**projekt** do předmětu **PGR – Počítačová grafika 2023**

**Zobrazení Země v OpenGL**

řešitel: **Ladislav Ondris**, xondri07

# Zadání

Cílem je vytvořit interaktivní vizualizaci Zeměkoule na styl Google Earth.

Obecné požadavky:

* **Interaktivita**: použití klávesnice a myši k interakci se scénou
  + přibližování a oddalování
  + pohyb kolem Země táhnutím
  + vypnutí či zapnutí jednotlivých prvků (noci, mřížky, simulace, apod.)
* **Simulace**: Součástí by měla být simulace dne a noci a to i v průběhu roku.
* **Textury**: Použití skutečných textur Země pro den i noc.
* **Terén**: Kromě textur by se měl i vyobrazit terén.
* **Osvětlení**: Umístění Slunce do odpovídají vzdálenosti osvlětující odpovídající část Země
* **Reálnost**: Cílem je reprezentovat Zemi dostatečně reálně
  + Země je ve skutečnosti Elipsoid
  + Země má osu rotace nakloněnou o 23.44˚
  + Země obíhá kolem své osy a zároveň kolem Slunce (rotace kolem své osy trvá méně než 24 hodin).

Implementačně zaměřené požadavky:

* **Rychlost**: aplikace by měla být responzivní a nemělo by se čekat na načtení zdrojů.
  + **Tiling**: rozdělení světa do dlážnic, což dokáže ušetřit zdroje (není nutné načítat jednu obří texturu)
  + **Culling**: použití různých typů cullingu
  + **Level of detail**: renderování zdrojů (textur a meshe) odpovídající vzdálenosti Země od kamery a dalších parametrů
  + **Out of order rendering**: načítání zdrojů mimo hlavní vlákno

# Nejdůležitější dosažené výsledky

Figure 1: Zobrazení Země v OpenGL včetně zobrazení terénu a přechodu dne a noci.

Nejdůležitější části výsledného řešení jsou následující:

### 1. Tiling & Culling

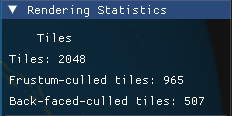
**Prvnotní naivní řešení**: Prvotním řešením byl jediný mesh reprezentující Zemi, na nějž se nanesla jedna velká textura. Textury jsou na disku uloženy komprimovaně. Při načítání se do paměti načte její celá velikost. V tomto případě textury zabírají i jednotky GiB.

**Rozdělení do dlážnic**: Typickým přístupem řešící výše zmíněný problém je rozdělení do dláždic (tiling). V tomto případě je celý mesh rozdělen do fixního počtu dláždic. Stejně tak textury musí být rozděleny do dláždic tak velkých, aby pokryly jednu či více dláždic. Při vyobrazování dláždic se odpovídající zdroje načtou z paměti.

[Foto dláždic]

**Asynchronní načítání zdrojů**: Načítání zdrojů je časově náročné. Načítání v hlavním vlákně by způsobilo neresponzivní aplikaci. Tudíž jsem načítání zdrojů implementoval asynchronně ve vedlejším vlákně. Komunikace mezi vlákny probíhá pomocí sdílených front, ke kterým přistupují s výlučným přístupem. Hlavní vlákno si vytváří požadavky na načtení zdrojů a načítácí vlákno je načte do paměti. Výsledkem je responzivní aplikace.

**Culling**: Hlavní vykreslovací cyklus iteruje po dláždicích. Před vyobrazením se kontrolují zda jsou načtené zdroje pro onu dláždici. Velké množství dláždic leží na odvráceném povrchu Země od kamery, nebo mimo frustum kamery. Načítání zdrojů a vyobrazování těchto dláždic by znamenalo plýtváním zdrojů. Je důležité, že se tento culling provádí ještě na CPU před samotným vykreslováním. V případě cullingu ve vykreslovacím řetězci OpenGL by se trojúhelníky zahodily až před samotnou rasterizací a zbytečně by se prováděl vertex shader a teselační shadery, a taktéž by se pro ně načítaly textury.

