnieje dopiero po wciśnięciu przycisku Start), zmienne typu double – rotSpeed, moveSpeed, które określają szybkość obrotu i poruszania się oraz posX, posY które określają położenie gracza na mapie, zmienna typu time_t startTime, która przechowuje informacje o czasie rozpoczęcia rozgrywki.

4.4 Specyfikacja interfejsu użytkownika

Program udostępnia użytkownikowi interfejs okienkowy. Z lewej strony okna znajduje się pole w którym wyświetlana będzie animacja poruszania się po labiryncie i minimapka. Po prawej stronie znajdują się elementy umożliwiające wczytanie mapy, rozpoczęcie rozgrywki i wybranie opcji jej dotyczących (minimapa i pole widzenia).

- Start – umożliwia rozpoczęcie rozgrywki, po jego kliknięciu w lewym obszarze pojawi się widok gracza. Jeśli przycisk zostanie kliknięty przed wybraniem mapy, pojawi się komunikat informujący o takiej konieczności a widok nie pojawi się.
- Wybierz planszę – otwiera okno dialogowe, które umożliwia wybór pliku konfiguracyjnego. Wyświetlane są tylko pliki o rozszerzeniu .map (Linker Adress Map), ponieważ jest to wymagany przez aplikację format.
- Zakończ – umożliwia przerwanie gry w dowolnym momencie, wyświetla komunikat informujący o zakończeniu gry.
- FOV – slider, który umożliwia wybór pola widzenia, z zakresu wartości 45-120 stopni. Przesuwanie w lewo zmniejsza pole widzenia, a w prawo – zwiększa.
- Minimapka – checkbox, umożliwia włączanie i wyłączanie minimapy.
- Dodatkowo na ekranie znajduje się timer, który informuje o czasie gry. Timer zaczyna odliczanie w momencie rozpoczęcia nowej rozgrywki.

4.5 Wyodrębnienie i zdefiniowane zadań

Projekt można podzielić na moduły:

- Stworzenie klas obiektów
- Odczyt danych z pliku wejściowego i ich zapisanie w instancji MapReader
- Komunikacja z użytkownikiem – wybór map, opcji, rozgrywka
- Prezentacja labiryntu w formie 3D – implementacja odpowiedniego algorytmu
- Poruszanie się po labiryncie
- Prezentacja minimapy
- Zmiana field of view
- Możliwość przyspieszenia

4.6 Wybór narzędzi programistycznych

Projekt zrealizowany jest w języku programowania C++. Do stworzenia interfejsu graficznego wykorzystano bibliotekę klas C++ wxWidgets dedykowaną do tworzenia oprogramowania różnych środowisk graficznych.

