Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

Grafika nieeuklidesowa na przestrzeni dwuwymiarowej

Maciej Hajduk
Nr indeksu: 236596

Praca inżynierska napisana
pod kierunkiem
Prof. dr hab. Jacka Cichonia

Politechnika Wrocławska

Wrocław 2019

Wydział Podstawowych Problemów Techniki

Maciej Hajduk Politechnika Wrocławska

Wrocław 2019

[section] [theorem] theorem]{Lemat}

Spis treści

Wstęp

Kontekst historyczny

Geometria jest nauką o mierze. Nazwa ta narzuca silne skojarzenia z nauką niemalże przyrodniczą. Nauczana we wszystkich szkołach od dwóch i pół tysiaca lat - wydawałoby się jest już czymś bardzo dobrze poznanym. Nowe teorie matematyczne doprowadziły jednak do podważenia tej pewności i powstania geometrii alternatywnych.

O życiu Euklidesa wiemy bardzo niewiele, a przecież to jemu zawdzięczamy nazwę *naszej* geometrii. Ani data urodzenia, ani pochodzenie nie są nam znane, a wszystkie informacje o nim czerpiemy z antycznych dzieł w których opisana jest matematyka. Około 300 roku przed naszą erą, Euklides - dyrektor Biblioteki Aleksandryjskiej, wydał swoje największe dzieło - *Elementy Geometrii*, na które składa się 13 ksiąg zawierających właściwie całą wiedzę matematyczną tamtych czasów. Początkowe definicje pierwszej księgi posiadają 5 stwierdzeń, które według Euklidesa są tak proste, że nie wymagają uzasadnienia. Euklides nazwał je aksjomatami:

1. Od dowolnego punktu do dowolnego innego można poprowadzić prostą.
2. Ograniczoną prostą można dowolnie przedłużyć.
3. Z dowolnego środka dowolnym promieniem można opisać okrąg.
4. Wszystkie kąty proste są równe (przystające).
5. Jeśli 2 proste na płaszczyźnie tworzą z trzecią kąty jednostronne wewnętrzne o sumie mniejszej od 2 kątów prostych, to proste te, po przedłużeniu, przetną się i to z tej właśnie strony. [1]

Piąty aksjomat mówi o tym, że z jednej strony przecinanej linii dwie proste będą się przybliżać. Zaczął on dość szybko wzbudzać podejrzenia. Jest znacznie bardziej skomplikowany od pozostałych, a już na pewno nie tak intuicyjny. Nawet Euklides unikał używania go w swoim dziele tak długo, jak to było możliwe i użył go dopiero w dowodzie własności 29.

Można śmiało powiedzieć, że piąty aksjomat w kolejnych wiekach spędzał uczonym sen z powiek. Przez kolejne 1500 lat matematycy próbowali udowodnić, że o wiele bardziej skomplikowany postulat musi wynikać z pozostałych czterech. Jednym z pierwszych zajmujących się tym problemem uczonych, był żyjący w V wieku naszej ery Proklos. Stwierdził on w swoim komentarzu do dzieł Euklidesa:

> Nie jest możliwe, aby uczony tej miary co Euklides godził się na obecność tak długiego postulatu w aksjomatyce – obecność postulatu wzięła się z pospiesznego kończenia przez niego Elementów, tak aby zdążyć przed nadejściem słusznie oczekiwanej rychłej śmierci; my zatem – czcząc jego pamięć – powinniśmy ten postulat usunąć lub co najmniej znacznie uprościć. [2]

Wyzwanie usunięcia piątego aksjomatu podjęło wielu matematyków w kolejnych wiekach. Prowadziło to do postania wielu nowych twierdzeń, które w istocie były piątemu

[1] Geometria euklidesowa. Encyklopedia PWN
[2] Najgłupiej postawiony problem matematyki. Marek Kordos - Delta, maj 2012

aksjomatowi równoważne. Prowadziło to do sprzeciwu innych uczonych. W szczególności Immanuel Kant w swoim dziele *Krytyka czystego rozumu* stwierdził, że intuicja geometryczna jest wrodzona, więc nie może istnieć wiele równoległych geometrii, a każdy kto chciałby zajmować się alternatywnymi geometriami nie nadaje się do nauki. Nie wszyscy zgodzili się z tym stwierdzeniem. Udano się do największego w tamtym czasie autorytetu - Carla Friedricha Gaussa, który jednak wycofał się, bojąc się - jak pisał - wrzasku Beotów. Do problemu należało się jednak odnieść. Odważyło się na to dwóch młodych ludzi, którzy uparli się nie tylko na uprawianie tej geometrii, ale wręcz głosili jej równoprawność. Rosjanin, Nikołaj Łobaczewski oraz Węgier - Janos Bolyai, niezależnie od siebie opublikowali prace w których - chociaż odmiennie - nowa geometria była konsekwentnie wyprowadzona. Obu odkrywców spotkała też za to kara, Łobaczewski został wręcz zmuszony do opuszczenia katedry.

Sprawę nowej geometrii (nazywanej już geometrią Bolyaia-Łobaczewskiego) przejął Felix Klein. Postawił on tezę, że jeżeli za pomocą geometrii euklidesowej jesteśmy w stanie przedstawić tę nieeuklidesową - i odwrotnie, to oba modele są sobie w istocie równoważne. Opublikował też w 1870 roku dzieło, w którym dowiódł równoprawności obu modeli.

Dosadnie do nowego modelu odniósł się fizyk - Hermann Helmholtz, publikując pracę, w której określił matematykę jako skrzynkę z narzędziami dla nauk przyrodniczych, czym odebrał jej walor nauki przyrodniczej jako takiej.