Figure 2: GUI ukazuje počet dláždic nevykreslených díky frustum a back-face cullingu.

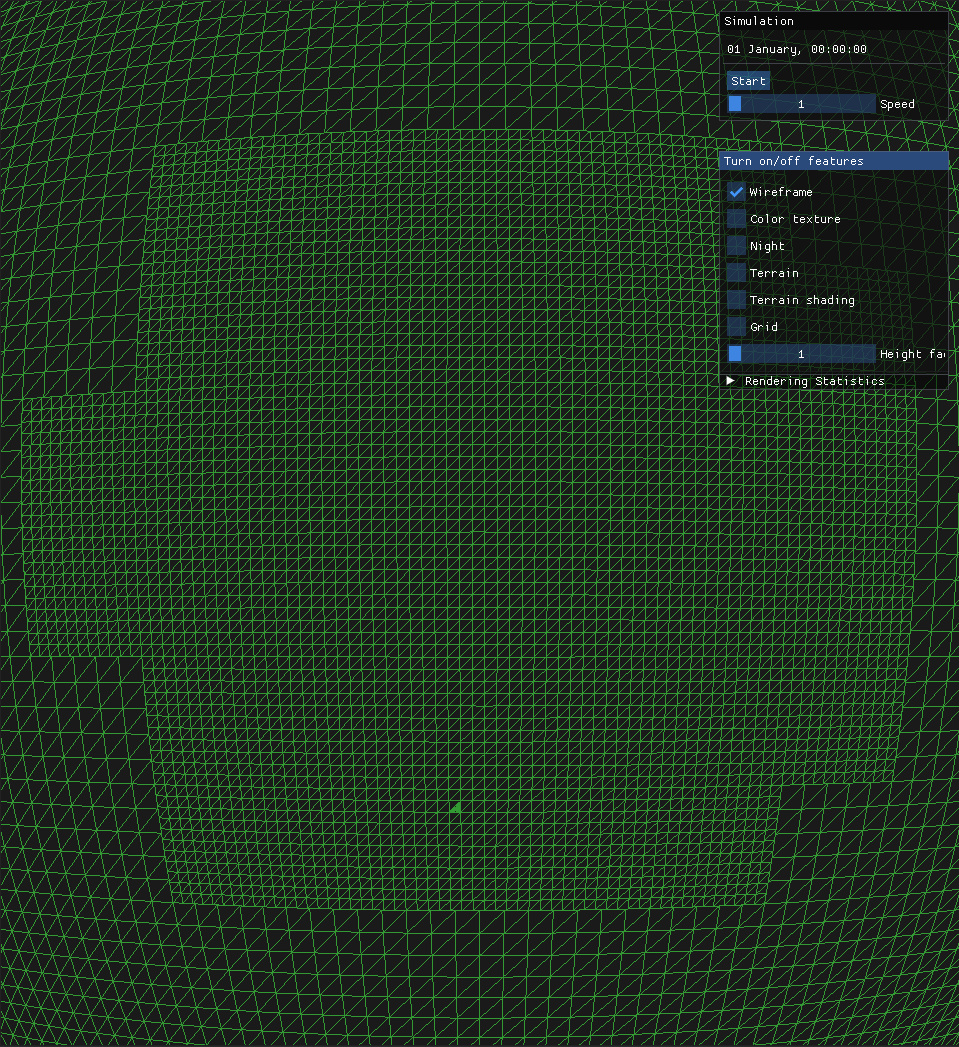
Na obrázku je vidět wireframe Země bez použití cullingu. Je tam možné vidět wireframe i odvrácených dláždic.

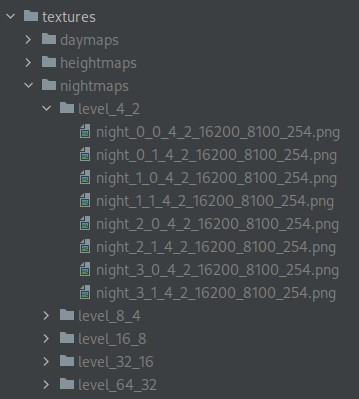
Na obrázku je vidět wireframe Země při použití cullingu. Wireframe odvrácených dláždic zde vidět není.

