VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

3D DEMO PRO HOLOGRAFICKÝ DISPLEJ

3D DEMO FOR HOLOGRAPHIC DISPLAY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ KULDA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ CHLUBNA**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Kulda Lukáš**
Program: Informační technologie
Název: **3D demo pro holografický displej**
3D Demo for Holographic Display
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s vhodným API pro vývoj 3D aplikace (OpenGL, Unity, Unreal)
2. Navrhněte vhodnou scénu v omezeném prostoru využívající efektivně zaostřovací roviny holografického displeje
3. Implementujte aplikaci
4. Zhodnoťte zobrazení aplikace na displeji a proměřte efektivitu
5. Publikujte demo v oficiální knihovně
6. Vytvořte video reprezentující výsledky vaší práce

Literatura:

- Dle zadání vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, experimenty vedoucí k bodu 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 15. listopadu 2019

Abstrakt

Výsledkem této práce je 3D demo pro holografický displej vizualizující družicová data na povrchu zeměkoule. Pro získání vizualizovaných dat bylo využito Earth Engine API od společnosti Google. Scéna 3D modelu a uživatelské rozhraní byly vytvořeny pomocí vývojového prostředí Unity, a zobrazeny na holografickém displeji Looking Glass, který scénu uvádí do prostoru.

Abstract

The result of this thesis is a 3D demo for holographic display visualizing satellite data on Earth surface. The Earth Engine API from Google was used to obtain visualized data. The 3D model scene and user interface were created using the Unity development environment and displayed on the Looking Glass holographic display to bring the scene into space.

Klíčová slova

vizualizace družicových dat, C#, Python, Earth Engine, družice, Unity, holografický displej, Looking Glass, souřadné systémy, projekce map

Keywords

satellite data visualization, C#, Python, Earth Engine, satellite, Unity, holographic display, Looking Glass, coordinate systems, map projections

Citace

KULDA, Lukáš. *3D demo pro holografický displej.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Chlubna

3D demo pro holografický displej

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Chlubny. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Lukáš Kulda
6. května 2020

Poděkování

Rád bych poděkoval vedoucímu této bakalářské práce, panu Ing. Tomáši Chlubnovi, za odbornou pomoc a cenné rady, které mi věnoval během této práce. Dále bych chtěl poděkovat i Ing. Tomáši Miletovi za konzultace v ranném počátku této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Družice a především informace, které získávají a poskytují, se staly nedílnou součástí dnešní doby. Pomocí těchto družic lze získávat velké množství informací o Zemi, které se využívají napříč mnohými odvětvími. Tyto informace lze například využít v meteorologii pro určování denních teplot, srážek, bouřek a jiných aspektů ovlivňující počasí. V geologii mohou být využity pro přesné mapování terénu, jakkoliv složitého a těžce dostupného. Využívají se i v globálních polohovacích systémech (GPS), které dokáží určit přesnou polohu na Zemi. Využití je opravdu obrovské, nicméně proces získávání těchto informací, a jejich interpretace není vůbec jednoduchý. A proto jsou vytvářeny nástroje, které data zobrazují do takové podoby, aby i pouhý laik z nich byl schopen číst.

Dochází i k vývoji zobrazovacích zařízení, v dnešní době je standardním zobrazovacím zařízením 2D displej. Nicméně, nový holografický displej Looking Glass zavádí nový způsob zobrazování 3D modelů. Tento displej vytváří dojem, že se člověk opravdu pohybuje okolo zobrazovaného 3D modelu.

Cílem této práce je navrhnout a implementovat aplikaci, která bude vizualizovat datové sady družic na povrchu 3D modelu zeměkoule, a zobrazí je na displeji Looking Glass. Jednou z předností této aplikace by měla být možnost, si nahrát různé datové sady pro jejich vizualizaci.

Kapitola 2 seznámí čtenáře s principy, které jsou potřebné k vytvoření výše zmíněného nástroje. Sekce 2.1 se zabývá problémy spojenými s reprezentací povrchu Země. Jsou zde popsány modely pomocí, kterých je povrch reprezentován. Dále se zabývá souřadnými systémy Země a jejich transformacemi. Nakonec jsou v této sekci popsány projekce map Země. Následuje sekce 2.2, která se zabývá družicemi a principy získávání a interpretace družicových dat, včetně jejich dostupnosti. Sekce 2.3 popisuje princip displeje Looking Glass. Na konci této kapitoly je sekce 2.4 popisující herní engine Unity a balíček pro zobrazení scény na displeji Looking Glass. V kapitole 3 je představen návrh řešení. Sekce 3.1 popisuje tvorbu modelu zeměkoule. Hned poté v sekci 3.2 je popsán způsob vizualizace dat na povrchu modelu zeměkoule. Poslední sekce 3.3 této kapitoly se zabývá návrhem scény pro holografický displej Looking Glass. Kapitola 4 popisuje konkrétní implementaci. Exportem dat se zabývá sekce 4.2. Vytvoření modelu Země popisuje sekce 4.3. Následuje sekce 4.4 popisující vizualizaci dat. Poté je v sekci 4.5 popsáno zobrazení scény na displeji Looking Glass. Konec této kapitoly je věnován optimalizacím. V kapitole 5 jsou popsána provedená měření aplikace. A v poslední kapitole jsou shrnuty dosažené výsledky této práce.

Kapitola 2

Teorie

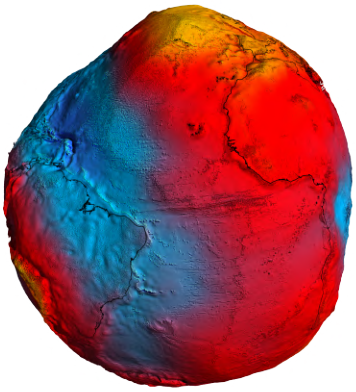
V této kapitole jsou popsány způsoby, metody a principy zabývajícími se Zemí. Ty jsou rozebírány v sekci 2.1, kde je konkrétně rozebírán povrch zeměkoule a jaké modely se používají pro jeho reprezentaci. Dále je součastí této kapitoly rozbor problematiky souřadných systému Země, včetně jejich transformací. Poté jsou popsány různé projekce map v souvislosti s povrchem Země. Sekce 2.2 se zabývá družicemi planety Země, kde je popsána především problematika získávání dat, jejich zpracování a dostupnost. Součástí této kapitoly je i představení aplikačního rozhraní Google Earth Engine. Následující sekce 2.3 se pak zabývá holografickým displejem Looking Glass, pro něhož je implementována výsledná aplikace v rámci této práce. A nakonec poslední sekce 2.4 pojednává o herním enginu Unity, který byl použit pro vývoj výsledné aplikace.

2.1 Matematický popis planety Země

Pro vytvoření 3D modelu Země je nejdříve zapotřebí se seznámit, jak lze takovou planetu matematicky popsat a aproximovat. Právě pomocí těchto popisů a technik lze následně vytvořit objekty, jenž budou, pokud možno, co nejpřesněji reprezentovat reálnou podobu Země. Tento popis je klíčový pro jakýkoliv nástroj pracující s určením polohy na Zemi. Pokud by byl model popsán chybně, nástroj se stává nevěrohodný a nepoužitelný.

2.1.1 Geoid

Země je téměř kulatá planeta jejíž povrch je pokryt nepravidelnostmi jako jsou nížiny, moře, oceány, hory a údolí [14]. Planeta Země je tedy velice členitá a nepravidelná. Kvůli dlouhodobým rotačním účinkům je poloměr Země větší u rovníku než u pólů, a proto je zdeformovaná. Pro tento přesný tvar Země, znázorněný na obrázku 2.1, se v geodezii obvykle využívá termín „geoid“. Geoid je tedy povrch, který definuje nulovou nadmořskou výšku – to znamená – že geoid je povrch ideálního globálního oceánu bez přílivu a odlivu, formovaný pouze gravitační silou. Jedná se spíše o fyzikální model využívaný především k preciznímu měření nadmořské výšky [15].

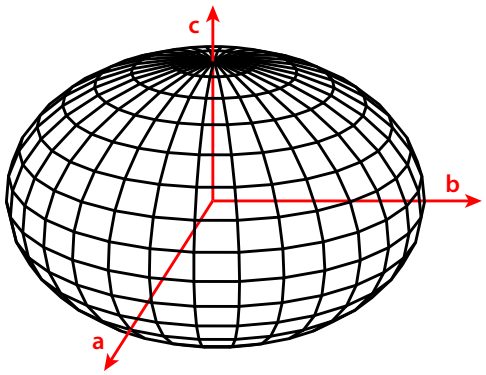


Obrázek 2.1: Nejaktuálnější geoid

Byl zaznamenaný roku 2018 družicí GOCE znázorňující povrch zeměkoule. Barvy na obrázku představují odchylky výšky (–100m až +100m) od ideálního geoidu. Modré barvy představují nízké hodnoty, zatímco červené a žluté představují vysoké hodnoty, převzato a upraveno[1].

2.1.2 Referenční elipsoid

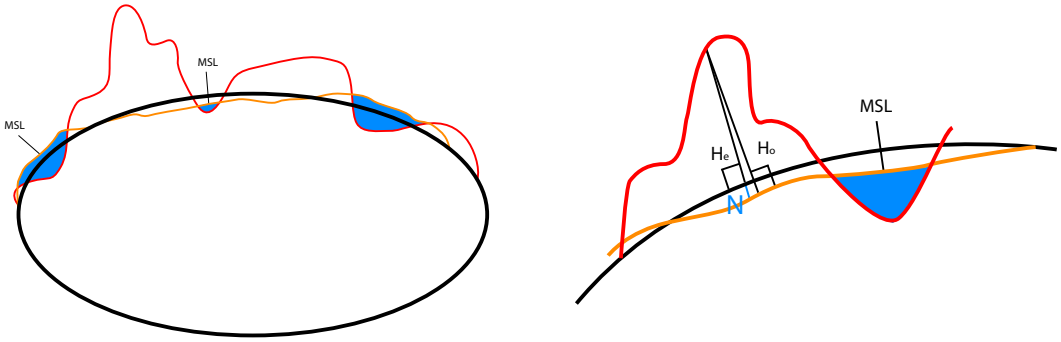
Elipsoid je uzavřený povrch, jehož všechny rovinné průřezy jsou buď elipsy, nebo kruhy. Je symetrický kolem tří vzájemně kolmých os, které se protínají ve středu. Elipsoid je možné vidět na obrázku 2.2. Referenční elipsoid se poté nazývá elipsoid, který se používá k reprezentaci Země při geodetických výpočtech. Poskytuje jednodušší výpočty, než jiné, složitější matematické modely [17]. Jeho vztah vzhledem k reálnému povrchu Země a geoidu lze vidět na obrázku 2.3. Referenční povrch lze použít například pro projekce map nebo satelitní navigace. Slouží jako základ pro souřadné systémy závisející na zeměpisné šířce, délce a elipsoidální výšce – známé spíše jako „nadmořská výška“.



Obrázek 2.2: Elipsoid

Na obrázku jsou a, b a c hlavní poloosy. Pokud $a = b = c$, tak povrch je definován jako koule.

[1] http://www.esa.int/ESA_Multimedia/Images/2011/03/New_GOCE_geoid2



(a) Rozdíl mezi geoidem a referenčním elipsoidem (b) Rozdíly ve výškách modelů

Obrázek 2.3: Porovnání geoidu a referenčního elipsoidu

Na obrázku (a) je znázorněn rozdíl mezi geoidem, referenčním elipsoidem a terénem. Obrázek (b) pak zobrazuje rozdíly dvou zmíněných modelů, přičemž H_e je výška elipsoidu, H_o výška geoidu a N je jejich rozdíl, MSL představuje průměrnou hladinu moře. Černě je zbarvený elipsoid, oranžově geoid a červeně „reálný“ povrch Země, převzato a upraveno[2].

2.1.3 Souřadné systémy

Souřadný systém umožňuje jednoznačně popsat polohu bodu pomocí hodnot souřadnic. Souřadných systémů existuje mnoho a každý je vhodný pouze pro popis určitého problému. K vytvoření planety Země v trojrozměrném prostoru mohou posloužit následující souřadné systémy, pomocí, kterých lze popsat povrch zeměkoule.

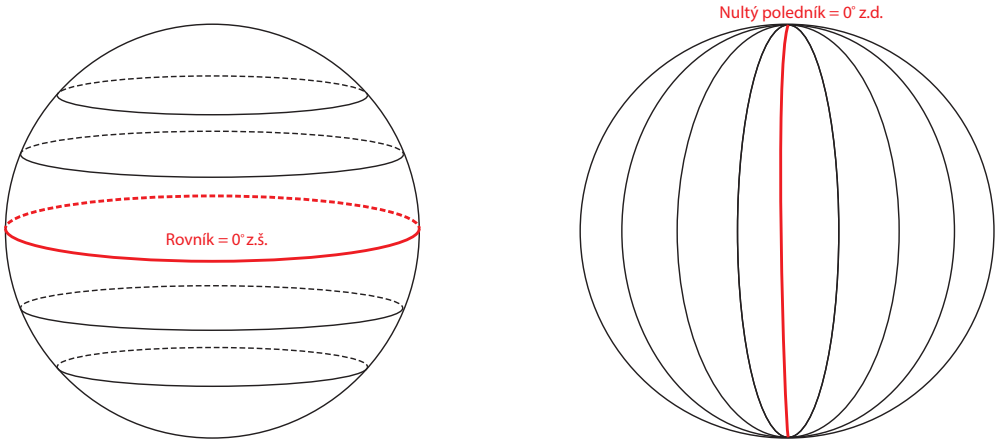
Geografický souřadný systém

Geografický souřadný systém definuje polohu na Zemi pomocí sferoidu (elipsoidu). Bod na povrchu se určuje pomocí zeměpisné šířky a zeměpisné délky - jedná se o úhly meřené od středu Země. Horizontální linie neboli rovnoběžky jsou řádky zeměpisné šířky a vertikální linie jsou linie zeměpisné délky.

Rovnoběžka ve středu mezi póly Země se nazývá rovník, a definuje nulovou zeměpisnou šířku, viz obrázek 2.4a. Svislá osa, která definuje nulovou zeměpisnou délku je nazývána jako nultý poledník. Počátek souřadného systému je definován na místě, kde se protíná rovník s nultým poledníkem, viz obrázek 2.4b. Zeměpisná šířka se měří relativně k rovníku a nabývá hodnot od $-90°$ (jižní pól) do $90°$ (severní pól). Zeměpisná délka se měří relativně k nultému poledníku a nabývá hodnot od $-180°$ (směrem na západ) do $180°$ (směrem na východ).

I přestože lze díky zeměpisné šířce a délce určit přesnou polohu na Zemi, tak neposkytují uniformní jednotnou měrnou jednotku. Toho si je možné povšimnout na obrázku 2.4, kde kruhy definující rovnoběžky se postupně zmenšují, až dojdou k pólům, kde se z nich stane pouhý bod, ve kterém se sbíhají poledníky, jenž mají vždy stejnou velikost. Což znamená, že jeden stupeň zeměpisné délky u rovníku má přibližně $111.321km$, zatímco při $60°$ zeměpisné šířky má pouze $55.802km$ [12].

[2] https://support.pix4d.com/hc/en-us/articles/202559869-Orthometric-vs-ellipsoidal-height



(a) Zeměpisná šířka (b) Zeměpisná délka

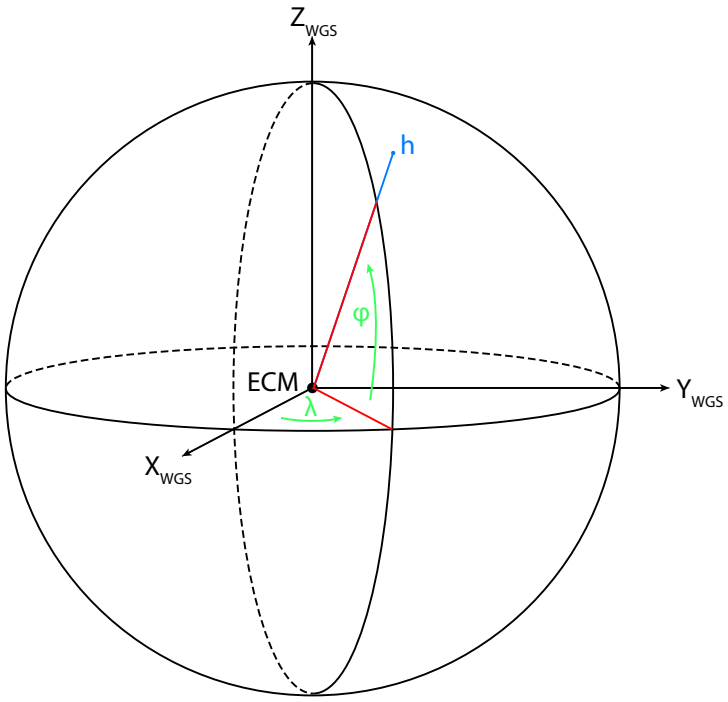
Obrázek 2.4: Zeměpisná šířka a délka

Geodetický souřadný systém

Geodetický souřadný systém (anglicky „World Geodetic System“) WGS je standard užívaný v kartografii, geodezii a satelitové navigaci. Standard WGS byl již několikrát revidovaný, aktuálním standardem je WGS 84 (také známý jako EPSG:4326[3]), jenž byl naposledy aktualizován roku 2004 a uvádí se jako referenční souřadný systém pro GPS [11]. Tento souřadný systém je definován referenčím elipsoidem popsaném v podsekci 2.1.2, a je zobrazen na obrázku 2.5.

Počátek souřadného systému leží v těžišti Země. Osa X je kladná směrem k průsečíku rovníku s nultým poledníkem, kladná osa Z směřuje k severnímu pólu, záporná k jižnímu pólu a osa Y je kolmá těmto osám, a vytváří pravostranný souřadný systém.

[3] http://www.epsg.org/



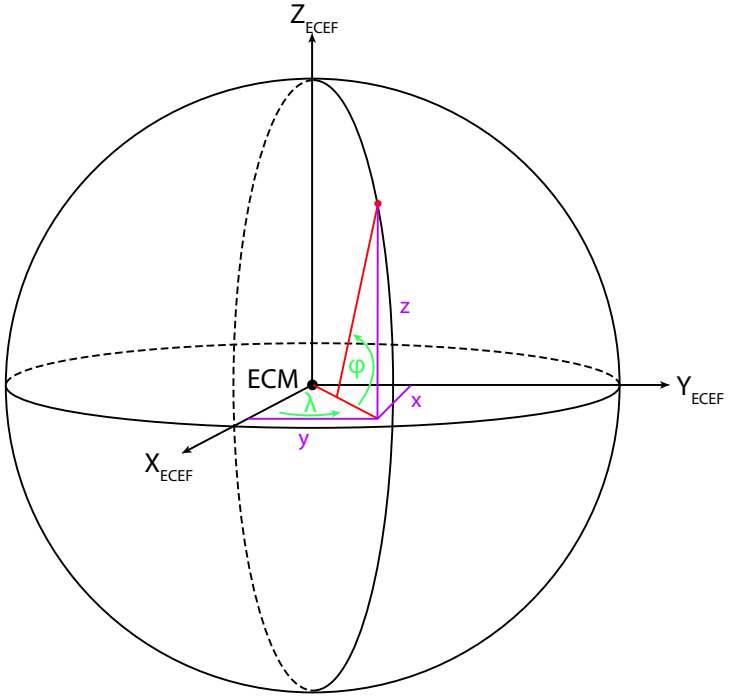
Obrázek 2.5: Souřadný systém dle standardu WGS 84

Poloha bodu na povrchu se určuje pomocí zeměpisné šířky ϕ, délky λ a výšky h. Zeměpisná délka nabývá úhel v rozmezí $-180°$ až $180°$, zeměpisná šířka pak úhel v rozmezí $-90°$ až $90°$ a zeměpisná výška se uvádí jako vzdálenost bodu od povrchu elipsoidu.

