Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

GEOVIZUALIZACE ZALOŽENÉ NA INTEGRACI 3D FYZICKÝCH MODELŮ TERÉNU A ROZŠÍŘENÉ REALITY

Diplomová práce

Bc. Richard LÁZNA

Vedoucí práce RNDr. Jan BRUS, Ph.D.

Olomouc 2023
Geoinformatika a kartografie

ANOTACE

Cílem diplomové práce je návrh technického řešení pro integraci 3D fyzických modelů terénu a rozšířené reality, se zaměřením na možnost doplnění jednobarevných modelů o textury, animace a další prostorové objekty. Za tímto účelem je vytvořena aplikace využívající principy rozšířené reality, schopná o překrytí daného 3D modelu virtuálním obsahem. Dílčím cílem práce je vytvoření a optimalizace nejvhodnějšího tvaru a struktury markeru.

Hlavním výsledkem práce je mobilní aplikace pro operační systém Android, obsahující dvě hlavní scény a několik doprovodných scén. První hlavní scéna funguje na principu detekce markeru pomocí kamery mobilního telefonu či jiného zařízení s AR funckionalitou, a zobrazení obsahu, který je promítnut nad vytištěný 3D model. Druhá z hlavních scén funguje na principu detekce horizontálně umístěných virtuálních rovin v reálném prostředí a zobrazení odpovídajícího obsahu po dotknutí se obrazovky. Aplikace obsahuje doplňkovou funkcionalitu jako možnost manipulace se zobrazeným obsahem či změnu jazyka aplikace.

Výsledná aplikace a vytištěné 3D modely reliéfu mohou sloužit edukativním účelům. Díky širokému záběru různých geovizualizací aplikace demonstruje různé jevy či objekty atraktivní, a do jisté míry interaktivní, formou. Aplikace také může sloužit jako demonstrace možností rozšířené reality.

KLÍČOVÁ SLOVA

Unity; rozšířená realita; geovizualizace; 3D model

Počet stran práce: 66

Počet příloh: 15 (z toho 7 volných a 8 elektronických)

ANOTATION

The diploma thesis proposes a technical solution for integrating 3D physical terrain models and augmented reality, with the focus on enhancing single colour models with textures, animations and other spatial objects. For this purpose, an augmented reality application which is able to overlay virtual content on top of a 3D model, is developed. The design process and optimisation of the most suitable shape and structure of an augmented reality marker was a secondary goal of the thesis.

The main outcome is a phone application for the Android operating system which consists of two main scenes and several supporting scenes. The first scene detects a marker with the camera of the mobile phone or any other device with AR capabilities. Upon detection, the virtual content is displayed on top of a 3D model as an overlay. The second scene detects horizontally oriented virtual planes in real environment and displays corresponding content upon touching the screen of the device. The application has additional functionality which allows for manipulating with the displayed content or changing language of the application.

The resulting Android application and the printed 3D terrain models can be utilized as education material. Complemented by the broad range of different geovisualisations, the application can demonstrate various natural phenomena or objects in an appealing and interactive way. The application may also serve as a demonstration of the possibilities of augmented reality technology.

KEYWORDS

Unity, augmented reality, geovisualisation, 3D model

Number of pages: 66
Number of appendixes: 15

Prohlašuji, že

- diplomovou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.

- jsem si vědom, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,

- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3),

- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP,

- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona,

- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

Děkuji vedoucímu práce RNDr. Janu Brusovi, Ph.D. za podněty a připomínky při vypracování práce. Dále děkuji Svatopluku Místeckému za poskytnutí dat pro jednu z případových studii zpracovaných v rámci práce. Děkuji také Mgr. Radku Barvířovi, Ph.D. za výpomoc s 3D tiskem. V neposlední řadě děkuji své rodině a přátelům za podporu během studia.

UNIVERZITA PALACKÉHO V OLOMOUCI

Přírodovědecká fakulta

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Richard LÁZNA**
Osobní číslo:	**R21852**
Studijní program:	**N0532A330009 Geoinformatika a kartografie**
Téma práce:	**Geovizualizace založené na integraci 3D fyzických modelů terénu a rozšířené reality**
Zadávající katedra:	**Katedra geoinformatiky**

Zásady pro vypracování

Cílem diplomové práce je vytvoření technického řešení schopného integrovat rozšířenou realitu a vytištěné 3D modely reliéfu. Student se v práci zaměří na možnosti doplnění jednobarevných modelů o textury, animace a další prostorové objekty. Vizualizace budou využívat také GIS analýzy s využitím daného modelu reliéfu. K práci bude student využívat primárně software Blender a Unity. Dílčím cílem práce bude vytvoření a optimalizace nejvhodnějšího tvaru a struktury markerů. Výstupem práce bude několik vytištěných modelů reliéfu a aplikace zobrazující několik scén pro každý specifický model.
Student vyplní údaje o všech datových sadách, které vytvořil nebo získal v rámci práce do Metainformačního systému katedry geoinformatiky a současně vytvoří zálohu údajů ve formě validovaného XML souboru. Celá práce (text, přílohy, výstupy, zdrojová a vytvořená data, XML soubor) se odevzdá v digitální podobě na CD (DVD) a text práce s vybranými přílohami bude odevzdán ve dvou svázaných výtiscích na sekretariát katedry. O diplomové práci student vytvoří webovou stránku v souladu s pravidly dostupnými na stránkách katedry. Práce bude zpracována podle zásad dle Voženílek (2002) a závazné šablony pro diplomové práce na KGI. Povinnou přílohou práce bude poster formátu A2.

Rozsah pracovní zprávy:	**max. 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

CARMIGNIANI, Julie, Borko FURHT, Marco ANISETTI, Paolo CERAVOLO, Ernesto DAMIANI a Misa IVKOVIC, 2011. Augmented reality technologies, systems and applications. In: Multimedia Tools and Applications [online]. ISSN 13807501.

LINOWES, Jonathan; BABILINSKI, Krystian. Augmented reality for developers: Build practical augmented reality applications with unity, ARCore, ARKit, and Vuforia. Packt Publishing Ltd, 2017.

MA, Wei, Shuai ZHANG a Jincai HUANG, 2021. Mobile augmented reality based indoor map for improving geo-visualization. In: PeerJ Computer Science [online]. ISSN 23765992.

NGUYEN, Vinh T. a Tommy DANG, 2017. Setting up Virtual Reality and Augmented Reality Learning Environment in Unity. In: Adjunct Proceedings of the 2017 IEEE International Symposium on Mixed and Augmented Reality, ISMAR-Adjunct 2017 [online]. ISBN 9780769563275.

NOWACKI, Paweł a Marek WODA, 2020. Capabilities of ARCore and ARKit Platforms for AR/VR Applications. In: Advances in Intelligent Systems and Computing [online]. ISBN 9783030195007.

PARK, Youngmin, Vincent LEPETIT a Ntack WOO, 2008. Multiple 3D object tracking for augmented reality. In: Proceedings – 7th IEEE International Symposium on Mixed and Augmented Reality 2008, [online]. ISBN 9781424428403.

Vedoucí diplomové práce: **RNDr. Jan Brus, Ph.D.**
Katedra geoinformatiky

Datum zadání diplomové práce: **9. prosince 2021**
Termín odevzdání diplomové práce: **5. května 2023**

UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA GEOINFORMATIKY
17. listopadu 50, 771 46 Olomouc

L.S.

doc. RNDr. Martin Kubala, Ph.D.
děkan

prof. RNDr. Vít Voženílek, CSc.
vedoucí katedry

V Olomouci dne 16. prosince 2021

OBSAH

SEZNAM POUŽITÝCH ZKRATEK

Zkratka	Význam
3DEP	3D Elevation Program
6DOF	Six Degrees of Freedom
API	Application Programming Interface
AR	Augmented Reality
FOSS	Free and Open Source Software
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMD	Head-Mounted Display
HUD	Heads-Up Display
IDE	Integrated Development Environment
IL2CPP	Intermediate Language to C++
ISS	International Space Station
LiDAR	Light Detection and Ranging
LTS	Long-Term Support
MAR	Mobile Augmented Reality
MARS	Mobile Augmented Reality Systems
MR	Mixed Reality
NASA	National Aeronautics and Space Administration
OOP	Object-Oriented Programming
OpenGLES	OpenGL for Embedded Systems
PDA	Personal Digital Assistant
SDK	Software Development Kit
SLAM	Simultaneous Localisation and Mapping
UI	User Interface
URP	Universal Render Pipeline
USGS	United States Geological Survey
VR	Virtual Reality
WYSIWYG	What You See Is What You Get

ÚVOD

Technologie 3D tisku představuje zajímavý nástroj pro tvorbu prostorových objektů na bázi prostorových dat. Jedná se o relativně levnou výrobní metodu, schopnou vytvořit rozličné objekty, které mohou představovat například architektonické stavby z dob dávno minulých či 3D model ukazující charakter krajiny. Možnosti 3D tisku jsou omezeny jen dostupnými daty a z části také specifickými vlastnostmi 3D tiskárny a materiálu. Nespornou výhodou 3D tisku je fakt, že dokáží uživateli zprostředkovat haptický vjem. Ve své implicitní podobě jsou však vytištěné 3D modely poměrně nezajímavé z hlediska vzhledu.

