**Szegedi Tudományegyetem**

**Informatikai Intézet**

**Szakdolgozat**

**Pullai Szilárd**

**2024**

**Szegedi Tudományegyetem**

**Informatikai Intézet**

**Interaktív 3D Technológiák a Weben:**

**Dinamikus Modell Megjelenítő Fejlesztése React Three Fiberrel**

Szakdolgozat

|  |  |
| --- | --- |
| Készítette: | Témavezető: |
| **Pullai Szilárd** | **Dr. Iván Szabolcs** |
| Üzemmérnök-Informatikus Bprof Szakos hallgató | Tanszékvezető, egyetemi tanár |

**Szeged**

**2024**

**Feladatkiírás**

A kitűzőtt feladat egy webalkalmazás elkészítése amely képes részletes  
3D modellek megjelenítésére. A projekt elkészítése során cél a Three.js, React.js könyvtárak alapos elsajátítása és átfogó elemzés készítése az alkalmazásban implementált optimalizálásokról.

A teljesítménynek skálázódik, mivel a programnak futnia kell mobil eszközökön, laptopokon és asztali számítógépeken, változó felbontásokat és képarányokat támogatva.

Tömörített 3D modelleket kell kezelnie az alkalmazásnak, mivel cél a gyors működés, mobilhálozatról is gyorsan és gördülékenyen kell történnie a betöltésnek.  
Az optimalizálási módokról részletes elemzést kell készíteni és a feltárt összefüggéseket ábrákon keresztül szemléltetni.

Az alkalmazásnak rendelkezni kell felhasználói felülettel, amin keresztül interakív módon változtatható a betöltött 3D modell megjelenése, a szintér környezete és a virtuális kamera forgatása.

A renderelésnél használt shader programok működésének megismerése, különböző fény számítási algoritmusok elemzése és összehasonlítása. Utófeldolgozás során alkalmazott algoritmusok megértése és ismertetése.

**Tartalmi összefoglaló**

* **Téma megnevezése:**Interaktív 3D Technológiák a Weben:  
  Dinamikus Modell Megjelenítő Fejlesztése React Three Fiberrel
* **A megadott feladat megfogalmazása:**Egy webes környezetben működő, 3D modellek realisztikus megjelenítésére alkalmas alkalmazás fejlesztése, amely interaktív funckiókat tartalmaz és optimalizáltságnak köszönhetően mobilokon is jól működik.  
  A fejletszési folyamat alatt használt techinkák elemzése, valamint az optimalizálási módok szemléltetése.
* **A megoldási mód:**  
  Az alkalmazás React Three Fiber könyvtár felhasználásával készült JavaScript programozási nyelvben írva, a Visual Studio Code fejlesztői környezeten belül.
* **Alkalmazott eszközök, módszerek:**Az alkalmazás alapját a ***React Three Fiber*** könytár adja, továbbá a kamera és irányítás funkciók, illetve a 3D modell importáló modul a ***drei*** kollekcióból lett felhsaználva. Állapotok kezelését a ***Zustand*** rendszerével kéeszült. ***Blender*** szoftverrel történt a modellek optimalizálása, szekesztése, tömörítése.
* **Elért eredmények:**Az elkészült program gyorsan és gördülékenyen fut, illetve betölt gyengébb eszközökön is, az optimalizálásoknak köszönhetően. A fejlesztések lehetőséget biztosítanak az eszköz továbbfejlesztésére. A dokumentált optimalizáslások pedig felhasználhatóak további 3D-s alkalmazásokban.
* **Kulcsszavak:**  
  Interaktív, 3D, React, webalkalmazás, React Three Fiber, Drei, Zustand, Blender, optimalizáció, modell megjelenítő, Three.js, JavaScript, Web

**Tartalomjegyzék**

todo

**Bevezető**

Azért szerettem volna 3D grafikához kapcsolódó szakdolgozatot készíteni, mert az egyetemi évek alatt a ***Számítógépes grafika*** kurzus keltette fel legjobban az érdeklődésemet, és tanulmányaim utolsó évében már dolgoztam is ezen a szakterületen.

Az utóbbi időben egyre jobban elterjedtek a 3D grafikát használó alkalmazások, a kis méretű és egyre erősebb hardvereknek köszönhetően. Gondolok itt mobil, AR és VR eszközökre amelyek mind rendelkeznek böngészővel és azon keresztül tudnak tartalmat megjeleníteni. Vannak felhasználási módok ahol különösen hasznos, például 3D animációk az oktatásban, épületek terve három dimenziós térben, körbeforgatható termékek megjelenítése weboldalon stb.