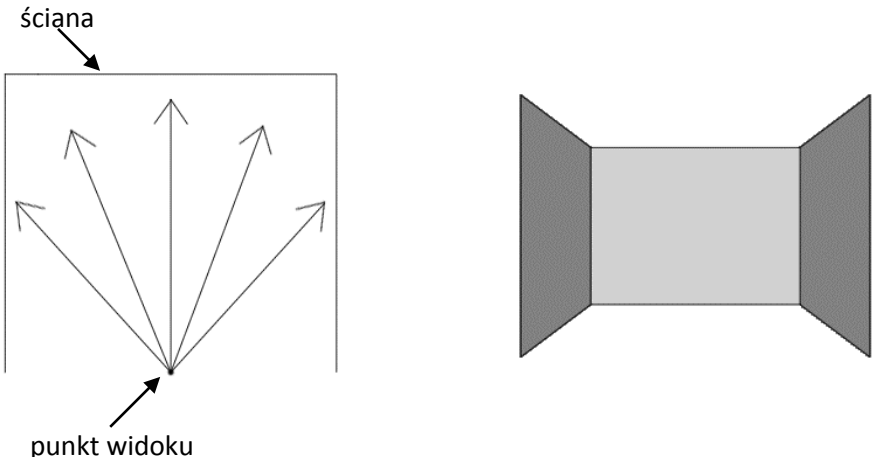
5 Podział pracy i analiza czasowa

Planowanie	
Wybór środowiska programistycznego	20.12.2016 – 21.12.2016
Projekt wyglądu interfejsu	20.12.2016 – 21.12.2016
Analiza realizowanych funkcjonalności	21.12.2016 – 23.12.2016
Opracowanie odpowiednich algorytmów	21.12.2016 – 23.12.2016
Kodowanie	
Przygotowanie przykładowych plików wejściowych	26.12.2016
Stworzenie layout-u interfejsu	26.12.2016
Implementacja klasy MapReader + odczyt danych z pliku	27.12.2016
Implementacja algorytmów wyświetlających widok 3D	28.12.2016 – 31.12.2016
Implementacja poruszania się	02.01.2017 – 05.01.2017
Implementacja wyświetlania minimapy	06.01.2017 – 08.01.2017
Implementacja timera	08.01.2017 – 10.01.2017
Implementacja field of view	11.01.2017 – 14.01.2017
Implementacja przyspieszenia	14.01.2017
Testowanie	
Testy funkcjonalne	15.01.2017 – 16.01.2017
Dokumentacja	
Opracowanie dokumentacji	17.01.2017 – 18.02.2017

6 Opracowanie i opis niezbędnych algorytmów

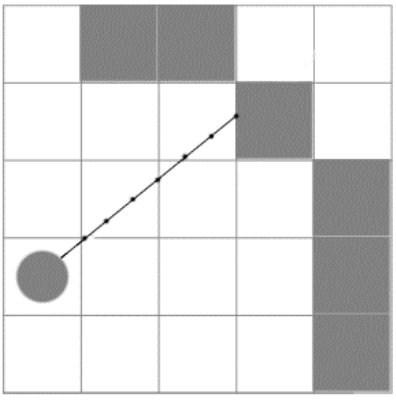
Aby zamienić mapę 2D labiryntu na widok trójwymiarowy, wykorzystany został algorytm o nazwie ray-casting.
Ray-casting umożliwia transformację dosyć ograniczonych danych (np. prostych map) w projekcję 3D poprzez wykorzystanie promieni, poprowadzonych od punktu widoku. W momencie napotkania ściany rysowana jest pionowa linia reprezentująca przeszkodę. W zależności od tego jak daleko od punktu widoku znajduje się ściana zmienia się długość rysowanej linii. Im dłuższy jest rzucony promień, tym krótsza będzie rysowana linia.

Dzięki wykorzystaniu algorytmu można przekształcić widok 2D, do widoku 3D

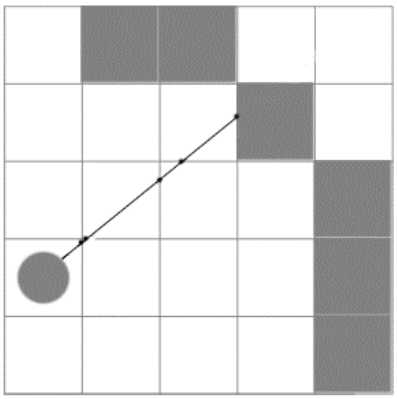


W algorytmie wykorzystane zostały współrzędne kartezjańskie.
Aby „umieścić" gracza na planszy, musimy znać jego współrzędne x,y. Przyjmujemy że „wzrok" znajduje się na wysokości połowy ściany.

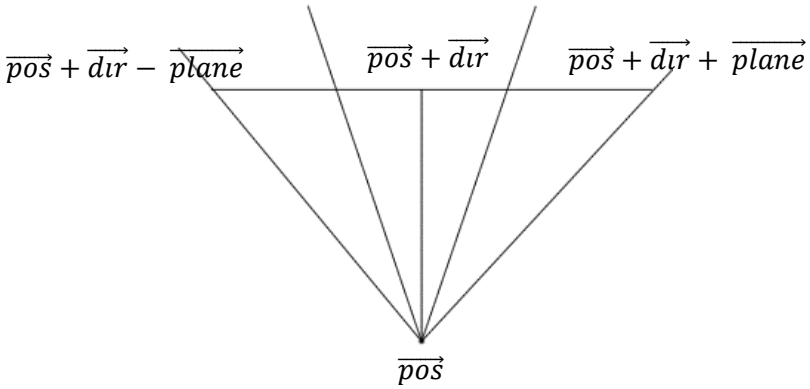
Aby znaleźć ścianę od punktu widoku, sprawdzane jest co pewien odstęp czy promień znajduje się w ścianie, jeśli tak (hit), to poszukiwanie kończy się, obliczana jest odległość i na jej podstawie rysowana linia o odpowiedniej wysokości:



Widać jednak, że aby nie pominąć żadnej ze ścian (to znaczy aby sprawdzenie nie dokonało się np. przed i za ścianą) należy przyjąć odpowiednio mały krok sprawdzania. Aby ominąć ten problem wykorzystuje się sprawdzanie czy ściana występuje w miejscu przecięcia się promienia z siatką mapy:

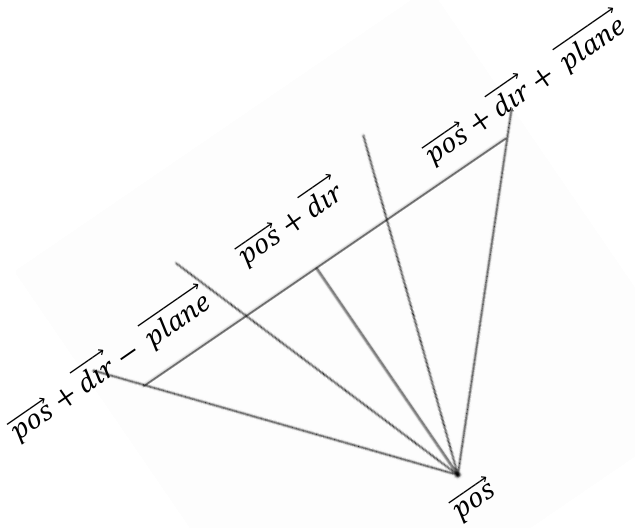


W ten sposób każda ściana zostanie wykryta.
Ponieważ pozycja gracza na planszy się zmienia, potrzebne są wektory określające zarówno jego położenie na mapie jak i kierunek w którym „patrzy" gracz. Do linii wyznaczającej kierunek gracza prowadzona jest prostopadle jeszcze jedna linia, która reprezentuje ekran komputera:



Linia pozioma reprezentuje ekran komputera (wektor plane), linia do niej prostopadła reprezentuje kierunek (wektor dir), linie ukośne to kilka promieni poprowadzonych z punktu widzenia (wektor pos). Kąt między wektorem z najbardziej z lewej i najbardziej z prawej determinuje pole widzenia.

Kiedy gracz obraca się wszystkie wektory także się obracają:



Obortu dokonać można poprzez wykorzystanie macierzy obrotu:

$$\begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix}$$

Poruszanie odbywa się poprzez modyfikowanie wektoru pos, reprezentującego pozycję gracza na mapie.

7 Kodowanie

Program stworzony został przy wykorzystaniu szablonu standardowego okna wxWidgets.
Struktura:

- Project1Frm – klasa dziedzicząca po klasie wxFrame. Odpowiada zarówno za funkcjonowanie GUI jak i za realizację algorytmu i wyświetlanie animacji na ekranie. Metody:
  - StartButtonClick – umożliwia rozpoczęcie rozgrywki
  - LoadButtonClick – umożliwia wybranie mapy
  - ExitButtonClick – umożliwia zakończenie gry
  - FOVSSliderScroll – odpowiada za zmianę pola widzenia
  - MiniMapClick – odpowiada za zmianę opcji wyświetlani a minimapy
  - OnTimerTimeout – odpowiada za licznik czasu rozgrywki
  - StartGame – odpowiada za stworzenie instancji klasy MapReader i zainicjowanie widoku
  - Draw() – odpowiada za rysowanie rozgrywki na ekranie
- MapReader – klasa odpowiedzialna za odczyt z pliku danych i przechowywanie ich w czytelnej formie. Metody:
  - Konstruktor – odpowiada za wczytanie danych z pliku
  - GetEement – umożliwia pobranie wartości z macierzy w miejscu przekazanych współrzędnych
  - getWidth – umożliwia pobranie szerokości mapy
  - getHeight – umożliwia pobranie wysokości mapy
  - getPlayerPosX – zwraca początkową współrzędna x gracza
  - getPlayerPosY – zwraca początkową współrzędną y gracza

8 Testowanie

Program poddany został testom funkcjonalnym. Dla map zawartych w folderze Maps sprawdzono:

- Czy mapa wyświetlana jest prawidłowo i czy można się po niej poruszać
- Czy użytkownik nie może wchodzić w ściany
- Czy czas jest liczony prawidłowo
- Czy dotarcie do czerwonej ściany powoduje zakończenie gry

Podczas testowania nie wykryto żadnych nieprawidłowości.

9 Wdrożenie, raport i wnioski

Wszystkie podstawowe wymagania projektu zostały zrealizowane. Dodatkowo udało się zrealizować podgląd mapy z góry, wyświetlany w postaci minimapy, możliwość zmiany pola widzenia i przyspieszania. Podczas pracy nie pojawiły się większe problemy. W przyszłości program mógłby zostać rozszerzony o dodatkowe funkcjonalności, np. dodanie opcji (zmiana klawiszy sterowania, itp), wprowadzenie różnych kolorów ścian.