Technologie rozšířené reality je technologií, která dokáže, ať už pomocí referenčních značek či na základě prostorových informací, zobrazit v reálné scéně virtuální obsah ve formě 3D objektů, animací, textur, audia a dalších. V dnešní době, zejména díky rychlému technologickému vývoji, se rozšířená realita přesunula ze specializovaných zařízení a specializovaných vědeckých pracovišť do běžného mobilního telefonu.

Tato diplomová práce si dává za cíl spojit přednosti technologie 3D tisku a rozšířené reality. Díky rozšířené realitě může být zmíněná nevýhoda 3D tisku – sice vizuální nezajímavost – alespoň do určité míry upozaděna. Rozšířená realita totiž dovoluje na vytištěném 3D modelu reliéfu, či jakémkoliv jiném modelu, zobrazit obsah, který jej jistým způsobem vylepšuje. Z jednobarevného 3D modelu se tak stává model z jistého pohledu interaktivní, se schopností pomocí rozšířené reality zobrazit kupříkladu vzhled objektu v reálných barvách či doplnit daný objekt o animace demonstrující přírodní jev. Dochází tak k využití výhod obou technologií. 3D tisk dodává haptický vjem, a do jisté míry i vizuální vjem. Ten je však vylepšen právě o obsah zobrazený pomocí rozšířené reality, a tak nabývá na atraktivnosti.

1 CÍLE PRÁCE

Cílem diplomové práce je vytvoření technického řešení schopného integrovat rozšířenou realitu a vytištěné 3D modely reliéfu. Modely jsou doplněny o textury, animace, text a další 3D objekty. Textury zobrazují i vybrané GIS analýzy pro jednotlivé modely. Doplňující obsah je připraven v softwaru Blender a zobrazen pomocí referenčních značek (markerů), na které se obsah uchytí. Za tímto účelem je vytvořena mobilní aplikace, pro jejíž vývoj je využit program Unity, který slouží pro vytváření různorodých aplikací a disponuje prostředky pro vývoj na více platformách, včetně systému Android. Každý z výsledných modelů reliéfu je logicky napojen na jeden marker, aplikace po detekci markeru zobrazuje několik scén pro každý specifický model. Dílčím cílem práce je vytvoření a optimalizace nejvhodnějšího tvaru a struktury markeru. Vytištěné modely a doprovodná aplikace obohacující modely o obsah mohou posloužit jako edukativní materiál demonstrující různorodé jevy a objekty či jako ukázka využití technologie rozšířené reality.

2 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Rozšířená realita (Augmented Reality, AR) znamená technologii, která kombinuje informace ve virtuální podobě s realitou. Principem AR je nasazení počítačově vygenerovaných informací, které mohou být v podobě například textu, videa či 3D modelu, do skutečného světa. Prezentované informace tak existují ve dvou úrovních, a sice ve virtuální a v reálné, které se navzájem doplňují (Chen et al., 2019). AR využívá množství technických prostředků a principů. Mezi stěžejní technologie patří zejména technologie inteligentního displeje (v angličtině Intelligent Display Technology), 3D registrace či inteligentní interakce. Za dobu své existence se AR postupně posunula od teoretického výzkumu v laboratorním prostředí až k hromadnému využití v různorodých oblastech lidské činnosti, jelikož figuruje jako prostředník mezi virtuálním a skutečným světem a nabízí tak uživateli nové způsoby poznávání a vnímání věcí kolem sebe.

Historie rozšířené reality sahá až do období 60. let 20. století, kdy americký výzkumník Ivan Sutherland přišel s myšlenkou displeje umístěného na hlavě (HMD, Head-Mounted Display). V roce 1968 vytvořil přístroj pro rozšířenou realitu, využívající opticky průhledný HMD (Carmigniani et al., 2011), který však z důvodu omezeného výpočetního výkonu byl schopný v reálném čase vykreslit a promítat pouze jednoduché geometrické útvary (Arth et al., 2015). Se zvyšujícím se výpočetním výkonem v 80. letech a na začátku 90. let 20. století se otevřel prostor pro to, aby se rozšířená realita stala samostatným předmětem výzkumu. Právě v tomto období experimentovali výzkumníci jako Myron Krueger, Dan Sandin či Scott Fisher s interakcemi mezi lidmi a prvky vygenerovanými pomocí počítačů (Hollerer a Schmalstieg, 2016). V roce 1992 byl představen prototyp zařízení, jehož cílem bylo napomáhat při výrobě letadel pomocí promítání informací na HUD (Heads-Up Display). Zařízení sloužilo k augmentaci skutečně dostupné informace pomocí počítačové grafiky, pro použitou technologii byl tak zaveden pojem Augmented Reality (Caudell a Mizell, 1992). Uvedené zařízení bylo schopno registrovat virtuální informace nezávisle na perspektivě uživatele s využitím principu 6DOF (Six Degrees of Freedom), bylo tedy možné kompenzovat změny umístění a orientace objektu. Ve zmíněné studii byl také uveden rozdíl mezi technologií rozšířené reality a virtuální reality (Virtual Reality, VR), který spočíval zejména v otázce výpočetního výkonu, a s tím souvisejícími možnostmi promítání objektů. V případě virtuální reality bylo nutné zpracovat každý pixel obrazu a s tím se zvyšovala výpočetní náročnost. V případě rozšířené reality se promítaly pouze výpočetně nenáročné objekty, navíc jen v části zorného pole uživatele, a bylo tak možné využít tehdejší běžné mikroprocesory.

Významným milníkem ve vývoji AR byl rok 1994, kdy bylo zavedeno tzv. virtuální kontinuum (anglicky Virtuality Continuum), které reprezentuje množinu objektů zobrazovaných pomocí displeje (Milgram a Kishino, 1994). Zjednodušená verze kontinua představuje smíšenou realitu (Mixed Reality, MR), která vůči sobě staví skutečný a virtuální svět. Reálná část kontinua obsahuje výhradně skutečné objekty, které mohou být pozorovány skrze displej. Virtuální část obsahuje výhradně objekty reprezentované počítačovou grafikou. Mezi reálným světem a virtuální reprezentací jsou umístěny virtuální a rozšířená realita, přičemž je uvedeno, že AR má blíže skutečnosti a VR má blíže k plně virtuálnímu zobrazení. V roce 1997 Ronald T. Azuma definoval stěžejní charakteristiky AR (Azuma, 1997). Ve své studii uvádí, že rozšířená realita pojí skutečné a virtuální, je interaktivní v reálném čase a je zpracována ve 3D. Uvedené charakteristiky se nepojí s žádnou specifickou technologií jako je HMD, ale zachovávají stěžejní principy AR. Azuma v uvedené studii také uvedl několik oblastí využití AR, mezi které patří medicína, armádní technika či zábavní průmysl. V roce 1999 byl představen ARToolKit, první open-source

software pro práci s AR (Hollerer a Schmalstieg, 2016). Stěžejní funkcí byla knihovna pro 3D sledování objektů pomocí černobílých referenčních značek.

Významným pojmem v oblasti AR je MARS (Mobile Augmented Reality Systems). Rozšířená realita dokáže promítat informace uživateli ve formě uživatelského rozhraní (UI), historicky však byla tato funkcionalita omezena pouze na pracoviště se specifickým vybavením. Koncept MARS dovoluje využít AR bez uvedeného omezení (Höllerer a Feiner, 2004). Zmíněná kapitola knihy uvádí základní komponenty potřebné pro fungování MARS systému, mezi které se řadí výpočetní platforma schopná vykreslit virtuální obsah (tedy v podstatě PC nebo ekvivalentní zařízení), displej pro zobrazení obsahu, schopnost registrace okolního obsahu za účelem zobrazení, přenositelné zařízení pro interakci s virtuálním obsahem, bezdrátová síť pro komunikaci s ostatními zařízeními a také technologie pro ukládání dat a přístup k uloženým datům. V uvedených komponentech lze sledovat určité paralely s funkcionalitou nabízenou dnes běžně dostupným mobilním telefonem. Jedním z případů využití principů mobilní rozšířené reality je prototyp sloužící k navigaci po univerzitním kampusu s využitím přenositelného počítače ve formě batohu (tzv. wearable computer) a HMD (Feiner et al., 1997). Ve studii představený prototyp využíval americký GNSS (Global Navigation Satellite System) systém GPS (Global Positioning System) pro sledování polohy uživatele v prostoru a magnetometr a sklonoměr ke sledování orientace v prostoru. Přenositelností AR do venkovního prostředí s využitím přenositelných systémů a navigace pomocí GPS se zabývali také Thomas et al. (1998).