Szakdolgozatomban azt szeretném leírni, hogy milyen módszerekkel kell egy 3D-s alkalmazást elkészíteni, amely működik bármilyen modern böngészővel rendelkező eszközön, legyen az PC, mobil vagy VR eszköz. Ahogy a címben is látható a React függvénykönvtár és a Three.js segítségével fogom mindezt elérni.

Kiemelkedő szerepet fog kapni az optimalizálás, mivel az alkalmazás betöltésnél és a renderelésnél is sok lehetőségünk van elérni egy adott megjelenést állapotot, de ezek nem mindig a legköltséghatékonyabban történnek, ha a fejlesztő nem ismeri a pontos működést és a teljesítmény növelésére alkalmas módszereket.

A különböző megvalósítási módszerek összehasonlításával és fejlesztési eszközök jellemezésével az a célom, hogy a jövőbeli ilyen jellegű alkalmazások fejlesztése könyebb legyen és a megvalósított alkalmazások jobb teljesítménnyel fussanak gyengébb eszközökön is.

**1. 3D Webes alkalmazások**

Webes alkalmazások fejlesztéséhez elengedhetetlen a HTML, CSS és JavaScript technológiák ismerete, viszont ha 3D-s webalkalmazást csinálunk tisztában kell lenni a WebGL működéséről is. Ezt leegyszeűsítik a különböző függvénykönyvtárak esetünkben a Three.js ami már előre megírt WebGL implementálásokat tartalmaz. Modern fejlesztéseknél keretrendszereket alkalmaznak, mint pl. Angular, Vue vagy React amik lehetővé teszik könnyen újrafelhasználható komponensekké szerveznünk a megírt kódot. Ebben a fejezetben ezek a technológiák lesznel bővebben kifejtve, hogy tisztában legyünk vele, mi is a története a szakdolgozat-projektemben felhasznált technológiáknak.

**1.1 WebGL Technológia**

Angolul Web Graphics Library, egy alacsony szintű, renderelő API, amely közvetlenül a böngészőben fut, és lehetővé teszi a 3D grafika megjelenítését egy HTML5 szabványból ismert Canvas elem segítségével. Ennek a technológia felhasználásával a fejlesztők készíthetnek olyan interaktív 3D alkalmazásokat, mint játékok, vizuális szimulációk, oktatási eszközök, és egyéb dolgok, amelyek elérhetőek bármilyen modern webböngészőn keresztül.

Több verziója van jelen, most a WebGL 2.0 a legnépszerűbb amely az elődje által lefektetett alapokon, és az OpenGL ES 3.0-ra épít. Ennek köszönhetően számos új funkciót és API-t kínál, amelyek jelentősen bővítik a grafikai lehetőségeket, mint például jobb textúrázási technikák és komplexebb árnyékolási modellek. Ezen felül a WebGL 2.0 garantálja az előző verzió sok opcionális kiterjesztésének elérhetőségét, ami nagyobb rugalmasságot és jobb teljesítményt biztosít a fejlesztőknek.

A shaderek, amelyek a WebGL-ben GLSL nyelven íródnak, kulcsfontosságú szerepet játszanak a grafikus renderelésben. Ezeket a shadereket szövegsorozatként adja át a fejlesztő a WebGL API-nak, amely fordítási folyamaton keresztül a grafikus feldolgozó egység gépi kód jává alakítja át őket. Ez a GPU kód végzi el a számításokat minden egyes 3D objektum csúcspontjára és a képernyőn megjelenített pixelek RBG értékét, ami lehetővé teszi a fejlesztők számára, hogy létrehozzanak vizuális effekteket és részletesen kidolgozott 3D modelleket közvetlenül a webböngészőben.

A shader kódok az OpenGL ES Shading Language (GLSL ES) használatával készülnek, ami egy olyan nyelv, ami hasonlít a C vagy C++ programozási nyelvekre. Egy webalkalmazáson belül a JavaScript a WebGL API-t használva irányítja az alkalmazás logikáját és kezeli a bemeneteket a felhasználó és a weboldal között, míg a GLSL kódok, amelyek a grafikus műveletekért felelősek, közvetlenül a GPU-n futnak. A GPU-ra írt kódok lehetővé teszik a számítási és renderelési műveletek gyors végrehajtását, kihasználva a grafikus processzor párhuzamos számítási képességét, ami sokkal gyorsabb, mint a JavaScript kód, ami a fő processzorn futtatná az utasításokat.

A fejlesztők a WebGL-ben definiált két fő típusú shader kódot használhatnak: a vertex shader-eket, amelyek minden egyes 3D modell csúcspontjára lefutnak, és a fragment shader-eket, amelyek minden egyes pixel számításáért felelősek a képernyőn. Ezzel a két shader-rel lehetőség nyílik arra, hogy a fejlesztők részletesen kontrollálják a grafikus kimenet minden aspektusát, beleértve a fények és árnyékok, textúrák, és egyéb vizuális effektek kezelését.

