QB - React Three Fiber | Projekt Grafika 3D

Tomasz Gajda

SPIS TREŚCI

1. Problem i założenia projektu **1**

2. Krótki wstęp teoretyczny **2**

3. Zakres funkcjonalności **2**

4. Opis implementacji **4**

5. Podsumowanie i pomysły na rozbudowę **5**

6. Literatura **6**

7. Kod źródłowy **7**

1. Problem i założenia projektu

Celem projektu było stworzenie prostej gry, która dostępna byłaby z poziomu różnorodnych urządzeń. Jako że najłatwiejszą opcją do stworzenia gry międzyplatformowej jest skorzystanie z technologii internetowych - tak właśnie uczyniłem.

Stos technologiczny:

- Typescript
- React
- SASS
- @react-three/fiber
- zustand
- netlify

Pomniejsze wykorzystane biblioteki:

- react-icons
- react-device-detect

2. Krótki wstęp teoretyczny

Stworzona gra korzysta z frameworka **React** i wrappera **React-Three-Fiber** który obudowuje zawarte w **Three.js** klasy w komponenty wykorzystywane w React. Biblioteka ta oferuje prostszy interfejs obsługi 3D w Javascript/Typescript nie tracąc przy tym na wydajności.

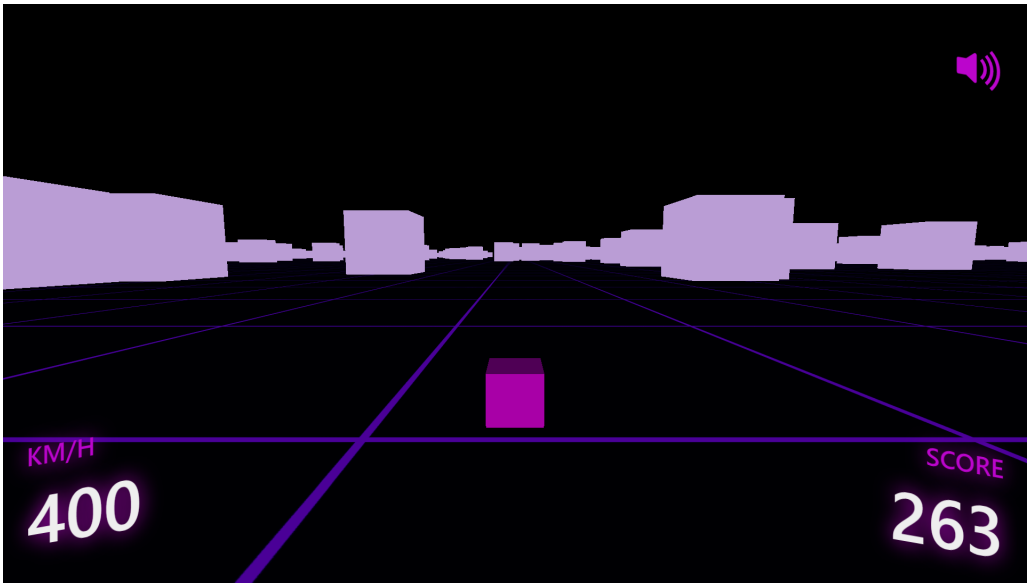
Oprócz tego wykorzystałem preprocesor SASS do stylowania elementów, które tego wymagały w aplikacji (np. **HUD**). Na koniec za pomocą **netlify** i konsolowego interfejsu szybko deployowałem grę do sieci.

3. Zakres funkcjonalności

Zasady gry są dość proste - gracz ma za zadanie omijać sześciany znajdujące się na trasie i przebyć w ten sposób jak **największy** dystans.

3.1 Proces gry

Kontrolując blok za pomocą strzałek bądź klawiszy “A” oraz “D” (czy też specjalnych przycisków na ekranach mobilnych) gracz musi omijać przeszkody znajdujące się na płaszczyźnie.



Rys. 1 - Główny ekran gry

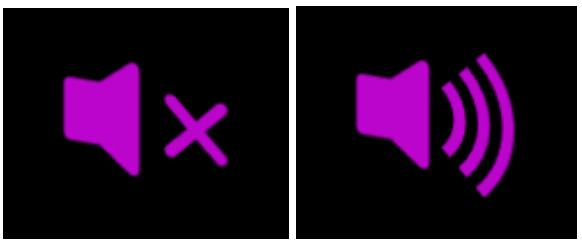
3.2 Zapisywanie wyników

Uzyskany wynik jest widoczny w trakcie gry jak i na ekranie po przegranej. Trzy najlepsze wyniki są zapisywane do **localStorage** i wyświetlane na ekranie “Game Over”.

Rys. 2 - Zestaw wyników na ekranie końcowym

3.3 Manipulacja muzyką

W grze dodany jest również Soundtrack - za pomocą przycisku w prawym górnym rogu ekranu możemy go włączać/wyłączać w trakcie gry.