Na obrázku 2.5 je modelován trojrozměrný souřadný systém. Nicméně, souřadnice lze vyjádřit jiným způsbem, bez potřeby transformace z jednoho systému na jiný, zatímco je zachován vztažný bod systému. [10].

ECEF

Kartézský souřadný systém ECEF (zkratka pro „Earth-Centered Earth-Fixed“) je používaný v mnoha satelitních systémech pro určení polohy na Zemi [3]. Stejně jako u souřadného systému WGS, jeho počátek leží v těžišti Země, a jeho osy jsou totožné, přičemž osa Z se shoduje s osou rotace Země, viz obrázek 2.6.



Obrázek 2.6: Souřadný systém ECEF
Poloha bodu je určena pomocí tří souřadnic: X, Y a Z.

2.1.4 Transformace souřadných systémů

Mezi jednotlivými souřadnými systémy lze provádět vzájemné transformace. Je tedy možné vyjádřit konkrétní bod několika různými souřadnými systémy. V této kapitole jsou popsány nezbytné transformace pro reprezentaci 3D modelu zeměkoule.

Transformace z geodetického souřadného systému do ECEF

Souřadnice ECEF (X, Y, Z) libovolného bodu P, reprezentovaného geodetickými souřadnicemi z podsekce 2.1.3 (ϕ, λ, h), mohou být určeny pomocí transformačních rovnic 2.1 [20]:

$$\begin{aligned} x &= (R + h)\cos\phi\cos\lambda \\ y &= (R + h)\cos\phi\sin\lambda \\ z &= (R + h - e^2 R)\sin\phi \\ R &= \frac{a}{\sqrt{1 - e^2\sin^2\phi}} \end{aligned} \tag{2.1}$$

kde:

a	:	rovníkový poloměr
ϕ	:	zeměpisná šířka
λ	:	zeměpisná délka
h	:	nadmořská výška

2.1.5 Projekce mapy

Povrch Země lze reprezentovat mnoha způsoby v různých měřítkách, ať už v prostoru či rovině. Mapová projekce je způsob, jakým se převádí trojrozměrný povrch zeměkoule do roviny a vytváří tak mřížku, která zobrazuje mapu povrchu.

Zde ale nastává problém, jak rozbalit povrch Země do roviny a zploštit jej, aniž by se zdeformoval. Reprezentace povrchu v rovině se bez deformací neobejde. Projekcí se napříč historií vytvořilo již nespočet, a vzájemně se liší, přičemž každá způsobuje různé typy zkreslení (tvaru, plochy, vzdálenosti nebo směru) [12].

Existují však tři základní metody projekcí, přitom každá z nich zkresluje a zachovává různé aspekty na mapě:

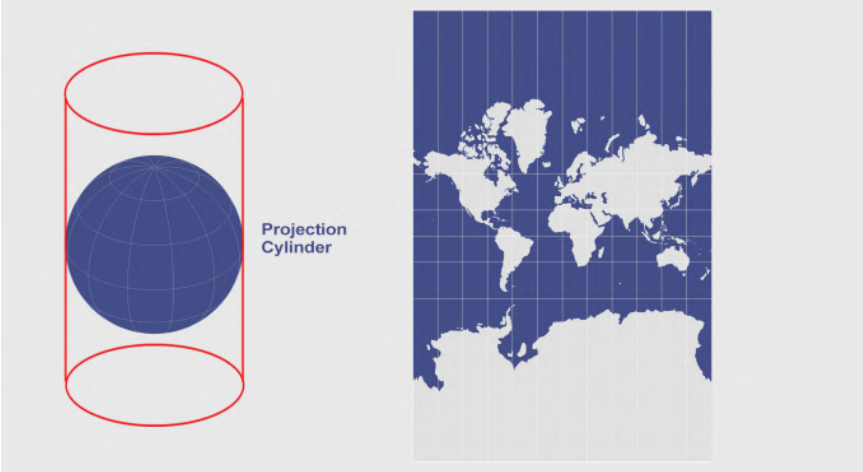
- Válcová projekce – zachovává tvar a směr, zkresuluje velikost oblastí a vzdáleností
- Kónická projekce – zachovává velikost oblastí a vzdáleností, zkresluje tvar a směr
- Azimutální projekce – zachovává velikost oblastí a vzdáleností podél bodu kontaktu roviny se zeměkoulí

Vzhledem k velkému množství projekcí, bude v této práci zmíněno pouze Mercatorovo a přímočaré zobrazení, které je relevatní k řešení, kterým se tato práce zabývá.

Mercatorova projekce

Tato projekce mapy byla představena Gerardem Mercatorem roku 1569. Původně sloužila pro navigaci po moři, protože jakákoli přímka na projektované mapě je i přímkou skutečného konstantního směru, jenž umožňuje zakreslit přímočarý kurz [4] – zobrazuje přesné kompasové směry [12].

Využívá výše zmíněnou válcovou metodu projekce, a vzhledem k tomu, že zachovává tvar a směr, tak se hodí právě pro navigaci. Nicméně vzdálenosti i velikost oblastí jsou zkresleny. Směrem k pólům se velikosti oblastí více zkreslují, například Grónsko se jeví na mapě větší než Jižní Amerika, i přestože je osmkrát menší [12]. Způsob, jakým je mapa projektována a jaké jsou její aspekty, lze vidět na obrázku 2.7.

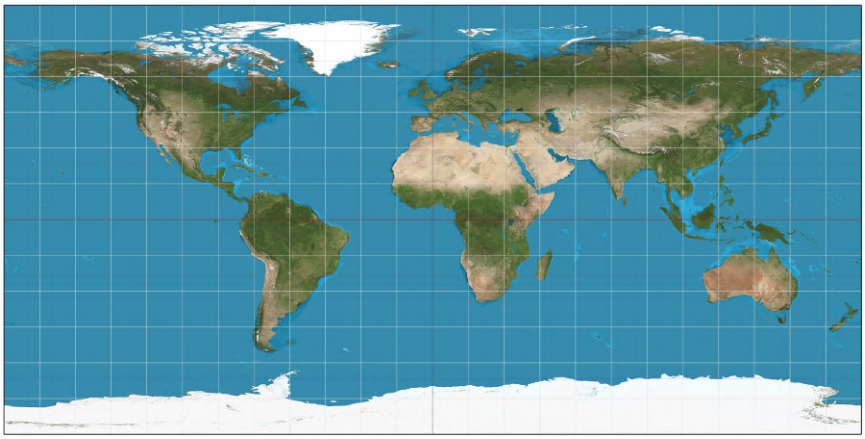


Obrázek 2.7: Mercatorova válcová projekce

Na obrázku lze vidět metodu projekce, včetně jejích vlastností, tvary a směry jsou zachovány, kdežto velikost oblastí a vzdálenosti nejsou, převzato[4].

Přímočará projekce

Stejně jako Mercatorova projekce využívá válcovou metodu projekce. Také je jednodušší, jelikož její mřížka je tvořena ze stejně velkých obdélníků. Tvary a velikosti oblastí jsou zkreslenější, čím dále jsou od standardních rovnoběžek. Vektory směřující na sever, jih, západ a východ jsou přesné, nicméně obecně směry zachovány nejsou. Vzdálenosti jsou pak zachovány podél poledníků a standardních rovnoběžek. Spíše se využívá pro mapy v globálním měřítku nebo s větším meřítkem, při kterých se snižuje zkreslení projekce [12]. Projekci lze vidět na obrázku 2.8.



Obrázek 2.8: Přímočará projekce

Na obrázku lze vidět metodu projekce, včetně jejích vlastností, tvary a směry jsou zachovány, kdežto velikost oblastí a vzdálenosti nejsou, převzato[5].

[4] https://gisgeography.com/cylindrical-projection/
[5] https://en.wikipedia.org/wiki/Equirectangular_projection

2.2 Družice planety Země

Umělá družice (anglicky „artificial satellite“) je objekt, který je vypuštěn do vesmíru na oběžnou dráhu planety Země. Umělé se jim říká z toho důvodu, aby se rozlišily od přírodních družic, jako je například Měsíc [19]. První umělá družice (dále jen „družice“) se nazývala Sputnik 1[6] a byla vypuštěna na oběžnou dráhu již roku 1957.

K dnešnímu dni se na oběžné dráze nachází již stovky družic, přičemž každá z nich je vybavena různými speciálními přístroji a má určitý úkol v závislosti na jejím typu. Může se například jednat o astronomické, pozorovací, komunikační, navigační, výzkumné a jiné [19]. Tato práce se zabývá pouze pozorovacími družicemi, jejichž misí je pozorovat planetu Zemi. Jedná se o tzv. „Dálkový průzkum Země“, kterým se zabývá následující kapitola.

2.2.1 Dálkový průzkum Země

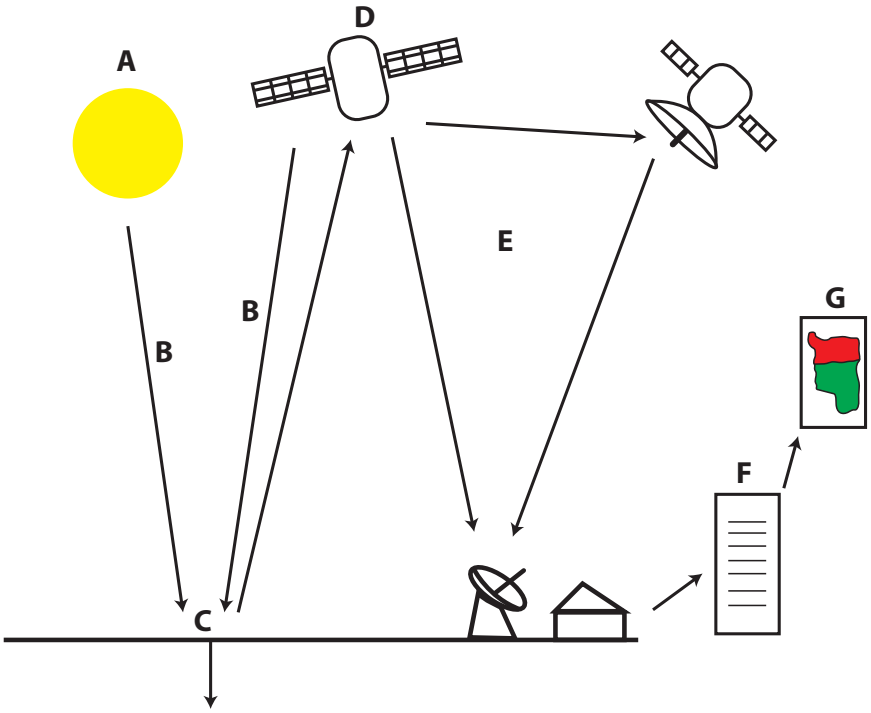
Dálkový průzkum Země (dále jen „DPZ“) je metoda získávání informací o objektech na zemském povrchu bez přímého kontaktu s ním. DPZ zahrnuje kompletní proces získávání informací od pořízení dat, jejich zpracování, analýzu, až po výslednou vizualizaci a interpretaci obrazu.

Jedná se o moderní geoinformační technologii, která se v současnosti dostává do povědomí stále širšího okruhu odborné i laické veřejnosti. K tomu přispívá zejména fakt, že tato technologie má oproti klasickým pozemním měřením řadu výhod. Především je možné pořizovat snímky z těžko dostupných oblastí, kde by to jinak s použitím pozemní techniky bylo příliš složité nebo nemožné. Druhým důvodem je pak neustálý rozvoj družicových technologií a výpočetní techniky, a tím i zjednodušení dříve náročných postupů [2].

2.2.2 Principy získávání dat dálkovým průzkumem Země

Základní princip průzkumu spočívá v měření množství elektromagnetického záření vyzařovaného nebo odráženého zemským povrchem. Každý objekt na Zemi, který má teplotu větší, než je absolutní nula ($-273.15°$), je zdrojem tohoto záření. V DPZ se používají záření vydávané třemi konkrétními zdroji, jsou jimi: záření vydávané samotným povrchem Země, sluneční záření povrchem odrážené, případně záření vydávané umělým zdrojem, které následně zemský povrch opět odráží. Každý objekt na Zemi, ale i samotná Země mají různé fyzikální vlastnosti, přičemž dochází k vzájemné interakci dopadajícího a odráženého záření. Právě na základě odráženého záření jsme schopni klasifikovat objekty na zemském povrchu [2]. Proces DPZ znázorňuje obrázek 2.9.

[6] https://www.nasa.gov/multimedia/imagegallery/image_feature_924.html



Obrázek 2.9: Princip DPZ

A Zdroj elektromagnetického záření, které bude odráženo objektem sledování.

B Energie záření přichází do kontaktu s atmosférou nejprve před dopadem, až poté po odrazu.

C Interakce objektu s energií dopadaného záření v závislosti na fyzikálních vlastnostech objektu.

D Zaznamenání elektromagentického záření senzorem (radiometrem), kterým je družice vybavena.

E Přenos mezi družicí a stanicemi, kde následně dochází ke zpracování dat.

F Vyhodnocení zpracovaného obrazu – zde jsou uchovány informace o sledovaném objektu – datové sady.

G Využití informací z datových sad, může se jednat například o jejich další zpracování, aplikace algoritmů nebo vizualizaci – posledním zmíněným využitím se zabývá tato aplikace.

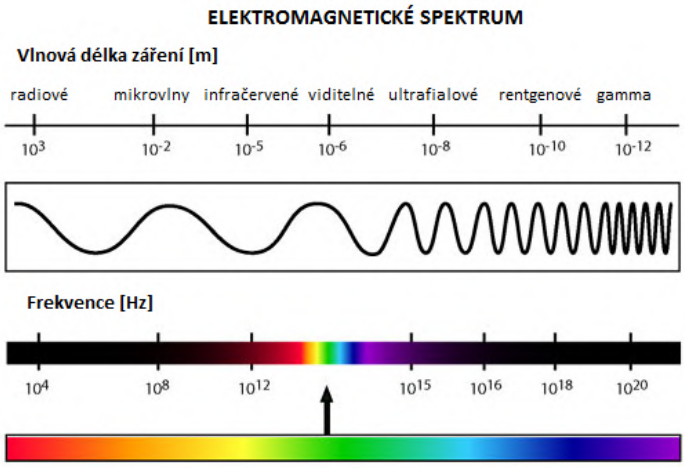
Elektromagnetické záření

Jak bylo zmíněno v předchozí kapitole, pomocí elektromagnetického záření jsme schopni získávat informace o zemském povrchu. Velkou předností DPZ je možnost záznamu vlnové

délky pro člověka neviditelné [2]. Elektromagnetické záření obsahuje dva důležité parametry, které jsou nezbytné pro pochopení DPZ [1]:

- Vlnová délka (λ) – meřená v jednotkách metrů, případně v nanometrech (nm), mikrometrech (μm) nebo centimetrech (cm).
- Frekvence (v) – meřená v hertzech (Hz).
- Jejich vztah je dán vzorcem: $c = \lambda v$, kde c je rychlost světla.

Čím kratší vlnová délka, tím vyšší frekvence a čím delší vlnová délka, tím nižší frekvence. Elektromagnetické spektrum pak zahrnuje elektromagnetická záření všech možných vlnových délek, přičemž lidské oko je schopné vidět pouze rozsah od cca $380nm$ - $720nm$, viz obrázek 2.10. V rámci DPZ se využívají délky v rozsahu cca $300nm$ - $1m$ [2].



Obrázek 2.10: Elektromagnetické spektrum
Rozsahy jednotlivých úseků elektromagnetického spektra, převzato [2].

Na obrázku 2.10 si lze všimnout, že spektrum se dělí na několik oblastí. Pro většinu účelů se využívá ultrafialové (dále jen „UV“) záření, pod kterým jsou zřetelně vidět specifické materiály na Zemi, jako jsou horniny a minerály. Dalším je infračervené (dále jen „IR“) záření, jenž se využívá například pro zjišťování teplot oceánů, rozlišení vegetace a mnoho dalších. Dále lze využít mikrovlnné záření například pro studium geologie, ledovců nebo lesnictví [1].

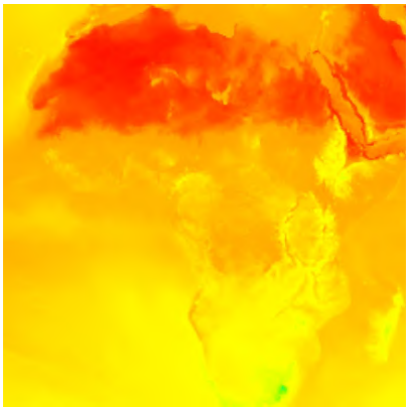
2.2.3 Parametry družicových dat

Jak již bylo řečeno v sekci 2.2, každá družice je jinak vybavená a její přístroje mají určité parametry. Při výběru z datových sad pro danou aplikaci je důležité dbát na tyto parametry, jelikož se mohou značně lišit, tudíž by nemuselo být vhodné použít je pro stejný účel.

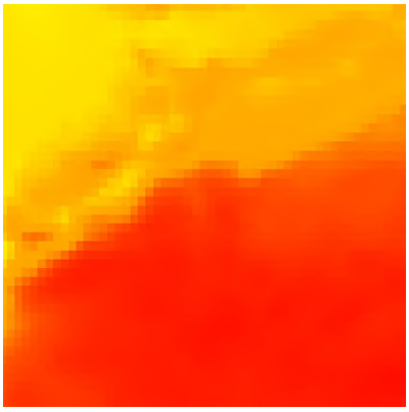
Prostorové rozlišení

Prostorové rozlišení definuje, jak velká plocha na zemském povrchu odpovídá jednomu pixelu na snímku. S větším rozlišením jsou snímky detailnější a lze rozeznat více podrobností. Rozlišují se družice, které zaznamenávají snímky s rozlišením 1 kilometr a více na pixel.

V tomto případě se často jedná o meteorologické záznamy, příklad lze vidět na obrázku 2.11. Dále se rozlišují družice se středním rozlišením 100 - 250 metrů na pixel a družice s vysokým rozlišením 10 - 50 metrů na pixel. Ty jsou využívány například v mapování v regionálním měřítku, sledování oceánů nebo zdravotního stavu vegetace. Družice s velmi vysokým rozlišením 5 metrů i méně, se pak používají například pro podrobné lokální mapování [2], viz obrázek 2.11. Někdy se rozlišení uvádí i v obloukových stupních[7].



(a) Nízké rozlišení – zachycení celého kontinentu



(b) Nízké rozlišení – detail kontinentu

(c) Vysoké rozlišení – zachycení části města

(d) Vysoké rozlišení – detail budov

Obrázek 2.11: Prostorové rozlišení snímků.

Na obrázku (a) s nízkým rozlišením (0.25 obloukových stupňů) jsou zachycena teplotní data nad Africkým kontinentem. Obrázek (b) zachycuje detail obrázku (a). Na obrázku (c) s vysokým rozlišením ($1m$) je zachycena část městské čtvrti a obrázek (d) zobrazuje opět detail předchozího obrázku.

