Symulacja dyskretna systemów złożonych
Hokej z ładunkiem elektrycznym

Emil Sroka Roman Yatsuniak Hubert Miziołek

Spis treści

1 Wstęp **2**

1.1 Opis problemu 2

1.2 Możliwości rozwiązania - analiza wstępna 2

2 Analiza literatury **3**

2.1 Kinematyka 3

2.2 Dynamika 3

2.3 Zderzenia sprężyste na płaszczyźnie 4

2.4 Rotacja wektora 5

2.5 Detekcja kolizji 5

2.6 Pole elektryczne 5

2.7 Źródła 6

3 Propozycja modelu rozwiązania **7**

3.1 Zastosowane uproszczenia 7

3.2 Dane wejściowe i interakcja z użytkownikiem 7

4 Implementacja modelu **8**
4.1 Użyte narzędzia 8
4.2 Implementacja obiektów fizycznych i zarządzanie nimi 8
4.3 Implementacja Symulacji 10
4.4 Wizualizacja 11
4.5 Skrócona całość implementacji 12

5 Wyniki wizualizacji **13**
5.1 Przykładowa konfiguracja bez lini pola 13
5.2 Przykładowa konfiguracja z liniami pola 13
5.3 Przygotowane plansze 14

6 Podsumowanie **17**

1 Wstęp

1.1 Opis problemu

Celem naszego projektu jest symulacja ruchu ładunku elektrycznego w polu elektrycznym i zaprezentowanie tego w postaci gry w hokeja ładunkiem elektrycznym.

1.2 Możliwości rozwiązania - analiza wstępna

Mamy tu do czynienia z prostym problemem pod względem fizycznym - do przedstawienia ruchu ładunku wystarczy wyznaczyć siłę wypadkową, z której już łatwo określić przyśpieszenie i jego kierunek dla danego ładunku. Do jej wyznaczenia należy zastosować zasadę superpozycji. Problemem tej metody jest dość szybko rosnąca złożoność obliczeniowa dla coraz to większej liczby ładunków.
Jedynym dodatkowym zdarzeniem, które może wystąpić podczas takiego ruchu jest zderzenie z innym obiektem. Korzystając z zasady zachowania pędu oraz energii można wyliczyć skutek takiego zderzenia.

2 Analiza literatury

2.1 Kinematyka

Zależność położenia ciała od czasu w ruchu jednowymiarowym:

$$x(t) = x_0 = v_0 t + \frac{at^2}{2}$$

Ruch na płaszczyźnie można traktować jak dwa niezależne ruchy jednowymiarowe.

2.2 Dynamika

Druga zasada dynamiki Newtona:

$$\overrightarrow{F} = m\overrightarrow{a}$$

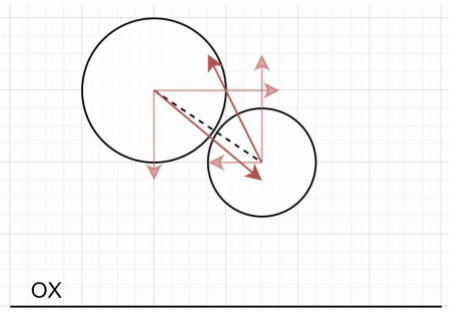
Zderzenia sprężyste w jednym wymiarze (One-dimensional Newtonian):

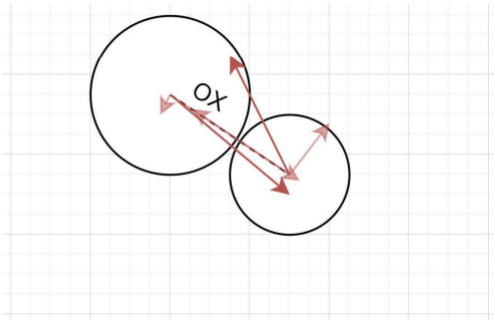
$$v_1 = \frac{m_1 - m_2}{m_1 + m_2} u_1 + \frac{2m_2}{m_1 + m_2} u_2$$