WebGL programok esetében kritikus a teljesítmény optimalizálása, mivel a komplex 3D grafikák számítási igénye magas lehet. A fejlesztőknek gondoskodniuk kell arról, hogy a kódjaik hatékonyan használják a rendelkezésre álló erőforrásokat, minimalizálva ezzel a böngészők és a végfelhasználói eszközök terhelését. A WebGL kiterjedt eszköztárat és technikákat kínál a teljesítmény mérése és profilozására, segítve a fejlesztőket a hatékonyabb grafikai alkalmazások létrehozásában.

**1.2 three.js függvénykönyvtár**

A Three.js egy magas szintű, JavaScript-alapú 3D grafikai könyvtár, amelyet kifejezetten a WebGL API-ra építettek. A célja, hogy megkönnyítse a fejlesztők számára a 3D grafikák létrehozását a webes környezetben.

A Three.js egy absztrakciós réteget biztosít a WebGL fölött, így a fejlesztőknek nem kell közvetlenül a WebGL alacsony szintű API-jával foglalkozniuk. Ez a könyvtár egyszerűsíti a bonyolult grafikai programozási feladatokat. Előre definiált shaderek biztosítják az objektumok megjelenését, a fények kezelését és a színek tónusleképezését.

A Three.js rengeteg beépített funkciót és komponenst tartalmaz, mint például kamera típusok, fényforrások, geometriai alakzatok, animációs eszközöket és 3D modell beolvasó funkciókat. Ezek a komponensek előre megírtak és optimalizáltak, így jelentősen csökkentik a fejlesztési időt és erőfeszítést.

A Three.js könnyen integrálható más webes technológiákkal és keretrendszerekkel. Támogatja a különféle kimeneti formátumokat is, beleértve a WebGL mellett a Canvas 2D, SVG, és CSS3D renderereket. Ez a rugalmasság teszi lehetővé a fejlesztők számára, kiválasszák a projekthez szükséges technológiát. Ezekhez kapcsolódva jön ide a React függvénykönyvtár is amit elég effektíven tudunk használni a Three.js által előre definiált funckiókkal.

**1.3 react.js mint keretrendszer**

A React egy JavaScript könyvtár, amelyet dinamikus adatokkal rendelkező, webalkalmazások felhasználói felületeinek létrehozására fejlesztettek ki. A Meta amerikai cég alkotta meg és tartja karban a nyílt forráslódú kódbázist, közreműködő felhasználók segítségével.

A React alkalmazások alapvető elemei a komponensek, amelyek újrafelhasználható, önálló elemek, és definiálják a felhasználói felület vizuális és interaktív aspektusait. Egy React alkalmazás több, egymásba ágyazott komponensből áll.

A React JSX-et használ ami egy JavaScript szintaxis kiterjesztés, ami HTML-re hasonlít. A JSX segítségével a fejlesztők HTML szerkezeteket írhatnak a JavaScript kód mellé, ami olvashatóbbá és kifejezőbbé teszi a kódot. A forrásfájlokba komponenseket hozhatunk létre, amelyek a React alkalmazásunk építőelemei, lehetnek osztály alapúak vagy funkcionálisak.

Az adatokat a szülő komponensektől a gyermek komponensek felé lehet átadni. Ezek olvasásra szántak és nem módosíthatók a gyermek komponensek által. React fejlesztők ezt „prop” –nak nevezik, mivel tudjuk ezzel tudjuk kezelni egy komponens viselkedését. Emellett még van állapota is komponenseknek, módosítható, és a React új létrehozza a komponenst az állapot változásakor, ami újrarajzolást eredményez.

A React működése egy virtuális DOM-on, Document Object Modell –en alapszik. Ez egy memóriában létrehozott másolat a valós DOM-ról, ahol a React először minden manipulációt ezen a virtuális DOM-on végez. A virtuális DOM használatának több előnye is van. Először is, a React kiszámítja az aktuális oldalszerkezet és az új szerkezet közötti különbségeket, majd csak a szükséges változásokat hajtja végre a valós DOM-on, ami hatékony frissítéseket tesz lehetővé. Másodszor, a React több frissítést összegyűjt a virtuális DOM-ban, majd egyetlen újrarendereléssel frissíti a valós DOM-ot, optimalizálva ezzel a teljesítményt. Amikor egy komponens állapota vagy tulajdonsága megváltozik, akkor a React összehasonlítja az újonnan visszaadott elemet az előzővel. Ha van különbség, akkor a React ennek megfelelően frissíti a DOM-ot. Ez az összehasonlítási folyamat egy speciális algoritmus segítségével történik, amely feltételezi, hogy különböző típusú elemek különböző fastruktúrákat eredményeznek, így a DOM frissítése hatékonyan történik.