AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE

OPIS PROBLEMU, WIZJA ROZWIĄZANIA, KONCEPCJA SYSTEMU

AUTOR: MICHAŁ FLAK

DOKUMENTACJA WYKONANA W RAMACH PRZEDMIOTU PRACOWNIA PROJEKTOWA

PROWADZĄCY: MGR INŻ. WITOLD RAKOCZY

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI
KRAKÓW, 2020

OPIS PROBLEMU

Innowacyjne mechaniki są zawsze mile widziane wśród graczy. Zauważyłem jedną obiecującą alejkę która nie została dostatecznie dobrze eksplorowana: Przestrzenie nieeuklidesowe.

Proponuję projekt stworzenia prostej gry z widokiem pierwszej osoby opartej na tzw. Raycastingu z grafiką 2.5D, która odbywa się na zakrzywionej płaszczyźnie – konkretnie hiperbolicznej, o krzywiźnie ujemnej. Gra da graczowi doświadczenie zupełnie nowe I nieintuicyjne – nie ma linii równoległych, trójkąty mają mniej niż 180 stopni, idąc w prawo, przód, lewo, tył, nie wraca się na to samo miejsce I wiele podobnych zjawisk. Wyzwaniem będzie też projektowanie poziomów które zademonstrują odmienność tego świata.

Z przeszukania Internetu wynika że do tej pory tylko jedna gra – HyperRogue – podjęła się tego tematu, jest tam jednak widok z góry, co psuje immersję. W produkcji jest też inna – Hyperbolica – nie została jednak jeszcze wydana, więc będzie to poniekąd nowatorskie.

WIZJA ROZWIĄZANIA

OGÓLNA WIZJA

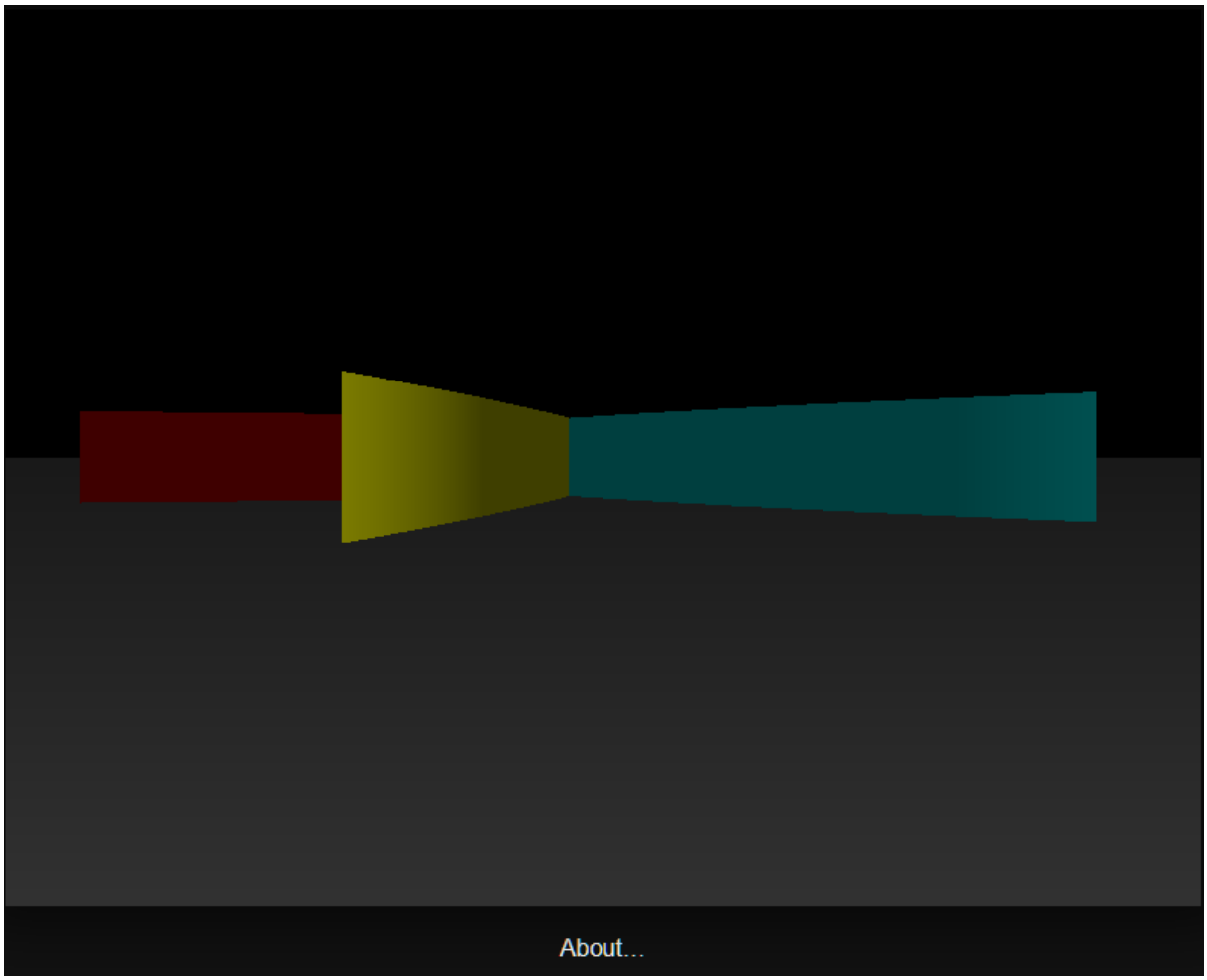
Chcę stworzyć prostą grę, uruchamiającą się na desktopie I w przeglądarce, która byłaby strzelanką pierwszoosobową na wzór gier w rodzaju Doom czy Wolfenstein 3D.