$$v_2 = \frac{2m_1}{m_1 + m_2} u_1 + \frac{m_2 - m_1}{m_1 + m_2} u_2$$

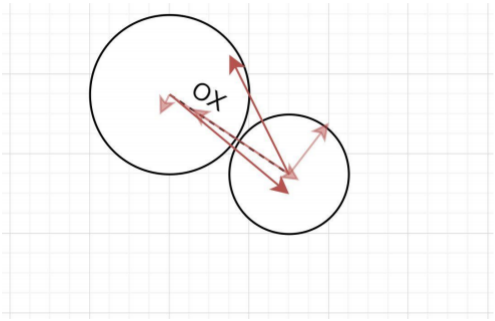
2.3 Zderzenia sprężyste na płaszczyźnie

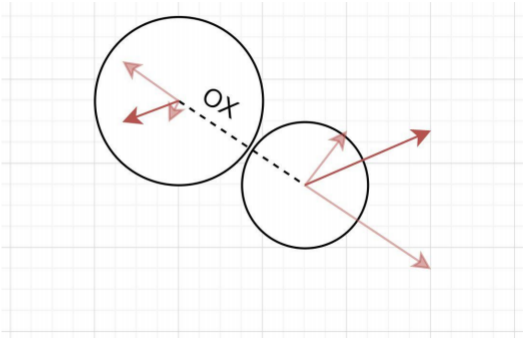
Rotacja wektorów prędkości obiektów o kąt między OX a prostą utworzoną między środkami obiektów.



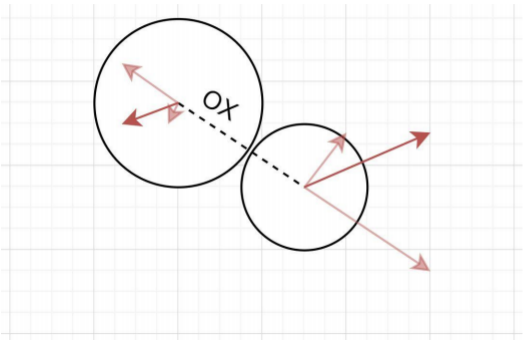


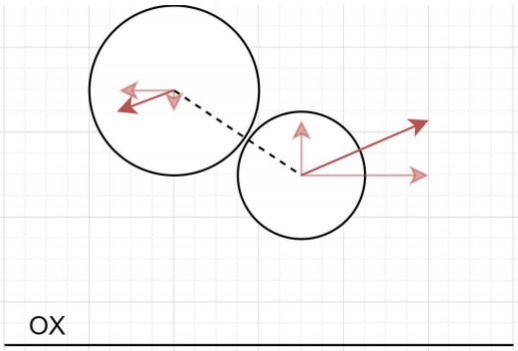
Obliczenia kolizji w osi OX:





Rotacja odwrotna:





YouTube: "How to Code: Collision Detection Part II"

2.4 Rotacja wektora

Rotację wektora można dokonać z pomocą macierzy obrotu:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2.5 Detekcja kolizji

Koło i koło:
Dwa koła kolidują kiedy dystans między ich środkami jest mniejszy lub równy sumie ich promieni
Koło i prostokąt:
Koło i prostokąt kolidują kiedy środek koła jest wewnątrz prostokąta lub jedna z krawędzi prostokąta ma punkt wspólny z okręgiem

2.6 Pole elektryczne

Ładunek elementarny: $e = 1,6 \cdot 10^{-19} C$
Wszystkie realnie istniejące ładunki są wielokrotnością ładunku e (skwantowanie)
Prawo Coulomba:

$$F = k\frac{q_1 q_2}{r^2}, k = \frac{1}{4\pi\epsilon_0\epsilon_r}$$

Zasada superpozycji:
Siłę wypadkową, obliczamy dodając wektorowo poszczególne siły dwuciałowe. Natężenie pola elektrycznego:

$$\overrightarrow{E} = \frac{\overrightarrow{F}}{q}$$

2.7 Źródła

1. Kinematyka:
   Zbigniew Kąkol - e-Fizyka, Ruch jednowymiarowy, Ruch na płaszczyźnie

2. Dynamika:
   Zbigniew Kąkol - e-Fizyka, Podstawy dynamiki, Zderzenia

3. Zderzenia:
   YouTube: "How to Code: Collision Detection Part II"

4. Rotacja wektora:
   Wikipedia: Rotation matrix Rotation matrix

5. Detekcja kolizji:
   YouTube: Coding Math: Episode 14 - Collision Detection
   Stackoverflow: Circle-Rectangle collision detection (intersection)