Wybrane zagadnienie

W niniejszej pracy zaimplementowany zostanie prosty silnik graficzny skupiający się na renderowaniu wizualizacji płaszczyzny dysku w modelu Poincarégo geometrii hiperbolicznej.

Praca swoim zakresem obejmie obsługę rysowania lini, okręgów, wielokątów na tejże płaszczyźnie oraz implementacje przykładowych programów obejmujących wizualizacje bardziej skomplikowanych struktur. Na tle innych implementacji, aplikacja wyróżnia się dostarczanymi możliwościami i realizacją problemu z pomocą matematycznego opisu pewnego modelu. Przykładowe demonstracje możliwości aplikacji są dostarczone razem z kodem źródłowym, jest to, poza możliwością narysowania dowolnego wielokąta, rysowaniem figur foremnych czy prostych animacji, także interakcja z urządzeniami peryferyjnymi i tesselacja przestrzeni hiperbolicznej. Niewątpliwą zaletą dostarczonej aplikacji jest prostota implementacji własnych rozwiązań, na co składa się silne typowanie języka Typescript wraz z dokładnymi interfejsami dla klas oraz funkcje dostarczone przez silnik, pozwalające na łatwe manipulowanie wyświetlającymi się obiektami, nie wymagające przy tym zrozumienia modelu.

Praca składa się z czterech rozdziałów:

Rozdział pierwszy: W rozdziale omówiono analizę wybranego problemu, przedstawiono motywacje podjęcia tego tematu oraz uzasadniono wybór modelu płaszczyzny Poincaré. Rozdział zawiera poza tym komentarz do różnych rodzajów geometrii nieeuklidesowych, oraz krótki opis i porównanie innych modeli geometrii hiperbolicznej.

Rozdział drugi: Rozdział zawiera szczegółową charakterystykę systemu wraz z opisem poszczególnych plików oraz przeznaczeniem klas i funkcji składających się na program. Opisane w nim zostały algorytmy przekształcające byty w geometrii Euklidesowe na odpowiadające im elementy geometrii hiperbolicznej, funkcje pomocnicze, reprezentacje punktów i linii w obu modelach.

Rozdział trzeci: W rozdziale wymieniono technologie użyte do implementacji projektu: wybrany język programowania, środowisko składające się na aplikację oraz biblioteki wykorzystane w programie.

Rozdział czwarty: Rozdział zawiera instrukcje instalacji i wdrożenia systemu w środowisku docelowym. Końcowy rozdział stanowi podsumowanie uzyskanych wyników i ewentualne możliwości rozwoju projektu.

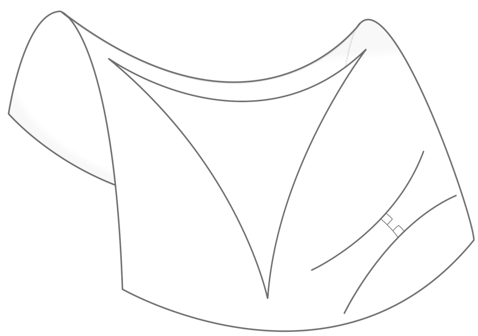
Analiza problemu

W niniejszym rozdziale przedstawiona będzie analiza problemu, opis matematyczny modelu płaszczyzny dysku Poincaré oraz przegląd kilku wybranych modeli geometrii nieeuklidesowej.

Odkrycie, że piątego aksjomatu nie można udowodnić na podstawie pozostałych czterech aksjomatów, było dla naukowców niespodzianką. Zrobiono to, demonstrując istnienie geometrii, w której pierwsze cztery aksjomaty utrzymywały się, ale piąty nie. Debata nad piątym postulatem Euklidesa stworzyła problem, jak alternatywna geometria powinna wyglądać. Umiano pokazać zaledwie poszczególne właściwości takich geometrii. Pierwszy model geometrii nieeuklidesowej został stworzony przez Kleina. W sprawę zaangażowało się wielu matematyków, w tym również Bernard Rieman. Stwierdził on, że można opisać nieskończenie wiele struktur matematycznych, które nie będą spełniały postulatów Euklidesa, będąc dalej geometriami.

Podstawowy podział

Geometria nieeuklidesowa to każda geometria, która nie spełnia przynajmniej jednego z postulatów Euklidesa. Geometrie nieeuklidesowe możemy podzielić na dwa rodzaje:



Rysunek 1: Trójkąt oraz dwie proste przedstawione na powierzchni o geometrii hiperbolicznej

Geometria Łobaczewskiego-Bólyaia (hiperboliczna)

Geometria hiperboliczna jest bliżej związana z geometrią euklidesową, niż się wydaje: jedyną różnicą aksjomatyczną jest postulat równoległy. Po usunięciu postulatu równoległego z geometrii euklidesowej geometria wynikowa jest geometrią absolutną. Wszystkie twierdzenia o geometrii absolutnej, w tym pierwsze 28 twierdzeń zaprezentowanych przez Euklidesa, obowiązują w geometrii i euklidesowej i hiperbolicznej.

W modelu hiperbolicznym, w płaszczyźnie dwuwymiarowej, dla dowolnej linii L i punktu X, który nie jest na L, istnieje nieskończenie wiele linii przechodzących przez X, które się nie przecinają L.

Geometria Riemanna (eliptyczna)

Geometria eliptyczna jest geometrią nieeuklidesową o dodatniej krzywiźnie, która zastępuje postulat równoległy stwierdzeniem “przez dowolny punkt na płaszczyźnie, nie ma linii równoległych do danej linii”. Geometria eliptyczna jest czasem nazywana również geometrią Riemannowską. Model można wizualizować jako powierzchnię kuli, na której linie przyjmowane są jako wielkie koła. W geometrii eliptycznej suma kątów trójkąta wynosi >180 stopni.