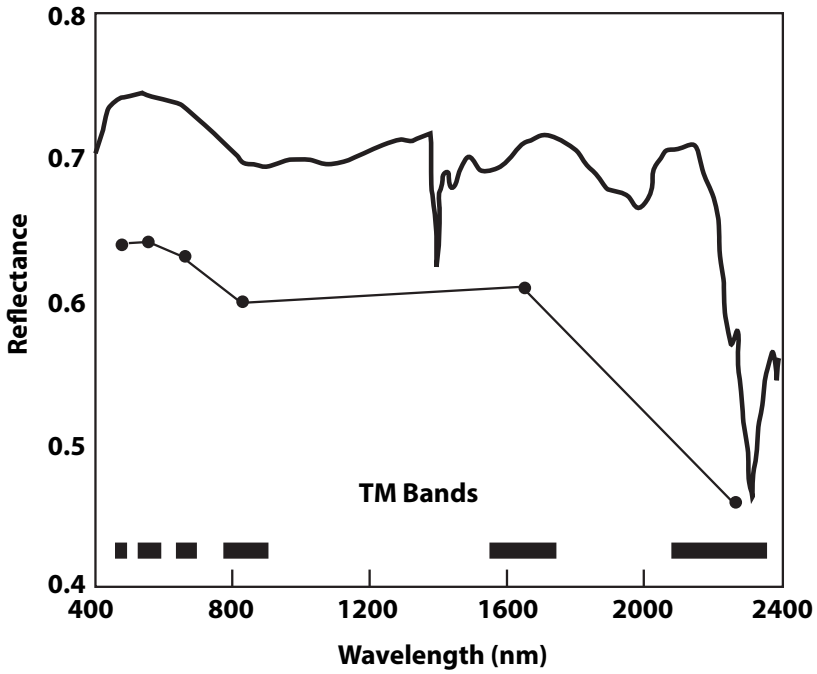
Časové rozlišení

Časové rozlišení udává interval v jakém jsou data periodicky pořizována na stejném území. Družice s nízkým prostorovým rozlišením mohou zaznamenávat až několikrát za den, zatímco družice se středním a vyšším rozlišením pořizují záznamy méně často [2]. Časové rozlišení tak může záviset právě na prostorovém rozlišení, čím větší je, tím menší pravděpodobně bude časové rozlišení družice. Časté pořizování příliš detailních snímků by mohlo být velice nákladné a paměťově náročné.

[7] https://www.opendem.info/arc2meters.html

Spektrální rozlišení

Spektrální rozlišení udává jakou šířku elektromagnetického spektra je radiometr schopen snímat, ale i počet pásem, na které je spektrum rozděleno. V tomto ohledu je možné rozlišovat multispektrální data, která obsahují několik jednotek pásem o šířce v řádu desítek nm, a hyperspektrální data, která obsahují několik desítek až stovek spektrálních pásem o šířce několika nm. Hyperspektrální snímky proto poskytují velmi podrobnou informaci o zemském povrchu viz. obrázek [2]. Rozdíly mezi snímky lze vidět na obrázku 2.12. Při pořizování snímků však může zasahovat do měřeného radiačního signálu atmosféra. Hyperspektrální zobrazovače jí z něj dokážou odstranit, zatímco multispektrální zobrazovače to nedokáží. Proto multispektrální zobrazovače mohou pořizovat snímky pouze v atmosferických oknech [7].



Obrázek 2.12: Rozdíl mezi multispektrálními (spodní křivka) a hyperspektrálními daty (horní křivka)

Vertikální osa udává odrazivost materiálu. Horizontální osa udává interval spektra, nad kterým byl snímek pořízen. V horní křivce je zachyceno mnohem více vzorků, než v dolní křivce, převzato a upraveno [7].

2.2.4 Správa a přístup družicových datových sad

V dnešním světě je poptávka po družicových datech opravdu veliká, využívají se napříč různými odvětvími. Jenže se nejedná o levnou záležitost, alespoň tedy z počátku ne. Vesmírné programy už nějakou dobu existují a s nimi i získávaná data. Přístup k družicovým datům se liší dle poskytovale daných dat, respektive provozovatele příslušné družice. Existují komerční družice, u kterých se pohybuje cena za snímek v řádech tisíců až desetitisíců korun. Díky programu Copernicus a dalším iniciativám však dochází k postupnému posunu směrem k otevřené datové politice [2].

Data pro širší veřejnost poskytuje například Evropská kosmická agentura (ESA)[8]. Data však nejsou „úplně“ veřejná – je zapotřebí se registrovat a sdělit účel, za jakým budou data využívána. Na základě požadavku agentura daná data zpřístupní nebo ne, případně dojde k další domluvě. ESA není jediným zdrojem takto veřejných dat, například National Aeronautics and Space Administration (NASA) nebo National Oceanic and Atmospheric Administration (NOAA) také poskytují družicová data. Vždy je třeba respektovat a dodržovat politiku a podmínky použití těchto dat.

Google Earth Engine

Google Earth Engine je platforma pro vědeckou analýzu a vizualizaci geografických dat pro akademické, neziskové, obchodní a vládní uživatele. Obsahuje družicové snímky, které ukládá do veřejného archívu dat[9]. Zahrnuje i historické snímky, získané až 40 let zpět. Snímky jsou přijímány denně a jsou k dispozici pro celosvětovou těžbu dat [9].

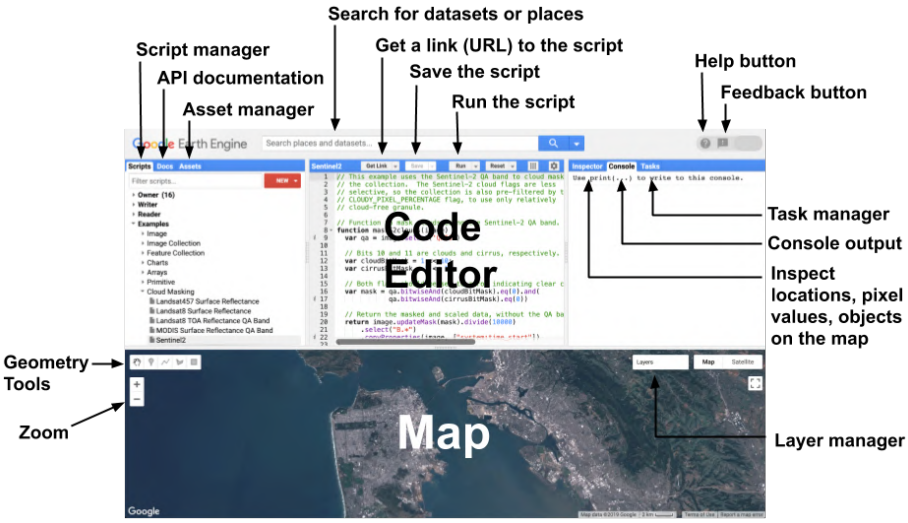
Poskytuje také programovatelné aplikační rozhraní (API), společně s nástroji pro získávání a rozsáhlé analýzy dat. Pro přístup k API je třeba vyplnit formulář s žádostí, která musí projít schvalovacím procesem [9].

Datových sad jsou již v archívu stovky. Každá datová sada nese pak svůj specifický název na základě účelu. Obsahuje vždy stručný popis, k jakému účelu slouží, kdo stojí za jejich vznikem, a jaké družice byly využity při pořizování snímků. Dále jsou uvedena jednotlivá pásma, kde každé pásmo má vlastní název, datový typ, škálu, masku a projekci. Například snímky s velmi vysokým prostorovým rozlišením, které zobrazují povrch Země, tak jak by jej viděl člověk pouhým okem, mají hodnotu RGB rozdělenou do tří pásem, pro každý barevný kanál, ale mohou obsahovat i další pásma. Uvádí se i prostorové rozlišení snímků a podmínky použití.

Datové sady lze vizualizovat přímo ve webovém rozhraní **Earth Engine Code Editoru** a pracovat s nimi přímo v editoru, viz obrázek 2.13. S těmito daty lze operovat i mimo editor a je možné je například exportovat pomocí skriptu napsaném v **Pythonu** nebo **Javascriptu**. API poskytuje bohatou dokumentaci, jak pracovat s metodami, objekty a podobně. Právě tento editor může vývojáři usnadnit práci, zvláště pokud chce využít data mimo editor.

API funguje na principu **Client-Server** a je velice důležité rozlišovat klientskou část kódu od serverového. Jakýkoliv objekt, který patří do serverové části, má před sebou `ee` jako „earth engine“.

[8] https://www.esa.int/Applications/Observing_the_Earth/How_to_access_data
[9] https://developers.google.com/earth-engine/datasets



Obrázek 2.13: Earth Engine Code Editor
Webové rozhraní Earth Engine API, převzato[10].

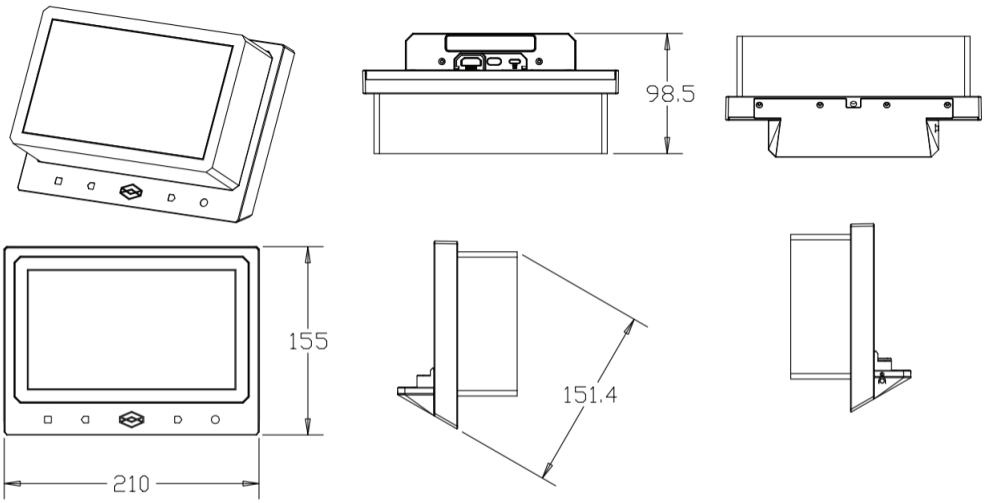
2.3 Holografický displej Looking Glass

Zobrazovacích zařízení je dnes spousta, od mobilních displejů, LCD monitorů až po virtuální realitu. Dalším a relativně novým zobrazovacím zařízením je holografický displej Looking Glass (dále jen „LKG“) od společnosti Looking Glass Factory, která přišla na trh s tímto displejem s hlavním cílem a tím je – uvést 3D obsah opravdu do reálného prostoru, bez potřeby nasazovat jakýkoliv další headset. Zařízení je ve vývoji již od roku 2014.

Společnost nabízí displej ve třech velikostech – 8.9″ displej, na kterém byla testována výsledná aplikace. Schéma lze vidět na obrázku 2.14. Dále nabízí 15.6″ displej společně s 8K displejem, který je osmkrát větší než 8.9″. Poměrové porovnání je na obrázku 2.15.

Prozatím se praktické využití tohoto displeje v průmyslu ukázalo zejména v 3D modelování, zobrazení chemických sloučenin, lékařských vizualizací a v mapování. Nicméně díky poměrně vysoké ceně tento displej zatím není standardem.

[10] https://developers.google.com/earth-engine/playground



Obrázek 2.14: Holografický displej 8.9″
Na obrázku jsou vidět rozměry a perspektivy malého displeje LKG, převzato a upraveno [5]

Obrázek 2.15: Porovnání velikostí displejů Looking Glass, převzato[11]

2.3.1 Princip fungování displeje

Oproti běžným 2D displejům, LKG přidává třetí rozměr a tím je hloubka displeje Displej využívá patentovanou technologii světelného pole, které poskytuje 45 diskrétních pohledů na 3D scénu. Tyto pohledy je možné vidět pod zorným úhlem 50°, právě díky takovémuto uspořádání pohledů vytváří 3D dojem dvěma způsoby [5]:

- Změnou pozorovacího úhlu, pod kterým uživatel displej pozoruje – zjednodušeně pohybem okolo LKG
- Představením různých perspektiv každému oku uživatele

[11] https://lookingglassfactory.com/product/8k

Nutno zmínit, že pohledy na LKG se mění pouze horizontálně, nikoliv vertikálně. Pro zobrazení a vytvoření scény, včetně jejího 3D dojmu, LKG používá „quilt“, o kterém pojednává sekce následující podsekce.

2.3.2 Quilt

Quilt je standard, který LKG využívá pro vytvoření 3D dojmu a slouží k několika účelům – k ukládání a načítání obrázků zobrazovaných na LKG, a ve vývojářských balíčcích (podsekce 2.3.4) jako dílčí krok pro vytvoření výstupu na displej. Dále slouží jako formát, podle kterého je možno manuálně vytvářet obrázky či videa, které bude možné následně korektně zobrazit na displeji. Vývojáři si mohou tedy vytvářet quilty samostatně, nicméně obecně, software LKG nese zodpovědnost za konverzi variabilních 3D prostředí na quilty [5].

Formát quiltu

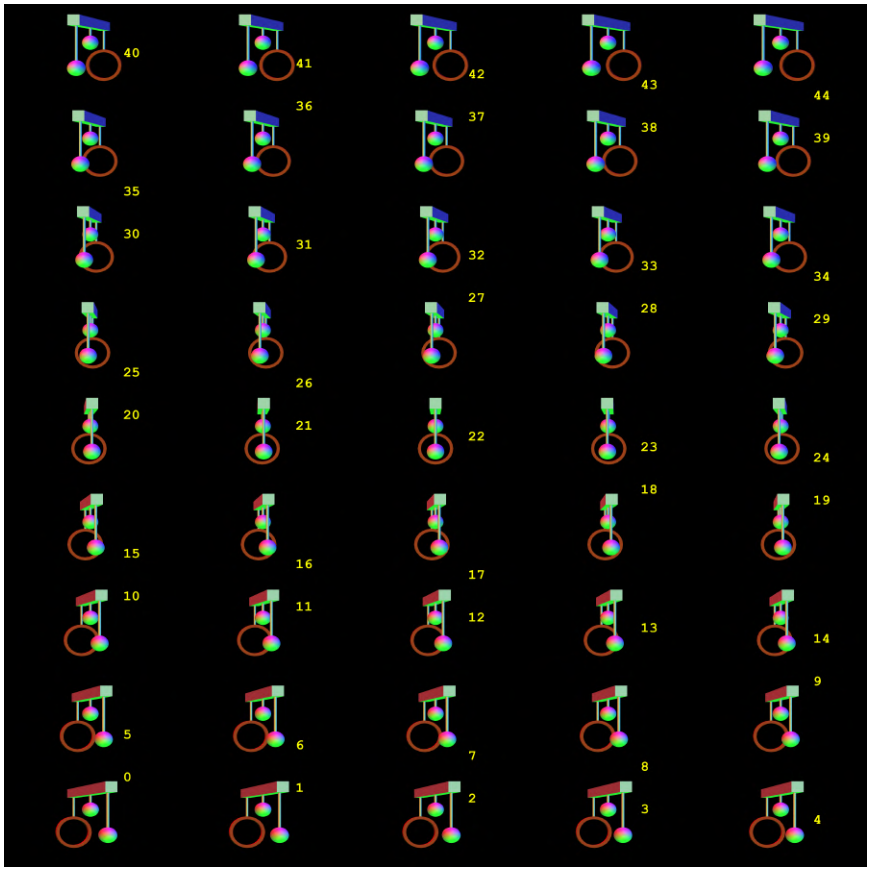
Quilt je obrázek složený ze sekvence standardních 2D obrázků uspořádaných do řádků a sloupců. Lze si jej tedy představit jako datovou strukturu pole, kde každý index představuje jeden pohled na scénu, viz obrázek 2.16. Ukázkový quilt je zobrazen na obrázku 2.17. Obrázek je sestaven systematicky – v levém dolním rohu se nachází obrázek, který představuje pohled zleva na scénu, neboli 0° úhel a v pravém horním rohu je pak obrázek s pohledem zprava na scénu, tedy maximální úhel [5].

Standardní formáty pro LKG běžně obsahují 45 pohledů o 9 řádcích a 5 sloupcích nebo 32 pohledů o 8 řádcích a 4 sloupcích. Ostatní formáty jsou sice validní, nicméně nemusí se korektně zobrazovat na LKG. Tento standard může být aplikován prakticky na jakýkoliv obrázkový nebo video soubor typu `jpg`, `png`, `gif`, `mp4`, `mov` a jiné [5].

40	41	42	43	44
35	36	37	38	39
30	31	32	33	34
25	26	27	28	29
20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

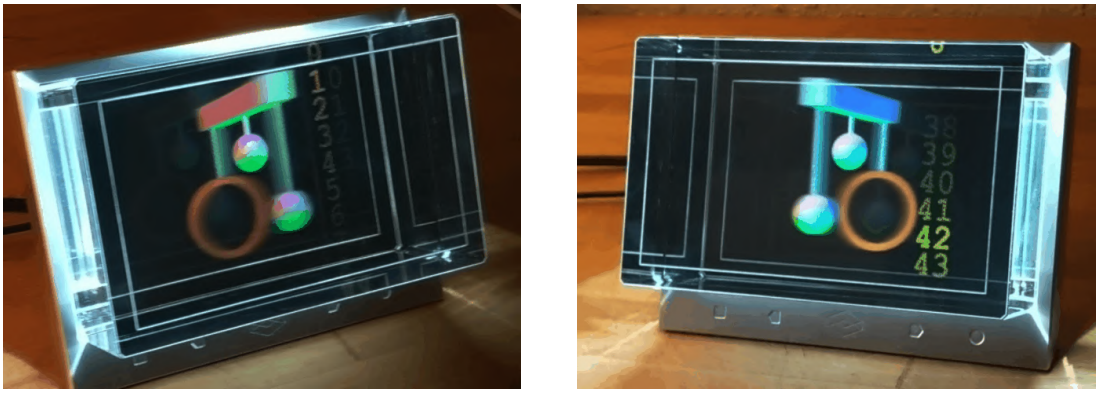
Obrázek 2.16: Uspořádání a indexování obrázků v quiltu

Takto jsou uspořádány v jednom obrázku jednotlivé pohledy na 3D scénu, červeně je vyznačený levý pohled a modře pravý pohled.



Obrázek 2.17: Quilt

Na obrázku je vidět quilt ve standardním formátu. Scéna je složená z jednoduchého objektu pro lepší demonstraci – násada s kruhem v pozadí, s koulí v popředí a ještě jednou koulí mezi nimi. V levém dolním rohu lze objekt scény vidět zleva a v pravém horním zprava, převzato [5].



(a) Pohled zleva

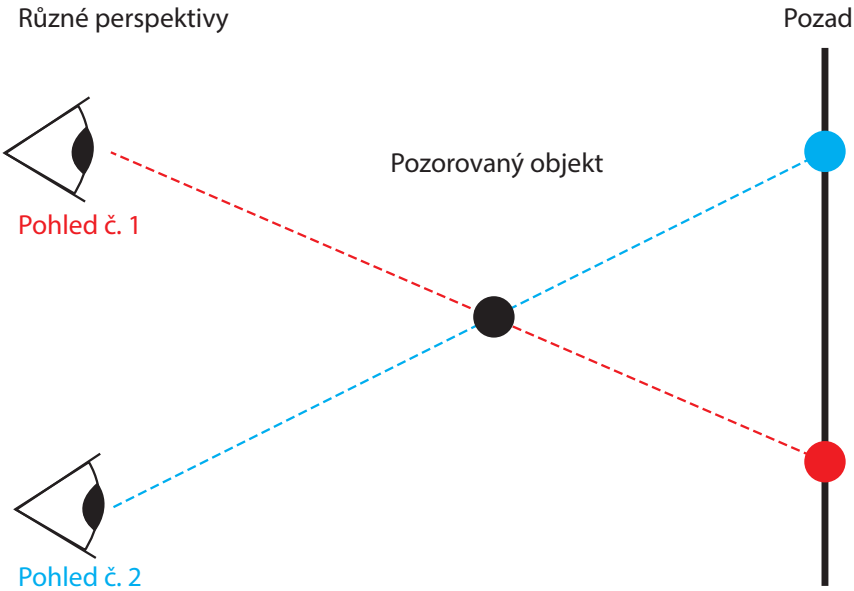
(b) Pohled zprava

Obrázek 2.18: Zobrazení quiltu na displeji LKG

Na obrázcích lze vidět levý (a) a pravý (b) pohled na scénu quiltu z obrázku 2.17, převzato [5].