6. Pole elektryczne:
   Zbigniew Kąkol - e-Fizyka, Pole elektryczne

3 Propozycja modelu rozwiązania

3.1 Zastosowane uproszczenia

W naszym modelu zdecydowaliśmy się na następujące uproszczenia:

1. Tylko elementy posiadające ładunek oddziałują między sobą za wyjątkiem zderzeń.
2. Każde zderzenie jest idealnie sprężyste
3. Jedyna siła która działa na ładunki to siła elektrostatyczna
4. Każdy ładunek znajduje się w stanie spoczynku w chwili startu
5. Przeszkody są nieruchome i mogą oddziaływać z ładunkami tylko poprzez zderzenia

3.2 Dane wejściowe i interakcja z użytkownikiem

Na potrzeby symulacji przygotowaliśmy kilka bazowych boisk do rozgrywki w hokeja w tym także "Free mode", gdzie użytkownik jest w stanie przeprowadzić symulacje na czystym boisku dla maksymalnie 100 ładunków.

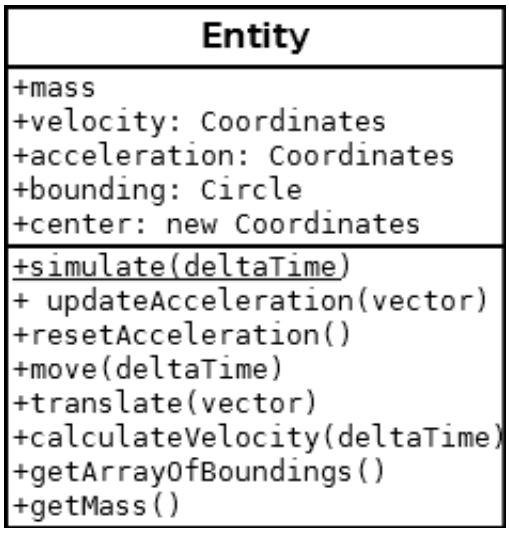
4 Implementacja modelu

4.1 Użyte narzędzia

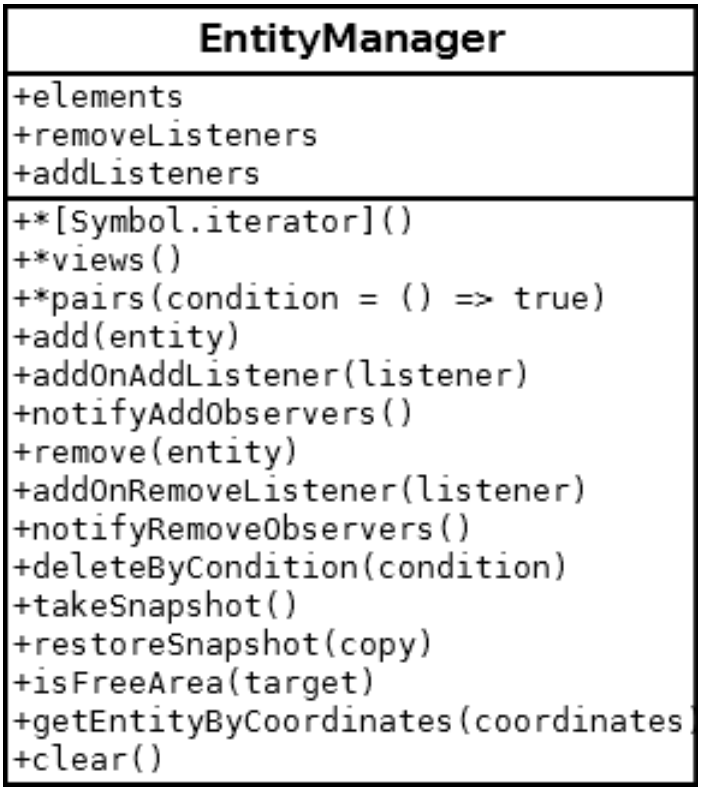
W naszym proponowanym rozwiązaniu zdecydowaliśmy się na użycie javascript'u ze względu na łatwość wizualizacji w przeglądarce. Jest on także stosunkowo szybki jak na język interpretowany dzięki "just in time compilation".