W modelu eliptycznym dla dowolnej linii L i punktu X, który nie jest na L, wszystkie linie przechodzące przez X przecinają się L.

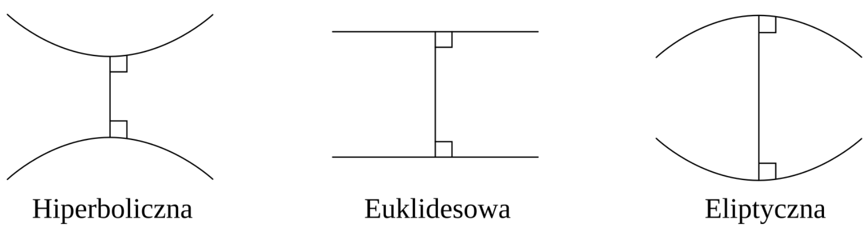
Różnice pomiędzy geometriami

Sposobem opisania różnic między tymi geometriami jest rozważenie dwóch linii prostych rozciągniętych w nieskończoność w płaszczyźnie dwuwymiarowej, które są prostopadłe do trzeciej linii:

- W geometrii euklidesowej linie pozostają w stałej odległości od siebie (co oznacza, że linia narysowana prostopadle do jednej linii w dowolnym punkcie przecina drugą linię, a długość odcinka linii łączącego punkty przecięcia pozostaje stała) i są znane jako równoległe.

- W geometrii hiperbolicznej linie *zakrzywiają się* od siebie, zwiększając odległość w miarę przesuwania się dalej od punktów przecięcia ze wspólną prostopadłą; linie te są często nazywane ultraparallelami.

- W geometrii eliptycznej linie *zakrzywiają się* do siebie i w końcu przecinają.



Rysunek 2: Zachowanie linii ze wspólną prostopadłą w każdym z trzech rodzajów geometrii

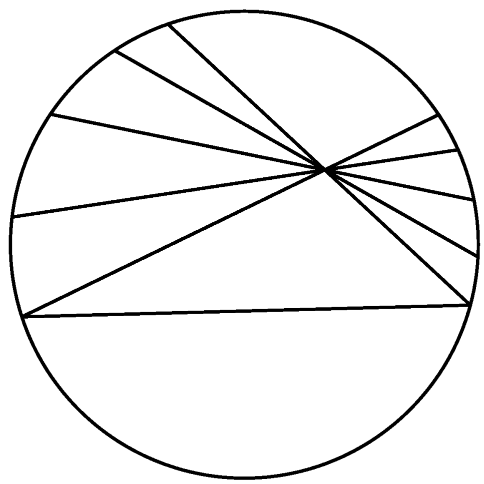
Ta praca skupia się na geometrii hiperbolicznej. Istnieje kilka możliwych sposobów wykorzystania części przestrzeni euklidesowej jako modelu płaszczyzny hiperbolicznej. Wszystkie te modele spełniają ten sam zestaw aksjomatów i wyrażają tę samą abstrakcyjną płaszczyznę hiperboliczną. Dlatego wybór modelu nie ma znaczenia dla twierdzeń czysto hiperbolicznych, jednak robi to różnicę podczas wizualizacji geometrii hiperbolicznej. Następne podrozdziały są poświęcone krótkiemu omówieniu najpopularniejszych z nich.

Popularne modele geometrii hiperbolicznej

Geometria hiperboliczna została opisana za pomocą wielu modeli. Najpopularniejsze przedstawiono poniżej.

Model Kleina

Model Kleina - a w zasadzie model dysku Beltrami–Kleina jest modelem geometrii hiperbolicznej, w którym punkty są reprezentowane przez punkty we wnętrzu dysku. Przyjmuje on następujące założenia:



Rysunek 3: Model Kleina

- **Płaszczyzną hiperboliczną** jest wnętrze koła bez krawędzi.
- **Prostymi hiperbolicznymi** są cięciwy tego koła (końce prostej).
- **Proste będą prostopadłe** wtedy, gdy przedłużenie jednej z nich przechodzi przez punkt przecięcia stycznych do obu linii.

Linie w modelu pozostają proste, a cały model można łatwo osadzić w ramach rzeczywistej geometrii rzutowej. Model ten nie jest jednak zgodny, co oznacza, że kąty są zniekształcone, a okręgi na płaszczyźnie hiperbolicznej na ogół nie są okrągłe w modelu.

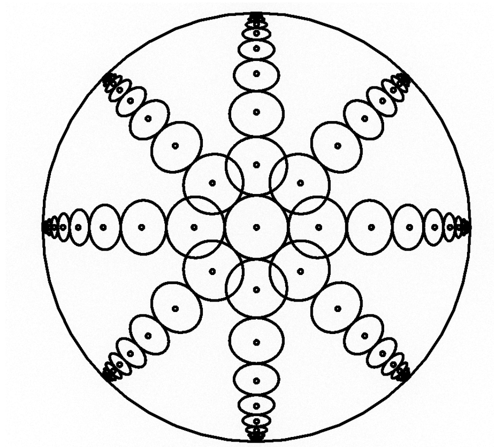
Model półpłaszczyzny Poincaré

Model półpłaszczyzny Poincaré to płaszczyzna:

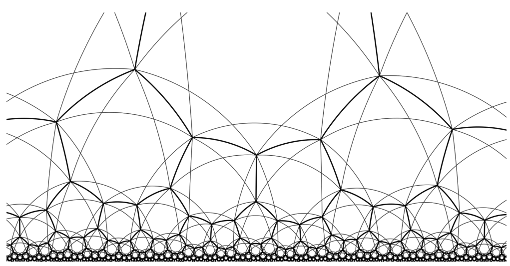
$$\{(x, y) \mid y > 0; x, y \in \mathbb{R}\}$$

Jest to model dwuwymiarowej geometrii hiperbolicznej.