Zamierzam również zaprojektować zestaw poziomów które zaskakiwałyby intuicyjne oczekiwania gracza, wykorzystując I eksponując to co w tym świecie wyjątkowe I nieprzewidywalne.

Istnieje wiele mechanik które są już rozwiązanymi problemami które mógłbym zawrzeć – teksturowanie, zmienna wysokość ścian, multiplayer, dynamiczne oświetlenie przychodzą na myśl.

STUDIUM WYKONALNOŚCI

Wykonalność projektu została już sprawdzona I potwierdzona poprzez wykonanie prototypu:



Dostępny on jest do wypróbowania w przeglądarce: https://elo-siema.github.io/hyperbolic-raycaster-rust/index.html

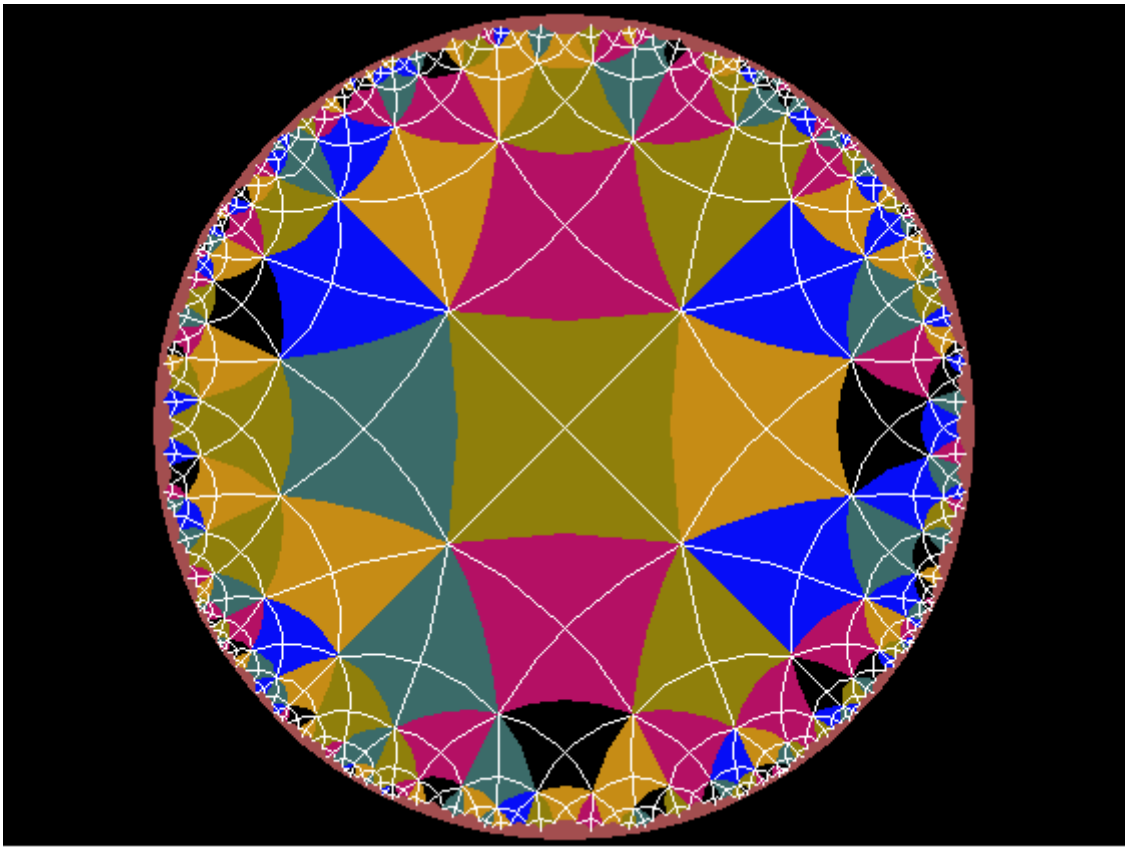
Kod źródłowy: https://github.com/elo-siema/hyperbolic-raycaster-rust

Napisany został w języku Rust, z wykorzystaniem bibloteki SDL2. Jest kompilowany do przeglądarki (WebAssembly / Emscripten) oraz do desktopowej aplikacji okienkowej.

Podczas tworzenia prototypu ujawniło się kilka problemów, do których znalazłem rozwiązania:

1. Wybór modelu przestrzeni / układu współrzędnych do łatwej edycji mapy

Początkowo rozważałem system oparty na siatce powstałej w wyniku tesselacji dysku Poincaré – numerując poszczególne kwadraty, ale szybko stało się jasne, że to rozwiązanie nie skaluje się oraz jest niezwykle niewygodne do projektowania poziomów:



Następnym pomysłem było stworzenie mapy jako zbioru ścian, z czego każda ma 2 punkty – początek I koniec. Punkty reprezentowane są we współrzędnych dysku Poincaré – pozwala to na łatwe tworzenie poziomu w edytorze graficznym, widząc świat w rzucie z góry. Przykładowy poziom z dwiema ścianami:

```
[
    {
        "beginning":  [0.0, 0.032],
        "end":  [0.0, 0.31],
        "color": {
            "red": 255,
            "green": 0,
            "blue": 0
        }
    },
    {
        "beginning":  [0.0, 0.032],
        "end":  [0.270, 0.129],
        "color": {
            "red": 255,
            "green": 255,
            "blue": 0
        }
    }
]
```

Tworzy to pewien problem – postrzegane odległości zmniejszają się wraz z oddalaniem od środka układu współrzędnych, jest to jednak ciągle najlepsze rozwiązanie jakie widzę.

