3D grafikus motor fejlesztése Vulkan alapokon

Önálló labor II.

Oczot Balázs  
PRAMRX

# Bevezetés

A feladatom egy 3D grafikus motor fejlesztése volt Vulkan és OpenGL alapokon, úgy hogy a program indulása előtt el lehet dönteni hogy melyik fajta render-engine-t szeretném használni.

Volt már korábbról egy OpenGL-es játékmotorom, de ez eléggé az olasz konyha stílusában készült, tehát a kód egy része igen spagetti volt, ezeket újraírtam. Mivel a vulkánnal előtte még nem találkoztam, így ez idegen terep volt, de mindenképpen ki akartam próbálni, így ennek az implementációja biztos vagyok hogy sok helyen nem hatékony és lehetne jobban is csinálni ha több tapasztalatom lenne az API-val.

Megvalósított funkciók:

* Tetszőlegesen nagy háromszögháló modell betöltése
* Textúrák használata
* Blinn-Phong árnyalás
* Gamma korrekció
* MSAAx4 alapú élsimítás
* GPU generált mipmap-ek

Cél volt még hogy a rendszer rugalmas és könnyen használható legyen, illetve eredetileg egy minecraft szerű világotot szerettem volna elkezdeni implementálni benne, ám ez a terv túl ambíciózusnak bizonyult, és így ennek megvalósítása még várat magára.

A projekt vulkan-os részét legfőképpen a vulkan-tutorial.com oldalon található tutorial alapján készítettem el, és teljesen meg voltam elégedve vele, mert nagyon érthetően és követhetően magyaráztott mindent. Az OpengGL-es részek leginkább a learnopengl.com és a opengl-tutorial.org alapján készültek.

# Használt könyvtárak

Számos függősége van az elkészült projektnek:

* *glfw3*

Ablak létrehozáshoz és a felhasználói bemenetek és események kezelésére

* *glm*

Header-only matematikai könyvtár kifejezetten a számítógépes grafikához szükséges műveletek és struktúrák támogatására

* *vulkan-sdk*

Biztosítja a vulkan api-ban elérhető függvényeket a gpu vezérlésére

* *vulkan-hpp*

Header-only c++ nyelnyű csomogaló a c nyelvű vulkan apihoz, ami megkönnyíti a fejlesztők elétét és még csak futásidejű cpu költsége sincs

* *glad*

Az eredeti specifikáción alapuló OpenGL függvény betöltő generátor

* *fmt*

Modern string formázó könytár

* *tinyobjloader*

Obj modell fájlok hatákony és gyors betöltésésre képes könyvtár

* *rapidjson*

Egy json fájl betöltő és generáló

* *stb-image*

Képek betöltésére és írására használható könyvtár

# Fejlesztőkörnyezet

A projektet windows-on fejlesztettem windows-ra, de az összes függősége cross-platform és belül sem használok semmilyen platform specifikus elemet, így valószínűleg egyből vagy kevés munkával még linuxon is lefordulna.

A buildsystem cmake alapú, és legalább 3.17-es cmake kell, mivel használom a precompiled header funkcióját a cmake-nek.

A függőségek legkönnyebben a vcpkg program használatával telepíthetők fel, most éppen a ”2020.11-1”-es kiadást használom, ha minden függőség innen van telepítve, akkor ezzel biztosan fordul a programom.

Mivel a Visual Studio 2019 már natívan támogatja a CMakeList.txt betöltését, így elég csak megnyitni a projekt mappáját, és már lehet is fordítani.

Alapvetően C++20 standarddal fejlesztettem, de lehet elég neki a C++17 is hogy leforduljon, de a 16.8.3-mas verziójú Visual Studio-val ennek is biztosan működnie kell.

# Vulkan API

TODO

# Továbbfejlesztési lehetőségek

Eredetileg egy minecraft-szerű játékot szerettem volna elkezdeni fejleszteni, de a vulkan api mélyebb és időigényesebb volt mint ahogy gondoltam igy erre még nem jutott idő. Azonban a közeljövőben ha arra kerülne a sor, akkor implementálni lehetne kezdésnek egy 16x16-os chunk-on alapuló procedurálisan generált világot, úgy tudom hogy eredetileg is ezzel a chuck-os technikával készült a játék és a mai napig is ezt használják.

Akár egy skeletal alapó karakter animáció is hasznos és érdekes munka lehetne, ezáltal a világ kicsit életteltelibb lenne, mivel így nem csak statikus objektumok lennenek benne, hanem akár állatok is ugrándozhatnának a szabadban.

Árnyékok megjelenítése is szintén hasznos lenne.

# Referenciák

1. <https://vulkan-tutorial.com/>
2. <https://github.com/KhronosGroup/Vulkan-Hpp>
3. <https://github.com/tinyobjloader/tinyobjloader>
4. <https://developer.nvidia.com/vulkan-memory-management>
5. <https://learnopengl.com/>
6. <http://www.opengl-tutorial.org/>