4.2 Implementacja obiektów fizycznych i zarządzanie nimi

Entity jest podstawą każdego obiektu fizycznego stworzonego na potrzeby symulacji. Dodatkowe cechy, takie jak statyczność czy elektryczność, są dodawane poprzez obiekty decorator.



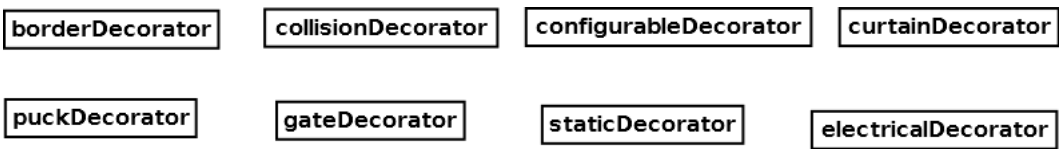
Rysunek 1: Implementacja entity



Rysunek 2: Implementacja EntityManager'a

EntityManager odpowiada za zarządzanie istniejącymi już obiektami w symulacji. Obiekt jest reprezentowany przez parę obiektów: Entity i View.

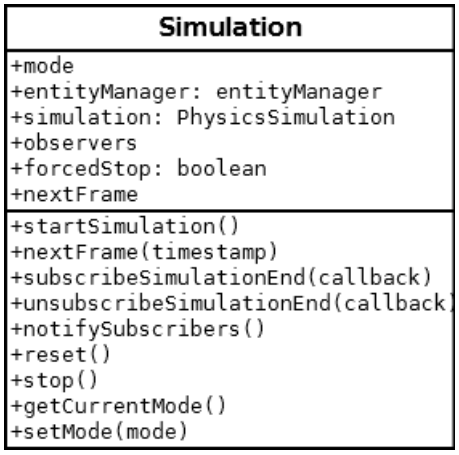
Klasa View korzystając z danych zawartych w Entity oraz obiektu Painter rysuje reprezentację graficzną obiektu.



Rysunek 3: Zaimplementowane dekoratory

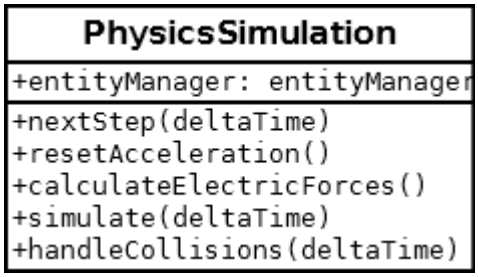
4.3 Implementacja Symulacji

PhysicsSimulation odpowiada za silnik fizyczny. Oblicza oddziaływania między obiektami oraz kolizje. Obiekt mode ustawia stan początkowy obiektu EntityManage. Pozwala też przywrócić stan EntityManage po resecie symulacji oraz zmianie trybu.



Rysunek 4: Implementacja symulacji

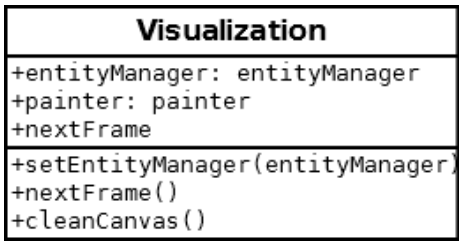
Klasa Simulation spaja całość symulacji. Za pomocą obiektu mode sprawdza wynik symulacji i podejmuje decyzję o następnym kroku symulacji.



