AKADEMIA GÓRNICZO-HUTNICZA IM.
STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji

Projekt i realizacja gry w przestrzeni nieeuklidesowej

Autor: Michał Flak (294309)

Opiekun: dr. inż. Witold Alda

Dokumentacja specyfikacyjna

Komentarz ogólny

Specyfikacja aplikacji typu „gra” będzie różniła się od typowego przykładu systemów opisywanych na zajęciach, zwykle aplikacji webowych. Na przykład przypadek użycia będzie tylko jeden – granie, więc nie ma sensu opisywać go przesadnie. Gracz może poruszać się w przód, w tył, na boki, obracać się oraz zmieniać widok na widok z góry – rzeczy raczej trywialne.

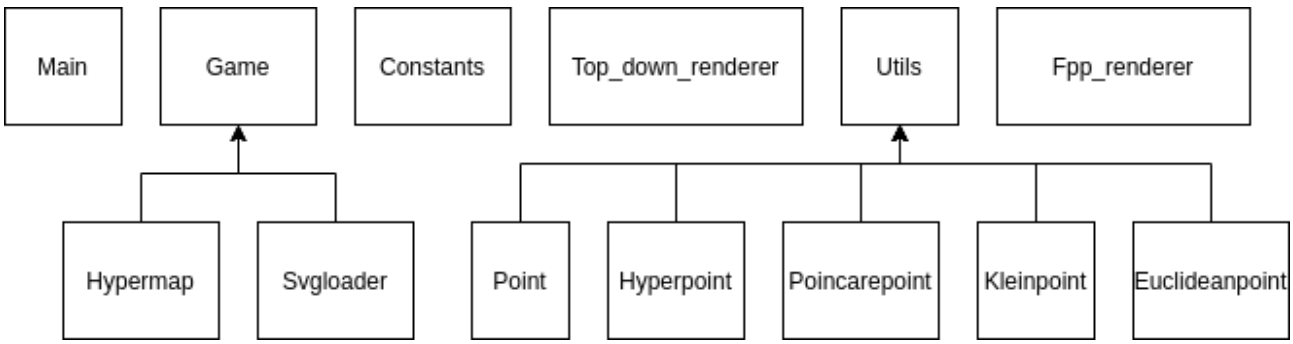
Język, w którym zdecydowałem się napisać projekt (Rust), nie jest też językiem obiektowym – jak również nie używałem go w sposób zorientowany obiektowo - trudno więc miejscami zamapować zaimplementowane koncepcje, na koncepcje poznanie np. w UML, jak na przykład diagram klas.

Kod gry jest ustrukturyzowany w moduły, które są jedynym poziomem organizacji kodu w języku Rust. Opiszę więc zależności między modułami tak jak występują one w projekcie.

Należy mieć na uwadze, że pojęcie Rustowego modułu w żaden sposób nie odpowiada pojęciu klasy – moduł może zawierać:

- definicje struktur
- stałe
- definicje funkcji
- definicje traitów (interfejsów)
- implementacje traitów na strukturach

Hierarchia modułów:



Rysunek: Hierarchia modułów

Specyfikacja modułów:

1. Main

Główny moduł programu. Zawiera funkcję main.

Odpowiada za:

- Inicjalizację okna i biblioteki graficznej

- Wczytanie czcionki

- Inicjalizację modułu mapy i gry

- Inicjalizację modułów rendererów

- Obsługę wejścia-wyjścia

- Obsługę głównej pętli gry

- Wywoływanie odpowiedniego renderera

2. Game

Moduł rozgrywki.
Odpowiada za:

- Obsługę ruchu i obrotu gracza

- Obliczanie i wyświetlanie aktualnego wyniku

- Obsługę kolizji

2.1. Hypermap

Wewnętrzna reprezentacja świata gry. Trzyma stan wszystkich ścian i obiektów. Gracz jest trzymany zawsze w środku układu współrzędnych.
Zawiera:

- Listę ścian HyperWall

- Listę obiektów HyperObject

2.2. Svgloader

Odpowiada za:

- Tworzenie HyperMap na podstawie podanego pliku SVG

3. Utils

Zawiera implementacje struktur:
- Punktu

- Ściany

- Obiektu

w różnych modelach przestrzeni, oraz matematyczne funkcje pomocnicze i umożliwiające na zamianę pomiędzy reprezentacjami.