Na snahy výzkumníků v poli mobilních AR systémů navázala v roce 2000 aplikace ARQuake, AR modifikace videohry Quake z roku 1996, využívající knihovnu ARToolKit pro umisťování virtuálních objektů do dané scény a GPS pro určení pozice a orientace uživatele (Thomas et al., 2000). Aplikace využívala přenositelný počítač Tinmith, obsahující množství knihoven a modulů (např. knihovnu pro převod souřadnic či síťovou komunikaci) pro implementaci AR funkcionality. Cílem aplikace bylo přenést intuitivní povahu uživatelského rozhraní nabízeného AR a VR systémy do formy AR videohry, kterou uživatel může provozovat nezávisle na své lokalitě. Dalším významným milníkem v poli rozšířené reality byl rok 2003, kdy vznikl první samostatně fungující (bez nutnosti využití wearable hardwaru) AR systém využívající PDA (Personal Digital Assistant) k implementaci 3D aplikace pro navigaci v budově (Wagner a Schmalstieg, 2003). Jak je uvedeno ve zmíněné studii, wearable hardware do značné míry limitoval možnost pohybu uživatele a bránil tak praktickému využití. Další vývoj tedy logicky směřoval k využití kompaktních zařízení v oblasti mobilní AR. Využité zařízení bylo rozšířeno o optickou kameru, která sloužila pro sledování objektů. Výsledná aplikace sloužila k navigaci uživatele pomocí zvýrazňování východů a jiných orientačních prvků či zobrazení síťového modelu budovy na displeji zařízení. Studie poukazuje také na výpočetní náročnost sledování objektů, kterou lze významně ovlivnit přesunutím části zátěže na server. Na některá úskalí mobilních telefonů, jako právě nedostatečný výpočetní výkon oproti stolním počítačům, poukazují Wagner a Schmalstieg (2009). Zmíněna je také v té době nedostatečná schopnost registrace přirozených objektů.

V současné době, díky rozvoji možností a funkcí mobilních zařízení, jsou MAR (Mobile Augmented Reality) aplikace stále více dosažitelnější (Chatzopoulos et al., 2017). Mobilní telefony, přestože stále nenabízí výkon ekvivalentní stolním počítačům, disponují množstvím prvků, které usnadňují vývoj MAR aplikací. Navíc lze využít technologie jako cloud či strojové učení pro rozdělení výpočetních úkonů. Zmíněná studie uvádí, že MAR aplikace mají tři základní komponenty: vstup, zpracování a výstup. Vstupem je myšlena schopnost využití různých senzorů zařízení, jakými jsou kamera či gyroskop. Zpracování je schopnost zpracování informace, která bude vykreslena na displeji zařízení. Výstup pak

znamená zobrazení informací na displeji, resp. rozšíření okolních reálných objektů o virtuální informaci. Jak je zjevné z uvedených informací, rozšířená realita prošla významným historickým vývojem, od počátků v laboratořích se specializovaným vybavením, přes přenositelné počítače až po současnou dobu, kdy AR funkcionalitu dokáže obstarat mobilní telefon.

2.1 Formy rozšířené reality

Technologii rozšířené reality lze využít na velkém množství zařízení, mezi něž patří chytré telefony, tablety či brýle Hololens. Kromě rozdílných zařízení se AR vyskytuje v rozdílných formách (Softtek, 2021). Mezi hlavní formy se řadí AR využívající značky pro umístění obsahu (Marker-based AR) a AR využívající přirozeně se vyskytující objekty (Markerless AR). Následující odstavce budou věnovány bližšímu představení jednotlivých forem AR.

2.1.1 Marker-Based AR

Jedná se o přístup k AR využívající rozpoznávání určitého vzoru v podobě umístěné značky (marker) za účelem přenesení virtuálního 3D objektu či animace do skutečného prostředí (Sinha, 2021). Rozpoznávání markeru probíhá pomocí kamery mobilního telefonu či podobného zařízení, která neustále skenuje objekty v blízkém okolí. V případě, že kamera není zaměřena na specifické místo, či marker, obsah spojený s daným markerem se nevykreslí. Systém rozpoznávání obrazu pro potřeby Marker-Based AR má, mimo nutnosti využití kamery, více částí nutných ke správnému fungování, jako je snímání obrazu, zpracování obrazu, vykreslení obsahu a sledování markerů. Výhodou Marker-Based AR je snadná implementace a nízké náklady. Tvar objektů sloužících jako markery musí být jasně vymezen, tak aby bylo možné je rozpoznat nezávisle na okolních objektech (Aircards, 2021). Mezi běžně využívané markery patří QR kódy či loga společností.

Implementaci Marker-Based AR se věnovali např. Gherghina et al. (2013). Ve studii poukázali na fakt, že chytré mobilní telefony jsou nejlépe cenově dostupná zařízení pro rozšířenou realitu, nesou s sebou však určitá úskalí, zejména spojené s dostupným výpočetním výkonem. Poukázáno bylo také na nevýhodu QR kódů, kterou je pomalé rozpoznávání obrazu a malé rozpoznávací úhly. Výsledkem studie byla aplikace pro dynamické zobrazení obsahu na základě rozpoznání QR kódů, využívající architekturu klient-server pro přesunutí části úkolů aplikace (ukládání a stahování obsahu) na stranu serveru.

2.1.2 Markery pro rozšířenou realitu

Markery jsou referenční značky pro umístění obsahu, které využívá Marker-Based AR. Mohou být rozličných tvarů a charakteristik. Jedná se o obrázky nebo objekty, které mohou být rozpoznány pomocí mobilní aplikace s AR funkcionalitou a použity pro zobrazení prvků v rámci rozšířené reality (Klavins, 2021). Markery by měly být umístěny na rovných plochách, jelikož při umístění na křivé a nepravidelné povrchy dojde k deformaci markeru (a tedy zhoršení schopnosti rozpoznat daný marker). Z technického hlediska (z pohledu computer vision) funguje rozpoznávání markerů na základě detekce referenčních bodů (tzv. key points nebo feature points). Čím více daný marker obsahuje referenčních bodů, tím více bude zobrazený AR obsah stabilnější. Důležitou vlastností pro dostatek referenčních bodů je unikátnost vzorů obsažených v daném markeru.

Specifickým typem markeru, používaného pro potřeby rozšířené reality, je ArUco marker. Jedná se o marker čtvercového tvaru s černými okraji a binární maticí, jež hraje roli určujícího identifikátoru pro daný marker (OpenCV, 2023). Díky černému okraji

markeru je možné jej rychle detekovat a binární vzor uvnitř markeru umožňuje samotnou detekci. ArUco markery také nabízí detekci chyb a opravu chyb.

Ačkoliv zmíněná funkcionalita je zjevně využitelná jen samotnou OpenCV knihovnou a nikoliv v práci využitou knihovnou AR Foundation, vlastnosti jako snadná detekce markeru a charakteristický vzor by měly být univerzální napříč implementacemi AR. Právě ArUco markery byly ve finále využity pro vytvořenou mobilní aplikaci v rámci diplomové práce, vytvořeny pomocí nástroje pro jejich snadné generování.

2.1.3 Markerless AR

Na rozdíl od přístupu, který byl zmíněn v podkapitole 2.1.1, Markerless AR, jak vypovídá z názvu, nevyužívá rozpoznávání markeru pro umístění objektu do scény, ale pracuje se skutečným prostředím, do kterého umisťuje objekty na základě výpočtů z dat v reálném čase (Sinha, 2021). Výpočty probíhají s využitím hardwaru běžného mobilního telefonu, jako je kamera, GPS, kompas či akcelerometr, které napomáhají efektivitě výpočtů a umístění virtuálního objektu do scény. Markerless AR využívá technologii SLAM (Simultaneous Localisation And Mapping), která funguje na principu skenování okolního prostředí za účelem tvorby map (nejedná se o mapy v kartografickém slova smyslu), které slouží jako podklad pro umisťování 3D objektů.