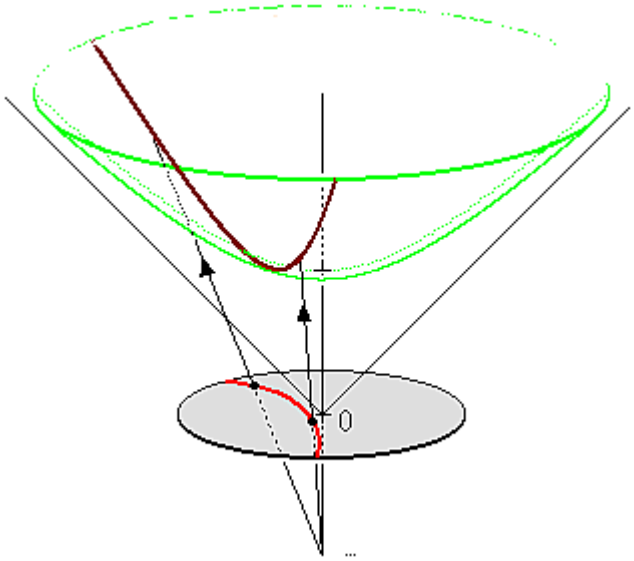
2. Wybór modelu przestrzeni do reprezentacji wewnętrznej świata gry

Tym razem priorytetem była łatwość dokonywania transformacji. Początkowo posiłkowałem się materiałami developerskimi z tworzenia gry Hyperbolica, gdzie reprezentacja świata trzymana jest w modelu dysku Poincare, tworzy to jednak spory problem z transformacjami – wymagają użycia żyrowektorów, które nie są jednak wspierane przez biblioteki oraz są zupełnie nieintuicyjne w użyciu.

Zdecydowałem się na reprezentację w postaci modelu hiperboloidu Minkowskiego, ze względu na analogiczne transformacje do przestrzeni euklidesowej – zmieniają się tylko macierze przekształcenia oraz funkcje trygonometryczne na ich hiperboliczne odmiany.

3. Rzutowanie świata na ekran

Tutaj wybór był już prosty – ponowne rzutowanie na dysk Poincaré:



Następnie, używając odpowiedniej dla tego modelu metryki, wypuszczamy promienie ze środka układu współrzędnych I mierzymy odległość do punktów przecięcia ze ścianami. Odległości te są odwrotnie proporcjonalne do wysokości ściany rysowanej na ekranie w tej kolumnie – klasyczny raycaster. Dodatkowa transformacja została dodana żeby naprawić efekt “rybiego oka”.

4. Wybór technologii

Zdecydowałem się na język Rust oraz bibliotekę graficzną SDL2, ze względu na osobiste doświadczenie, szybkość, doskonały system typów, wsparcie społeczności oraz wsparcie platform na których chciałbym, żeby gra działała.

Wykonanie prototypu udowodniło, że wybór ten spełnia swoje założenia.

WIZJA SZCZEGÓŁOWA – KIERUNKI ROZWOJU, POTENCJALNE PROBLEMY

Rozgrywka:

- Dodanie jakiejś mechaniki, celu gry – na przykład walka z komputerowymi przeciwnikami
- Dodanie trybu multiplayer
- Rozwiązanie problemu AI przeciwnika na płaszczyźnie hiperbolicznej

Grafika:

- Teksturowanie ścian
- Różne wysokości ścian
- Oświetlenie – źródła? Mapy?
- Teksturowanie podłogi I sufitu
- Sprite'y przeciwników

Dodatkowo warte rozważenia:

- Zrealizowanie projektu jako modyfikacji do istniejącej gry, na przykład DOOM?

ANALIZA RYZYKA

Ryzyko nieukończenia projektu oceniam jako bardzo niewielkie. Główne problemy, które wymagały rozwiązań, rozwiązania te już mają. Zostaję więc w komfortowej sytuacji dodawania kolejnych funkcjonalności żeby uatrakcyjnić rozgrywkę i grafikę.

Pewnym ryzykiem jest to, że tworzenie świata będzie znacznie utrudnione przez brak odpowiednich narzędzi – konieczne może okazać się stworzenie edytora poziomów.