Rysunek 5: Implementacja zmian fizycznych

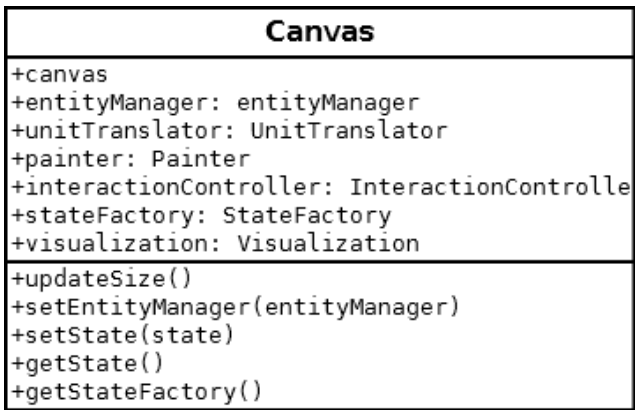
4.4 Wizualizacja

Visualization odpowiada za rysowanie klatki symulacji.



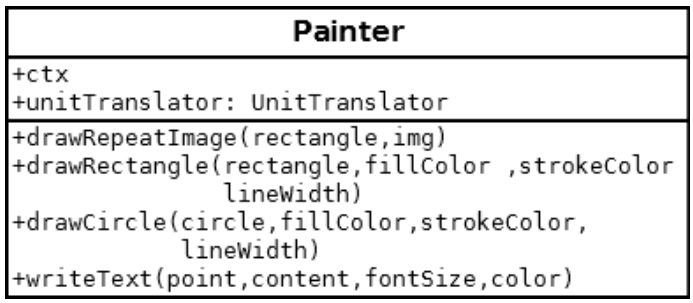
Rysunek 6: Implementacja wizualizacji

InteractionController odpowiada za interakcję z użytkownikiem w zakresie elementu HTML5 canvas. Korzystając ze wzorca projektowego state, pozwala na inne czynności użytkownikowi w różnych sytuacjach.



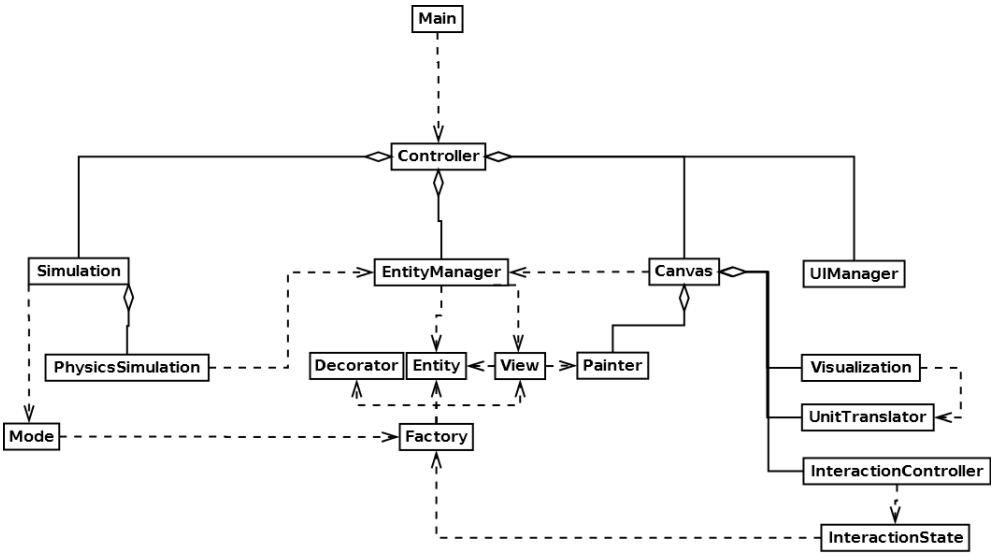
Rysunek 7: Główny element modułu wizualizacji

Painter umożliwia obiektom View, rysowanie udostępniając konkretny zestaw metod. Korzysta przy tym z obiektu UnitTranslator, pozwalającego przeliczać jednostki między układem odniesienia użytym w symulacji a układem współrzędnych na stronie internetowej.