Zmíněná technologie je tak schopna detekovat objekty či charakteristiky daného prostředí, a to i v případě, že se jedná o prostředí neznámé (Softtek, 2021). Výhodou Markerless AR je zvýšení rozsahu pohybu uživatele, který je schopna tato forma poskytnout, jelikož není vázána na markery umístěné na specifických místech. Funkcionalita Markerless AR je implementována do frameworků ARKit a ARCore SDK (Software Development Kit). Využitím Markerless AR se zabývali Ufkes a Fiala (2013). Ve zmíněné studii definovali mapu pro potřeby AR jako seznam deskriptorů prvků a jejich odpovídající 3D souřadnice. Výsledkem studie byl pracovní postup využívající algoritmy pro dekódování obrazu, detekci prvků či sledování objektů za účelem aktualizace obsahu mapy v reálném čase a následně vykreslení objektů v prostoru.

Markerless AR rozlišuje několik dalších kategorií. Mezi nejpodstatnější kategorie patří AR pracující s geografickou informací (Location-Based AR), nejběžněji s polohou uživatele, pro potřeby umisťování obsahu (Softtek, 2021). Příkladem aplikace využívající přístup Location-Based AR je Pokémon GO. Dalším významným druhem je AR založená na promítání obsahu ve vymezeném prostředí (Projection-Based AR). Tato implementace AR tak dovoluje uživateli volně se pohybovat v předem dané oblasti, kde jsou umístěny kamery, které vykreslují daný obsah v reálném čase.

2.1.4 Web AR

Jedná se o poměrně nový způsob využití rozšířené reality, který není vázán na dedikované aplikace, jako tomu bylo v případě 2.1.1 a 2.1.3. V této formě je obsah uživatelům dostupný přímo, pouze za použití webového prohlížeče (Rock Paper Reality, 2021). Stejně jako ostatní formy, Web AR využívá různé senzory mobilních zařízení k umístění virtuálního obsahu do skutečného prostředí. Výhodou je snadná přístupnost AR obsahu. Podrobněji se aspektům Web AR věnovali Qiao et al. (2019). V uvedené studii poukazují na stále přítomnou výpočetní náročnost AR aplikací a tedy jedno z úskalí Web AR. Web jako platforma má totiž omezené možnosti výpočetních operací a vykreslování obsahu, je tak vhodné přesunout část zátěže (např. využitím cloudu). Další nevýhodou Web AR může být nedostatečně rychlá síťová komunikace či nedostatečná kapacita baterií mobilních telefonů. V neposlední řadě může být problémem diverzita využívaných zařízení. V dnešní době existují webové

technologie podporující požadavky Web AR. Jednou z takových technologií je WebRTC, která poskytuje webovým prohlížečům možnost výměny informací v reálném čase, což je pro Web AR důležitým aspektem. Podstatnou technologií je také WebAssembly, sloužící pro zrychlení výpočetních operací v prostředí webu. Mezi další důležité technologie patří tzv. Web Workers, přinášející možnost paralelizace webových procesů, a WebGL, napomáhající operacím vykreslování obsahu na webu.

2.2 Možnosti tvorby aplikací pro rozšířenou realitu

S rozvojem rozšířené reality se pojí také množství sad nástrojů, které lze využít pro tvorbu AR aplikací. Mezi hlavní představitele dostupných nástrojů patří sady nástrojů ARCore SDK a ARKit. Dále existuje množství nástrojů, zejména v prostředí Unity, které nabízejí podpůrné funkce pro implementaci AR. Následující odstavce blíže představí a srovnají jednotlivé nástroje.

2.2.1 ARCore SDK

Jedná se o sadu nástrojů (SDK) od firmy Google, sloužící ke tvorbě aplikací pro rozšířenou realitu. ARCore využívá množství API (Application Programming Interace) pro porozumění informacím a předávání těchto informací mezi mobilním telefonem a skutečným prostředím (Google, 2022). Mezi základní složky ARCore, starající se o schopnost umístění virtuálních objektů do skutečného prostředí, patří schopnost sledování pohybu, schopnost rozpoznání skutečného prostředí či schopnost odhadnout světelné podmínky. Technologie ARCore je schopna detekovat body zájmu, které, v kombinaci se senzory telefonu, slouží k určení umístění a orientace zařízení. Další stěžejní funkcí je schopnost rozpoznat některé povrchy a také odhadnout okolní osvětlení. ARCore tak v podstatě sestavuje obraz světa, do kterého jsou poté umístěny objekty, anotace či další doplňující informace. ARCore SDK nabízí API pro Unity i Unreal Engine. Výhodou ARCore je podpora zařízení jak se systémem Android, tak se systémem iOS.

2.2.2 ARKit

Alternativou k ARCore SDK je sada nástrojů ARKit, shodně zaměřená na tvorbu aplikací pro účely rozšířené reality, avšak limitovaná pouze na zařízení se systémem iOS. Oproti ARCore nabízí pokročilejší funkcionalitu (Apple, 2023). Současně nejnovější plánovaná verze SDK, ARKit 6 nabízí např. snímaní pohybu člověka v reálném čase (tzv. Motion Capture), simultánní využití přední i zadní kamery zařízení a s tím spojenou možnost využití sledování více objektů (specificky snímaní obličeje a zároveň okolního prostředí). Mezi další funkcionality patří schopnost vyhodnocení hloubky reálné scény pomocí zabudovaného LiDAR (Light Detection and Ranging) skeneru a v důsledku, v kombinaci s vytvořením topologie scény, tak schopnost poskytnout přesnější umisťování virtuálních objektů do reálné scény. Díky LiDAR skeneru je také významně urychlena detekce roviny pro umístění obsahu (tzv. Plane Detection) a tedy i samotná rychlost umístění obsahu. Stejně jako ARCore je možné ARKit využít v prostředí Unity i Unreal Engine.

2.2.3 Unity

Jedná se o multiplatformní herní engine, jehož první verze byla vydána v roce 2005. V základní verzi je možné jej využívat zcela zdarma. Multiplatformita znamená, že v Unity lze vyvíjet projekty určené jak pro systémy Windows, Linux či Mac, tak i pro systémy, které jsou vyhrazeny pro mobilní zařízení (Android, iOS). Lze také vyvíjet aplikace pro web, využívající WebGL API. Obsažena je i podpora pro vývoj na herní konzole, avšak nikoliv ve

verzi zdarma. V prostředí Unity lze tvořit scény obsahující 3D i 2D grafiku a využít tzv. assety (různé objekty či funkcionalita, kterou je možno ve scéně zobrazit, např. kolekce 3D objektů a textur či různé frameworky, jako je rozšíření fyziky objektů či již předpřipravené ovládání). Unity využívá principy OOP (objektově orientované programování), logiku chování objektů lze programově určit s využitím programovacího jazyka C#. V neposlední řadě, v Unity lze využít množství frameworků či nástrojů právě pro podporu tvorby aplikací využívajících principy rozšířené reality. Unity lze získat v několika verzích, současně nejnovější LTS (Long Term Support) verzí je 2021.3.17f1 (Unity Technologies, 2023d).

2.2.4 AR Foundation

Framework pro vývoj aplikací pro rozšířenou realitu, který se, stejně jako Unity, vyznačuje multiplatformitou, jelikož je možné vyvíjet aplikace s obsažením funkcionality ARCore i ARKit, ale mimo to je obsažena i možnost přizpůsobení na wearable hardware, jako Magic Leap či HoloLens. Pro vývoj lze také využít přímo předpřipravenou funkcionalitu Unity, jako např. URP (Universal Render Pipeline) (Unity Technologies, 2023b). AR Foundation obsahuje množství tzv. manažerů (manager v angličtině), jako jsou AR Tracked Object Manager, AR Tracked Image Manager či AR Plane Manager. Pro zpřístupnění funkcionality je nutné v prostředí Unity doinstalovat zásuvné moduly (tzv. plug-in) korespondující s danou platformou (ARCore, ARKit, OpenXR) (Unity Technologies, 2022). Většina funkcionality je dostupná jak pro ARCore, tak ARKit, zajímavou funkcionalitou navíc pro ARKit je schopnost sledování objektů (Object Tracking) a také lidského těla (Body Tracking). Funkcionalita OpenXR, který využívá HoloLens, je velmi limitována. V současnosti nejnovější verzí frameworku je AR Foundation 5.0.3.