3.1. Hyperpoint

Reprezentacja Punktu, Ściany, Obiektu w modelu hiperboloidu Minkowskiego.
Używany do wewnętrznej reprezentacji świata gry.

Zawiera definicje struktur:

- Hyperpoint
- Hyperwall
- Hyperobject

3.2. Poincarepoint

Reprezentacja Punktu, Ściany, Obiektu w modelu dysku Poincarego.
Obecnie nieużywany.

Zawiera definicje struktur:

- Poincarepoint
- Poincarewall
- Poincareobject

3.3. Kleinpoint

Reprezentacja Punktu, Ściany, Obiektu w modelu dysku Beltramiego-Kleina.
Używany do:
- Renderownia widoku z góry

- Pośrednio do renderowania widoku z pierwszej osoby

Zawiera definicje struktur:

- Kleinpoint
- Kleinwall
- Kleinobject

3.4. Euclideanpoint

Reprezentacja Punktu, Ściany, Obiektu w zwykłej przestrzeni euklidesowej.
Używany do renderowania widoku z pierwszej osoby, w celu stworzenia sceny dla silnika Macroquad.

Zawiera definicje struktur:

- Euclidianpoint
- Euclidianwall

- Euclidianobject

3.5. Point

Definicja traitu (interfejsu) Point – opisuje zachowania, które powinna udostępniać każda z implementacji punktu, takie jak:

- Odległość do środka układu współrzędnych

- Funkcja do stworzenia punktu w środku układu współrzędnych

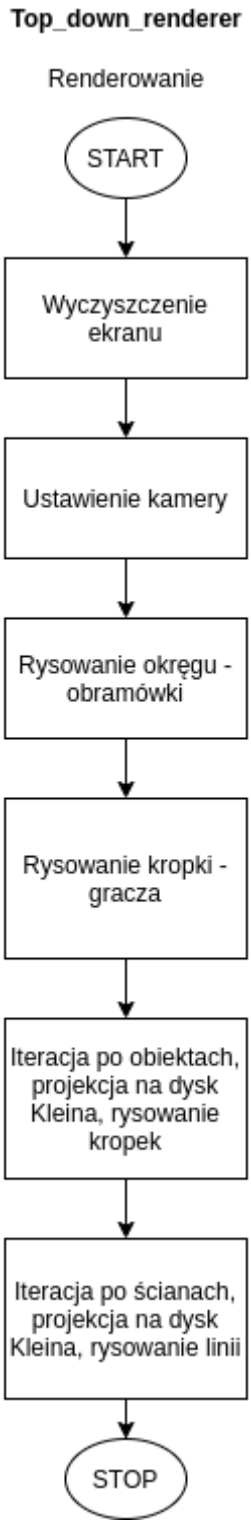
- Kąt względem osi X układu współrzędnych

4. Top_down_renderer

Odpowiada za renderowanie minimapki (widoku z góry).

Zawiera definicję struktury o tej samej nazwie co moduł.

Udostępnia konstruktor oraz jedną funkcję (render), przyjmującą strukturę Game i renrerującą stan gry na ekran, w postaci dysku Beltramiego-Kleina.



