

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

## Spis treści

|  |           |
|--|-----------|
| <b>Wstęp</b>   | <b>4</b>  |
| Kontekst historyczny . . . . .                             | 4         |
| Wybrane zagadnienie . . . . .                              | 5         |
| <b>Analiza problemu</b>                                    | <b>8</b>  |
| Podstawowy podział . . . . .                               | 8         |
| Geometria Łobaczewskiego-Bólyaia (hiperboliczna) . . . . . | 8         |
| Geometria Riemanna (eliptyczna) . . . . .                  | 9         |
| Różnice pomiędzy geometriami . . . . .                     | 9         |
| Popularne modele geometrii hiperbolicznej . . . . .        | 10        |
| Model Kleina . . . . .                                     | 10        |
| Model półpłaszczyzny Poincaré . . . . .                    | 10        |
| Model dysku Poincaré . . . . .                             | 11        |
| Model Hemisfery . . . . .                                  | 12        |
| Uzasadnienie wyboru modelu dysku Poincaré . . . . .        | 14        |
| <b>Projekt systemu</b>                                     | <b>16</b> |
| Cykl pracy silnika . . . . .                               | 16        |
| Klasy obiektów . . . . .                                   | 17        |
| Klasa Line . . . . .                                       | 19        |
| Klasa Point . . . . .                                      | 19        |
| Klasa Circle . . . . .                                     | 20        |
| Klasa Plane . . . . .                                      | 20        |
| Klasa HypLine . . . . .                                    | 20        |
| <b>Implementacja systemu</b>                               | <b>23</b> |
| Opis technologii . . . . .                                 | 23        |
| <b>Instalacja i wdrożenie</b>                              | <b>23</b> |
| Serwer deweloperski . . . . .                              | 23        |
| <b>Podsumowanie</b>  | <b>25</b> |

|                           |           |
|---------------------------|-----------|
| <b>Bibliografia</b>       | <b>27</b> |
| <b>Zawartość płyty CD</b> | <b>29</b> |



# 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 tysiąca 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. Dowolne dwa punkty można połączyć odcinkiem.
2. Dowolny odcinek można przedłużyć nieograniczenie (uzyskując prostą).
3. Dla danego odcinka można zaznaczyć okrąg o środku w jednym z jego końcowych punktów i promieniu równym jego długości.
4. Wszystkie kąty proste są przystające.
5. Dwie proste, które przecinają trzecią w taki sposób, że suma kątów wewnętrznych po jednej stronie jest mniejsza od dwóch kątów prostych, przetną się z tej właśnie strony.

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 pośpiesznego 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.

Wyzwanie usunięcia piątego aksjomatu podjęło wielu matematyków w kolejnych wiekach. Prowadziło to do postania wielu nowych twierdzeń, które w istocie były piątemu 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 Boetów. 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 nauk 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 objemnie obsługuje rysowania linii, okręgów, wielokątów na tejże płaszczyźnie oraz implementacje przykładowych programów obejmujących wizualizację bardziej skomplikowanych struktur. Na tle innych implementacji, aplikacja wyróżnia się dostarczaniem 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 motywację 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 Euklidesowej 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.





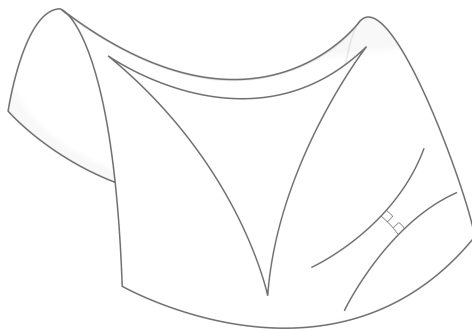
## Analiza problemu

W niniejszym rozdziale przedstawiona będzie analiza problemu, opis matematyczny modelu płaszczyzny dysku Poincarégo 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 postulat 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)