2.2.5 Vuforia

Jedná se o platformu pro vývoj AR a MR aplikací, stejně jako AR Foundation podporuje vývoj využívající funkcionalitu ARCore a ARkit a vývoj uzpůsobený na wearable hardware. Vuforia využívá tzv. drag and drop worklow (Unity Technologies, 2017). Platforma pracuje s běžnými formami AR, jako Marker-Based a Markerless AR. Je dostupná v rámci Unity Asset Store (digitální obchod s množstvím zdarma dostupných i placených 3D objektů, frameworků, textur, nástrojů, aj.). Vuforia nabízí množství řešení pro sledování objektů, které lze rozdělit do třech kategorií: snímky, objekty a prostředí (Vuforia Developer Library, 2021). Do první kategorie spadá, kromě klasické detekce jednoho referenčního snímku na rovné ploše a umístění odpovídajícího obsahu, např. schopnost pracovat v reálném čase s rozsáhlou datovou sadou snímků a tu postupně rozšiřovat. Mimo to lze umisťovat obsah i na takové referenční snímky, které jsou uspořádány do nepravidelného (př. válcového) tvaru. Specifickou formou první kategorie jsou pak VuMarks, což jsou přizpůsobitelné markery, do kterých je možno zakódovat množství datových formátů (př. URL adresa webu). Do druhé kategorie spadá možnost rozeznat objekt umístěný v reálné scéně podle existující 3D reference. Příkladem může být model automobilu a umístění obsahu na daný automobil ve scéně. Sledování objektů v prostředí pak plní funkce augmentace prostředí, které tak mohou obsahovat prvky navigace či jiné prostorové instrukce (kapacita baterie u přístroje, teplota, aj.).

2.2.6 Unity MARS

Sada nástrojů pro tvorbu aplikaci rozšířené reality, která usnadňuje vývoj a rozšiřuje funkcionalitu množstvím nástrojů. Shodně, jako Vuforia a AR Foundation, i Unity MARS je multiplatformní, s podporou systémů Android, iOS i wearable hardwaru (Unity Technologies, 2023a). Mezi hlavní rysy Unity MARS patří schopnost uživatele definovat pravidla a rozložení AR obsahu pomocí přirozeného jazyka, tj. bez nutnosti programovacího jazyka. Další významnou funkcionalitou je využití zástupných objektů a definice jejich chování. V případě, kdy pak aplikace detekuje daný reálný objekt, zareaguje na něj podle předdefinovaných pravidel. V neposlední řadě lze díky Unity MARS testovat AR aplikace přímo v editoru Unity, bez nutnosti aplikaci nejdříve zkompilovat a poté testovat v prostředí mobilního telefonu či jiného korespondujícího zařízení. Se zmíněným souvisí také fakt, že se jedná o WYSIWYG (What You See Is What You Get) systém, lze tak přímo nasimulovat, jak se bude aplikace chovat v reálné scéně (Chacko, 2020).

2.2.7 Unreal Engine

Stejně jako v případě Unity, i Unreal Engine je multiplatformním herním enginem. Lze tak vyvíjet projekty pro velké množství platforem, od Microsoft Windows právě až po mobilní operační systémy. První verze enginu vznikla již v roce 1998, v současnosti nejnovější verzí je Unreal Engine 5.1. Stejně jako v případě Unity je i zde možné programově určit logiku chování objektů, avšak primárně využívá programovací jazyk C++. Zajímavostí je možnost využití systému Blueprint Visual Scripting, který je založen na systému propojených uzlů. Jednotlivé uzly představují jednotlivé složky a procesy ve scéně (Epic Games, 2022b). Samozřejmostí je možnost vyvíjet AR a VR aplikace. Unreal Engine podporuje Andoid i iOS systémy a jejich korespondující sady nástrojů (Epic Games, 2022a). Pro tvorbu AR aplikací využívá vlastní API, které rozšiřuje jazyk C++ o novou knihovnu a systém Blueprint o korespondující funkce. Výhodou je využití stejného API, resp. knihovny, pro vývoj na obě dostupné platformy.

2.3 Role rozšířené reality ve vzdělávání

Rozšířená realita dokáže rozšířit informace, které uživatel dostává z reálného světa. Tohoto faktu může být využito pro marketing (lze si zobrazit např. nábytek v reálném prostoru před zakoupením), navigaci (zobrazení směrových šipek a dalších ukazatelů v prostoru) či právě vzdělávání, např. formou zobrazení doplňujících informací o určitém objektu přírodního nebo jiného charakteru. Následující odstavce shrnují možnosti rozšířené reality pro potřeby vzdělávání.

Příkladem využití technologie rozšířené a virtuální reality pro vzdělávání je tvorba frameworku pro prezentaci obsahu na základě interaktivní aplikace v enginu Unity, která slouží studentům k porozumění problematiky pomocí prostředí, které je obklopuje (Nguyen a Dang, 2017). Studie popisuje tři hlavní části návrhu AR a VR prostředí pro reprezentace dat, analýzy či implementaci řešení daného problému. První částí je samotný vytvořený framework, druhou pak integrace komponentů do enginu Unity. Ve studii jsou také popsány technické výzvy řešení. Aplikace, která framework obsluhuje, využívá software Vuforia a VR zařízení Google Cardboard. Výsledná aplikace funguje na principu kombinace VR a AR technologie. Pro vytvoření objektů ve virtuální realitě využívá uživatel objektů, které sesbírá v prostředí AR pomocí kamery mobilního telefonu. Hlavní myšlenkou aplikace je vizualizace možných dopadů lidské činnosti na chování povodí řeky a jejich různé implikace.

Výhodám a aplikacím rozšířené reality pro vzdělávání se věnovali Saidin et al. (2015). Ve studii popisují, že technologie ve vzdělání obecně může být nápomocna pro aktivní učení se a motivování studentů. AR proces učení pak učiní zajímavějším skrze schopnost interagovat virtuálním obsahem a také schopnost pohlcení uživatele v realistickém zážitku. Studie popisuje některé výhody AR technologie pro vzdělávání, jako interakci mezi reálným a virtuálním prostředím a schopnost navrhovat učební materiál tak, aby bylo možné se učit mimo výuku. Je zmíněna také schopnost AR přímého zapojení studentů do procesu učení a s tím spojené zlepšení schopností vizualizace.

Rozšířená (a virtuální) realita zprostředkovává, díky technologickým pokrokům více než kdy dřív, nové typy učení, které jsou vhodné pro studenty 21. století, kteří očekávají interaktivitu, participaci a manipulaci s objekty (Elmqaddem, 2019). Studie předkládá, že z pohledu VR je pro edukaci důležité, že lze s virtuálními objekty manipulovat pomocí ovladačů, což podporuje učení pomocí interakce. VR umožnuje využití různých simulátorů (které mohou mít využití např. v medicíně) či návštěvu muzea ve virtuální podobě. Naopak rozšířená realita nabízí edukaci pomocí virtuálního obsahu, který překrývá reálnou scénu. K synchronizaci reálné a virtuální informace dochází pomocí geolokalizace a senzorů zařízení. Specifickým využitím AR pro vzdělávání je např. učení se nových postupů v reálných podmínkách, s pomocí instrukcí promítaných v reálném čase. Dalším využitím je objevování historie objektu či místa, a to jen za pomoci kamery chytrého telefonu. Dle zmíněné studie mohou být výhody AR pro vzdělávání shrnuty následovně: Z pohledu studentů AR zvyšuje motivaci a zájem k dané problematice, nabízí lepší příležitosti k pokládání otázek, zvyšuje úroveň interakce mezi studenty či konkretizuje abstraktní koncepty. Z pohledu učitele pak AR podporuje kreativitu studentů a tím zvyšuje úroveň jejich aktivní participace v dané problematice.

Roli rozšířené reality ve vzdělávání se věnoval i Lee (2012). Ve studii popisuje roli AR pro případ K-12 vzdělávání (americký systém vzdělání, který zahrnuje roky od předškolního věku až do věku 17–18 let) a také pro případ vysokoškolského vzdělávání. Zmíněna je efektivita AR při pochopení složitých teorií či strojových mechanismů. Dále je zmíněn přehled oborů, ve kterých se dá AR využít, a to z pohledu jak edukace, tak z pohledu komerční sféry. Z pohledu edukace je zajímavým příkladem využití AR pro zobrazení planety Země a Slunce, což je zahrnuto v kategorii astronomie (ale vzhledem k přesahům zejména do planetární geografie zde lze vidět i využití při výuce geografie). Dalšími obory jsou matematika či fyzika. Z pohledu komerční sféry je zmíněna role AR pro oblast kulturního dědictví (Cultural Heritage), kde AR může posloužit pro zobrazení historických informací (např. jak vypadala daná budova napříč historií). Zmíněna je i role z pohledu muzejních zážitků. V neposlední řadě studie shrnuje výhody AR pro edukaci, jako je schopnost učinit proces učení více atraktivním, jednoduchost, využití kontextových informací nebo efektivita vzdělávání.

Dále se vzdělávání pomocí AR věnovali také Wu et al. (2013). Ve studii popisují tři základní přístupy k učení s využitím rozšířené reality, a sice role, lokace a úkoly. První přístup (role) se vyznačuje důrazem na to, aby participantu učícího se procesu zaujmuli různé role v AR prostředí, např. v parcipativních simulacích. Druhý přístup dává důraz na interakci studenta se skutečným prostředím, doprovozenou o AR. Poslední přístup se vyznačuje na plnění úkolů v prostředí AR. Studie zmiňuje také některé problémy rozšířené reality z hlediska pedagogického a z hlediska učení se. Z pedagogického hlediska hrozí, stejně jako u většiny nových technologií, odpor ze strany samotných vzdělávacích institucí a učitelů. Z pohledu učení se pak studentům hrozí přetížení kognitivních funkcí z důvodu přemíry informací v prostředí AR.