2.3.3 Rovina s nulovou paralaxou

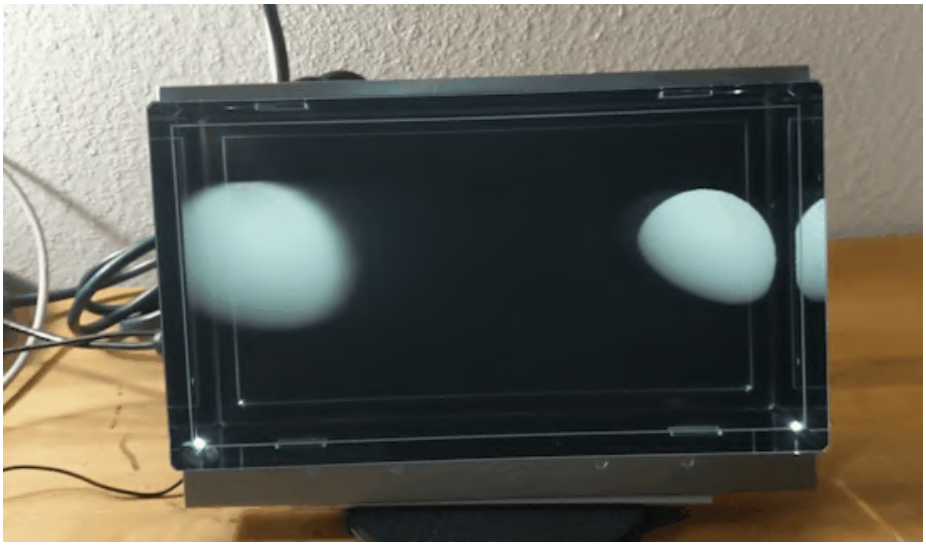
Na výstupu LKG displeje mají různé hloubky různé vlastnosti, které se odvíjejí od roviny s nulovou paralaxou neboli zaostřovací rovinou [5]. Obecná definice paralaxy zní následovně: zdánlivé přemístění objektu (posun jeho polohy na pozadí) je způsobené změnou pozorovací polohy, která poskytuje novou linii pohledu [21]. Pro představu definice slouží obrázek 2.19.



Obrázek 2.19: Ilustrace definice paralaxy
Na obrázku lze vidět pohledy z dvou různých perspektiv a místa na pozadí, kde se promítne pozorovaný objekt.

Objekty, které leží v zaostřovací rovině, zůstávají nehybné vůči pozorovateli, zobrazují se tedy na stejném pixelu. Objekty, které neleží v zaostřovací rovině se pohybují, pokud pozorovatel změní úhel, pod kterým na displej hledí.

Z obrázku 2.18 si lze všimnout, že na LKG je současně zobrazeno více perspektiv. Skutečnost, že je možné vidět více pohledů najednou má svá pro i proti. Výhodou je, že pokud se pozorovatel pohybuje okolo LKG, tak pohledy mezi sebou nepřeskakují a nemění se diskrétně, ale navzájem se prolínají, což dělá 3D zážitek přirozenější. Ačkoliv to z obrázku není příliš zřetelné, tak nevýhodou je rozmazání, které překrývání pohledů způsobuje. Objekty, jenž leží v zaostřovací rovině, vypadají ostřeji, tudíž jejich zobrazení pixelů je v mnoha pohledech konzistentní. A objekty, které jsou dále od roviny, se pak zobrazují na jiných pixelech na základě perspektivy, proto se jeví rozmazanější [5].



Obrázek 2.20: Demonstrace rozmazání scény na displeji LKG

Koule nalevo je více rozmazaná z důvodu, že se nachází příliš daleko od zaostřovací roviny, zatímco koule napravo je umístěna přímo na zaostřovací rovině, a proto je její obraz ostřejší, převzato [5].

2.3.4 Software a nástroje pro vývojáře

Looking Glass Factory poskytuje vlastní software pro procházení a načítání modelů vytvořených komunitou LKG. Mimo jiné na svém webu poskytuje i volně dostupné knihovny pro kohokoliv, kdo má zájem vytvořit jakoukoliv scénu pro LKG. Tyto knihovny jsou vytvořeny pro nástroje, které umožňují tvorbu 3D modelů. Doposud jsou zveřejněny knihovny pro Unity, Three.js, Unreal Engine, programovací jazyky C/C++, Blender a Voxatron.

2.4 Unity

Unity je multiplatformní herní engine pro tvorbu 2D a 3D her, a řadí se mezi jeden z nejpoužívanějších enginů v dnešní době. Poskytuje kompletní vývojové prostředí s grafickým rozhraním. Je volně dostupný, avšak firma využívající Unity za účelem zisku nesmí překročit obrat 100 000$ za rok. Aplikace v Unity lze vytvářet pomocí objektů a nástrojů, které vývojové prostředí poskytuje, nicméně je možné rozšiřovat vlastní aplikace pomocí dalších balíčků a assetů. Součástí těchto balíčků mohou být již předdefinované objekty, tzv. „Prefab“ – tyto objekty lze následně umístit do scény, kde jsou instanciovány. Objekty lze rozšiřovat pomocí komponent, které mohou popisovat jeho chování nebo například vzhled. Dále podporuje tvorbu skriptů napsaných v jazyce C# – tyto skripty lze následně aplikovat na objekty jako komponentu. Unity si lze propojit s vývojovým prostředím Microsoft Visual Studio[12] pro programování skriptů. Unity má poměrně rozsáhlou a udržovanou dokumentaci, jak k manuálu editoru, tak ke skriptovacímu API. Běh aplikace je pak zajištěn pomocí scén, které mohou obsahovat jednotlivé objekty, skripty atp. Unity poskytuje i celkem jednoduchou tvorbu uživatelského rozhraní, které je možné používat přímo během scén.

[12] https://visualstudio.microsoft.com/

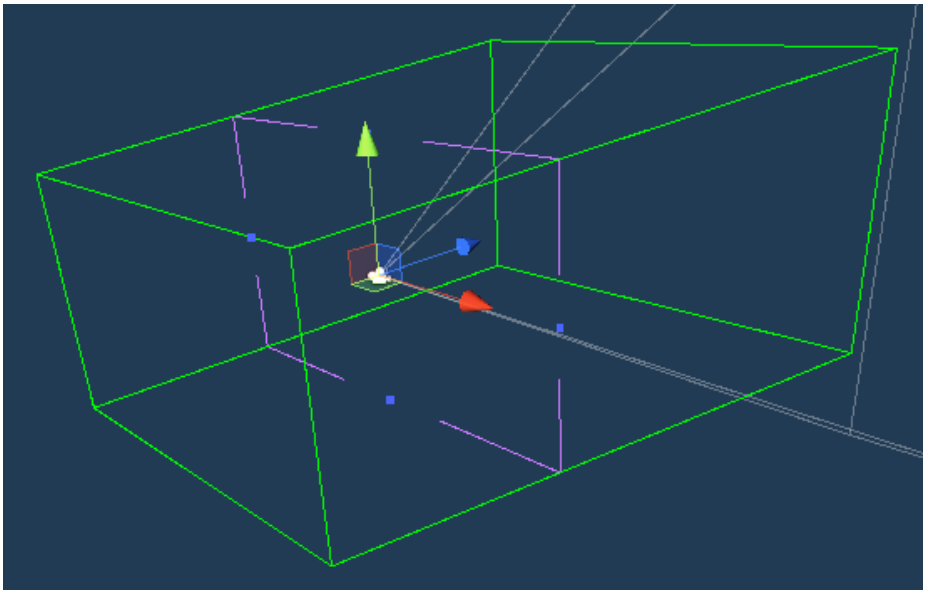
2.4.1 Vykreslování objektů ve scéně

Objekty, které se mají vykreslit ve scéně, jsou složeny minimálně ze dvou komponent, které vykreslení provádí. První komponentou je **Mesh Filter**, ten obsahuje referenci na **Mesh**, jenž definuje tvar objektu. Druhou komponentou je **Mesh Renderer**, který si vezme geometrii z **Mesh Filter**u a vykreslí objekt. **Mesh Renderer** obsahuje seznam materiálů, které jsou použity pro vykreslení objektu. Materiál je asset, který definuje, jak má být povrch vykreslen. Lépe řečeno, používá **Shader**, což je program běžící na GPU a obsahuje matematické výpočty a algoritmy pro určení barvy každého vykresleného pixelu. Unity již obsahuje předdefinované shadery, nicméně umožňuje i vytváření vlastních shaderů [18].

2.4.2 Looking Glass balíček pro Unity

Looking Glass Factory poskytuje balíček přímo pro Unity. Obsahuje objekty, skripty, speciální kameru pro zobrazení scény na displeji LKG. Dále jsou jeho součástí i příkladné scény, dodatečné shadery, balíček **Post Proccesing Stack v2**, který je v pozdějších verzích Unity i jeho součástí, nicméně v balíčku LKG je upravený přímo pro displej LKG.

Jak bylo zmíněno, balíček tedy obsahuje speciální kameru, jako prefab, který je nutné využít pro zobrazení scény na displeji LKG, viz obrázek 2.21. Kamera má spoustu proměnných, které určují, jak se výsledná scéna bude na displeji zobrazovat [5].



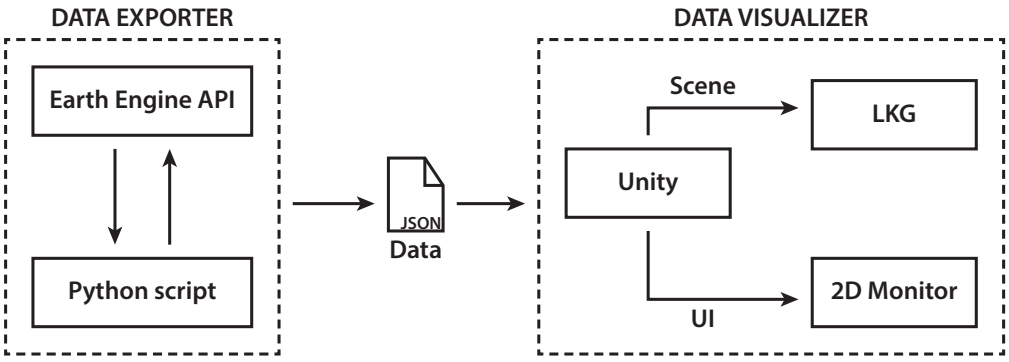
Obrázek 2.21: Prefab Holoplay Capture
Zeleně je označen prostor scény zachycený v displeji a fialově zaostřovací rovina.

Uživatelské rozhraní (dále jen „UI“) je možné umístit přímo do scény, kterou zabírá kamera z obrázku výše, nicméně pro zobrazení UI na standardním 2D monitoru lze použít prefab `Extended UI`. Do tohoto prefabu lze umísťovat standardní Unity UI objekty.

Kapitola 3

Návrh řešení

Tato kapitola se zabývá návrhem řešení pro vizualizaci družicových dat na zeměkouli v omezeném 3D prostoru. Během této disciplíny byla vytvořena již spousta aplikací, jak 2D, tak i 3D. Data jsou velice variabilní a mohou se využívat napříč odvětvími, podle toho jsou uzpůsobeny i aplikace, které se vizualizací zabývají. Popisované řešení v této kapitole je navržené přímo pro holografický displej LKG, viz sekce 2.3. Pro realizaci návrhu byl využit herní engine Unity, zmíněný v sekci 2.4 a Google Earth Engine API, viz podsekce 2.2.4. Navržené řešení se skládá ze dvou entit, první je aplikace, jejíž pomocí jsou data nejprve exportována, a druhou je aplikace vizualizující tato data – navržené schéma lze vidět na obrázku 3.1.

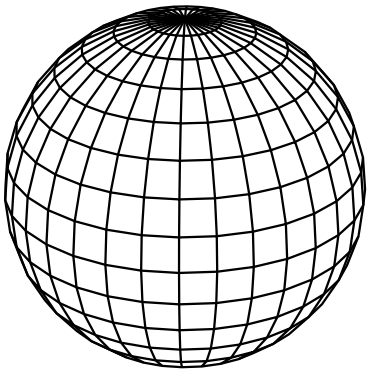


Obrázek 3.1: Schéma navrženého řešení

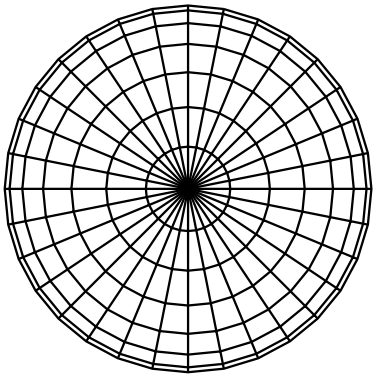
3.1 Model planety země

Základním stavebním kamenem aplikace je model zeměkoule, na jehož povrchu se následně generují objekty reprezentující data. Kouli lze vytvořit několika způsoby, přičemž každý způsob jejího vytváření může způsobit odlišnosti ve struktuře koule. Výsledná struktura může mít vliv na to, jakým způsobem se bude nanášet textura na její povrch nebo počet a distribuce vrcholů, od čehož se odvíjí i počet a velikost jednotlivých trojúhelníků. Pro reprezentaci planety je třeba zvolit přístup, díky kterému model koule bude mít v ideálním případě rovnoměrnou distribuci trojúhelníků, a povrch modelu by tak vypadal stejně z jakéhokoliv pohledu. Z tohoto důvodu je třeba vyřadit „UV sphere“, protože její počet trojúhelníků směrem k pólům se zvětšuje a jejich velikost naopak zmenšuje. Mnohem lepším

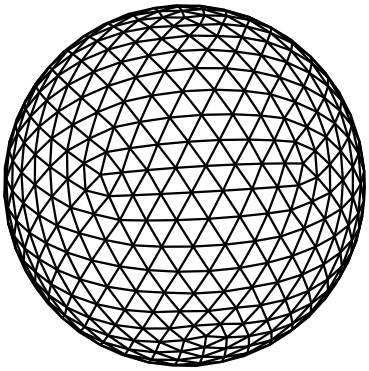
řešením by byla „Icosphere“, která nabízí mnohem lepší distribuci trojúhelníků, nicméně při menším detailu mřížky by nemusela mít dostatečně kulatý tvar a naopak, při detailní mřížce je generováno přiliš mnoho trojúhelníků. A při příliš velikém počtu trojúhelníků by se mohl právě snižovat výkon aplikace. Je tedy třeba použít přístup, který bude generovat rozumný počet podobně velkých trojúhelníků. Proto pro reprezentaci zeměkoule byla zvolena tzv. „Cube sphere“. Výsledné rozdíly mezi těmito modely jsou vidět na obrázku 3.2.



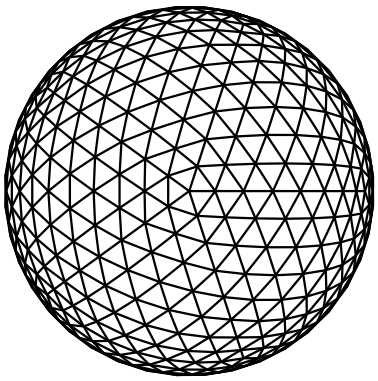
(a) UV sphere – pohled zepředu



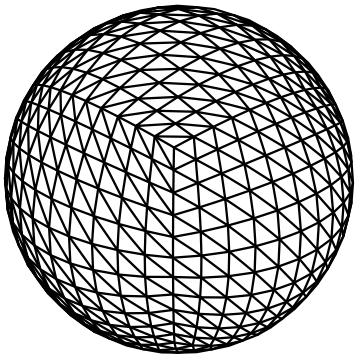
(b) UV sphere – pohled seshora



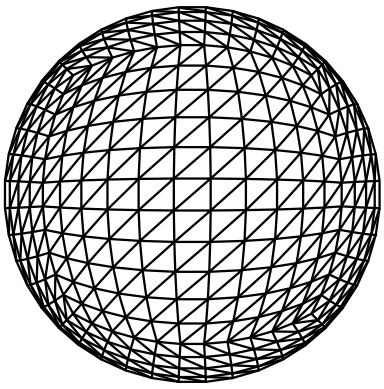
(c) Icosphere – pohled zepředu



(d) Icosphere – pohled seshora



(e) Cube sphere – pohled zepředu



(f) Cube sphere – pohled seshora

Obrázek 3.2: Modely koulí a jejich rozdíly

Znázornění rozdílů mezi přístupy generování modelu koule. Detaily modelů byly nastaveny tak, aby byly vidět dostatečně rozdíly mezi jednotlivými přístupy.

3.1.1 Generování koule

Cube sphere je výsledkem matematického přístupu, který mapuje body z krychle na body v jednotkové kouli. Pro výpočet polohy bodu slouží rovnice 3.1. Toto je odvozeno původně z mapování bodů ze čtverce na body v jednotkové kružnici [16].

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x\sqrt{1 - \frac{y^2}{2} - \frac{z^2}{2} + \frac{y^2 z^2}{3}} \\ y\sqrt{1 - \frac{z^2}{2} - \frac{x^2}{2} + \frac{z^2 x^2}{3}} \\ z\sqrt{1 - \frac{x^2}{2} - \frac{y^2}{2} + \frac{x^2 y^2}{3}} \end{bmatrix} \tag{3.1}$$

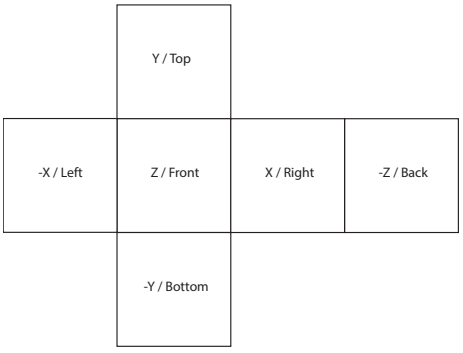
Koule je sestavena z šesti stejných ploch a každá plocha má stejný počet bodů. Kolik bodů je generováno, je nastaveno předem danou hodnotou detailu mřížky. Pokud bude tato hodnota například 10, znamená to, že se pro každou plochu vygeneruje mřížka 10×10 o 100 vrcholech. Detail mřížky je kontrolovatelný jedinou hodnotou, a případně je možné využít optimalizační metodu LOD (Level of Detail). LOD snižuje detail modelu (počet trojúhelníků) čím dále je pozorovatel (kamera) [13]. Poloměr koule lze nastavit pouhým pronásobením výsledného vektoru bodu libovolnou konstantou. Pronásobení se provádí pro všechny body.

3.1.2 Texturování koule

Další důležitou částí je textura, která bude nanesena na 3D model zeměkoule. Podsekce 2.1.5 se zabývala projekcemi ze zeměkoule do roviny. Tuto operaci je potřeba udělat zpětně – tedy z nějaké zdeformované projekce bude vytvořen povrch zeměkoule. Za účelem vizualizace dat na povrchu Země byla zvolena textura bez jakýchkoliv dalších překážek, jako jsou například mraky.

Texturu lze nanést na objekt jako klasickou 2D texturu, přičemž by se muselo, například pomocí geodetických koordinát transformovaných na ECEF koordináty (rovince 2.1), pro každý vrchol objektu přiřadit pixel textury. Nicméně v závislosti na projekci by mohlo docházet k chybám při mapování.

Nabízí se zde i vlastnost Cube sphere, jelikož se původně jedná o krychli, tak je možné na ni korektně aplikovat „cubemapu“ [18]. Cubemapa je typ textury, která se skládá z šesti jednotlivých textur a ty pak lze namapovat na stěny krychle, viz obrázek 3.3.



Obrázek 3.3: Cubemapa

Na obrázku lze vidět, jak by mělo vypadat rozložení obrázků na cubemapu tak, aby na sebe správně navazovaly.

3.2 Vizualizace dat na povrchu Země

Data reprezentují jednotlivé „sloupce“ na povrchu Země. Tyto sloupce jsou v podstatě jen jednoduché kostky, které se liší na základě hodnot dat získaných vždy z určitého bodu na mapě. Data jsou zobrazovány na místech zájmu, jako je právě pevnina. Navíc jsou mapovány po pevnině na celé Zemi, z tohoto důvodu jsou zvoleny datové sady, které obsahují data v globálním měřítku. Tato data obsahují přímo konkrétní hodnoty, jako je například průměrná teplota vzduchu, nadmořská výška a jiné. Pro získání dat slouží již dříve zmíněné Google Earth Engine API (podsekce 2.2.4). Tato data jsou pokud možno zobrazena se stejnou konstatní vzdáleností na modelu Země, s čímž se opět pojí i problémy ohledně projekcí.