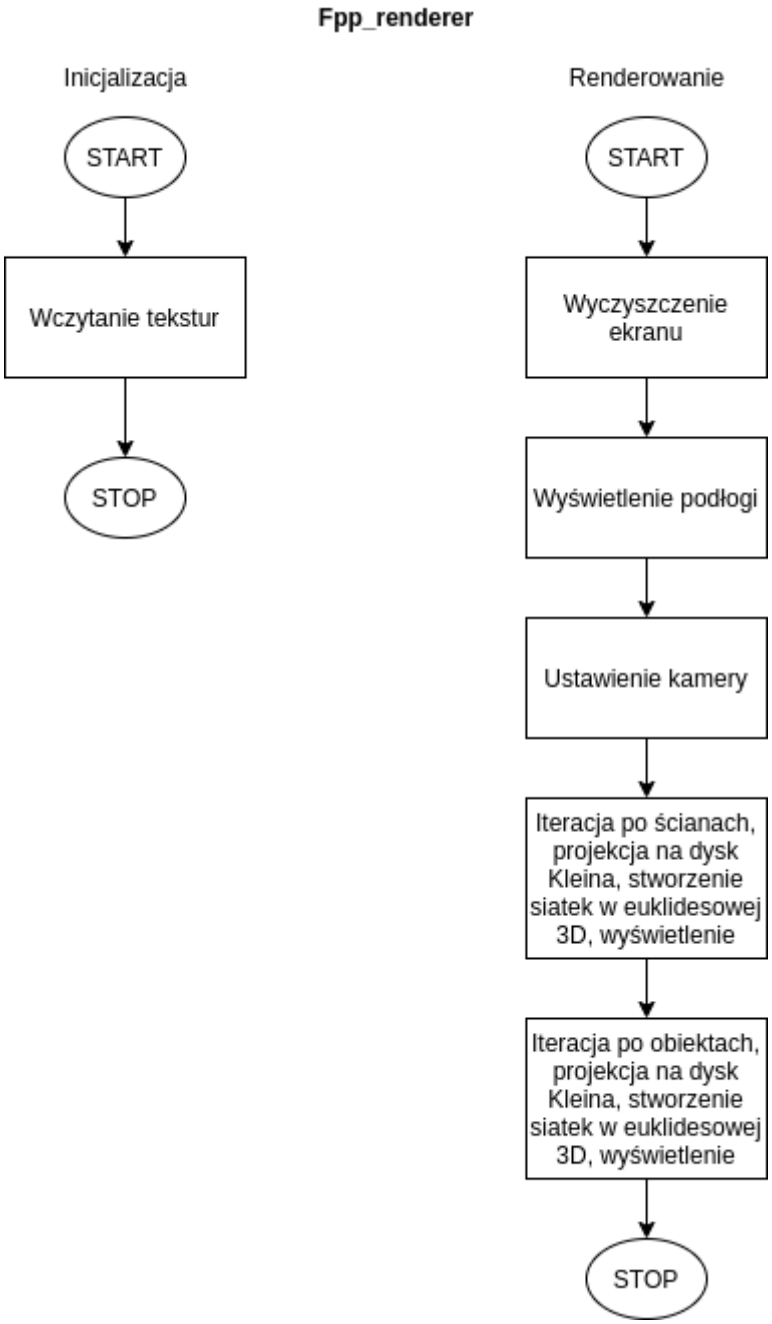
Rysunek: Działanie modułu Top_down_renderer

5. Fpp_renderer

Odpowiada za renderowanie widoku z pierwszej osoby.

Zawiera definicję struktury o tej samej nazwie co moduł.

Udostępnia konstruktor (ładujący tekstury) oraz jedną funkcję (render), przyjmującą strukturę Game i renrerującą stan gry na ekran, w postaci rzutu dysku Beltramiego-Kleina na trójwymiarową przestrzeń euklidesową.