Využití AR pro vzdělávání z pohledu geografických aplikací se věnovali např. Chitaniuc a Iftene (2018), kteří vytvořili aplikaci v enginu Unity, s využitím knihovny Vuforia. Úkolem aplikace bylo atraktivní cestou zprostředkovat učení se o geografii Evropy, včetně zemí, hlavních měst či vlajek států. Vyvinutá aplikace podporuje zapamatování informací pomocí rozšířené reality. Dalším využitím AR při vzdělávání z pohledu geografie se zabývali i Adedokun-Shittu et al. (2020) nebo Hong et al. (2022).

2.4 Geovizualizace s využitím rozšířené reality

Z principu fungování rozšířené reality vyplývá schopnost, alespoň virtuálně, zobrazit rozličné množství objektů. Takovými objekty mohou být například různé architektonické stavby, které zachycují vzhled určité budovy, a to ať už soudobý nebo historický. Lze také zobrazit krajinu, například s texturou povrchu planety, a udělat si tak představu o tom, jaký je svět z výšky, vše jen za pomocí mobilního telefonu. Rozšířená realita může také zobrazit informace využitelné při navigaci v neznámém prostředí, a to formou šipek či různých dalších prostorových ukazatelů. Rozšířená realita má však jednu značnou nevýhodu, a sice fakt, že virtuálně zobrazený obsah neposkytuje uživatelům haptický vjem. Z toho důvodu je zajímavým konceptem zobrazení virtuálního obsahu na vytištěném 3D modelu, který zprostředkovává, mimo haptickou odezvu (lze si na model sáhnout), také další vrstvu vizuální odezvy. Následující odstavce popisují možnosti rozšířené reality z pohledu různých geovizualizací.

Velmi podobnou problematikou jako tato práce se zabývala diplomová práce Petra Mužíčka (Mužíček, 2021). V práci se zaměřil na tvorbu aplikace pro systém Android s využitím technologie rozšířené reality a enginu Unity. Vytvořená aplikace obsahovala tři hlavní scény, a sice scénu MARKER, pracující na principu detekce markerů a zobrazení obsahu, scénu PLANE, pracující na principu detekce virtuálních rovin a umístění obsahu, a scénu EXTERIÉR, využívající principy Location-Based AR.

Klasifikaci geografických vizualizací s použitím rozšířené reality představují Lobo a Christophe (2020). Data se dle studie dělí nejprve na fyzická a abstraktní, kdy pod pojmem fyzická data si lze představit data s jasnou fyzickou reprezentací jako budovy, jezera nebo cesty. Abstraktní data pak jsou taková, která nemají fyzickou reprezentaci, jako jsou statistická, tematická nebo kvalitativní. Obě skupiny dat se dále dělí na geograficky umístěné (geographically situated) a geograficky vzdálené (geographically distant) (pozn.: Jedná se o vlastní překlad autora). Ještě podrobnějším dělením je pak v případě fyzických geograficky umístěných dat dělení na data historická, současná či plánovaná. V případě fyzických geograficky vzdálených dat se data dělí na historická a současná. Co se týče abstraktních geograficky umístěných dat, ty se dělí na současná a plánovaná data. Pro poslední skupinu poté existuje identické dělení. Z pohledu představené klasifikace tak v této diplomové práci využitý postup kombinace AR a 3D vytištěných modelů představuje primárně fyzická geograficky vzdálená současná data (která jsou představena daným vytištěným 3D modelem). Některé geovizualizace v rámci práce však ukazují historická data či tematická data v podobě barevné hypsometrie či sklonu svahu.

Specifickým příkladem využití technologie AR pro geovizualizace je vytvoření AR aplikace zobrazující fotorealistické 3D geovizualizace (Papadopoulou et al., 2020). V rámci studie byla vytvořena aplikace s pomocí enginu Unity a knihovny Vuforia, která zobrazuje AR 3D geovizualizaci s použitím dat ortofotomapy a digitálního modelu povrchu. Hlavní devízou vytvořené aplikace je schopnost výběru zobrazených informací a také možnost přepínání obsahu mezi třemi různými obdobími pomocí časové osy. Dalšími příklady

využití AR pro geovizualizace je metoda založená na Markerless AR, sloužící pro registraci, geovizualizaci, a interakci ve venkovních prostorách, s využitím algoritmů strojového učení (Rao et al., 2017) nebo využití AR pro potřeby urbánních geovizualizací pomocí nástrojů schopných např. odstranit část budovy nebo nahradit celou budovu, to vše na původním místě (in situ) (Devaux et al., 2018).

Zajímavou studii v oblasti geovizualizací využívajících rozšířenou realitu představili také Zhang et al. (2020). Uvedená studie se zaměřuje na integraci 3D fyzického modelu terénu a virtuálních dat o povodni. Za tímto účelem je ve studii navržena inovativní metoda FARV3DPT (Flood AR Visualization 3D Physical Terrain), která funguje na principu překrytí 3D modelu virtuální scénou povodně a s tím spojeno zobrazení vztahu mezi povodní a reliéfem oblasti. Pro zobrazení AR obsahu byl využit černobílý marker, umístěný u vytištěného modelu. Z technického hlediska byla vytvořena aplikace v enginu Unity, využívající knihovnu Vuforia. Využitím hardwarem však nebyl mobilní telefon. Namísto toho byly využity brýle Microsoft HoloLens, tedy typově se jednalo o HMD AR systém. Výsledná aplikace dokázala pomocí markeru překrýt 3D model AR obsahem a kompenzovat i pohled z jiné strany.

Ve zmíněné studii lze najít mnoho paralel s touto diplomovou prací. Shodně zde existuje snaha překrytí 3D vytištěných modelů obsahem vyvolaným s použitím rozšířené reality. Výhodou zmíněné studie je využití HMD AR systému, díky němuž má uživatel volné ruce a může veškerou pozornost věnovat zobrazenému obsahu. Naproti tomu, běžná implementace AR spoléhá na mobilní telefon či jiné zařízení se způsobilostmi AR. Nevýhodou ve studii představeného řešení je ale právě použití HMD AR systému. Microsoft Hololens není zařízením dostupným pro běžného uživatele, z pohledu ceny (ačkoliv studie zmiňuje, že existuje možnost adaptace i na jiné HMD systémy – Magic Leap, Cardboard AR). Z uživatelského pohledu je tak do jisté míry příjemnější AR aplikace, která není vázána na specifický hardware (i když vázána na specifický operační systém). Výhodou aplikace vytvořené v této diplomové práci je také zobrazení více jevů, nikoliv zaměření pouze na jeden jev, což ji dělá více univerzální.

3 METODY A POSTUP ZPRACOVÁNÍ

Práce započala akvizicí vhodných dat. Již od koncepční fáze byla práce zamýšlena pro celkem čtyři případové studie. Modely pro jednotlivé studie byly vybrány tak, aby mohly být vhodně rozšířeny o obsah v prostředí rozšířené reality a ukazovaly tak právě integraci dvou přístupů, tedy vytištěného 3D modelu a rozšířené reality. První vybranou oblastí byla Hooverova přehrada (Hoover Dam) nacházející se na hranicích států Nevada a Arizona v USA. Druhou vybranou oblastí se stal stratovulkán Mount. St. Helens, ležící ve státě Washington v USA. Třetí studie se zaměřila na planetu Zemi, která byla znázorněna pomocí globu. Poslední vybranou oblastí se pak stal hrad Trosky v okrese Semily v Libereckém kraji. Po výběru vhodných oblastí bylo nutné pořídit vhodná data. Zdroje dat se lišily napříč modely a jsou podrobněji rozepsány v podkapitole 3.2.

Následným krokem bylo předzpracování získaných dat. Zde se taktéž postup mírně lišil v závislosti na modelu. Pro první studii bylo nejprve nutné přesněji vymezit zájmovou oblast. Poté byly určeny souřadnice vrcholů zájmové oblasti, využité později pro další krok. Přesněji vymezená oblast byla vyexportována ve formátu TIF a nahrána do prostředí programu Blender, ve kterém z dat vznikla 3D mesh (síť vrcholů tvořící souvislý 3D objekt). Vytvořená mesh byla poté uložena do formátu STL a zjednodušena. Následně byly modelu přiřazeny textury a upravena UV mapa (rozložená síť 3D modelu do 2D plochy), načež byl model uložen do formátu FBX pro další práci. V prostředí programu Blender byly také připravena doprovodná data v podobě dalších 3D objektů. Analogický postup platil i v případě druhé studie, jelikož byla využita obdobná zdrojová data. Pro případ třetí studie byla data obstarána zakoupením již předpřipraveného 3D modelu Země, který měl připravenou i UV mapu a základní texturu povrchu. Taktéž byl model již ve formátu FBX, v základu tak byl připraven pro další práci. Vlastní práce spočívala v přípravě jednotlivých scén pro model, což je blíže rozepsáno v podkapitole 4.3. Čtvrtá a poslední studie byla zpracována obdobným způsobem jako první a druhá studie. Pro všechny studie byly také připraveny STL soubory pro 3D tisk.