### 2. Level of detail (LOD)

Využívání zdrojů efektivně vyžaduje zobrazování pouze nutných detailů. Ve větší vzdálenosti od Země je možné zobrazit pouze hrubější mesh a textury. Za tímto účelem je předdefinováno několik úrovní detailu, a to jak polygonové sítě pro dláždici, tak textur.

**Určení úrovně**: Úroveň detailu se určuje na základě spočtené chyby na obrazovce pro danou dláždici. Chyba se počítá na základě vzdálenosti od kamery, granularity meshe, úhlu zorného pole a šířky obrazovky. Se zvětšující se vzdálenosti se chyba každé úrovně detailu zmenšuje, což dovoluje zobrazit hrubější úrovně a naopak.

Figure 3: Dláždice jsou zobrazeny v úrovni detailu odpovídající vzdálenosti od kamery a dalším parametrům. Dláždice ve středu obrazovky jsou blíže kameře, tudíž jsou zobrazeny s větším detailem než ty po stranách.

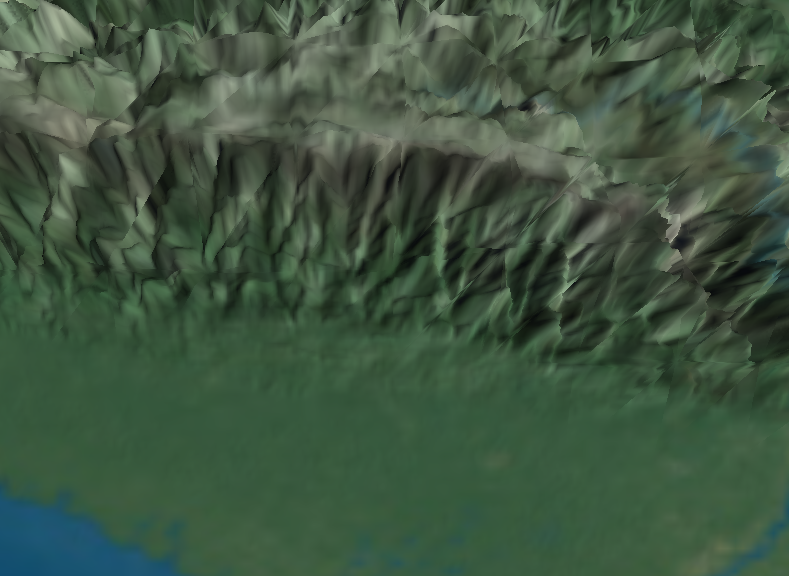
**Rozdělení textur na úrovně**: Textury jsou fyzicky uloženy na disku v různých úrovních detailu. Každá textura má stejnou velikost, avšak pokrývá menší či větší část Země dle úrovně detailu. V nejvyšší úrovni detailu pokrývá pouze malou část povrchu.

**Prodleva při náčítání textur**: V projektu jsou použity tři typy texture: denní, noční a výšková mapa. V případě, že nějaká z textur pro aktuální úroveň LOD ještě není načtena z disku, použije se textura z jiných LOD. Při přibližování se typicky použije texture z hrubější úrovně, a při oddalování se použije texture z jemnější úrovně.

### 3. Zobrazení terénu

Zobrazení terénu je provedeno za použití výškové mapy. Výšková mapa je použita ke třem různým účelům:

1. Polygonová síť je **nateselována** v teselačních shaderech. Teselační faktory jsou určeny dle velikosti gradientu (změny ve výšce) v daném trojúhelníku.
2. V teselačním evaluačním shaderu je použita technika ***displacement mapping***, kde výška každého vrcholu je pozměněna dle výšky ve výškové mapě.
3. Ve fragment shaderu je použita technika ***normal mapping***, která zkombinuje normály původního povrchu a vypočtených normál z výškové mapy. Úprava normál v závislosti na povrchu je důležité pro správné difuzní stínování.

Figure 4: Obrázek ukazuje vyobrazení terénu. Byly použity techniky teselace, displacement mappingu a stínování.