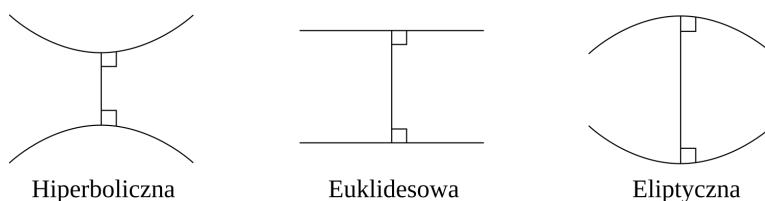
Geometria eliptyczna jest geometrią nieeuklidesową o dodatniej krzywiznie, która zastępuje postulat równoległy stwierdzeniem “przez dowolny punkt na płaszczyźnie, nie ma linii równoległych do danej linii”. Geometria eliptyczna jest czasem nazywana również geometrią Riemannowską. Model można zwizualizować 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 ultrarównoległymi.
- 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

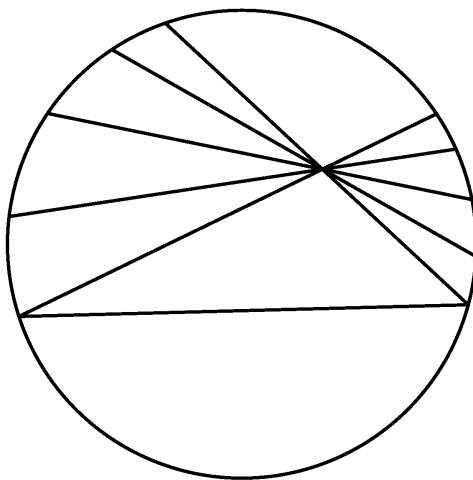
różnicę podczas wizualizacji geometrii hiperbolicznej. Następne podrozdziały są poświęcone krótkiemu omówieniu najpopularniejszych z nich.

## 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

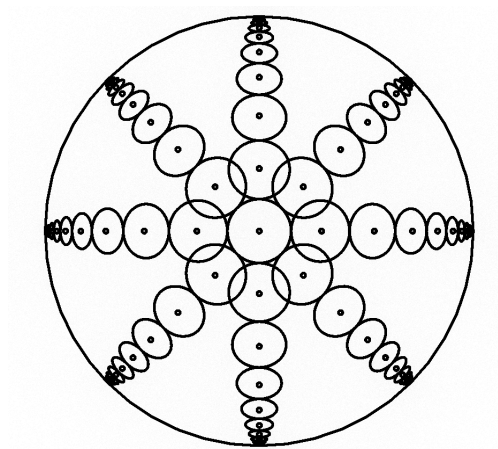
- **Płaszczyzną hiperboliczną** jest wnętrze koła bez krawędzi.
- **Prostymi hiperbolicznymi** są cięciwy tego koła (końce prostej).
- **Proste będą prostopadłe** wtedy, gdy przedłużenie jednej z nich przechodzi przez punkt przecięcia stycznych do obu linii.

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

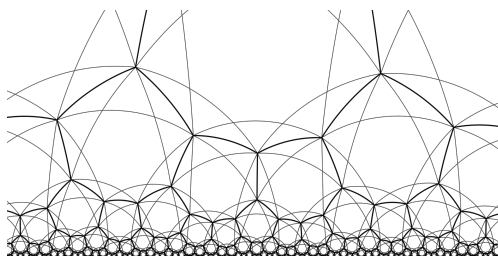
Model półpłaszczyzny Poincaré to płaszczyzna:

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



Rysunek 4: Koła w modelu Kleina

Jest to model dwuwymiarowej geometrii hiperbolicznej.