Model nosi imię Henri Poincaré, ale został stworzony przez Eugenio Beltrami, który użył go wraz z modelem Kleina i modelem dysku Poincaré, aby pokazać, że geometria hiperboliczna jest równie spójna, jak spójna jest geometria euklidesowa. Ten model jest



Rysunek 4: Koła w modelu Kleina



Rysunek 5: Tesselacja w modelu półpłaszczyzny Poincaré

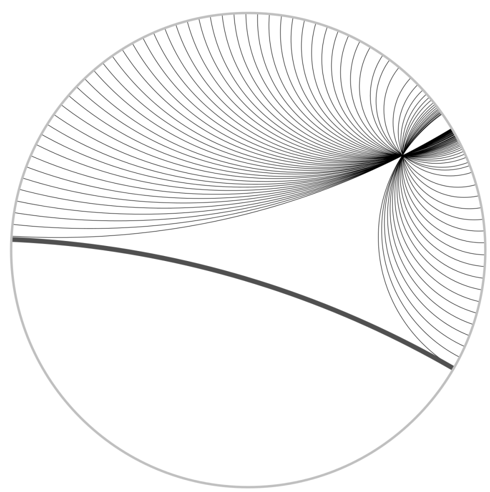
zgodny, co oznacza, że kąty zmierzone w punkcie modelu są równe kątom na płaszczyźnie hiperbolicznej.

Model dysku Poincaré

Model dysku Poincaré wykorzystuje wnętrze dysku jako model płaszczyzny hiperbolicznej. Najbardziej oczywistym wyborem dla dysku jest dysk jednostkowy, który będzie również przedmiotem dalszych rozważań.

- **Punkty hiperboliczne** to punkty wewnątrz dysku jednostkowego.
- **Linie hiperboliczne** to łuki koła prostopadłe do dysku. Linie hiperboliczne przechodzące przez początek degenerują się do średnic, o których można pomyśleć jako łuki kół o nieskończonym promieniu.
- **Kąty** są mierzone jako kąt euklidesowy między stycznymi w punkcie przecięcia.
- **Odległości** między punktami hiperbolicznymi można mierzyć w oparciu o normę euklidesową:

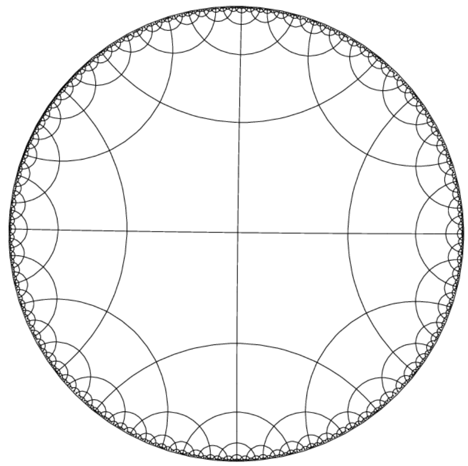
$$\delta(u, v) = 2\frac{\|u - v\|^2}{(1 - \|u\|^2)(1 - \|v\|^2)}$$



Rysunek 6: Wszystkie powyższe linie w dysku Poincaré są równoległe do siebie

Ponieważ rozpatrywany jest dysk jednostkowy, formuła nie zawiera w zmiennej dla promienia.

Model jest zgodny, to znaczy, że zachowuje kąty. Oznacza to, że kąty hiperboliczne między krzywymi są równe kątom euklidesowym w punkcie przecięcia. Wadą jest fakt, że ponieważ linia hiperboliczna jest modelowana przez łuk koła euklidesowego, linie proste wydają się zakrzywione.

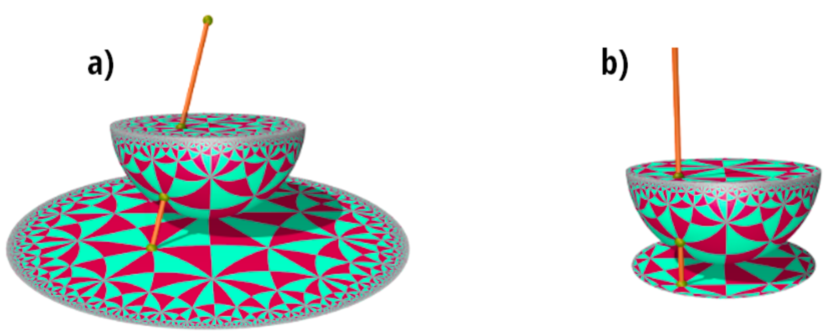


Rysunek 7: Tesselacja w modelu dysku Poincaré

Model Hemisfery

Hemisfera nie jest często używana jako model płaszczyzny hiperbolicznej jako taka. Jest to jednak bardzo przydatna w łączeniu różnych innych modeli za pomocą różnych rzutów, jak pokazano na poniższym rysunki.

- **Punkty hiperboliczne** to punkty na półkuli południowej.
- **Linie hiperboliczne** to półkola powstałe z przecięcia półkuli południowej z płaszczyznami prostopadłymi do równika.