# Ovládání vytvořeného programu

**Pohyb**

* Pohyb po zeměkouli
  + Tažením (držení levého tlačítka myši)
  + Klávesy "W", "A", "S", "D"
* Přiblížení a oddálení
  + Kolečkem myši
  + Klávesy "+" a "-"

**Ovládání simulace**

* Menu aplikace nabízí spuštění a zastavení simulace tlačítkem "Start" a "Stop".
* Lze nastavit rychlost simulace. Rychlost znamená faktor, o jaký běží simulace rychleji než skutečný čas.

**Zapnutí a vypnutí prvků**

* V GUI aplikace lze zapnout či vypnout různé prvky zobrazení jako například terén, přechod dne a noci, mřížku, a podobně.

# Zvláštní použité znalosti

Věrohodná ale přebližná simulace oběhu Země kolem Slunce vyžaduje znalost náklonu Země, jak dlouho Zemi trvá rotace kolem své osy­, nebo který den v roce nastává rovnodennost. Tyto znalosti jsou potřebné k správnému osvětlení dle data a času.

# Použité technologie

Jazyky

* C++
* GLSL

Knihovny

* OpenGL
* GLFW3: Vytvoření okenní aplikace pro OpenGL
* OpenGL Mathematics (GLM): Matematické výpočty pro OpenGL
* ImGUI: Tvorba prvků GUI

# Použité zdroje

**Hotová data**

* Textury (denní, noční): https://www.shadedrelief.com/natural3/pages/textures.html
  + Sekce "Earth without clouds"
  + Sekce "Earth at night"
* Výšková mapa: https://visibleearth.nasa.gov/images/73934/topography
  + Všechny GeoTIFF soubory s rozlišením 10800x10800
  + Jejich popis je uveden zde: https://visibleearth.nasa.gov/grid
* Databáze měst
  + Staženo z https://simplemaps.com/data/world-cities

**Hotový kód**

* Výpočty nad elipsoidem
  + Třída Ellipsoid obsahuje několik metod, jejichž implementace je založena na ukázkách v knize [1]. Některé z těchto výpočtů jsou zduplikovány v shaderech.
    - Projekce bodů na povrch elipsoidu
    - Převod geodetických a geocentrických souřadnic
    - Zjištění normály ze souřadnic
* Některé části shaderů
  + Zjištění zda fragment se nachází na mřížce, funkce "isGrid" je implementací knihy [1]
  + Blending dne a noci je taktéž inspirování knihou [1]
* Parsování csv souboru (include/fast-cpp-csv-parser/csv.h)
  + Pochází z https://github.com/ben-strasser/fast-cpp-csv-parser
* Adaptovaný kód v metodě *setVertexDataForText*pro zobrazování textu
  + Pochází https://en.wikibooks.org/wiki/OpenGL\_Programming/Modern\_OpenGL\_Tutorial\_Text\_Rendering\_02

**Studijní materiály**

* [1] P. Cozzi, K. Ring, 3D Engine Design for Virtual Globes. A. K. Peters, Ltd., 2011.
* [2] Webová stránka https://learnopengl.com/
* Způsob extrakce rovin zobrazovacího komolého jehlanu (frustum) z *view-projection* matice je popsán v [3] a samotný algoritmus je pak inspirován popisem na webové stránce [4].
  + [3] https://www8.cs.umu.se/kurser/5DV051/HT12/lab/plane\_extraction.pdf
  + [4] https://cgvr.informatik.uni-bremen.de/teaching/cg\_literatur/lighthouse3d\_view\_frustum\_culling/index.html
* Direct State Access in OpenGL: https://github.com/fendevel/Guide-to-Modern-OpenGL-Functions/blob/master/README.md

# Co bylo nejpracnější

V průběhu řešení jsem se potýkal s velkou řadou problémů.

**Tiling**: Ze všeho nejobtížnější bylo dlážnicování (*tiling)*, které vyžadovalo množství programování ve všech částech aplikace:

* tvorba polygonové sítě,
* rozstříhání textur,
* vyobrazování jednotlivých dláždic v hlavní smyčce,
* načítání odpovídajících textur,
* upravení shaderů tak, aby zobrazovaly správné části textur.