S připravenými daty přišel na řadu další krok, a sice prvotní prototypy mobilní aplikace v prostředí Unity, předcházející samotnému vývoji. V rámci vytvořených prototypů bylo vyzkoušeno několik postupů práce s daty v prostředí Unity a jejich správné načtení a zobrazení ve scéně na základě přečtení referenční plochy (tzv. marker) kamerou telefonu. V úvodních prototypech bylo pracováno s pouze s jednoduchými geometrickými tvary. Jelikož aplikace měla být navržena pro práci s celkem čtyřmi modely, prototypy se zabývaly i otázkou zobrazení více virtuálního obsahu (zobrazeného pomocí rozšířené reality) v jedné scéně, přepínáním mezi scénami či výběrem vhodných markerů. Pozornost byla věnována zejména datovým strukturám pro uchování dat a také možnostem nastavení offsetu zobrazeného obsahu, což bylo pro aplikaci zcela stěžejní. Z datových struktur pro uložení dat byl vyzkoušen slovník, pracující na principu klíč – hodnota nebo pole, do kterého se uložily všechny objekty, které mohly být součástí scény. Vyzkoušen byl také přístup, který využíval pouze funkcionalitu editoru bez rozšíření pomocí vlastních skriptů. Tento přístup se však ukázal být nedostatečným. Některá zjištění v rámci vytvořených prototypů byla využita při vývoji aplikace.

Následoval samotný vývoj aplikace. Nejprve bylo nutné správně nastavit prostředí Unity, za tímto účelem byla využita šablona pro vývoj AR aplikací, která obsahuje množství důležitých nastavení. Vývoj se skládal z několika dílčích problémů, které bylo nutné vyřešit, ve většině případů pomocí vlastních skriptů či pomocí zabudované funkcionality v editoru. Stěžejní funkcionalitou aplikace byla schopnost přečíst daný marker pomocí kamery telefonu a vyvolat odpovídající obsah na obrazovku. Za tímto účelem poskytuje

Unity množství manažerů starajících se o specifickou funckionalitu (přesněji rozšiřující framework ARFoundation). Bylo také nutné uložit do vlastního objektu podoby markerů, aby došlo k logickému propojení markeru s odpovídajícím obsahem. Při vývoji aplikace bylo nutné použít i vytištěné testovací 3D modely v menším měřítku pro účely testování korektního chování aplikace. V rámci tohoto kroku bylo také vyřešeno přičítání offsetu pro správné zobrazení obsahu, a to formou zadání vhodných hodnot v editoru. Dalším důležitým bodem při vývoji aplikace byl návrh UI (User Interface, uživatelské rozhraní), které bylo z velké části navrženo přímo v editoru Unity. S návrhem UI se pojila také problematika přepínání scén a zobrazení korektního modelu ve scéně. Ve finální podobě obsahovala aplikace hlavní menu, ve kterém je na výběr jeden ze čtyř modelů a poté ve vedlejším menu výběr dané scény, načež je uživatel přenesen do hlavní scény. Dalším úkolem byla příprava druhého módu v rámci aplikace, který nepracoval s markery, ale na základě detekce virtuální roviny a následné inicializaci vybraného modelu na této rovině. Za tímto účelem byl využit další z manažerů dostupných v prostředí Unity. Muselo také dojít k úpravě dat. Logika přepínání scén musela být upravena, aby dokázala načíst odpovídající model i pro případ výběru druhého módu.

Dalším krokem byl vývoj vedlejších funkcionalit, mezi které patřila možnost přepnout prostředí aplikace do anglického jazyka. Za tímto účelem byl napsán vlastní skript starající se přepnutí zobrazeného textu na základě zvoleného jazyka. Mezi vedlejší funkcionalitu patří mimo jiné i zobrazení informací o dostupných scénách v rámci vybraného modelu v odpovídajícím vedlejším menu či zobrazení vizuální reprezentace modelu. Samostatným problémem pak bylo zobrazení některých UI elementů přímo v hlavní scéně, zejména informačního panelu, obsahující dodatečné informace o vybraném modelu.

Následným krokem v implementační části práce bylo vytištění samotných modelů v odpovídajícím měřítku, které bylo potřeba vhodně určit. Poté byl použit slicer pro vygenerování pokynů pro tiskárnu (tzv. G-code) a modely vytištěny. Doprovodnými součástmi 3D modelů byly také plochy na umístění markerů, které byly taktéž vytištěny. S vytištěnými modely pak bylo nutné upravit hodnotu offsetu v editoru tak, aby byl zobrazený obsah rozšířené reality co nejvěrohodněji umístěn na vytištěný model. Ve finále proběhla série optimalizací aplikace, zaručující optimálnější chod.

3.1 Použité metody

Pro řešení práce byla využita zejména technologie rozšířené reality (AR), která byla implementována pomocí programu Unity. Na rozdíl od virtuální reality, ve které je uživatel pohlcen do zcela virtuálního prostředí, AR využívá kombinace skutečného světa a virtuálního obsahu. Technologii rozšířené reality lze využít na telefonu, tabletu, chytrých brýlích, či obdobném zařízení. Základní myšlenkou je schopnost zobrazení vybraného virtuálního obsahu na obrazovce zařízení, využívající za tímto účelem kameru zařízení. Uživatel tak vidí jak reálné prostředí kolem sebe, tak právě virtuálně zobrazený obsah. Zobrazený obsah může být např. ve formě textu, animací, 3D modelů či textur.

V práci byla využita zejména Marker-Based AR, fungující na principu rozpoznání vzoru na referenční značce (marker). V takovém případě je zobrazený obsah navázán na specifický marker, v případě že je odpovídající marker nalezen. Vhodným markerem může být QR kód nebo obdobné značky s charakteristickým vzorem. Existují zde určité limitace, z vlastního testování vyplynul zejména fakt, že AR technologie je velmi citlivá na světelné podmínky. Svoji roli hraje také fyzická velikost využitých markerů. Limitace existuje také na straně enginu Unity, jelikož je vyžadováno, aby daný využitý marker měl dostatek

identifikovatelných bodů. V ideálním případě dojde k identifikaci a přečtení markeru, načež je zobrazen odpovídající virtuální obsah na obrazovce zařízení.

Práce využívá také metodu umisťování obsahu nevyužívající markery (Markerless AR). V takovém případě je využita schopnost AR detekovat roviny v reálném prostředí a na tyto roviny umístit daný obsah. Stejně jako v případě Marker-Based AR, i Marklerless AR je velmi citlivá na světelné podmínky. Obě formy AR však shodně fungují na principu vizualizace dat, v případě této práce se jedná o data prostorová.

Pro přípravu dat byla hojně využita metoda 3D modelování, ať už pro případy jednoduché úpravy modelu pro další práci či vymodelování daného 3D objektu vyskytujícího se v některé ze scén aplikace. Pro tisk modelů byla využita technologie 3D tisku. Tisk probíhal s využitím dvou 3D tiskáren. První tiskárnou byla Creality CR-10 Max, vyznačující se velkým tiskovým prostorem a tedy možností tisknout prostorově rozměrnější 3D modely. Některé doprovodné díly, modely použité při testování a modely v menším měřítku byly vytištěny na tiskárně Prusa i3 MK3S.