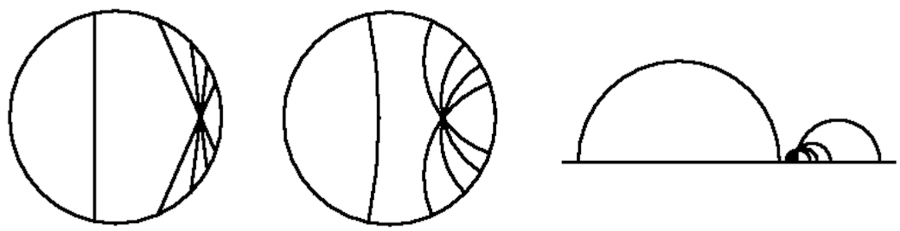


Rysunek 8: Rzut na dysk Poincarégo (a) i projekcja do modelu Klein-Beltrami (b)

Wadą tego rozwiązania, jest dodatkowy wymiar, jaki należy rozpatrywać przy pracy z tym modelem.

Uzasadnienie wyboru modelu dysku Poincaré

Jak stwierdzono na początku tego rozdziału, kolejne rozdziały, a także opisane implementacje będą prawie wyłącznie korzystać z modelu dysku Poincaré. Podczas renderowania geometrii hiperbolicznej wydaje się to być właściwym wyborem, z uwagi na wartości estetyczne i zgodność modelu.



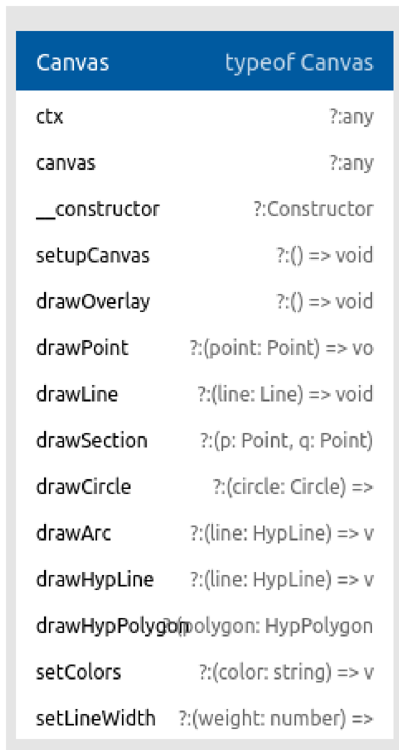
Rysunek 9: Porównanie modeli Kleina, dysku Poincaré i półpłaszczyzny Poincaré

Projekt systemu

W niniejszym rozdziale przedstawiony zostanie szczegółowy projekt systemu, jego matematyczną interpretacje, zależności pomiędzy klasami oraz podstawowe algorytmy składające się na logikę funkcjonowania silnika.

Cykl pracy silnika

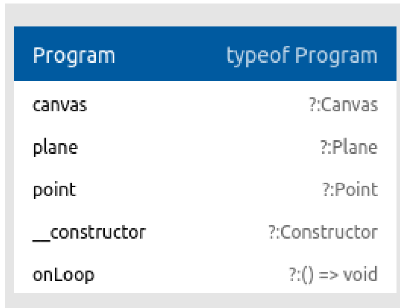
Głównym plikiem silnika jest `main.ts` znajdujący się w katalogu `/src`. Po załadowaniu programu, tworzy on instancje klasy `Canvas` odpowiedzialnej za rysowanie elementów na ekranie, ładuje konfiguracje wyświetlanego programu i tworzy pętlę silnika poprzez wywołanie metody `createLoop()` klasy `Engine`.



Rysunek 10: Diagram klasy Canvas

Moduł odpowiedzialny za renderowanie obrazu znajduje się w pliku `canvas.ts`. Konstruktor klasy `Canvas` przyjmuje element `canvas` ze strony oraz jego kontekst, oraz inicjuje się poprzez wywołanie funkcji `setupCanvas()`, która ustala szerokość i wysokość elementu. W każdym cyklu silnika, wywoływana jest funkcja `drawOverlay()`, która resetuje element do podstawowego widoku. Kolejne funkcje klasy odpowiadają za rysowanie punktów, liń, łuków i wielokątów. Poza tym klasa udostępnia też funcje zmiany koloru rysowanych elementów i grubości linii.

Klasa `Engine` przyjmuje konfigurację z pliku `/assets/config.json`, która ustala ilość FPS, wywołuje następnie metodę `drawOverlay()` klasy `Canvas` i odpala funkcję `onLoop()` z programu, konfigurację którego dostaje za pomocą *dependency injection* w parametrach konstruktora.