3.2.1 Získávání družicových dat

K získání dat je využito Earth Engine API, jehož pomocí je sestavena množina bodů (míst), ze kterých jsou získány informace konkrétní datové sady.

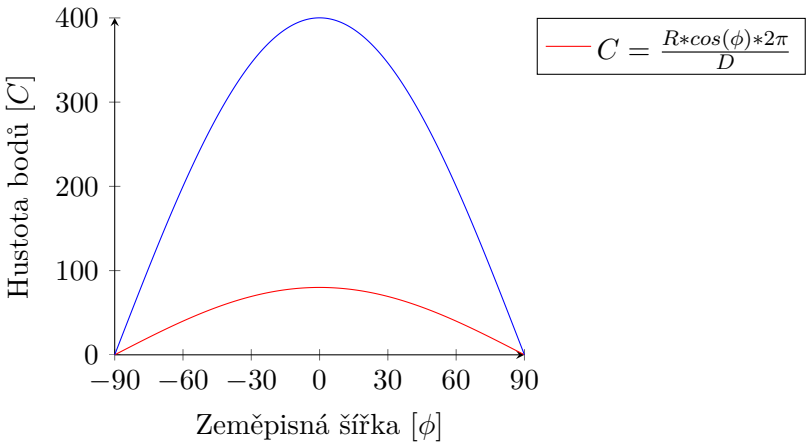
Sestavení množiny bodů

Earth Engine API využívá mercatorovu projekci (podsekce 2.1.5) jako výchozí. Je tedy třeba brát v potaz, že vzdálenosti blíže k pólům jsou markantně vyšší než u rovníku. Pokud by data byla získávána lineárně přes mřížku, tak by hustota sloupců na modelu byla podstatně vyšší právě u pólů. Poloha bodu se v tomto API určuje pomocí zeměpisných souřadnic. K výpočtu přibližně stejných vzdáleností se využijí rovnoběžky zmíněné v podsekci 2.1.3. Pro každou rovnoběžku s určitou zeměpisnou šírkou lze vypočítat její poloměr pomocí rovnice 3.2.

$$r = R \cos \phi \tag{3.2}$$

kde:
r : výsledný poloměr rovnoběžky
R : poloměr Země
ϕ : zeměpisná šířka

Data se budou získávat postupně od Severního pólu, až k Jižnímu pólu. Kružnice budou od sebe vždy vzdáleny o 1°, tedy jejich zeměpisná šířka se bude lišit o 1°. Stanoví se konstatní vzdálenost, jakou budou od sebe objekty vzdáleny na kružnici, přičemž by tato konstanta měla být nastavitelná. Nastavitelnost konstanty zde hraje důležitou roli, jelikož bude mít dopad na množství získávaných dat a tím i výkonnost aplikace. Navíc v souvislosti s prostorovým rozlišením snímků by se z některých oblastí získávaly zbytečně duplicitní hodnoty. A pro předem stanovený počet kružnic se vypočítají jejich obvody. Podílem obvodu vzdáleností je pak získán potřebný počet objektů pro danou kružnici, viz graf 3.4. Ale k získání zeměpisné délky je potřebný ještě jeden údaj a tím je úhel, o který se bude posouvat zeměpisná délka bodu. Tu lze zjistit snadno, jelikož zeměpisná délka nabývá hodnot z intervalu $\langle -180°, 180° \rangle$, což dává dohromady 360°, které se podělí počtem objektů, viz algoritmus 1.



Obrázek 3.4: Hustota bodů na základě zeměpisné šířky

Obvod kružnice rovnoběžky je podělen konstantou D, což dává za výsledek počet bodů C. Grafem funkce je parabola – čím blíž k pólům (menší poloměr rovnoběžky), tím je menší hustota bodů. Při modrém grafu byla konstanta D nastavena na 100 a při červeném na 500.

for $x \leftarrow 0$ **to** 180 **do**
  $latitude \leftarrow (90 - x - 1)$;
  `// compute circuit of current latitude`
  $circuit \leftarrow$ `ComputeCircuit(`$latitude$`)`;
  `// points count`
  $C \leftarrow circuit/D$;
  `// degrees step between points`
  $dist \leftarrow 360/C$;
  for $y \leftarrow 0$ **to** C **do**
    $longitude \leftarrow (-180 + (dist * y))$;
  end for
end for

Algoritmus 1: Algoritmus pro sestavení množiny bodů

Tento algoritmus sestaví celkovou množinu bodů, ze kterých se budou následně čerpat data. Výstupy se liší na základě předem zvolené konstanty D.

Interpretace dat

Pro získání dat v konkrétních bodech je nutné načíst datovou sadu, ze které se budou data získávat. Těchto sad je spousta, některé pokrývají celou Zemi a jiné zase jen určitou oblast. Tento návrh se zabývá načtením vždy pouze jedné sady v jednom okamžiku. Obecně tento návrh bude funkční s jakoukoliv datovou sadou, nicméně vzhledem k jejich různorodosti (hlavně v jednotkách), ne všechny sady budou mít vypovídající informační hodnotu. Především je zaměřená na meteorologická data nebo data s mapováním nadmořské výšky.

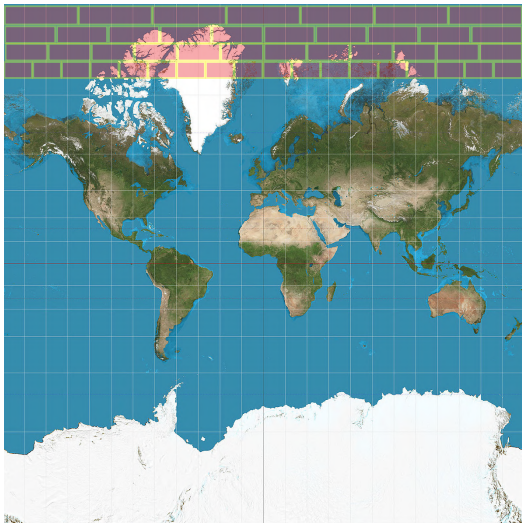
Po načtení dat dojde ke zvolení pásma, z jakého budou informace získány. Hodnoty z pásma jsou načítány pro celkovou množinu bodů. Hodnoty jsou získávány tak, aby bylo možné určit, jestli se nachází právě na pevnině nebo ne. Některé datové sady obsahují data taková, která jsou právě jen na pevnině, ale existují i takové sady, ze kterých toto vyčíst

nelze. Z tohoto důvodu se minimálně při těchto sadách bude muset načíst další sada, která rozděluje pevninu a oceány. Díky tomu nebude nutné dále řešit kontrolu dat, zda se mají vykreslit nebo ne – to lze řešit například pomocí masky. Maska je pouhá textura ve stupních šedi, která se běžně používá pro specifikování oblastí, jaké se mají vykreslit, masku Země lze vidět na obrázku 3.5. Přičemž maska by musela být vytvořena přímo z textury, která se používá pro model Země – je to hlavně z toho důvodu, že daná textura Země má určitou projekci, a proto kdyby byla maska vytvořena z jiné projekce, tak by docházelo k chybnému určení, zda se jedná o pevninu nebo ne.

Obrázek 3.5: Maska povrchu planety Země

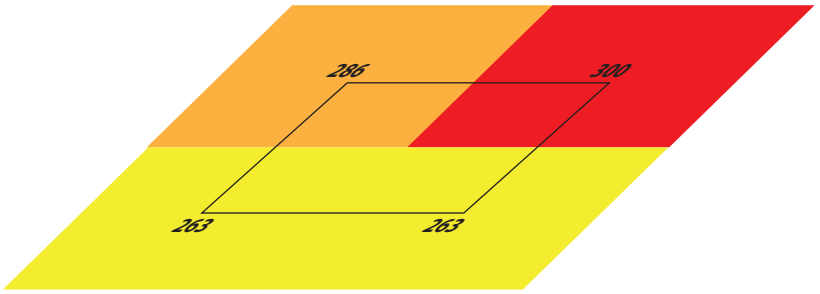
Černá barva reprezentuje pevninu a bílá vodní plochy. Maska byla vytvořena z textury s přímočarou projekcí, viz kapitola 2.1.5.

Hodnoty z bodů se získávají dvěma způsoby, buď se získá hodnota přímo z daného bodu – tento přístup je použit především u snímků s velmi vysokým prostorovým rozlišením – nebo se hodnoty získají z okolí bodu – ten se využije především u snímků s menším prostorovým rozlišením. Je třeba toto okolí tedy vytvořit, to je možné opět z několika dalších sousedních bodů, nebo API umožňuje vytváření jednoduché geometrie, jako jsou obdélníky, u kterých se specifikuje levý dolní roh a horní pravý roh, pomocí zeměpisné šířky a délky. V tomto případě se vytváří obdélníky, kde aktuální bod je jeho středem. Vzhledem k Mercatorově projekci (podsekce 2.1.5) velikosti oblastí jsou větší, čím blíž jsou k pólům, viz obrázek 3.6. Z této oblasti je možné pomocí reduktoru získat buď minimální, maximální nebo průměrnou hodnotu. Reduktor slouží k agregaci dat nad datovými strukturami v API. V implementaci se pak používá průměrná hodnota oblasti. Zde hraje svou roli i prostorové rozlišení dat (podsekce 2.2.3), sestavená oblast má určitou velikost a obsah, tudíž, pokud zrovna snímek bude mít malé rozlišení, tak se oblast může nacházet na jednom jediném pixelu nebo pouze jeho části. Na druhou stranu, pokud snímek bude mít velmi vysoké rozlišení, tak se v oblasti bude nacházet více pixelů. Demonstraci získávání dat z oblasti je vidět na obrázku 3.7

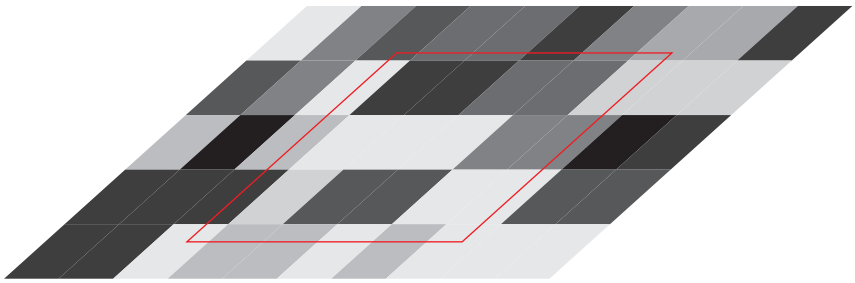


Obrázek 3.6: Velikost oblastí na Mercatorově projekci

Každá oblast je reprezentována červeným obdélníkem se žlutým ohraničením. Oblasti u pólů jsou větší, lépe řečeno širší, kvůli dané projekci. Velikosti oblastí také závisí na konstantě určující vzdálenost bodů na kružnici rovnoběžky.



(a) Oblast s menším prostorovým rozlišením dat



(b) Oblast s vyšším prostorovým rozlišením dat

Obrázek 3.7: Statistika oblastí

Na obrázcích lze vidět oblasti, ze kterých se získá hodnota dat. Předpokládá se, že oblasti na obou obrázcích jsou stejně veliké. Na obrázku (a) se předpokládá snímek s nízkým rozlišením – zde zasahují do oblasti tři pixely, minimální hodnota by byla 263, maximální 300 a průměrná 278. Na obrázku (b) se předpokládá pro změnu snímek s vyšším rozlišením, do stejně veliké oblasti zasahuje tedy více pixelů.

3.2.2 Vygenerování sloupců na 3D modelu zeměkoule

Vygenerování sloupců je posledním hlavním krokem generování scény. Každý sloupec bude mít tři vlastnosti – zeměpisnou šířku, délku a data. Pomocí těchto vlastností bude možné následně určit jeho polohu v kartézském prostoru, včetně jeho dalších aspektů (barva, výška, atp.). Barva a výška se pro všechna data budou vypočítávat dynamicky – to znamená, že se zjistí minimální a maximální hodnoty v získané datové sadě. Podle toho budou následně vygenerovány jednotlivé objekty.

Výběr barvy sloupce

Při zobrazování družicových dat se běžně využívají různé škály barev pro rozeznání jejich hodnot. Jak bylo řečeno v sekci 2.2 na straně 18, některé sady již obsahují jednotlivé barevné kanály v pásmech. Nicméně, v této práci jsou použity sady, které používají pásma, jejichž hodnoty o barvě nic nevypovídají. Z tohoto důvodu je využit gradient, ze kterého je vybrána barva pro jednotlivé sloupce, viz obrázek 3.8b. Barva z gradientu není vybrána náhodně, ale závisí právě na minimální a maximální hodnotě v datové sadě, jak již bylo zmíněno výše. Gradient je možné využít jakýkoli, nicméně je třeba experimentovat s barvami v závislosti na konkrétní datové sadě. Gradient je využit i v rámci uživatelského rozhraní, jako legenda popisující vizualizovaná data, včetně minimální a maximální hodnoty a měrné jednotky pásma.

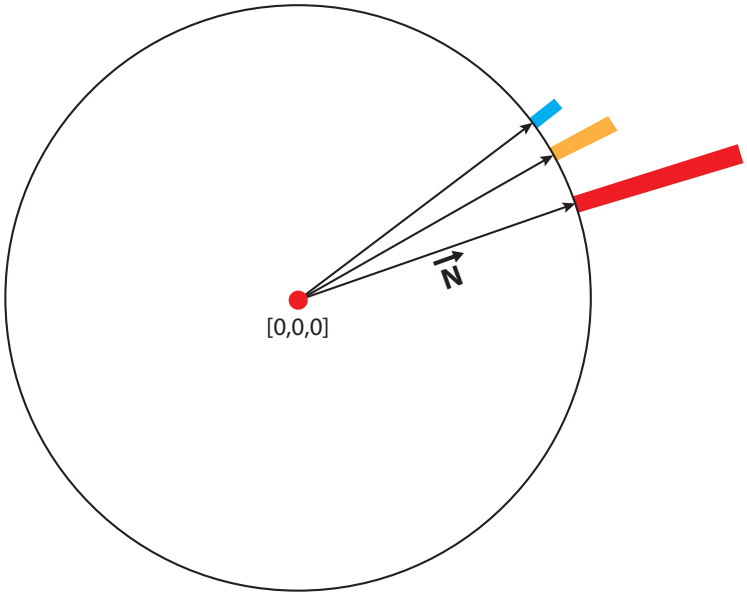
Pozicování sloupce

Pozicování sloupců na povrchu modelu zeměkoule lze opět vyřešit několika způsoby. Jedním z řešení je mapování přímo na vrcholy, s tím jsou ale spojeny i jisté nevýhody. Bylo by nutné udržovat v paměti pro každý vrchol jeho zeměpisné souřadnice. Pozicování sloupce by pak probíhalo buď přímo na vrchol modelu, pokud by zeměpisné souřadnice v datech sloupce odpovídaly souřadnicím vrcholu, nebo by se musel pozicovat mezi vrcholy, mezi jejichž souřadnice by sloupec patřil. Takové řešení je složitější a náročnější na paměť a výkon počítače. Vzhledem k tomu, že scéna bude posléze zobrazena na displeji LKG, je potřeba jiný přístup.

Jelikož při generování každého sloupce bude sebou nést i zeměpisné souřadnice, kde se nachází, lze využít transformačních rovnic 2.1 pro určení polohy v kartézském systému souřadnic. Tyto transformace lze provádět nezávisle, je však třeba znát jen počátek, ze kterého se bude pozice odvíjet – zde počátkem bude střed zeměkoule a poloměr, jak daleko od středu bude pozice vypočítána.

Výpočet výšky sloupce

U sloupců se bude ještě nastavovat výška, ta bude vypočítávána ze získaných dat. A aby bylo možné podporovat více datových sad, jejichž obor hodnot jednotlivých pásem se může značně lišit, bude třeba je nejprve normalizovat. S tím se musí brát v potaz i poměr velikostí sloupců k modelu zeměkoule. Příliš malé sloupce by nemusely mít dostatečnou informativní hodnotu, a rozdíly mezi daty by mohly velmi malé nebo žádné. Na druhou stranu, příliš vysoké sloupce by mohly zakrýt kompletně povrch Země a záběr kamery. Normalizace bude probíhat vždy pro jednu datovou sadu, a to ze všech jejích hodnot. Výška sloupce se vypočítává až po určení jeho polohy, která je jeho počátkem. Ten je následně škálován ve směru jeho vektoru, který směřuje od středu Země pryč. Výsledný koncept vizualizace sloupců je pomocí právě popsaných metod na obrázku 3.8.



(a) Vygenerované sloupce na povrchu Země

(b) Gradient pro výběr barvy sloupce

Obrázek 3.8: Koncept vizualizace sloupců

Pohled průřezem modelu Země, zde je vidět kombinace výše popsaných metod, od kterých se odvíjí implementace. Tento pohled je pouze 2D, pro jednodušší vizualizaci.

3.3 Zobrazení scény na holografickém displeji Looking Glass

Výsledná scéna – zeměkoule s namapovanými daty ze zvolené datové sady pak bude zobrazena v omezeném prostoru na holografickém displeji LKG, který popisuje sekce 2.3. Zobrazení na LKG by mělo uživateli dodat dojem, že se skutečně pohybuje kolem Země a umocňuje tak zážitek z 3D modelu.

Rozložení scény má veliký dopad na zážitek z LKG displeje. Scéna bude statická, tudíž objekty se nebudou nijak pohybovat. Kamera se může pohybovat kolem zeměkoule, včetně přiblížení nebo oddálení se. Pohyb kamery je v případě displeje LKG velmi důležitý, protože kamera musí pracovat se zaostřovací rovinou (podsekce 2.3.3). V blízkosti zaostřovací roviny by se měly nacházet objekty zájmu, kterými jsou v případě této práce dva – povrch Země a datové sloupce. Objekty se sice nebudou příliš pohybovat se změnou pozorovacího úhlu, ale na druhou stranu budou zřetelněji viditelné.

Dalším aspektem, jenž má dopad na hloubku scény je osvětlení – bez osvětlení scéna ztrácí na hloubce. Ale osvětlení je zcela závislé na povaze scény, v tomto konkrétním případě osvětlení může být kontraproduktivní, jelikož by mohlo zkreslovat barvy datových sloupců, nebo by povrch Země nemusel být z určitého úhlu příliš viditelný při špatném použití světel. Aplikace by v tomto případě mohla ztrácet informační hodnotu.

Vliv na hloubku scény může mít i pozadí, které se zobrazuje „za“ hlavní scénou. Jednobarevná a příliš tmavá pozadí ochuzují scénu o její hloubku.

3.3.1 Uživatelské rozhraní

Jakým způsobem a na jakém místě bude zobrazeno UI, při použití LKG, je velice důležité. To se odvíjí spíše od scény, zda její velikost bude zabírat celou šíři displeje či nikoliv. Pokud by se UI vmísilo do displeje, aniž by narušilo zobrazovanou scénu, tak v takovém případě je vhodné veškeré UI objekty umístit na zaostřovací rovinu, viz podsekce 2.3.3. Ale i přesto, že by UI bylo umístěno právě zde, tak může být nepříjemné pro oči, což by mohlo zkazit celkový dojem ze scény. Je spíše nežádoucí UI umístit právě přímo do scény displeje LKG, stává se tak zbytečně rušivým elementem.

Mnohem lepším řešením je zobrazovat veškeré prvky UI na standardním 2D monitoru. Tento přístup je využit i ve výsledné aplikaci této práce. Obsahuje menu pro přepínání scén, návod jak aplikaci ovládat, a během aktivní scény vizualizující data jsou zde i dodatečné informace o aktuální scéně.