Innym niebezpieczeństwem jest również możliwość tego, że projekt nie będzie się dobrze skalował przy zwiększaniu rozmiaru świata – obliczenia matematyczne przy transformacjach świata wykonują się co klatkę I nie są proste.

Oczywistym ryzykiem jest też że efekt końcowy będzie po prostu nieciekawy lub niegrywalny – I tak moim zdaniem będzie to warte poświęconej pracy jako eksperymentu czy eksploracji.

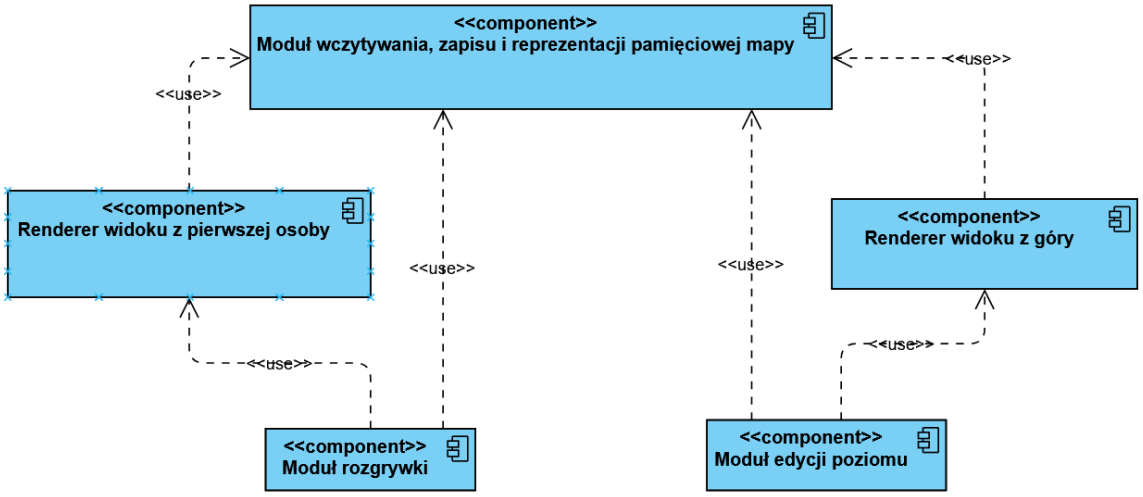
KONCEPCJA SYSTEMU

STOS TECHNOLOGICZNY

Nie powinien ulec większym zmianom względem prototypu:

Język	Rust
Biblioteka graficzna	SDL2
Kompilacja do przeglądarki	Emscripten
System kontroli wersji	Git

WYSOKOPOZIOMOWA LOGICZNA ARCHITEKTURA SYSTEMU



Podstawowym modułem będzie moduł wczytywania, zapisu I reprezentacji pamięciowej mapy. Jest to odrębna, zamknięta część systemu którą można wydzielić.

Z postaci mapy zdefiniowanej w wyżej wymienionym module korzystać będzie renderer, rzutujący tę reprezentację na ekran. Tutaj będzie odbywać się teksturowanie, renderowanie ewentualnych sprite'ów, aplikowanie oświetlenia.

Z obu powyższych będzie korzystać moduł rozgrywki. Będzie on przechowywał wewnętrzny stan gry, przyjmował polecenia od gracza, obsługiwał główną pętle gry w której będzie zajmował się logiką gry oraz wywoływaniem renderera.

Z racji braku dostępności narzędzi mogących służyć do łatwego manipulowania bądź wizualizacji obiektów w przestrzeni hiperbolicznej koniecznym może okazać się stworzenie również edytora poziomów. Architektura powinna być dość analogiczna, wymieniając jedynie renderer na taki wyświetlający rzut z góry, oraz mechanikę gry na mechanikę edycji poziomu.

Renderery mogą mieć wspólny interfejs, przyjmując jedynie aktualny stan świata I wyświetlając go.