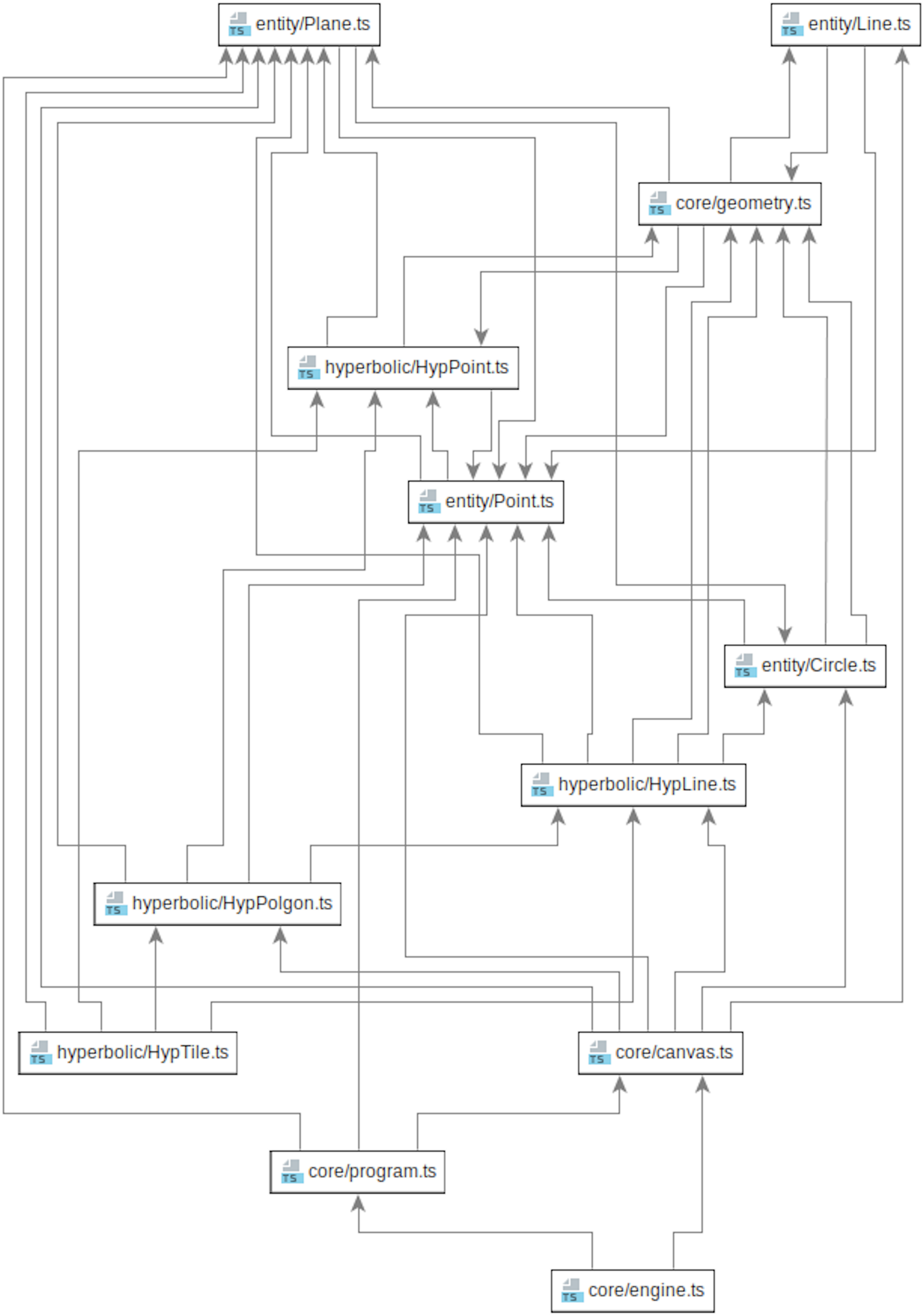


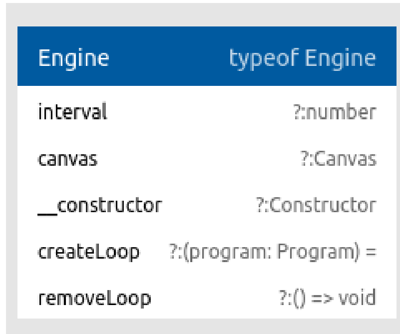
Rysunek 12: Diagram klasy Program

Odtwarzany program tworzony jest poprzez wywołanie instancji klasy programu, dziedziczącej po abstrakcyjnej klasie `Program`, udostępniającej metody takie jak `onLoop()`.

Typy obiektów renderowanych przez silnik

Każdy możliwy do narysowania obiekt jest instancją jednej z klas. W kodzie silnika istnieje wyraźny podział na klasy udostępniające obiekty rysowane w przestrzeni euklidesowej i hiperbolicznej. Wszystkie byty znajdują się w katalogu `/src/core/entity`. Kolejne rozdziały są poświęcone opisie i interpretacji poszczególnych klas.


entity/Plane.ts
entity/Line.ts
core/geometry.ts
hyperbolic/HypPoint.ts
entity/Point.ts
entity/Circle.ts
hyperbolic/HypLine.ts
hyperbolic/HypPolgon.ts
hyperbolic/HypTile.ts
core/canvas.ts
core/program.ts
core/engine.ts



Rysunek 11: Diagram klasy Engine

Obiekty geometrii Euklidesowej

Instancje klas opisanych poniżej są obiektami rysowanymi finalnie przez silnik, na płaskim ekranie całość sprowadza się do linii, łuków, kół i punktow w przestrzeni Euklidesowej.

Klasa Point

Konstruktor klasy `Point` przyjmuje dwie zmienne typu `number`, które są reprezentacją bezwzględnych koordynatów puntu na płótnie. Programista może skorzystać z metody `toHypPoint(plane: Plane): HypPoint`, która przyjmuje instancję klasy `Plane` i zwraca dla niej koordynaty punktu w interfejsie klasy `HypPoint`, oraz z metody `inversion(plane: Plane)`, zwracającej punkt odbity względem centralnego punktu obiektu klasy `Plane` (centrum sfery hiperbolicznej).

Klasa Line

Konstruktor klasy `Line` przyjmuje dwie zmienne typu `number`. Programista może skorzystać z metody `at(x: number): number`, która zwraca wartość w punkcie `x` oraz `intersectPoint(line: Line): Point`, która zwraca punkt przecięcia tejże linii z inną linią.

Alternatywnymi sposobami na stworzenie instancji klasy `Line` jest skorzystanie ze statycznych metody `fromPoints(p: Point, q: Point)`, która tworzy linię z dwóch punktów lub `fromPointSlope(p: Point, q: number)`, która do stworzenia linii potrzebuje podania punktu i kąta wyrażonego w radianach.

Klasa Circle

Konstruktor klasy `Circle` przyjmuje punkt centralny będący instancją klasy `Point` i średnicę typu `number`, oraz udostępnia metodę `intersectPoints(circle: Circle): [Point, Point]`, przyjmującą drugi okrąg i zwracającą parę punktów, w których przecinają się oba obiekty. Funkcja `fromPoints(p: Point, q: Point, r: Point)` umożliwia alternatywny sposób stworzenia okręgu z trzech obiektów klasy `Point`.