Rys. 3 - Przycisk, którym możemy manipulować muzyką

4. Opis implementacji

W aplikacji znajdziemy zbiór komponentów, które wraz z zapisywanym stanem tworzą całą grę:

4.1 Stan aplikacji

W folderze “store” znajduje się zapisany stan naszej aplikacji. Do manipulacji stanem w aplikacji wykorzystałem bibliotekę **“zustand”**, czyli rozwiązanie dużo prostsze w użyciu niż popularne aktualnie biblioteki Redux czy też oferowane przez samego React’a Context API. Możemy tam znaleźć wszystkie zmienne przechowywane w stanie. Są one aktualizowane w czasie działania aplikacji.

4.2 Player

Komponent ten przedstawia model sześcianu, którym steruje gracz oraz wszystkie operacje które manipulują jego pozycją, szybkością poruszania itp. Są tutaj też manipulację kamerą, która powinna podążać za sześcianem.

4.3 Obstacles

Komponent ten odpowiada za generowanie przeszkód na trasie - sześcianów o losowo wygenerowanej pozycji na płaszczyźnie Z.

4.4 Ground

Komponent generuje płaszczyznę o teksturze fioletowej siatki, po której porusza się gracz i na której pojawiają się przeszkody.

4.5 GameState

Pomimo przechowywania stanu gry w innym miejscu, stworzyłem również ten komponent by śledził aktualną sytuację w grze, aktualizował wynik i w przypadku końca gry zmieniał stan.

4.6 KeyboardControls

Komponent w pełni poświęcony śledzeniu inputa z klawiatury - na bieżąco zapisuje do stanu, które przyciski są naciskane.

4.7 Soundtrack

Komponent obsługujący muzykę zaimplementowaną w grze. Nasłuchuje zmiany na przycisku odpowiadającego za manipulację muzyką w HUD.

4.8 HUD

Komponent nakładający HUD na ekran gry - przedstawia wszystkie ważne dla użytkownika informacje w jasny i czytelny sposób. W tym komponencie również sprawdzane jest jakim urządzeniem posługuje się użytkownik - jeśli jest to urządzenie mobilne, renderowane są dodatkowe przycisku do sterowania.

4.9 GameOver

Komponent z ekranem “Game Over” - wyświetla wynik aktualnej próby oraz listę najlepszych wyników zapisaną w localStorage.

5. Podsumowanie i pomysły na rozbudowę

React-Three-Fiber daje niesamowite możliwości tworzenia gier i aplikacji internetowych przy użyciu całkiem prostego API. Bardzo przyjemnie mi się z nim pracowało, natomiast czas który chciałem poświęcić na stworzenie gry był ograniczony, dlatego też kod nie we wszystkich miejscach wygląda tak jak powinien, a i kilka funkcjonalności, które planowałem na początku nie zostało zaimplementowane:

5.1 Zmiana trudności

W planach miałem implementację zwiększającego się poziomu trudności - wraz z przekroczeniem kolejnych kamieni milowych (np. 5000 pkt.) ilość przeszkód lub prędkość kierowanego obiektu miały się zwiększyć (dlatego też mamy licznik prędkości w lewym dolnym rogu HUD).

5.2 Skybox

W tym momencie tło nad płaszczyzną jest czarne - dodaje to klimatu grze, natomiast można by było tam wstawić jakąś ładną grafikę, która pasowałaby do aplikacji.

5.3 Ładniejsze tekstury

Nie jestem największym fanem tego w jaki sposób wyglądają obecnie przeszkody - materiał klocków mógłby być ładniejszy. Można w tym miejscu nawet pobawić się shader'ami.

5.4 Lepsza detekcja kolizji

Pomimo że wszystkie obiekty są sześcianami, detekcja opiera się na odległości po promieniu okręgu o środku wewnątrz sześcianu - to powoduje że czasami pomimo dotknięcia krawędzi przeszkody uda się nam ujść z życiem.

5.5 Wgrywanie customowych modeli

Choć sześcian się wpisuje w kwadratowy wygląd świata przedstawionego, zdecydowanie ciekawym pomysłem byłaby możliwość dodania modelu, którym sterował by gracz zamiast sześcianu.

6. Literatura

Do stworzenia projektu wykorzystałem głównie **oficjalną dokumentację RTF**, oraz pomniejsze strony służące jako poradniki do obsługi tej biblioteki:

1. https://docs.pmnd.rs/react-three-fiber/getting-started/introduction
2. https://github.com/pmndrs/react-three-fiber
3. https://codesandbox.io/examples/package/react-three-fiber

7. Kod źródłowy

Kod źródłowy jest dostępny na platformie GitHub w repozytorium, które można znaleźć pod tym adresem: https://github.com/nerooc/3d-project