Rysunek 5: Tesselacja w modelu półpłaszczyzny Poincaré

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

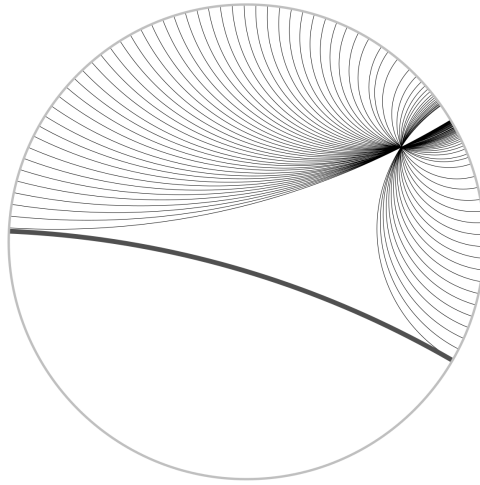
### Model dysku Poincaré

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.

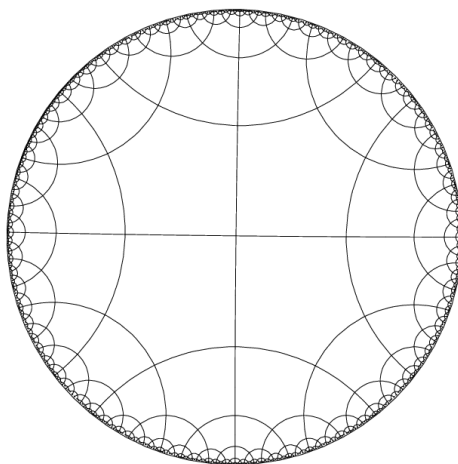
Model jest zgodny, to znaczy, że zachowuje kąty. Oznacza to, że kąty hiperboliczne między krzywymi są równe kątom euklidesowym w punkcie przecięcia. Wadą jest fakt, że ponieważ linia hiperboliczna jest modelowana przez łuk koła euklidesowego, linie proste wydają się zakrzywione.

### Model Hemisfery

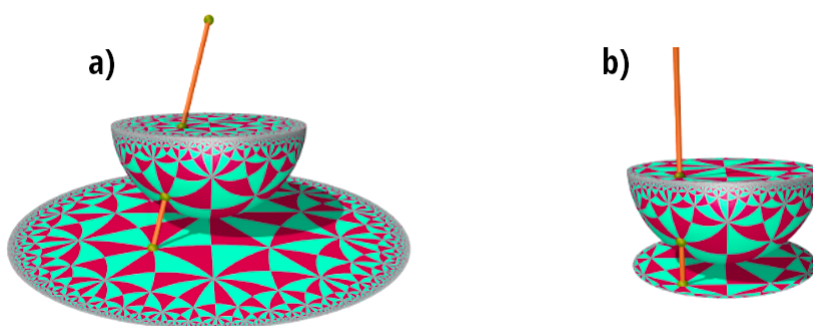
Hemisfera nie jest często używana jako model płaszczyzny hiperbolicznej jako taka. Jest to jednak bardzo przydatna w łączeniu różnych innych modeli za pomocą różnych rzutów, jak pokazano na poniższym rysunku.

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

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



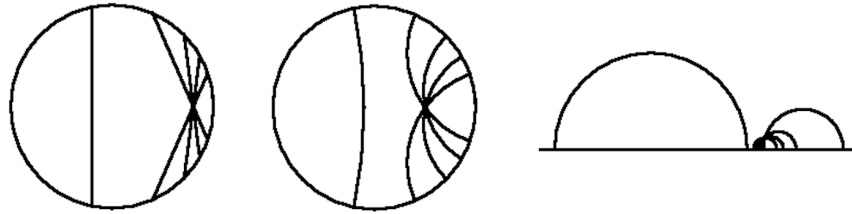
Rysunek 7: Tesselacja w modelu dysku Poincaré