Klasa Plane

Najważniejszym z pośród omawianych dotychczas bytów jest instancja klasy `Plane`, będąca singletonem i punktem odniesienia do wszystkich obiektów dla geometrii hiperbolicznej.

Klasa `Plane` dziedziczy po klasie `Circle`, podobnie jak ona posiada centrum i średnicę, liczone automatycznie na podstawie szerokości i wysokości ekranu przy pobraniu instancji klasy.

Obiekty geometrii hiperbolicznej

Kod źródłowy klas opisanych poniżej znajduje się w oddzielnym katalogu silnika: `/src/core/entity/hyperbolic`. Każdy z tych obiektów opisuje byt geometrii hiperbolicznej, rysowany następnie przez silnik w formie prostych linii, czy łuków.

Klasa HypLine

Klasa `HypLine` jest pierwszą z pośród klas obiektów hiperbolicznych. Konstruktor klasy przyjmuje, podobnie jak klasa Line, dwa punkty oraz dodatkowo instancję klasy `Plane`.

Pierwszym krokiem konstruktora jest wywołanie metody `calculateArc(p: Point, q: Point, plane: Plane): Circle`, która z pomocą algorytmu opisanego poniżej, zwraca instancję klasy `Circle`, będącą okręgiem, na obwodzie którego leży dana prosta hiperboliczna. Ustala jednocześnie punkty `p` i `q` wyznaczające końce odcinka, posługując się przy tym metodą `cutIfSticksOut(point: Point, circle: Circle, plane: Plane): Point`, sprawdzającą, czy punkt nie leży poza granicą koła wyznaczonego przez obiekt klasy `Plane` i ewentualnie przesuwającą go na punkt przecięcia.

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end

Algorithm 1: Algorytm wyznaczania okręgu na podstawie dwóch punktów i płaszczyny

Ostatnią nieomówioną funcją jest `countAngle(circle: Circle)`, określającą na podstawie wcześniej obliczonych punktów, początkowy i końcowy kąt łuku oraz kierunek, w jakim rysowany będzie ten łuk.

Klasa HypPoint

Klasa HypPoint to w rzeczywistości reprezentacja punktu wględem płaszczyzny hiperbolicznej w dziedzinie $(-1, 1) \times (-1, 1) \in \mathbb{R} \times \mathbb{R}$.

Klasa udostępnia metodę `toCanvasCoords(): Point`, zwracającą instancję tego samego punktu, zdolną do wyświetlenia przez aplikację, funcję `reflect(point: HypPoint): HypPoint` - zwracającą odbicie tegoż punktu względem innego i dwie prywatne, pomocnicze funkcje `times(point: HypPoint | number): HypPoint` oraz `over(point: HypPoint | number): HypPoint` służące kolejno do mnożenia lub dzielenia danego punktu przez stałą lub inny punkt.

Najważniejszą metodą tej klasy jest `moebius(point: HypPoint, t: number): HypPoint`. Aby zrozumieć jej działanie potrzebne będzie zdefiniowanie *Transformacji Möbiusa* i jej udziału w obliczaniu punktu na przestrzeni dysku Poincaré. Zdefiniowana jest ona na końcu tego rozdziału.

Klasa HypPolygon

Konstruktor klasy `HypPolygon` przyjjmuje dwie zmienne typu `Point` oraz instację klasy `Plane` i tworzy z nich wielokąt na przestrzeni hiperbolicznej.

Wielokąt może zostać rozszerzony o kolejne punkty z pomocą metody `addVerticle(point: Point)`. Funkcja `getCompletePolygonLines(): HypLine[]` zwraca wszystkie odcinki wchodzące w skład wielokąta, wraz z jednym dodatkowym odcinkiem, łączącym pierwszy i ostatni wierzchołek. Funkcje `moebius(point: HypPoint, t: number): HypPolygon` oraz `reflect(point: HypPoint): HypPolygon` wykonują kolejno transformację Möbiusa oraz odbicie względem punktu na wszystkich wierzchołkach wielokąta.

Programista może skorzystać ze statycznej metody `fromVerticles(verts: Point[], plane: Plane): HypPolygon`, która przyjjmuje tablicę punków oraz instację klasy `Plane` i zwraca gotowy wielokąt.

Klasa HypTile

Klasa `HypTile` jest nietypowa na tle swoich poprzedniczek. Konstruktor tej klasy jest prywatny, a stworzenie jej instancji odbywa się za pomocą jednej z trzech metod statycznych:

- `fromPolygon(polygon: HypPolygon, center: HypPoint, plane: Plane): HypTile` - funkcja tworzy obiekt klasy `HypTile` wykorzystując do tego instancję obiekty klasy `HypPolygon`

- `createNKPolygon(n: number, k: number, center: HypPoint, plane: Plane, quasiregular = false): HypTile` - Niech ABC będzie trójkątem w regularnym (n, k - kafelkowy), gdzie:
  - A jest środkiem n-gona (również środkiem dysku),
  - B jest wierzchołkiem n-gonu,
  - C jest punktem środkowym boku n-gonu sąsiadującego z B.

- `createRegularPolygon(numOfVerts: number, distance: number, center: HypPoint, plane: Plane, startAngle = 0): HypTile` - funkcja tworzy wielokąt foremnty o podanych parametrach.

Funkcje dodatkowe

Plik `geometry.ts` zawiera zestaw funkcji wspólnych dla wielu obiektów, lub nie powiązanych bezpośrednio z żadnym z nich. Są to głównie funkcje czysto matematyczno - geometryczne, takie jak odległość Euklidesowa lub szukanie dwusiecznej dwóch punktów.