Rysunek 8: Element pozwalający na "rysowanie"

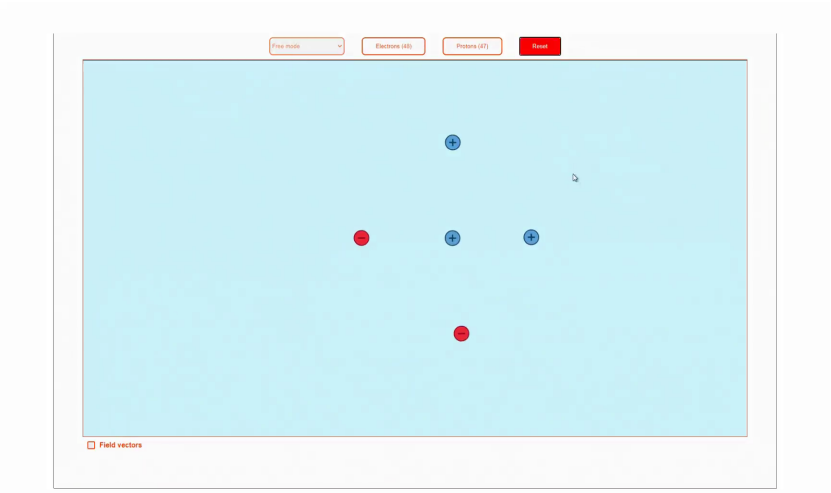
4.5 Skrócona całość implementacji



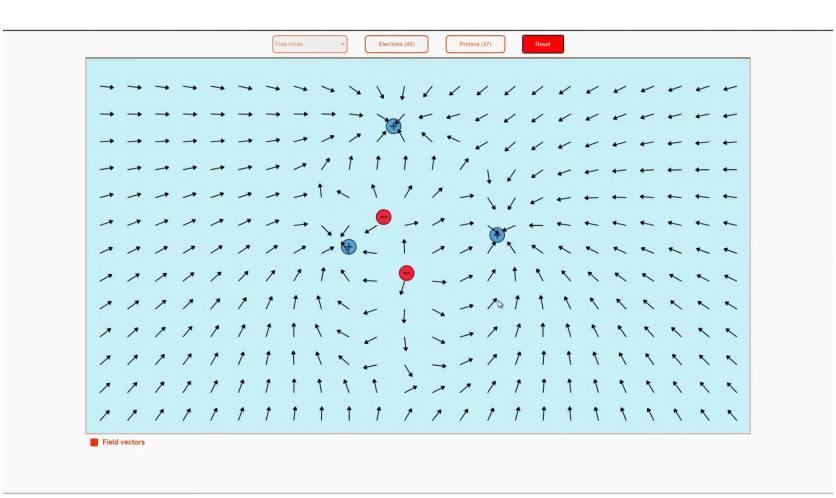
Rysunek 9: Skrócona implementacja zgodna z UML

5 Wyniki wizualizacji

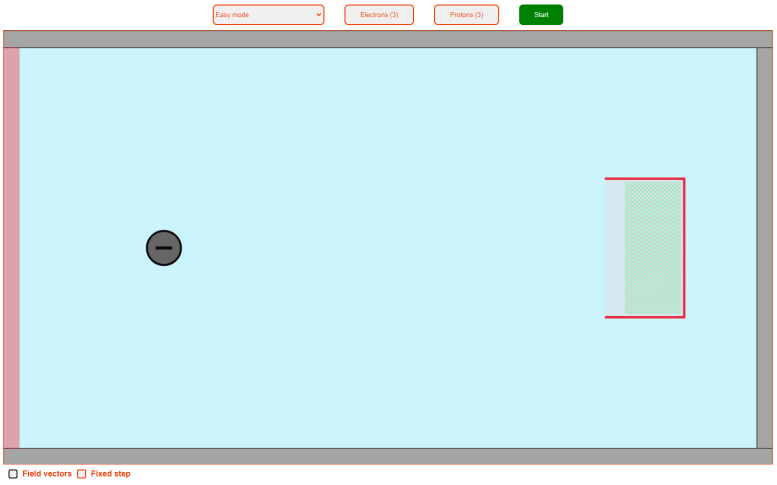
5.1 Przykładowa konfiguracja bez lini pola