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

## 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



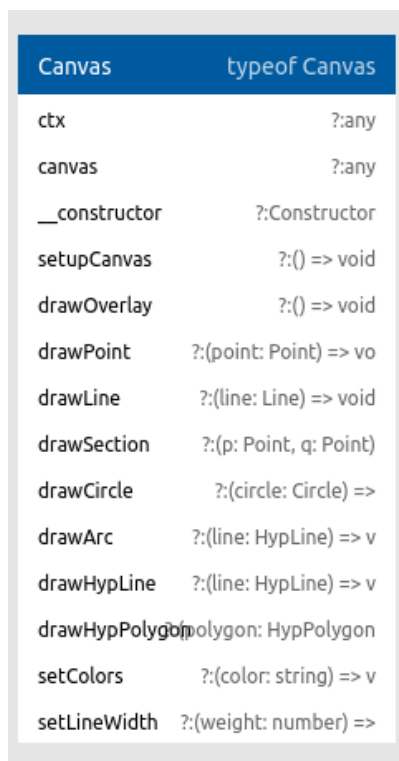


## Projekt systemu

W niniejszym rozdziale przedstawiony zostanie szczegółowy projekt systemu, jego matematyczną interpretację, 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 konfigurację wyświetlanego programu i tworzy pętlę silnika poprzez wywołanie metody `createLoop()` klasy `Engine`.



| Canvas         | typeof Canvas          |
|----------------|------------------------|
| ctx            | ?:any                  |
| canvas         | ?:any                  |
| __constructor  | ?:Constructor          |
| setupCanvas    | ?:() => void           |
| drawOverlay    | ?:() => void           |
| drawPoint      | ?:(point: Point) => vo |
| drawLine       | ?:(line: Line) => void |
| drawSection    | ?:(p: Point, q: Point) |
| drawCircle     | ?:(circle: Circle) =>  |
| drawArc        | ?:(line: HypLine) => v |
| drawHypLine    | ?:(line: HypLine) => v |
| drawHypPolygon | polygon: HypPolygon    |
| setColors      | ?:(color: string) => v |
| setLineWidth   | ?:(weight: number) =>  |

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, linii, łuków i wielokątów. Poza tym klasa udostępnia też funkcje zmiany koloru rysowanych elementów i grubości linii.

| Engine        | typeof Engine          |
|---------------|------------------------|
| interval      | ?:number               |
| canvas        | ?:Canvas               |
| __constructor | ?:Constructor          |
| createLoop    | ?:(program: Program) = |
| removeLoop    | ?:() => void           |

Rysunek 11: Diagram klasy Engine

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.