Kapitola 4

Implementace

Tato kapitola se zabývá konkrétní implementací 3D dema vytvořeného v rámci této práce. Pro vývoj byl zvolen herní engine Unity (sekce 2.4) verze 2019.3.0f5. Skripty využívané aplikací byly napsány v jazyce C#. V rámci této práce byla ještě vytvořena konzolová aplikace napsaná v jazyce Python verze 3.7. pro získání dat, která se využívají v implementovaném 3D demu. Při spuštění nástroje pro export dat je možné nastavit několik proměnných, které ovlivňují získávání dat, a poskytují tak jiné výsledky. Hlavní aplikace je strukturována poměrně jednoduše – při spuštění je načtena scéna s hlavním menu, kde si uživatel může zvolit další scény. Scény jsou v aplikaci čtyři – hlavní menu, dvě scény s příklady dat a scéna, kde si uživatel může nahrát vlastní data.

4.1 Použité Assety a knihovny

V rámci této práce byly použity knihovny, které jsou již součástí použitých technologií. Dále byly použity následující knihovny třetích stran:

Earth Engine API[1] **pro Python** – Poskytuje datové sady, včetně různých tříd a funkcí pro práci s těmito sadami.

HoloPlay Unity SDK for the Looking Glass[2] – Balíček obsahuje veškeré potřebné entity pro zobrazení scény na displeji LKG, viz podsekce 2.4.2.

4.2 Export dat z Earth Engine API

Export dat je prvním krokem k jejich vizualizaci. Aplikace zabývající se právě tímto exportem je oddělená od hlavní aplikace, která data vizualizuje – je to především z důvodu, že se tento export nepodařilo v dostatečné míře optimalizovat. Trvalo by tedy zbytečně dlouho, než by se scéna načetla. Proto pak ve výsledné aplikaci jsou již předpřipravené datové sady, které byly exportovány pomocí tohoto nástroje.

4.2.1 Inicializace aplikace a datové sady

Při spuštění aplikace má uživatel možnost si zvolit, mimo jiné datovou sadu a její pásmo. Následně pak dochází k inicializaci API voláním metody `ee.Initialize()`, kdy dochází k ově-

[1] https://developers.google.com/earth-engine/python_install
[2] https://lookingglassfactory.com/devtools/holoplay-unity-sdk

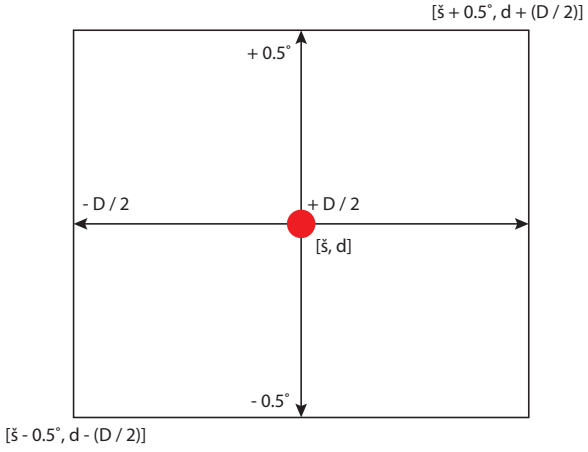
ření uživatele. Proto, pro užití tohoto nástroje je třeba vlastnit účet, který je zaregistrovaný k tomuto API[3]. Pokud je uživatel již zaregistrovaný, je následně vyzván k přihlášení se do Google mail účtu. Toto ověřování probíhá pouze při prvním spuštění.

Po ověření dochází k inicializaci zvolené datové sady, konkrétně tedy snímku nebo kolekci snímků. Ty si API uchovává jako objekt `ee.Image()` nebo `ee.ImageCollection()`, a přijímají řetězec jako argument, konkrétně tedy identifikátor datové sady. Validní identifikátory lze zjistit z katalogu dat zmiňovaném na straně 18, kde je na něj přímo i odkaz. V katalogu je vidět, zda se jedná o jednotlivý snímek, či jejich kolekci. To je v tomto nástroji nutné explicitně zadat parametrem `-i`, který udává, zda se jedná o kolekci. Pokud by toto nebylo specifikováno, mohlo by dojít k chybě, že daná sada neexistuje. Ke zvolené sadě se inicializuje ještě jedna sada **GPWv411: Land Area**[4], která slouží k oddělení vodních ploch od pevniny, jak bylo zmíněno v návrhu na straně 31. Tato sada má velice vysoké rozlišení, tudíž mapování je celkem přesné.

4.2.2 Získání dat pro celkovou množinu bodů

Implementované řešení vychází z jeho návrhu v podsekci 3.2.1. Sestavení množiny bodů je postaveno na základě algoritmu 1. Při jejím sestavování dochází navíc k získání hodnot v daném bodě nebo jeho okolí – jaký přístup se využije, už je v tuto chvíli stanoveno parametrem `-p` při spuštění aplikace.

Nejdříve se vytvoří `ee` objekt reprezentující geometrii, v případě oblasti je to obdélník `ee.Geometry.Rectangle()` a jeho poloha a velikost se nejdříve vypočítají z daného bodu v konkrétním průchodu cyklem. To probíhá tak, že se proměnná určující zeměpisnou délku, o kterou se body na kružnici rovnoběžky posouvají, podělí dvěmi a následně se oblast vytvoří ze dvou bodů. Pro sestavení prvního bodu se od původního odečte právě vypočtená poloviční vzdálenost zeměpisné délky a půl stupně zeměpisné šířky. Pro druhý bod se místo odčítání, vzdálenosti přičítají, viz obrázek 4.1.



Obrázek 4.1: Výpočet velikosti a pozice oblasti

Červený bod uprostřed reprezentuje aktuální bod, pro který výpočet probíhá. Šipky pak ukazují vzdálenost, o kterou se posunou body, které slouží pro vytvoření oblasti. Obdélník reprezentuje celou oblast. Výpočet probíhá v jednotkách stupňů.

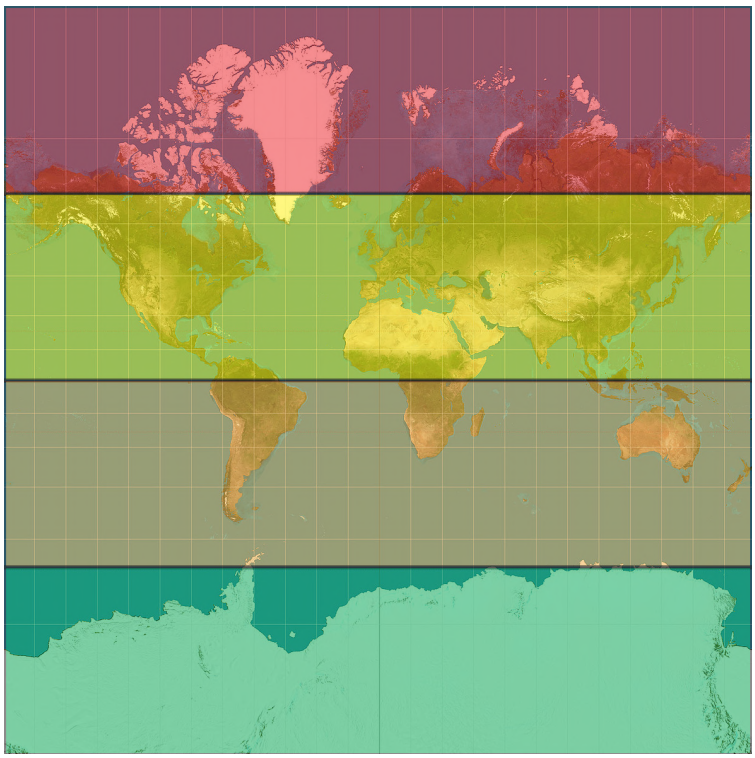
[3] https://signup.earthengine.google.com
[4] https://developers.google.com/earth-engine/datasets/catalog/CIESIN_GPWv411_GPW_Land_Area

V případě bodu je geometrie vytvořena ze zeměpisných souřadnic pomocí `ee.Geometry.Point()`, a není třeba provádět další výpočty míst, ze kterých budou následně získány hodnoty dat.

Následně dochází k vytvoření objektu zapouzdřující statistiku z bodu nebo oblasti. K jeho vytvoření je třeba použít metodu `reduceRegion()`, přičemž tato metoda se volá nad datovou sadou. Tedy pro vizualizovanou datovou sadu a sadu detekující povrch Země. Této metodě je předána vytvořená geometrie, ze které je získána statistika. A jakým způsobem je získána statistika geometrie, je určeno parametrem `reducer`, který přijímá reduktor, viz sekce 3.2 na straně 31. Zde, pro získání průměrné hodnoty, je použit reduktor `mean()`, který je součástí třídy `ee.Reducer`. Každý záznam dat je poté uložen jako `Dictionary` obsahující zeměpisné souřadnice a hodnotu dat, a je přidán do seznamu dat.

4.2.3 Dávkování požadavků

Dávkování bylo implementováno ze dvou důvodů. Jedním z nich je, že se využívají pro optimalizaci získávání dat – o tom více v podsekci 4.6.1. Druhým důvodem je, že při velkém množství dat, by aplikace byla ukončena z důvodu využívání příliš velké kapacity serverů – jedná se o ochranu API, aby jej jedinec příliš nepřetěžoval. Druhý důvod je způsoben především návrhem exportu dat této aplikace. Získání dat pro celkovou množinu bodů sice probíhá najednou, ale tato množina je rozdělena do jednotlivých seznamů, kde počet seznamů je roven počtu dávek, viz obrázek 4.2.



Obrázek 4.2: Rozdělení množiny bodů do dávek

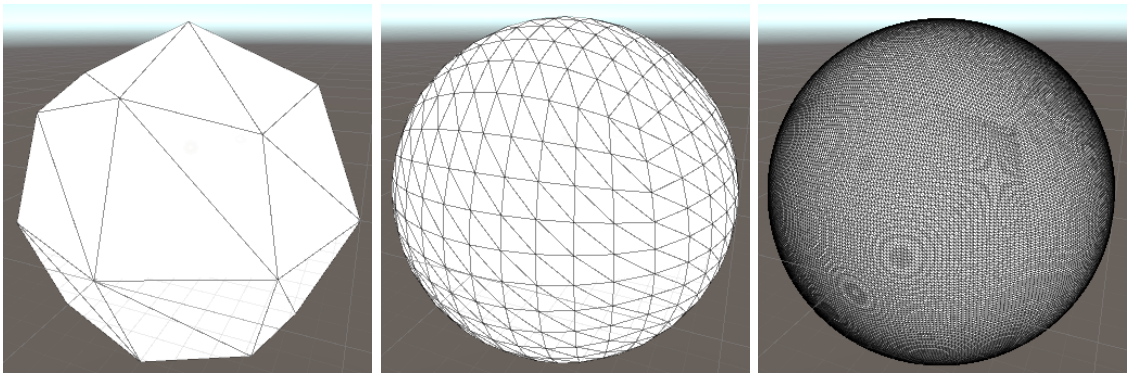
Na obrázku lze vidět rozdělení množiny bodů do dávek. Zde jedna barva reprezentuje jednu dávku a část celkové množiny bodů, které jsou v rámci dávky získány.

Tyto seznamy jsou doposud jen datové struktury uložené na klientské části, je třeba tedy odeslat požadavek ve formě dávky na server. Tyto dávky jsou ve for cyklu postupně poslány na server, a jedna dávka se rovná jednomu volání metody `getInfo()`. Nutno dodat, že všechny zmíněné `ee` objekty a třídy pouze zapouzdřují zprávu, která slouží jako instrukce, jak s danou strukturou má server naložit. Pro získání jakýchkoliv „čitelných" dat je třeba volat právě zmíněnou metodu, přičemž tato metoda je synchronní, a dle dokumentace je jí doporučeno využívat v menším množství [8]. Je tedy nezbytné tuto metodu použít. V původní implementaci se tato metoda volala pro každý bod, a vzhledem k tomu, že se jedná o synchronní metodu, tak získávání dat probíhalo velmi dlouho. Proto její volání bylo odstraněno při jednotlivých bodech, a místo toho se volá pouze pro jednu dávku. Poté co jsou obdrženy všechny odpovědi ze serveru, se data uloží do **JSON** souboru.

4.3 Vytvoření modelu Země

Základ scény tvoří model zeměkoule, který je vytvořen dle návrhu v sekci 3.1. Model je vytvořen ve scéně pomocí „prázdného objektu", což je objekt, který má jedinou komponentu **Transform**, která určuje globální polohu objektu ve scéně. Ten je umístěn v počátku scény, tedy na vektoru [0,0,0]. Tomuto objektu je přidána komponenta ve formě C# skriptu `EarthPlanet.cs`, který obsahuje třídu pro vygenerování modelu. Implementovaná třída dědí ze třídy `MonoBehaviour`, což je bázová třída pro C# skripty v Unity. U třídy je specifikovaný atribut `[RequireComponent(typeof(MeshFilter), typeof(MeshRenderer))]`, který automaticky po aplikování skriptu na objekt přidá vyžadované komponenty. Komponenty obsažené ve zmíněném atributu jsou stručně popsány v podsekci 2.4.1. Třída využívá návrhový vzor Singleton k zajištění jediné instance tohoto modelu. Model se generuje pouze jednou a to při spuštění scény. V Unity lze navíc nastavit pořadí, ve kterém se skripty provádí, přičemž skript `EarthPlanet.cs` se provede jako první, kvůli tomu, že jej následně využívá skript pro generování sloupců. Takto nedojde k chybám, jako například, že proměnné nebyly inicializovány.

Třída obsahuje veřejné proměnné pro kontrolu poloměru objektu a velikosti mřížky – tyto proměnné nelze nastavovat za běhu aplikace, jsou přístupny pouze v Unity editoru pro experimentování během vývoje. Velikost modelu v tomto případě není příliš důležitá, není tedy třeba nastavovat poloměr přesně na poloměr reálné Země, je ale třeba potom při mapování dat brát v potaz poměr velikostí sloupců k Zemi. Velikost mřížky je ale důležitější, příliš malá mřížka by vygenerovala kouli, jejíž detail by byl příliš nízký, a byly by vidět jednotlivé hrany. Na druhou stranu, příliš detailní mřížka by generovala příliš mnoho trojúhelníků, které by v této implementaci nebyly nijak využity, vzhledem k tomu, že se nejedná o velmi detailní mapování terénu s velmi vysokým rozlišením. Je tedy zvolen jakýsi zlatý střed, kdy model je dostatečně detailní a negeneruje příliš mnoho trojúhelníků, viz obrázek 4.3. Značná část kódu pro generování geometrie byla převzata z [6].



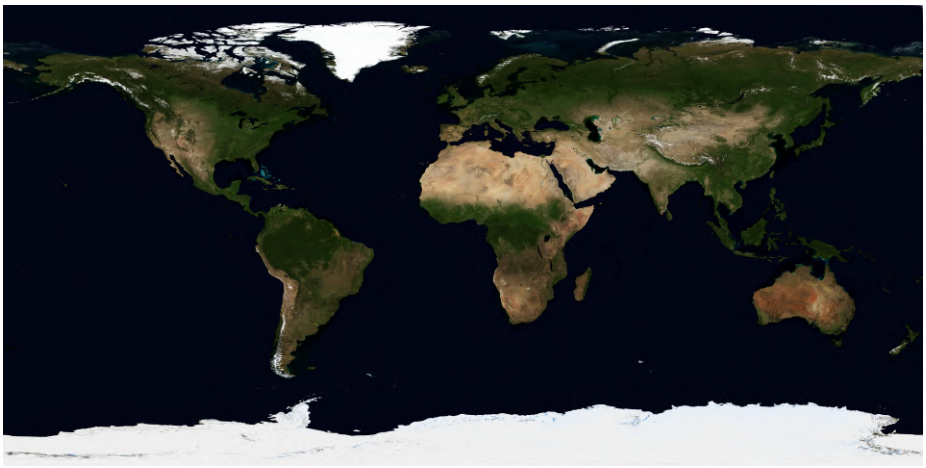
(a) Příliš malá mřížka (b) Adekvátní mřížka (c) Příliš detailní mřížka

Obrázek 4.3: Rozdíl v detailech mřížky

Model (a) má 48 trojúhelníků, model (b) má 1200 a model (c) má 120 000 trojúhelníků.

4.3.1 Texturování Země

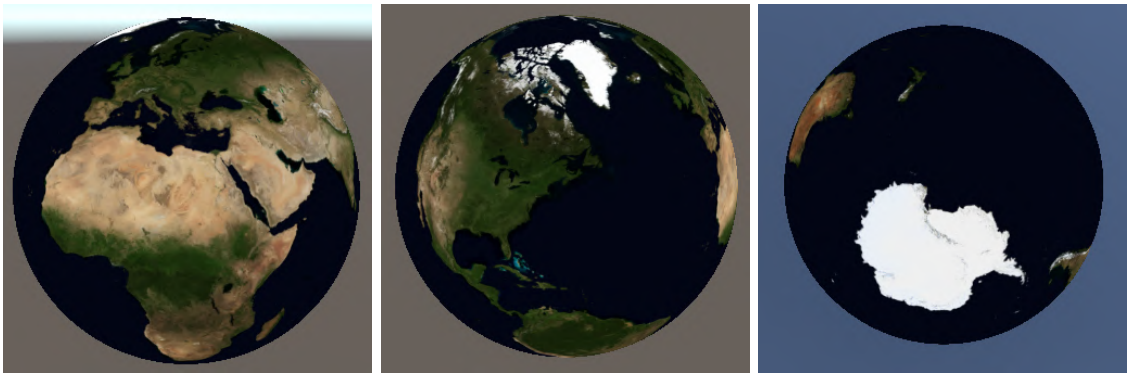
Za účelem vizualizace dat na povrchu Země byla zvolena textura bez jakýchkoliv překážek, které by znemožnily vidět přímo pevninu, například mraky. Textura použitá v této implementaci byla stažena z veřejného katalogu Visible Earth[5] od NASA, přičemž tato textura využívá přímočarou projekci (podsekce 2.1.5). Tato textura má prostorové rozlišení (podsekce 2.2.3) 500 metrů na pixel. Textura je zobrazena na obrázku 4.4.



Obrázek 4.4: Textura použitá pro 3D model Země v globálním měřítku.

Pro nanesení textury je vytvořen materiál `PlanetMat`, který používá shader, jenž nanáší texturu na model jako cubemapu (podsekce 3.1.2). Z textury na obrázku 4.4 by bylo nutné vytvořit nejdříve cubemapu, nicméně toto není třeba, vzhledem k tomu, že při importu textur do Unity, lze nastavit způsob, jakým bude textura uložena. Jedním z nastavení je **Mapping**, které obsahuje možnost **Latitude-Longitude Layout (Cylindrical)** – po aplikaci tohoto nastavení se textura importuje už jako přetvořená cubemapa. Výsledek mapování lze vidět na obrázku 4.5

[5] https://visibleearth.nasa.gov/collection/1484/blue-marble



(a) Afrika (b) Severní Amerika (c) Antarktida

Obrázek 4.5: Výsledek mapování textury

Mapování je bez chyb, lze si všimnout rozdílů ve velikostech a tvarech světadílů mezi těmito třemi obrázky a obrázkem 4.4.

4.4 Zobrazení dat

Implementace zobrazení dat na povrchu modelu vychází z návrhu v sekci 3.2 na straně 33. Celkový proces generování sloupců nastává po vygenerování modelu Země. Do scény je přidán opět prázdný objekt, na který je aplikován skript `ObjectGenerator.cs`, který generování objektu provádí. Třída `ObjectGenerator` dědí z `MonoBehavior`, a ze stejného důvodu je implementován jako Singleton. Třída obsahuje veřejné proměnné pro specifikování souboru, ze kterého budou načtena data, prefab (objekt, který není ve scéně, ale je v balíčku aplikace), který se bude instanciovat pro každý záznam dat, gradient a měřítko v jakém se bude objekt škálovat. Tyto proměnné jsou předem specifikovány pro každou scénu, přičemž všechny využívají gradient z návrhu, viz obrázek 3.8b na straně 34. A jako prefab je použita jednoduchá kostka, kterou lze vytvořit přímo v Unity. Pozice objektu s generátorem musí být stejná, jako pozice modelu Země.