**Kombinace rotací:** Poměrně dost času zabralo odladění simulace rotace Země kolem své osy a zároveň kolem Slunce. Potýkal jsem se s problémy pořadí rotací či nesprávných os. Ve výsledku se kombinují čtyři rotace (náklon, otočení kolem své osy, orbit, a zarování s počátečním časem simulace).

**Normály**: kombinace normál základního porvchu a normál spočtených z terénu bylo taktéž vyzývavé, a to především výpočet TBN matice pro převod z *tangent* prostoru.

# Zkušenosti získané řešením projektu

* Programování shaderů (vertex, tesselation control, tesselation evaluation, fragment), přičemž jsem se naučil přemýšlet nad tím, které operace je vhodné provést v kterých shaderech či případně na CPU.
* Mnoho konceptů z počítačové grafiky: tiling, teselace, frustum a back-face culling, texturování, osvětlování, normal mapping, displacement mapping, level-of-detail, zobrazování terénu pomocí výškové mapy,
* Práce s knihovnou ImGUI pro vytvoření uživatelského rozhraní.
* Zjistil jsem, že v počítačové grafice existuje mnoho způsobů, jak přistoupit k řešení. Každý přístup má své výhody a nevýhody a je třeba si vybrat konkrétní směr na základě několika parametrů. Často se jedná o nějaký kompromis.

# Autoevaluace

**Technický návrh: 90%** (analýza, dekompozice problému, volba vhodných prostředků, …)

K řešení problému jsem zvolil vhodný přístup, který dokáže řešit asynchronní načítání zdrojů. Textury by však mohly být řešeny jako *Sparse Textures* místo fyzického řezání textur.

**Programování: 90%** (kvalita a čitelnost kódu, spolehlivost běhu, obecnost řešení, znovupoužitelnost, …)

Řešení se snaží rozdělovat aplikační logiku od GUI. Kód je smysluplně rozdělen do tříd podle jejich zodpovědností. Některé části by si zasloužily refaktoring na úrovni metod.

**Vzhled vytvořeného řešení: 85%** (uvěřitelnost zobrazení, estetická kvalita, vhled GUI, …)

Řešení je vcelku pěkné. Textury na nejnižší úrovni by mohly být detailnější. Mezi dláždicemi se objevuje *T-junction* artefakt. Názvy měst jsou v noci nečitelné.

**Využití zdrojů: 100%** (využití existujícího kódu a dat, využití literatury, …)

Využil jsem k řešení knihu, která probírá problematiku vykreslování zeměkoulí, z čehož jsem se měl možnost se naučit mnoho.

**Hospodaření s časem: 100%** (rovnoměrné dotažení částí projektu, míra spěchu, chybějící části řešení, …)

Řešení jsem řešil dokončil výrazně s předstihem, což mi dovolilo implementovat věci dle mého názoru i nad rámec projektu.

**Celkový dojem: 100%** (pracnost, získané dovednosti, užitečnost, volba zadání, cokoliv, …)

Jedná se o perfektní zadání na kterém jsem si vyzkoušel mnoho konceptů z počítačové grafiky, jak bylo uvedeno v sekci výše. Osahal jsem si několik částí vykreslovacího řetězce včetně teselace.

Vytvořené řešení taktéž přináší jakési uspokojení z dosaženého výsledku, neboť je možné se scénou interagovat a dívat se na objekt skutečného světa.

Z mého pohledu se jedná o velice zdařený projekt, kterým mohu prezentovat svoje nabyté znalosti v počítačové grafice.

# Doporučení pro budoucí zadávání projektů

Jsem rád, že jsem si mohl zvolit vlastní zadání, díky čemuž jsem projekt vypracovával s chutí. Také je dobře, že každý má jiné zadání a děláme tak každý něco originálního.

Je skvělé, že nejsou žádné průběžné deadliny k projektu v průběhu semestru. Stejně tak jako mít na něj čas až do konce kalendářního roku, což dovolí vypracovat projekt kvalitně.