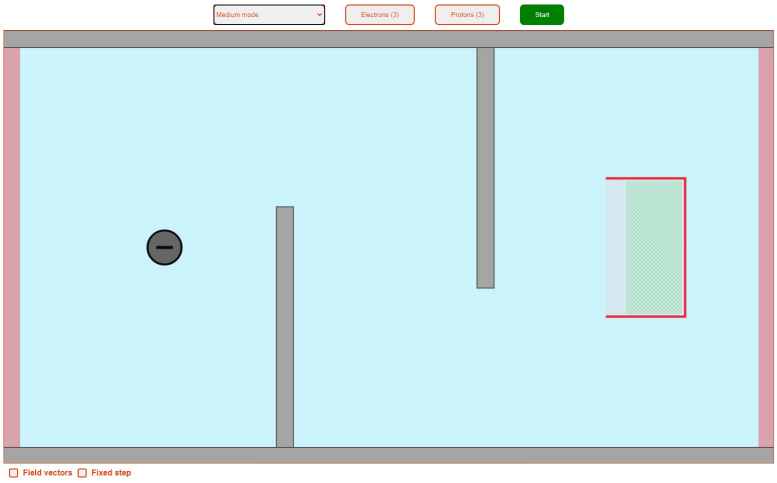
5.2 Przykładowa konfiguracja z liniami pola