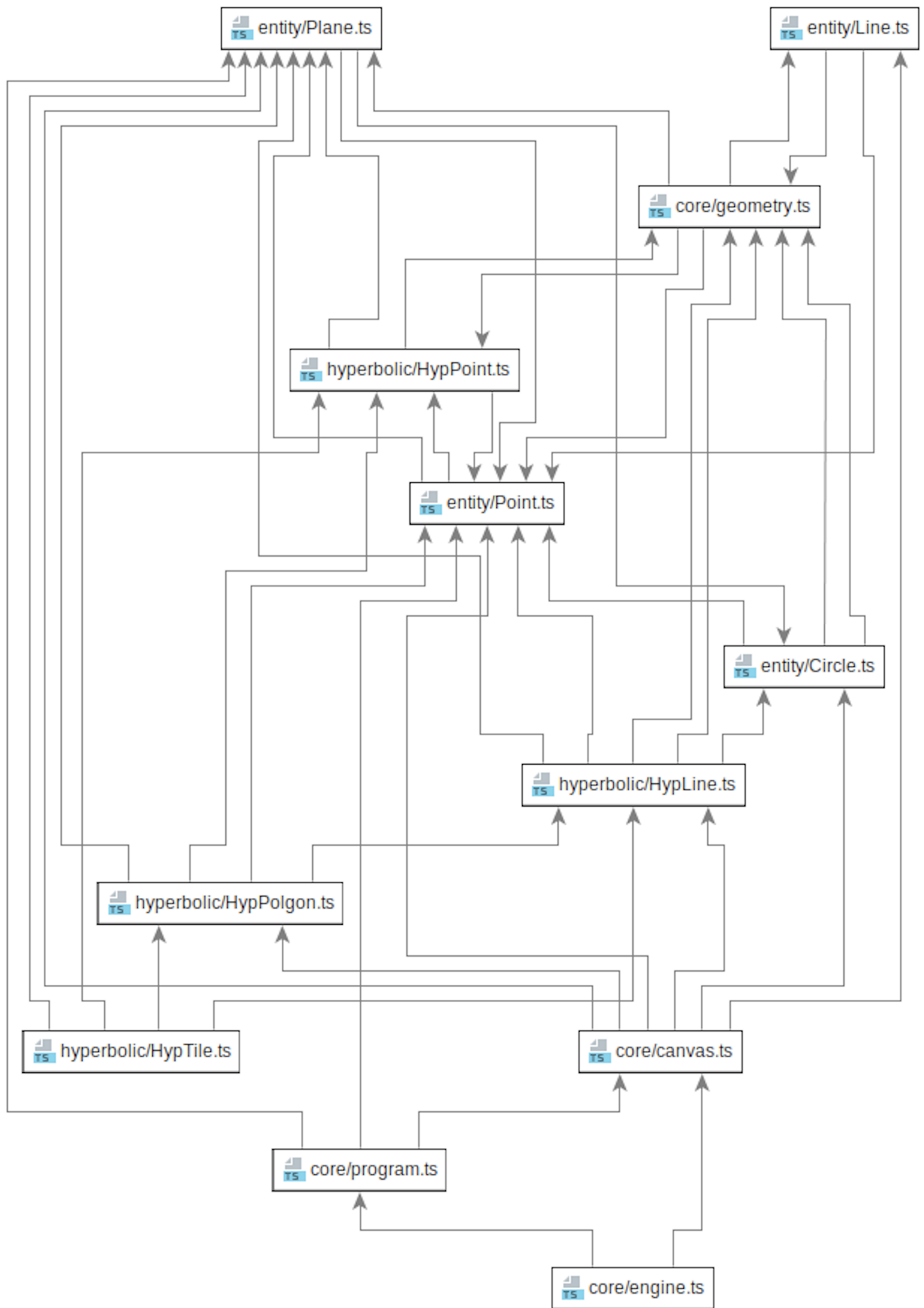
| Program       | typeof Program |
|---------------|----------------|
| canvas        | ?:Canvas       |
| plane         | ?:Plane        |
| point         | ?:Point        |
| __constructor | ?:Constructor  |
| onLoop        | ?:() => void   |

Rysunek 12: Diagram klasy Program

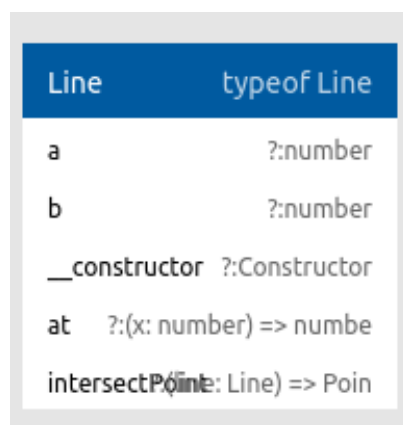
Odtwarzany program tworzony jest poprzez wywołanie instancji klasy programu, dziedziczącej po abstrakcyjnej klasie **Program**, udostępniającej metody takie jak `onLoop()`.

## Klasy obiektów

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 opisowi i interpretacji poszczególnych klas.