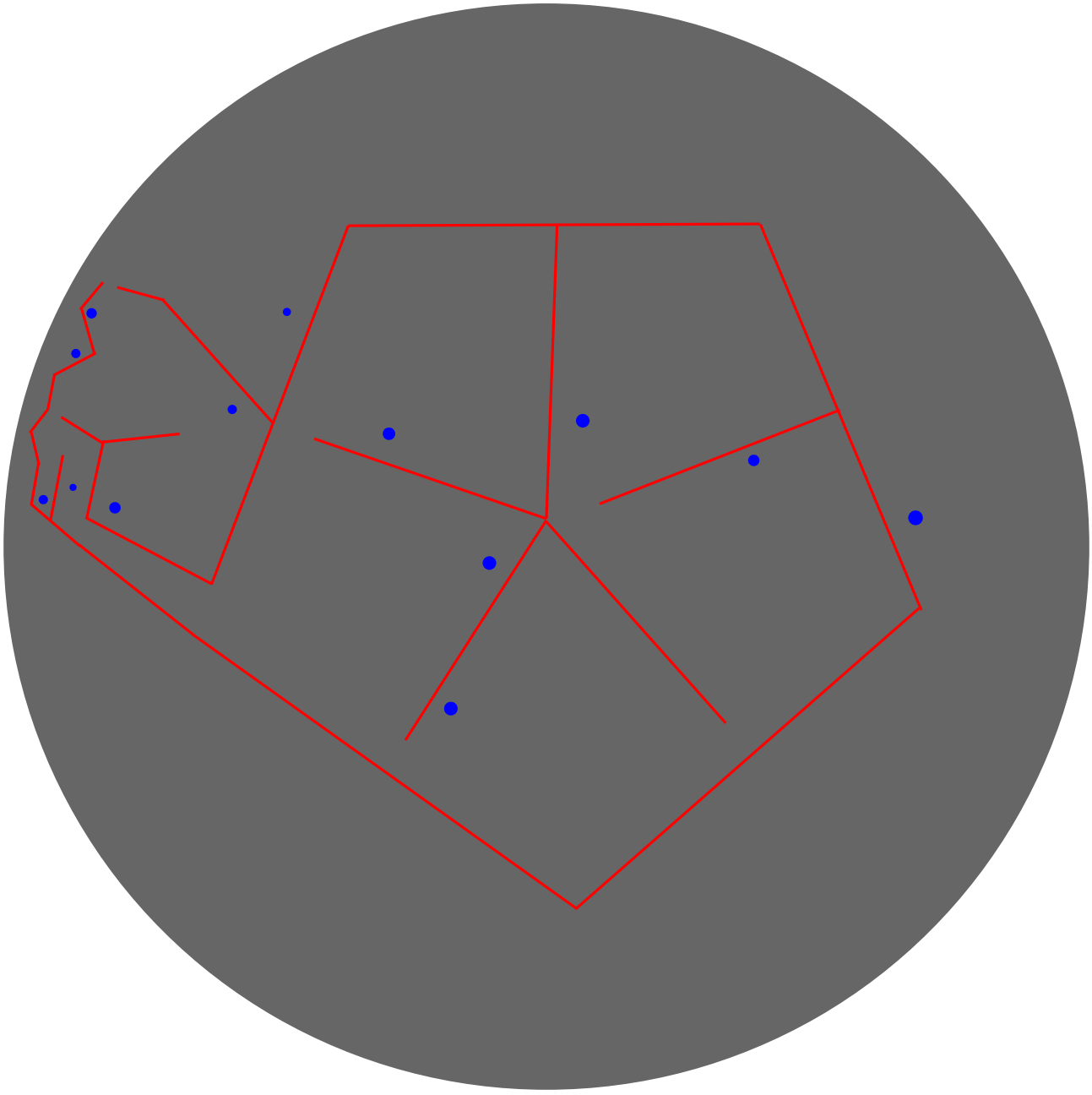


Rysunek: Działanie modułu Fpp_renderer

6. Constants

Zawiera stałe i parametry dostępne dla wszystkich pozostałych modułów:

Specyfikacja formatu danych - mapa (SVG)



Rysunek: Przykładowa mapa w postaci grafiki SVG.

Gra oczekuje pliku SVG o wymiarach 2000x2000mm.

Plik ten odpowiada dyskowi Beltramiego-Kleina, który ma promień 1 – współrzędne są więc odpowiednio przekształcane.

Gra iteruje po tagach <line>:

```
<line class="wall6" fill="none" id="svg_6" stroke="#ff0000" stroke-width="5" x1="635" x2="382.89449" y1="409.2105" y2="1069.73681"/>
```

tworząc na ich podstawie ściany. Tekstura ściany wybierana jest na podstawie atrybutu „class”. Możliwe wartości tego atrybutu:

- wall1
- CONCRETE
- wall2
- wall3
- wall4
- wall5
- wall6

Wartości x1, x2, y1, y2, odpowiadają współrzędnym początku i końca.

Znajdźki (obiekty) tworzone są następująco:

Gra iteruje po tagach <ellipse>:

```
<ellipse cx="1066.99976" cy="768.25" fill="#0000ff" id="svg_13" rx="10" ry="10" stroke="#0000ff" stroke-width="5"/>
```

cx, cy oznaczają współrzędne znajdźki.