4.4.1 Import dat z JSONu

Třída `ObjectGenerator` si nejdříve načte JSON data ze specifikovaného souboru. K tomu je použita třída `JsonUtility`, která je součástí Unity. Její použití by načetlo pouze jeden záznam, z tohoto důvodu je vytvořena třída `JsonSerializationHelper`, která umožňuje deserializovat pole objektů. Tato třída používá strukturovanou notaci JSONu, to znamená, že je třeba vytvořit další třídu, která specifikuje, jaké proměnné se mají deserializovat. Proto je v rámci třídy `ObjectGenerator` vytvořena privátní třída `PositionWithData`, viz obrázek 4.6, která musí přesně kopírovat strukturu JSON souboru s exportovanými daty.

```
{
  "dataset":
      [{
      "longitude": 162.2365,
      "latitude": 62.1546,
      "data": 300,
      "isOnLand": true
      }]
}
```

```
[Serializable]
private class PositionWithData
{
  public float longitude;

  public float latitude;

  public float data;

  public bool isOnLand;
}
```

Obrázek 4.6: Serializace a deserializace dat

Nalevo je ukázka exportovaných dat obsahující jeden záznam v datové sadě. Jeden záznam je deserializován do třídy `PositionWithData`, která je zobrazena napravo. Více záznamu znamená vytvoření pole této třídy.

Po deserializaci dat do pole dochází k vybrání minima a maxima datové hodnoty z tohoto pole. Pomocí těchto hodnot poté dochází k normalizaci datových hodnot všech prvků pole při generování sloupců.

4.4.2 Generování sloupců

Při generování sloupců nejprve dochází pouze k jejich inicializaci a přidání do seznamu. Počet vytvořených sloupců vychází z počtu záznamů ve vstupním JSON souboru. Objekty reprezentující sloupce jsou v hierarchii scény vytvářeny jako potomci objektu, který má `ObjectGenerator.cs` skript jako komponentu. Nastavování vlastností sloupce poté probíhá v metodě `SetPositionOnEarthSurface()`, ta se volá pro každý sloupec, je ale důležité zmínit, že se jí předává instance třídy z obrázku 4.6.

V této funkci nejdříve dochází k vyfiltrování sloupce, jestli se nachází na pevnině nebo nikoliv. K filtrování se používá hodnota `isOnLand`. Pokud ne, objekt je smazán a pokračuje se na další záznam.

Určení polohy objektu sloupce

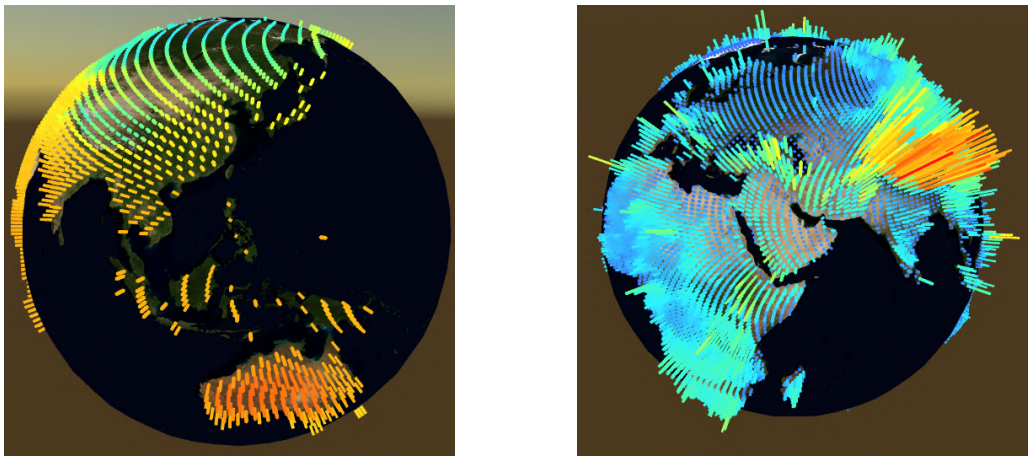
Nejdůležitější částí při generování sloupců, je určení jejich polohy. Tato část se shoduje s návrhem v sekci 3.2 a určení polohy je konkrétně na straně 33. Pro pozicování objektů je implementována statická třída `ECEFconverter`, která obsahuje metodu `GeodeticToECEF()`, které se předávají zeměpisné souřadnice a poloměr modelu Země. Tato funkce implementuje transformaci geodetického souřadného systému do ECEF, viz podsekce 2.1.4. K transformaci jsou využity rovnice 2.1. V této konkrétní implementaci je zanedbána proměnná h, je tedy vždy nulová a proměnná a je rovna poloměru objektu reprezentující model Země. Je ještě třeba provést malou úpravu. Vzhledem k tomu, že orientace os souřadného systému v Unity se liší od orientace os ECEF systému, je třeba rovnice přiřadit správným osám, jinak by se objekty zobrazovaly na jiných místech, než mají. Osa X v Unity se shoduje s osou X v ECEF a osy Y a Z jsou prohozené.

Určení vlastností sloupce

Před nastavováním vlastností, dochází nejprve k normalizaci datové hodnoty aktuálně nastavovaného sloupce na interval $\langle 0, 1 \rangle$. Následně může být nastavena barva a velikost sloupce. Pro jejich nastavení se používá již normalizovaná hodnota, nikoliv původní.

Barva sloupce se vybírá z gradientu, je to také jeden z důvodů, proč jsou hodnoty normalizovány. To, jaká barva se z gradientu pro daný sloupec vybere, probíhá pomocí funkce `Gradient.Evaluate()`, který přijímá hodnotu právě v intervalu $\langle 0, 1 \rangle$. Hodnota 0 je levá barva gradientu a hodnota 1 je pravá barva gradientu. Po vybrání barvy je třeba barvu ještě přiřadit materiálu, který sloupec používá. Jak a jakým způsobem se barva přiřazuje souvisí s optimalizací materiálu, více v podsekci 4.6.2.

Po nastavení barvy se určí škála sloupce – to probíhá tak, že se normalizovaná datová hodnota pronásobí měřítkem. A jako poslední se ještě nastaví rotace objektu, aby sloupce nesměřovaly ve výchozím směru nahoru, ale směrem od středu Země, podle konceptu 3.8. Výsledné vygenerováni sloupců může vypadat potom třeba takto:



(a) Teplotní data

(b) Data s nadmořskou výškou

Obrázek 4.7: Výsledné vygenerování sloupců
Demonstrace vizualizace dvou různých datových sad.

Na obrázku 4.7 lze vidět dynamiku vygenerovaných dat. Pro export datových sad byl využit nástroj, jehož implementace je popsána v kapitole 4.2.

Na obrázku 4.7a je využita datová sada **ERA5 Daily aggregates**[6], přičemž data byla exportována s menší hustotou bodů, proto oproti obrázku 4.7b jsou vidět větší rozestupy mezi sloupci. Jednotlivé hodnoty datové sady se mezi sebou příliš neliší, toho si je možné všimnout především na základě barev. Po experimentování s měřítkem, jakým se budou sloupce škálovat, byla u této konkrétní sady zvolena hodnota 0.6, jelikož se jevila jako nejrozumnější varianta.

Na druhém obrázku 4.7b je pro změnu použita sada **GMTED2010: Global Multi-resolution Terrain Elevation Data 2010**[7]. Data byla exportována s vyšší hustotou bodů, proto jsou rozestupy mezi sloupci menší. Zde jsou rozdíly mezi jednotlivými hodnotami markantnější, proto je zde škálování nastaveno větší, konkrétně na hodnotu 4 – výsledek vizualizace je v tomto případě mnohem atraktivnější.

[6] https://developers.google.com/earth-engine/datasets/catalog/ECMWF_ERA5_DAILY
[7] https://developers.google.com/earth-engine/datasets/catalog/USGS_GMTED2010

Uživatel má možnost nahrát si vlastní data do aplikace a nechat si je vizualizovat. To lze provést nahrazením obsahu souboru `CustomData.json`. Tento soubor se nachází ve složce, jejíž cesta je `Application.DataPath\LKGDataVisualizer_Data\Data`, kde `Application.DataPath` je umístění nainstalované aplikace. Je třeba, aby nahrazený obsah respektoval strukturu dat z obrázku 4.6, jinak se data nenahrajou. V menu následně stačí pouze zvolit scénu `Custom Data Scene`, která vizualizuje vlastní nahraná data.

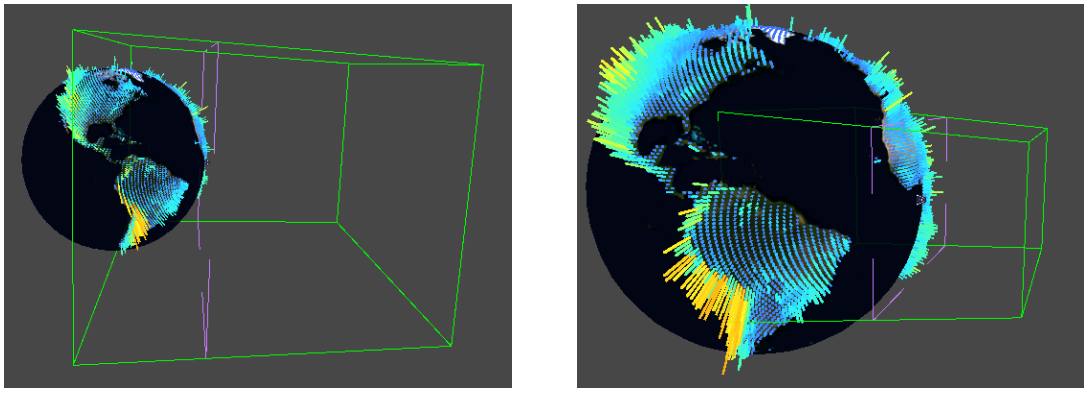
4.5 Zobrazení scény na holografickém displeji

Pro zobrazení scény je využit importovaný balíček **HoloPlay Unity SDK for Looking Glass** (dále jen „LKG balíček“), viz podsekce 2.4.2. Z balíčku je primárně využit prefab `Holoplay Capture`, pomocí kterého je scénu možné zobrazit na displeji.

Prefab kamery je umístěn ve scéně jako samostatný objekt a je na něj aplikován skript `CameraMovement.cs`, který umožňuje uživateli pohybovat se okolo Země, včetně přiblížení a oddálení. Výhodou je, že objekt má počátek přímo ve středu zaostřovací roviny. Umožňuje to snazší implementaci chování kamery. Zaostřovací rovina je tedy přímo na vektoru, který určuje polohu kamery. Objekt Země je tedy umístěn v počátku scény na vektoru [0,0,0], generátor sloupců taktéž, a jejich poloha se měnit nebude. Pro polohu kamery lze také využít zeměpisné souřadnice a jejich následnou konverzi na ECEF souřadnice. Zde je poloměr stanoven o trochu větší než poloměr modelu Země, aby zastřovací rovina byla těsně nad povrchem Země a místo, kam se uživatel právě dívá, bylo co nejostřejší. Tím je zajištěno, že obraz kolem středu zaostřovací roviny bude vždy dostatečně ostrý.

Dalším nastavením, které je třeba řešit, je **Near clip factor** a **Far clip factor**. První zmíněný pojem určuje vzdálenost, kam až bude zasahovat scéna před zaostřovací rovinou, a druhý pojem znamená to samé, ale za zaostřovací rovinou. Near clip factor musí být dostatečně daleko, aby „neusekával“ sloupce v popředí, k tomu by mohlo dojít třeba při přibližování. Far clip factor pro změnu nemusí být nastaven příliš daleko, zbytečně by se renderovaly sloupce, které by nebyly vidět. Nastavení far clip factoru je nastaveno takto ještě z důvodu, že displej má pozorovací úhel pouze 50°, nenastane tedy situace, kdy by uživatel viděl Zemi celou ze všech stran. Vidět bude vždy maximálně jedna strana polokoule.

Přibližování a oddalování Země funguje jinak než u standardní kamery – u té se mění její poloha. Se změnou polohy `Holoplay Capture` kamery by se měnila především poloha zaostřovací roviny. Z tohoto důvodu v tomto případě přibližování a oddalování znamená změnu velikosti scény (kamery), přičemž zaostřovací rovina zůstává stále na stejném místě. Vždy je renderováno pouze to co se nachází uvnitř `Holoplay Capture`. Výsledné nastavení a chování kamery lze vidět na obrázku 4.8.



(a) Výchozí stav kamery (b) Přiblížení kamery

Obrázek 4.8: Demonstrace chování kamery

Na obrázku (a) je vidět výchozí stav kamery, kdy kamera zachycuje polokouli Země. Na obrázku (b) je kamera ve stavu přiblížení, v tuto chvíli kamera zachycuje pouze část scény. Lze si všimnout toho, že zaostřovací rovina zůstává stále blízko povrchu Země.

Na scénu je aplikovaný skybox, což je statický obal scény, který vytváří dojem, že scéna je mnohem větší, než skutečně je. Textura skyboxu je tvořena cubemapou – v této konkrétní implementaci je použita textura vesmíru. Texturu je opět potřeba přiřadit materiálu, ale zde lze použít materiál již se standardním Unity shaderem `Skybox/Cubemap`, texturu použitou pro skybox lze vidět na obrázku 4.9.

Obrázek 4.9: Skybox vesmíru

Textura vesmíru použitá jako skybox scény. Skybox byl vygenerován pomocí nástroje Space 3D[8].

[8] https://wwwtyro.github.io/space-3d/

4.5.1 Uživatelské rozhraní holografického displeje

Implementace uživatelského rozhraní vychází z jeho návrhu v podsekci 3.3.1. V implementované scéně zobrazující planetu Země s datovými sloupci, už není místo, kam by šlo vhodně umístit uživatelské rozhraní aplikace. Do scény je umístěn další prefab `Extended UI` z LKG balíčku, díky kterému lze zobrazovat UI na klasickém 2D monitoru. Ten, jako potomka, obsahuje standardní Unity `Canvas`, do kterého lze umísťovat 2D objekty, jako text, tlačítka, slidery atd. V této aplikaci je pro UI doporučeno používat monitor s full HD rozlišením (1920x1080), jinak jednotlivé objekty UI mohou být nekorektně rozmístěné.

Při aktivní scéně s modelem Země a vizualizovanými daty se na UI nachází název scény, tlačítka pro návrat do hlavního menu a ukončení aplikace. Dále se zde nachází gradient, včetně minima a maxima. A ve dvou příkladných scénách je zde i měrná jednotka dat.

4.6 Optimalizace

Během vývoje aplikace docházelo občas k nepřijatelným výstupům, od příliš dlouhého exportu dat v prvotní implementaci, až po velmi nízké snímky za sekundu ve výsledné aplikaci. Z tohoto důvodu byly zavedeny optimalizace, které jsou popsány v této kapitole.

4.6.1 Multithreading

Tato optimalizace se týká pouze nástroje pro export dat a úzce souvisí s dávkováním požadavků na server, viz podsekce 4.2.3. Implementována byla kvůli synchronní metodě `getInfo()`, aby se neposílaly dávky postupně tak, že by se poslal jeden, na ten se čekalo, a až poté by se poslal další požadavek. Export dat by trval příliš dlouho. Místo toho se pro každou dávku vytvoří jedno vlákno, které běží na pozadí. Vlákna se vytvoří a spouští v jednoduchém for cyklu. Vlákna běží různě dlouho v závislosti na počtu vyžadovaných dat v dávce. Nicméně je vždy třeba čekat na dokončení všech vláken, pokud se nějaké nedokončí, tak ve výsledném JSON souboru budou chybět některá data. Proto se před uložením dat do souboru, vlákna připojují k hlavnímu vláknu. Až po dokončení všech vláken dochází k uložení dat do souboru.

4.6.2 GPU instancing

Pro vykreslení zobrazovatelných objektů ve scéně je třeba poslat informace grafické kartě, tzv. **Draw Call**. Při příliš velkém počtu objektů by tedy rostl i počet požadavků na vykreslení, což by celý proces vykreslování zpomalovalo, a klesala by tak výkonnost aplikace. Jedním z řešení je právě **GPU instancing**, pomocí kterého lze stejné nebo velice podobné objekty vykreslit v menších dávkách.

V implementované aplikaci se tato optimalizace hodí pro sloupce, jelikož se jedná o jednoduché objekty, které se mají lišit pouze v barvě. Tedy i ve výšce, ale ta tu nehraje žádnou roli. Všechny sloupce mají na sobě aplikovaný stejný materiál, který používá shader `ConstantColorShader`. Tento shader je implementovaný tak, aby podporoval právě GPU instanciování. To lze povolit v kódu shaderu přidáním direktivy
`#pragma multi_compile_instancing`. Ještě je třeba stanovit proměnné, které se budou instanciovat – ty se definují v instancing bufferu. V Unity na to existují makra.

Implementovaný shader instancuje a zobrazuje pouze jednu barvu, a to nehledě na pozici kamery. Ignoruje i světla nebo stíny – za prvé z důvodu, že by při nedostatečném osvětlení mohlo dojít ke zkreslení barev, to by způsobilo, že by aplikace ztrácela na informativní

hodnotě. A za druhé, pokud by byly využity především světla, muselo by se jich použít více pro rovnoměrné osvětlení všech sloupců. Jenže sloupců ve scéně je příliš velké množství, a to by vyžadovalo větší množství světel, které není možné zredukovat do rozumného počtu vykreslovacích dávek.

K přiřazení barvy každému sloupci pak dochází pomocí třídy `MaterialPropertyBlock`. Díky této třídě lze měnit vstupní proměnné pro shader, aniž by bylo nutné instanciovat nový materiál. Pokud by se měnila přímo barva materiálu, GPU instancing by nefungoval.

4.6.3 Interpolace

V implementaci je použita ještě jedna optimalizace, a tou je interpolace pohledů. Bez interpolace kamera renderuje všech 45 pohledů na scénů zároveň, což při velkém počtu trojúhelníků výrazně sníží výkon aplikace. U `Holoplay Capture` kamery lze nastavit interpolaci, která vynechá přechodné rendery a interpoluje ostatní rendery pohledů, ze zachycených [5]. Konkrétně se nastavuje počet renderů, ze kterých se poté interpolace provádí.

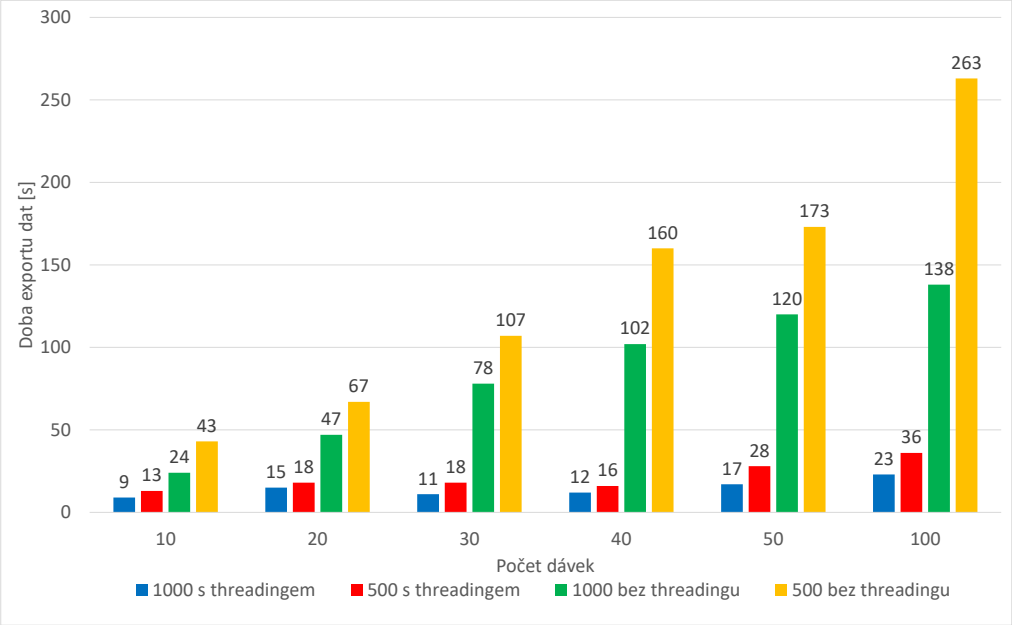
Kapitola 5

Měření

Měření probíhalo na počítači s procesorem Intel Core i5-7300HQ 2.50GHz, grafickou kartou Nvidia GeForce GTX 1050 a 8GB RAM. Měření hodnot aplikace implementované v Unity probíhalo přímo v editoru, tudíž v build verzi tyto výsledky mohou být o něco lepší.