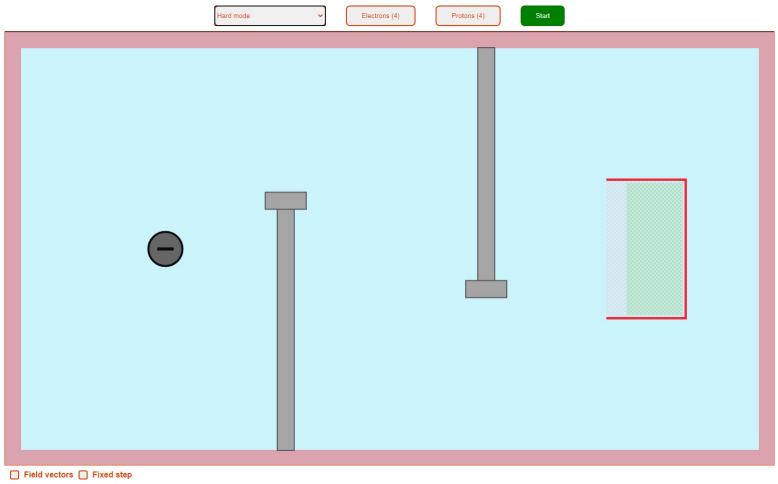
5.3 Przygotowane plansze



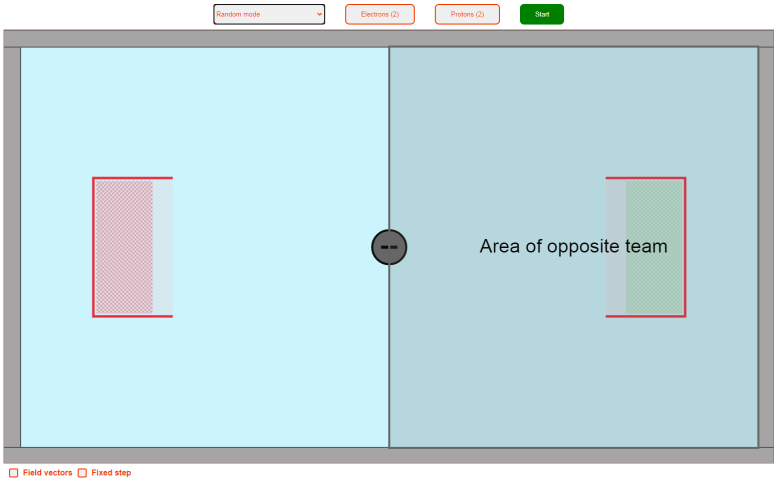
Rysunek 10: Easy mode



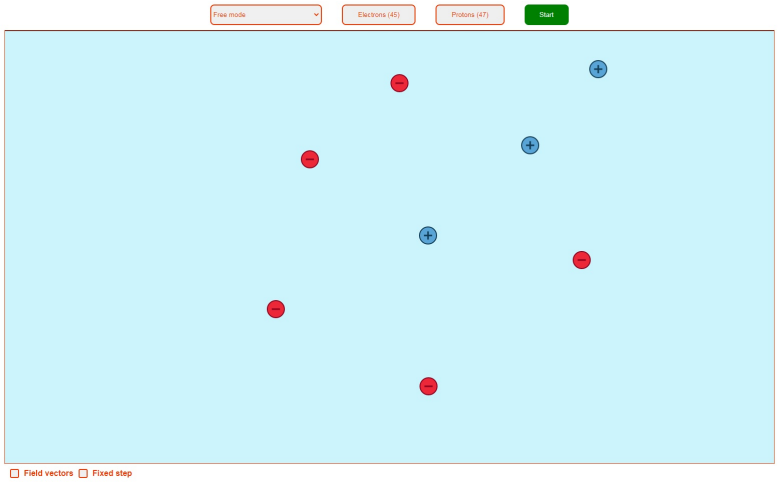
Rysunek 11: Medium mode



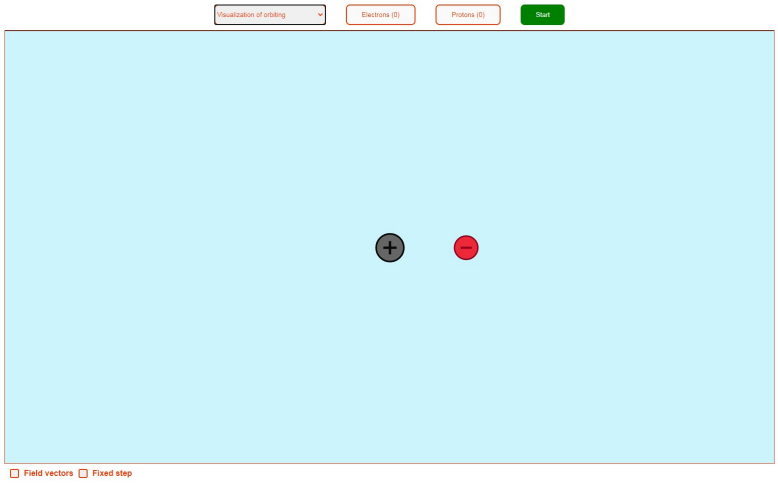
Rysunek 12: Hard mode



Rysunek 13: Random mode



Rysunek 14: Free mode



Rysunek 15: Przygotowana plansza z gotowym orbitowaniem ładunku

6 Podsumowanie

Symulacja w czasie rzeczywistym w poprawny sposób pokazuje ruch elektronu - nasz cel został osiągnięty.Jedynym powtarzającym się błędem jest "łączenie" się ładunków, kiedy te znajdą się za blisko. W wyniku niedokładności obliczeń powstaje wtedy sytuacja, gdzie połączone ładunki zaczynają się bardzo szybko obracać wokół siebie nabierając dużych prędkości.
Użyta przez nas technologia ma też swoje minusy - wybór kroku. Dynamiczne dobieranie kroku przez API przeglądarki może zmienić wynik symulacji dla tych samych warunków początkowych, a ustawienie go statycznie sprawia, że tempo wyświetlania jest zależne od przeglądarki oraz sprzętu.
Możliwym ulepszenie jest zoptymalizowanie algorytmu odpowiadającego za wyliczanie poszczególnych przyśpieszeń oraz zwiększyć odporność symulacji na błędy numeryczne np błąd zaokrąglenia.
Dzięki tej symulacji można łatwo zaobserwować niektóre zjawiska fizyczne np. ruch ładunku po orbicie, zderzenia sprężyste itp. Przystępność takiego zobrazowanie może zostać użyta do wytłumaczenia tych zjawisk.