## Klasa Line

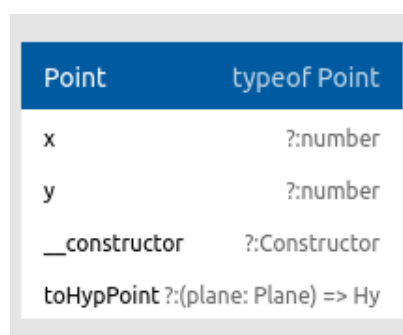


| Line           | typeof Line               |
|----------------|---------------------------|
| a              | ? :number                 |
| b              | ? :number                 |
| __constructor  | ? :Constructor            |
| at             | ? : (x: number) => number |
| intersectPoint | ? : (line: Line) => Point |

Rysunek 13: Diagram klasy 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 Point



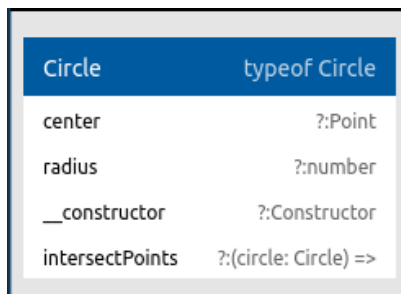
| Point         | typeof Point             |
|---------------|--------------------------|
| x             | ? :number                |
| y             | ? :number                |
| __constructor | ? :Constructor           |
| toHypPoint    | ? : (plane: Plane) => Hy |

Rysunek 14: Diagram klasy Point

Konstruktor klasy **Point** przyjmuje dwie zmienne typu **number**, które są reprezentacją bezwzględnych współrzędnych punktu na płótnie. Programista może

skorzystać z metody `toHypPoint(plane: Plane): HypPoint`, która przyjmuje instancję klasy `Plane` i zwraca dla niej koordynaty punktu w interfejsie `HypPoint`, oraz z metody `inversion(plane: Plane)`, zwracającej punkt odbity względem centralnego punktu obiektu klasy `Plane`.

## Klasa Circle



Rysunek 15: Diagram klasy 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 dotycznas 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.

## 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, jednocześnie ustalając punkty `p` i `q` wyznaczające końce odcinka, posługując się przy tym metodą `cutIfSticksOut(point: Point, circle: Circle, plane: Plane)`:

**Point**, sprawdzając, czy punkt nie leży poza granicą koła wyznaczonego przez obiekt klasy **Plane** i ewentualnie przesuwając go na punkt przecięcia.

**Data:** this text

**Result:** how to write algorithm with L<sup>A</sup>T<sub>E</sub>X2e 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:** How to write algorithms



## 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 **CSS3** i **HTML5** wraz z elementem `<canvas>` odpowiedzialnym za rysowanie grafiki na ekranie. Pełna lista wszystkich bibliotek wraz z ich wersjami znajduje się w pliku `package.json`, w katalogu głównym projektu.

## 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 plikiem `index.html` składają się na gotowy program możliwy do uruchomienia 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
```





## Podsumowanie



## Bibliografia

- Martin Freiherr von Gagern, Creation of Hyperbolic Ornaments Algorithmic and Interactive Methods, Technischen Universitat Munchen
- Tam, gdzie proste są krzywe, Geometrie nieuklidesowe, Joan Gómez, RBA, 2010
- Bjørn Jahren, An introduction to hyperbolic geometry, MAT4510/3510
- Izabela Przedzink, Geometria Poincarégo i Kleina. Skrypt do zajęć: Podstawy geometrii i elementy geometrii nieuklidesowej, Wrocław 2010, Uniwersytet Wrocławski Wydział Matematyki i Informatyki Instytut Matematyczny
- Mateusz Kłeczek, Geometria hiperboliczna, Chrzanów 2016
- Caroline Series With assistance from Sara Maloni, Hyperbolic geometry MA448
- Steve Szidlik, Hyperbolic Constructions in Geometer's Sketchpad, December 21, 2001
- Marek Kordos, Geometria Bolyaia–Łobaczewskiego, <http://www.deltami.edu.pl>, Sierpień 2018



## **Zawartość płyty CD**