5.1 Doba exportu dat

První provedené měření, je měření doby exportu dat. Měření probíhalo bez implementace multithreadingu a následně s jeho implementací. Vždy v jednu chvíli probíhal jediný export dat, naměřené hodnoty jsou tak na sobě nezávislé. V každém měření byla nastavena konstanta určující vzdálenost mezi objekty a počet dávek, do kterých byl export dat rozdělen, výsledná doba měření je uvedena v sekundách. Pro každé nastavení proběhlo měření celkem třikrát.



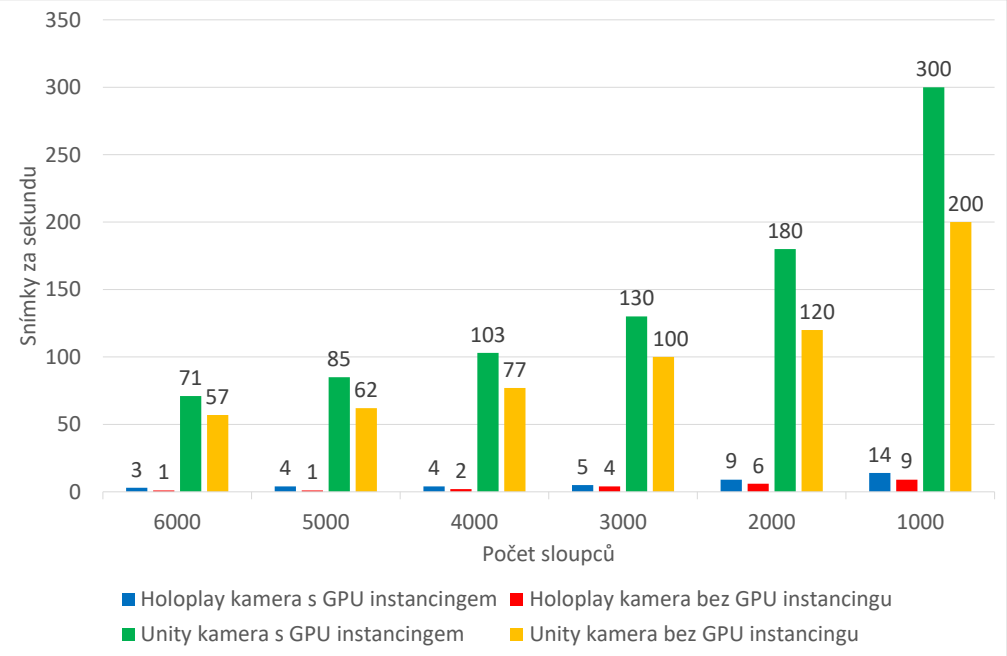
Obrázek 5.1: Doba exportu dat s použitím multithreadingu a bez

Výsledné hodnoty v grafu znázorňují dobu exportu dat – tedy sestavení množiny bodů, získání informací z bodů, zaslání požadavků a uložení do souboru.

V grafu 5.1 jsou naměřeny hodnoty s použitím multithreadingu i bez jeho použití. V implementaci bylo zmíněno, že jedno vlákno se rovná jedné dávce – logicky by tedy vyšší počet vláken měl zajistit rychlejší export dat. Jenže z grafu je vidět, že toto neplatí, je to z toho důvodu, že vlákna se nespouští všechna v jednu chvíli najednou, ale postupně. Zvláště při větších vzdálenostech objektů, jinak řečeno s menším počtem exportovaných dat, je efektivnější využít méně vláken. Toto platí i při menších vzdálenostech, ale při malém počtu dávek, se některá vlákna, alespoň v jednom měření nedokončila. Z grafu je také vidět, že naměřené hodnoty bez použití multithreadingu jsou značně vyšší. Implementace multithreadingu byla tedy úspěšná.

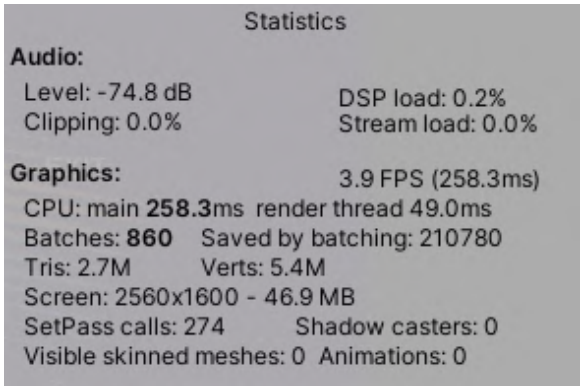
5.2 Výkon aplikace na displeji Looking Glass

V této sekci jsou popsány hodnoty, které byly naměřeny před a po optimalizacích. První aplikovanou a měřenou optimalizací je **GPU Instancing**, viz podsekce 4.6.2. Toto měření probíhalo jak pro standardní Unity kameru, tak pro kameru `Holoplay Capture`.



Obrázek 5.2: Počet snímků za sekundu s použitím GPU instancingu a bez

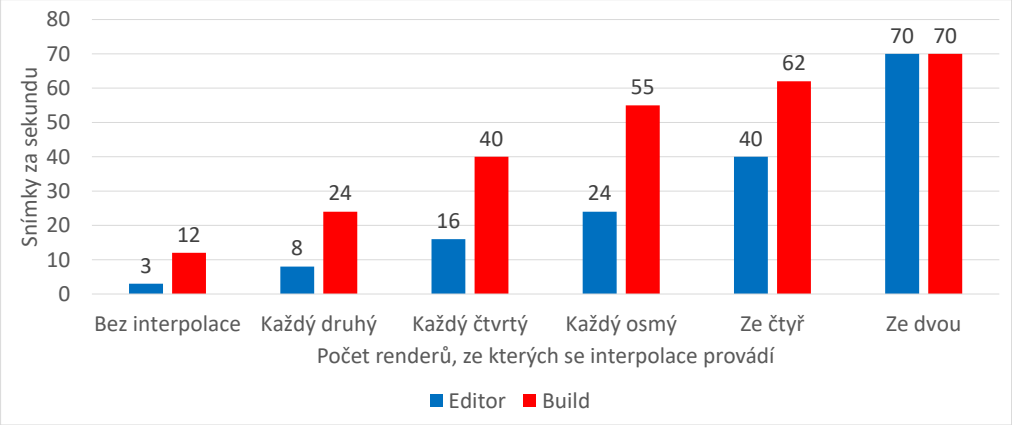
Na grafu 5.2 jsou naměřeny snímky za sekundu (FPS) vždy pro určitý počet sloupců, které jsou vykresleny najednou. Jelikož sloupec je reprezentován kostkou, která má 12 trojúhelníků, znamená to, že při 6000 sloupcích je v jednu chvíli vykresleno 72 000 trojúhelníků a ještě model Země. Z grafu je jasné, že standardní Unity kamera zvládá vykreslovat větší množství mnohem lépe. Kamera pro displej LKG, to zvládá mnohem hůř, to proto, že renderuje 45krát víc trojúhelníků, takže jich je při 6000 sloupcích celkem 3 240 000. Optimalizace GPU instancingem sama o sobě nepřinesla příliš přínosné výsledky ohledně FPS, ale i přesto markantně snížila počet Draw Callů, viz obrázek 5.3.



Obrázek 5.3: Statiska scény v editoru

Na obrázku je třeba si všimnout hodnot „Batches“ a „Saved by batching“. Je vidět, že počet zaslaných instrukcí na GPU byl výrazně zredukován. Statistika byla pořízena při velmi velkém počtu sloupců.

Další měření probíhalo pro interpolaci renderů, viz podsekce 4.6.3. To probíhalo jak v Unity editoru, tak v build verzi aplikace.



Obrázek 5.4: Počet snímků za sekundu s použitím GPU instancingu a interpolace

Na grafu 5.4 jsou vidět FPS při různém počtu interpolací renderů v kombinaci s GPU instancingem za použití kamery `Holoplay Capture`. Tyto hodnoty byly naměřeny ve chvíli, kdy kamera zabírala celou polokouli Země s co největší hustotou bodů – toto jsou tedy nejhorší výsledky, jakých bylo možné v rámci implementované aplikace dosáhnout. Hodnoty FPS se liší celkem výrazně mezi editor a build verzí aplikace, kromě interpolace ze dvou renderů – zde jsou FPS stejné. S větším počtem interpolací může docházet k nekonzistentní reprezentaci pixelů – tedy k jevu paralaxy, viz podsekce 2.3.3. Rozmazání se v aplikaci projevilo především po okrajích scény, kdy objekty jsou právě vzdálenější od zaostřovací roviny. Objekty na zaostřovací rovině to sice také ovlivnilo, nicméně rozdíl nebyl skoro vůbec znatelný. Ve výsledné aplikaci je použita interpolace z každého osmého renderu.

Kapitola 6

Závěr

Cílem práce bylo implementovat 3D demo vizualizující družicová data v omezeném prostoru pro holografický displej LKG. Demo vizualizuje družicová data pomocí různě barevných a vysokých sloupců na základě hodnot dat dle jejich polohy na zeměkouli. V aplikaci jsou scény již s připravenými datovými sadami, nicméně uživatel má možnost nahrát si vlastní data. Hlavním cílem byl plynulý běh i na běžných strojích, zatímco estetická stránka výsledné vizualizace je tématem pro budoucí práci.

Pro získání a export dat byla implementována vedlejší aplikace. Tato aplikace umožňuje export dat z různých datových sad, přičemž je nastavitelné jaké množství bude exportováno. Výstupem exportu je soubor, který lze importovat do výše zmíněné aplikace.

Ohledně exportu dat by se dala aplikace zrychlit například spouštěním všech vláken najednou. A v 3D demu, vzhledem k tomu, jak je implementována geometrie modelu zeměkoule, by bylo možné implementovat techniku Level of Detail. S tím by bylo i vhodné získat velké množství textur planety Země, které by se mohly na zeměkouli přemapovávat na detailnější, vzhledem k vzdálenosti kamery od modelu. Téma této práce bych si vybral znovu, ale tentokrát bych se soustředil na co největší využití vlastností displeje, na které jsem narazil v průběhu vývoje, ale řadu z nich nemohl použít, protože by narušovaly principy vizualizace dat. Také bych dbal na lepší návrh, jelikož ve výsledné aplikaci, vzhledem k navrženému řešení, bylo třeba řešit složité optimalizace.

Díky této práci jsem se mohl stát součástí inovativní technologie holografických displejů. Tyto zařízení mě fascinují a myslím si, že mají obrovský potenciál do budoucna. Získal jsem spoustu zkušeností s herním enginem Unity, ale i s 3D grafikou obecně. A především jsem dostal úplně nový pohled na vizualizaci družicových dat a odvětví kosmonautiky celkově, a to jsem určitě objevil pouze špičku ledovce.

Literatura

[1] Canada, N. R. *Remote sensing tutorails* [online]. 2019 [cit. 16. ledna 2020]. Dostupné z: https://www.nrcan.gc.ca/maps-tools-publications/satellite-imagery-air-photos/tutorial-fundamentals-remote-sensing/9309.

[2] CENIA. GEO/Copernicus v České republice. *Dálkový průzkum Země* [online]. 2016 [cit. 8. ledna 2020]. Dostupné z: http://copernicus.gov.cz/dalkovy-pruzkum-zeme.

[3] Clynch, J. R. Earth coordinates. *Electronic Documentation*. Únor 2006.

[4] Encyclopaedia Britannica, T. E. of. Mercator projection. [online]. Encyclopædia Britannica, inc. Září 2018, [cit. 9. dubna 2020]. Dostupné z: https://www.britannica.com/science/Mercator-projection.

[5] factory, L. glass. *The Looking Glass User Guide* [online]. 2020 [cit. 10. dubna 2020]. Dostupné z: https://docs.lookingglassfactory.com/.

[6] Flick, J. Cube Sphere. *Unity C# Tutorials* [online]. 2020 [cit. 19. dubna 2020]. Dostupné z: https://catlikecoding.com/unity/tutorials/cube-sphere/.

[7] Goetz, A. F. Three decades of hyperspectral remote sensing of the Earth: A personal view. *Remote Sensing of Environment*. Elsevier. 2009, sv. 113, s. S5–S16.

[8] Google. *Google Earth Engine* [online]. 2020 [cit. 19. dubna 2020]. Dostupné z: https://developers.google.com/earth-engine.

[9] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D. et al. Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*. Elsevier. 2017. DOI: 10.1016/j.rse.2017.06.031. Dostupné z: https://doi.org/10.1016/j.rse.2017.06.031.

[10] Iliffe, J. *Datums and map projections for remote sensing, GIS, and surveying*. CRC Press, 2000.

[11] Image, N. a Agency, M. *Department of Defense World Geodetic System 1984: its definition and relationships with local geodetic systems*. Defense Mapping Agency, 1997. Dostupné z: https://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html.

[12] Kennedy, M., Kopp, S. et al. *Understanding map projections*. Esri Redlands, CA, 2000.

[13] Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B. et al. *Level of detail for 3D graphics*. Morgan Kaufmann, 2003.

[14] Maling, D. H. *Coordinate systems and map projections.* Elsevier, 2013.

[15] NOAA. *What is geoid?* [online]. 2020 [cit. 11. prosince 2019]. Dostupné z: https://oceanservice.noaa.gov/facts/geoid.html.

[16] Nowell, P. Mapping a Cube to a Sphere. *Math Proofs* [online]. 2005 [cit. 19. dubna 2020]. Dostupné z: http://mathproofs.blogspot.com/2005/07/mapping-cube-to-sphere.html.

[17] Osserman, R. Ellipsoid. [online]. Encyclopædia Britannica, inc. Srpen 2006, [cit. 20. dubna 2020]. Dostupné z: https://www.britannica.com/science/ellipsoid.

[18] Technologies, U. *Unity User Manual* [online]. 2020 [cit. 18. dubna 2020]. Dostupné z: https://docs.unity3d.com/Manual/index.html.

[19] Welti, C. R. *Satellite basics for everyone.* IUniverse, Inc., 2012. ISBN 978-1-4759-2593-7.

[20] Zhu, J. Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates. *IEEE Transactions on Aerospace and Electronic Systems.* July 1994, sv. 30, č. 3, s. 957–961. ISSN 2371-9877.

[21] Žižek, S. *The parallax view.* Mit Press, 2009.

Příloha A

Obsah CD

`Doc\` – Složka obsahující zdrojové i zkompilované soubory technické dokumentace práce pro LaTeX.

`Exporter\` – Složka obsahující soubory aplikace pro export dat.

`Visualizer\` – Složka obsahující soubory aplikace pro vizualizaci dat.

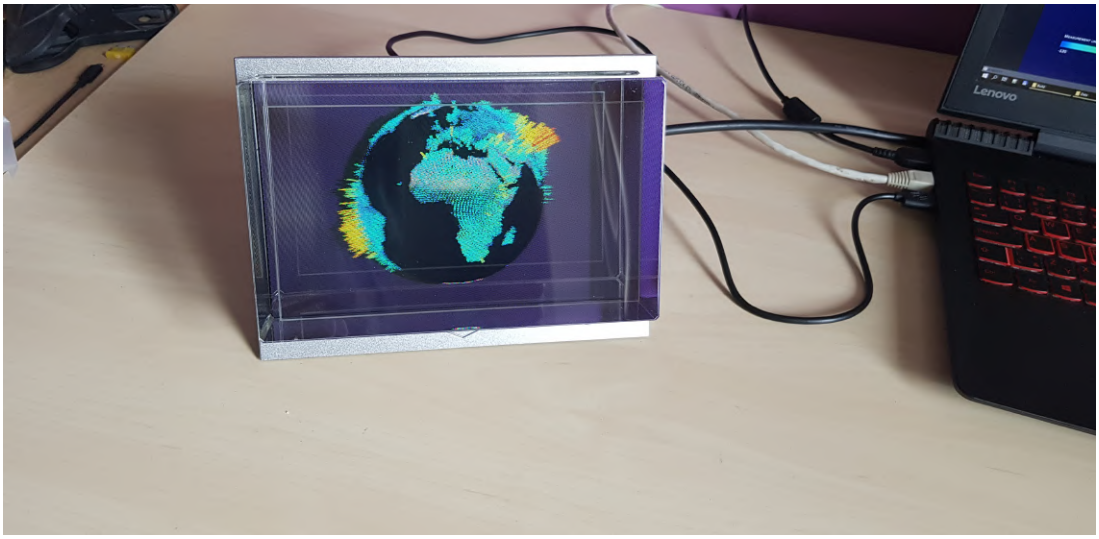
- `Build\` – Složka s přeloženou a spustitelnou aplikací.
- `Source\` – Složka se zdrojovými soubory aplikace.

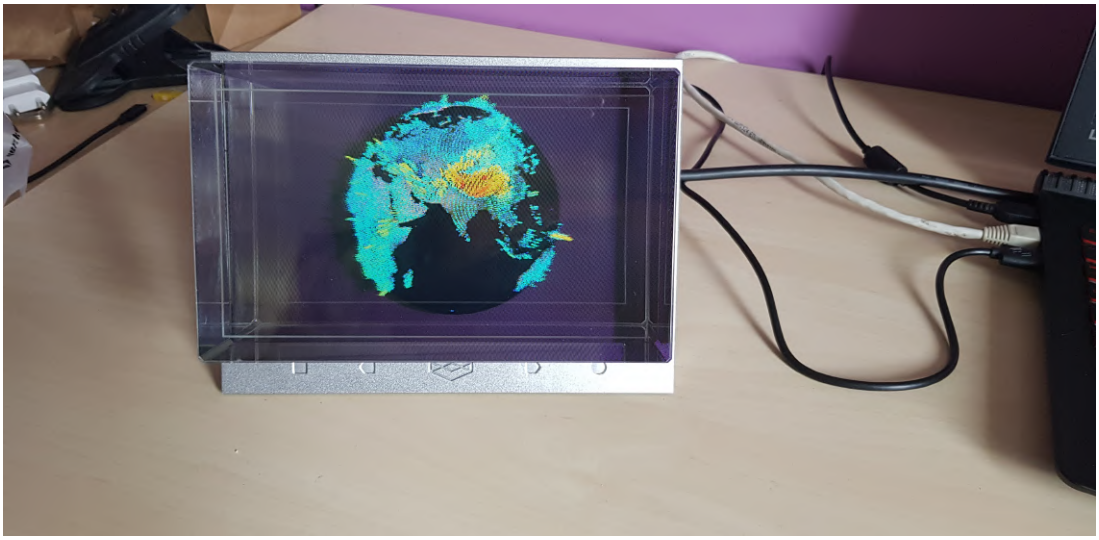
`Datasets\` – Složka s několika variabilními exportovanými datovými sadami.

`Intro\` – Složka obsahující krátké intro.

Příloha B

Obrázky výsledné aplikace






Lenovo


GLOBAL
ELEVATION
Lenovo


GLOBAL
ELEVATION
MAIN MENU
EXIT
MEASUREMENT UNIT: METERS
-120
6189
Lenovo