Transformacja Möbiusa

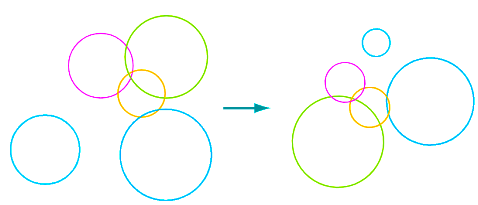
Twierdzenie 1 *Transformacja Möbiusa jest funkcją na rozszerzonej płaszczyźnie zespolonej określoną równaniem*

$$f(z) = \frac{az+b}{cz+d},\ gdzie\ ad - bc \neq 0$$

$$transformacja\ M\ddot{o}biusa = złożenie\ inwersji = izometrie\ hiperboliczne$$

Hiperboliczne symetrie są modelowane jako przekształcenia Möbiusa: [3]

Transformacje Möbiusa (zwane również homografiami) tworzą grupę geometryczną. Odwrócenie przestrzeni przez sferę ze środkiem w punkcie O i promieniu r, odwzorowuje na siebie wszystkie promienie pochodzące z tego, że iloczyn punktu na tym promieniu wraz z jego obrazem jest równy r^2. Transformacje Möbiusa zachowują również kąty. Izometria geometrii hiperbolicznych to własnie transformacje Möbiusa. W ten sposób, z ich pomocą możemy nawigować po przestrzeni hiperbolicznej, płynnie przesuwając punkt widzenia modelu dysku Poincaré.



Rysunek 13: Transformacja Möbiusa

[3]HyperbolicTransformations, Chapter 17

Implementacja systemu

W niniejszym rozdziale omówiona zostanie technologia, konfiguracja oraz wdrożenie systemu wraz z krótkim opisem poszczególnych części systemu i kodu źródłowego.

Opis technologii

Do implementacji systemu użyto języka `TypeScript` w wersji `3.6.3`, bundlera (transpilatora nowoczesnych wersji języka `JavaScript` do wersji zrozumiałych dla przeglądarek) `webpack` w wersji `2.3.3` oraz `SCSS` i `HTML5` wraz z elementem `<canvas>` odpowiedzialnym za rysowanie grafiki na ekranie. Użyta została również funkcyjna biblioteka `ramda` w formie pomocniczej biblioteki *utilsowej*. Pełna lista wszystkich bibliotek wraz z ich wersjcami znajduje się w pliku `package.json`, w katalogu głównym projektu.

Poszczególne składowe systemu

Kolejne paragrafy zawierają opisy i przeznaczenie poszczególnych plików.

Instalacja i wdrożenie

Rozdział ten zawiera informacje o sposobie zbudowania aplikacji w celu jej uruchomienia i opcjonalnie - wdrożenia na serwerze WWW.

Do zbudowania aplikacji konieczny będzie menager pakietów `npm` w wersji przynajmniej `6.5.0` oraz środowisko uruchomieniowe języka `JavaScript` - `node.js` w wersji `10.6.0` lub nowszej. Instalacja wymaganych pakietów odbywa się poprzez wpisanie w konsoli polecenia

```
npm install
```

w katalogu głównym projektu. Następnie należy zbudować aplikację poleceniem

```
npm run build
```

Po zbudowaniu aplikacji, w katalogu głównym pojawi się folder `dist` z plikami, które wraz z plikem `index.html` składają się na gotowy program możliwy do uruchomiania w przeglądarce.

Serwer deweloperski

Aplikacja wspiera tryb deweloperski, w którym bieżące zmiany w kodzie automatycznie są budowane do plików wynikowych. Do uruchomienia trybu deweloperskiego potrzebne są te same pakiety instalowane poleceniem:

```
npm install
```

Wywołanie trybu odbywa się komendą:

```
npm run build-watch
```

Wdrożenie na serwerze WWW

Podsumowanie

Bibliografia

- Joan Gómez, Tam, gdzie proste są krzywe, Geometrnie enieuklidesowe, RBA, 2010
- Martin Freiherr von Gagern, Creation of Hyperbolic Ornaments Algorithmic and Interactive Methods, Technischen Universitat Munchen
- Mateusz Kłeczek, Geometria hiperboliczna, Chrzanów 2016
- Bjørn Jahren, An introduction to hyperbolic geometry, MAT4510/3510
- Izabela Przezdzink, Geometria Poincaré i Kleina. Skrypt do zajęć: Podstawy geometrii i elementy geometrii nieeuklidesowej, Wrocław 2010, Uniwersytet Wrocławski Wydział Matematyki i Informatyki Instytut Matematyczny
- Stefan Kulczycki Biblioteka Problemów Geometria NieeuklidesowaWarszawa 1960, Państwowe Wydawnictwo Naukowe
- Marek Kordos, O różnych geometriach, Warszawa 1987, Wydawnictwa Alfa
- Caroline Series With assistance from Sara Maloni, Hyperbolic geometry MA448
- Marshall Bern, Optimal Möbius Transformationfor Information Visualization and Meshing
- Steve Szydlik, Hyperbolic Constructions in Geometer's Sketchpad, December 21, 2001
- Douglas N. Arnold and Jonathan Rogness, Möbius Transformations Revealed
- Frank Nielsen1and, Richard Nock, Hyperbolic Voronoi diagrams made easy
- Marek Kordos, Geometria Bolyaia–Łobaczewskiego, http://www.deltami.edu.pl, Sierpień 2018

Zawartość płyty CD

Płyta CD zawiera cały kod źródłowy programu, zbudowany w katalogu `/dist` projekt oraz katalog `/docs` zawierający źródła tej pracy oraz jej końcową wersję w postaci pliku `pdf`.