3.2 Použitá data

V práci využitá data se liší podle konkrétní studie. Data se taktéž dělí na primární (představující data, ze kterých je vytvořena primární mesh daného modelu, využita i pro případ 3D tisku) a sekundární (doplňující objekty a textury v určitých scénách aplikace). Pro první dvě studie byla využita, pro případ hlavních dat, data DMR (digitální model reliéfu), poskytována USGS (United States Geological Survey) (USGS, 2023). Specificky se jednalo o datové sady z kategorie 3DEP (3D Elevation Program). Konkrétní subkategorií poté byla datová sada DMR o rozlišení 1 m na pixel. V případě první studie (Hoover Dam) se jednalo o oblast 10 × 10 km ve formátu TIF, která byla poté upravena na oblast 2 × 2 km. Datum uveřejnění souboru na uvedeném portálu USGS bylo 11. 12. 2021. V případě druhé studie (Mount St. Helens) se jednalo o celkem čtyři TIF soubory, každý pokrývající oblast 10 × 10 km. Tyto soubory byly spojeny v jeden TIF soubor a poté upraveny na oblast 5 × 5 km. Datum uveřejnění souborů bylo v tomto případě 30. 11. 2021. Třetí studie (globus) využila primárně zakoupená data z webu Sketchfab (Studio Ochi, 2018). Datum uvedení modelu na webu je 30. 1. 2018. Zakoupený model byl poskytnut i v nativním formátu pro software Blender, nebylo tak nutné model dále zásadně upravovat. Tento model byl využit pro potřeby aplikace z důvodu nízkého počtu trojúhelníků a tedy nižší náročnosti na výpočetní výkon. Nebyl však vhodný pro 3D tisk. Pro potřeby 3D tisku byl využit 3D model Země z platformy Thingiverse, obsahující batymetrická data (Hartloff, 2018). Data k poslední studii (Trosky) byla poskytnuta vedoucím práce po dohodě s původním autorem dat Svatoplukem Místeckým. Data obsahovala, mimo jiné, model hradu Trosky ve formátu OBJ a odpovídající textury. Pro reliéf byl využit DMR 5G, zprostředkovaný pomocí programu ArcGIS Pro. Zmíněné datové sady byly vybrány z důvodu vhodnosti integrace rozšířené reality.

V rámci první studie byla využita vedlejší data ve formě 3D objektů představujících stavby vytvořené člověkem. Specificky se jednalo o samotnou Hooverovu přehradu a také dva mosty nacházející se ve vybrané oblasti. Model Hooverovy přehrady byl zakoupen na webu Sketchfab (Desertsage, 2022) a obsahoval kromě samotného 3D modelu i už připravenou UV mapu a textury. Dále, 3D model dominantního mostu, který se nachází před samotnou přehradou (Mike O'Callaghan–Pat Tillman Memorial Bridge) byl obstarán na webu 3D Warehouse, spadající pod software SketchUp (jjasper123, 2014). 3D model druhého z mostů na vybraném území byl vytvořen autorem. V případě druhé studie

představovala sekundární data DMR zájmového území před rokem 1980 (Greenberg, 2013), o prostorovém rozlišení 30 m na pixel.

Pro třetí studii byla jako sekundární data využita volně dostupné textury a modely od NASA (National Aeronautics and Space Administration), zahrnující textury povrchu Měsíce (Wright 2019), model ISS (Inernational Space Station) (NASA, 2019b) či modely měsíců Phobos a Deimos planety Mars (NASA, 2019c), (NASA, 2019a). Využity byly také planetární textury s uměle zvýšeným nasycením barev, částečně vycházející ze zdrojů NASA (INOVE, 2019). Ostatní využité textury napříč všemi studiemi jsou vytvořeny autorem práce z původních dat v prostředí ArcGIS Pro nebo, v případě textur ze satelitního snímkování, zprostředkovány rozšířením Blender-OSM za využití ArcGIS API klíče.

3.3 Použité programy

Pro naplnění hlavního cíle diplomové práce, a sice vývoje mobilní aplikace, byl využit software Unity ve verzi 2021.3.11f. Jedná se o multiplatformní herní engine, který nabízí širokou škálu nástrojů pro vývojáře a také možnost vyvíjet aplikace na různorodá zařízení, včetně mobilních telefonů. Unity podporuje vývoj aplikací využívající 2D i 3D grafiku a funkcionalitu aplikací lze rozšířit množstvím placeného i bezplatného obsahu. Pro konkrétní případ této práce byl program využit pro implementaci AR funkcionality s využitím frameworku AR Foundation ve verzi 4.2.3. AR Foundation obsahuje množství komponent (manažerů), které zastřešují specifickou funkcionalitu AR systémů. Stejně tak byl použit na záležitosti, které nesouvisely přímo s rozšířenou realitou, ale s obecným návrhem aplikací. Tedy, záležitosti jako návrh a implementace uživatelského rozhraní či další podpůrné funkcionality.

Skripty starající se o většinový chod aplikace byly napsány v jazyce C# s využitím IDE (Integrated Development Enviroment) Visual Studio 2022 Community Edition od firmy Microsoft. Visual Studio poskytuje škálu nástrojů pro asistenci s vývojem mnoha typů aplikací. V základu má vestavěnou podporu pro vývoj v programovacích jazycích jako C, C++ a C#. Pokročilejší podporu těchto a dalších technologií je možné přidat pomocí doplňků a vývojových nástrojů z oficiálních zdrojů nebo zdrojů třetích stran. V konečném důsledku tak umožňuje efektivní vývoj mnoha typů aplikací. Visual Studio obsahuje nástroje a funkce jako ladící program (debugger), monitorování využití systémových zdrojů, statickou analýzu kódu pro detekci potenciálních chyb či uživatelské rozhraní zjednodušující organizaci zdrojových souborů a jejich verzování.

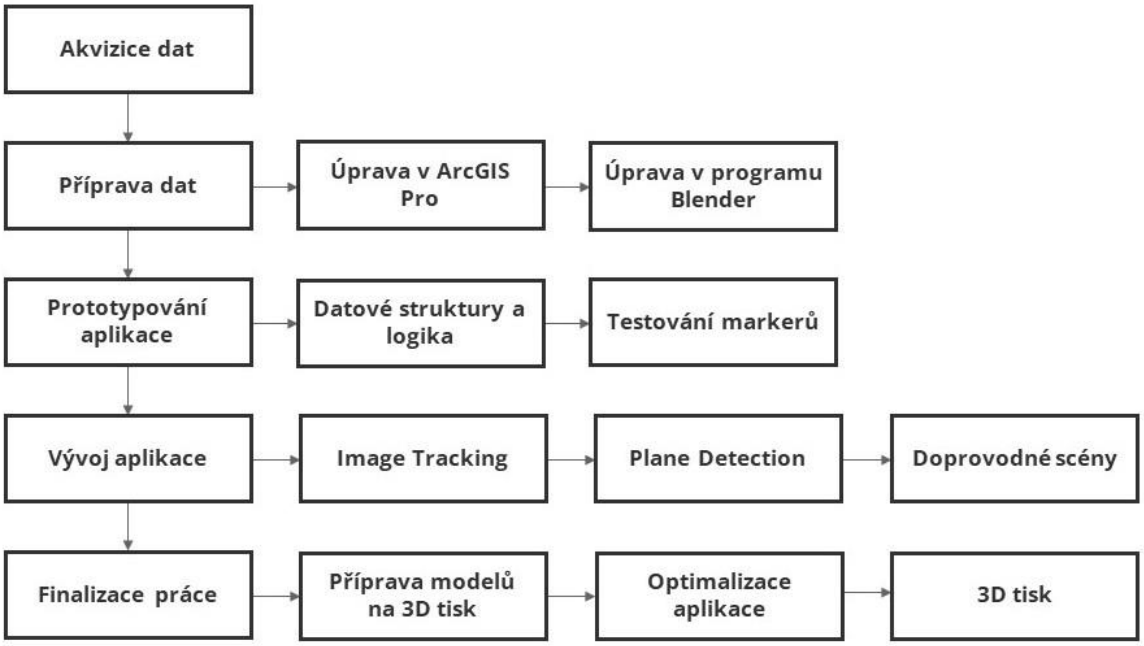
Majoritní část přípravy dat probíhala s pomocí programu Blender, ve verzi 3.3.0. Jedná se o FOSS (Free and Open Source Software) program vhodný pro širokou škálu činností, od 3D modelování a renderování (vykreslení obsahu) až po animace. Blender byl využit pro zpracování zdrojových dat, zahrnující zejména texturování a drobné úpravy modelů, které byly vyžadovány pro další práci. Příprava dat probíhala také s využitím programu ArcGIS Pro 3.0.2, který byl využit zejména pro prvotní vymezení zájmových oblastí či výpočet prostorových analýz, které byly následně využity jako textury. Pro tvorbu některých 3D modelů, zejména pro potřeby 3D tisku, byl využit program QGIS 3.16.16.

Příprava dat obnášela v některých případech také jejich zjednodušení, specificky dosaženo zmenšením počtu polygonů. Za tímto účelem byl využit program Netfabb Premium od firmy Autodesk ve verzi 2023.1. Zmíněný software slouží především pro různorodé opravy 3D modelů před tiskem, jako zacelení děr či invertování nesprávně orientovaných stěn. Jednou z dostupných oprav je právě i zredukování počtu trojúhelníků, ze kterých se skládá daný 3D model.

Pro potřeby tvorby 3D modelů některých objektů, jakož i finální úpravy modelů před tiskem byl využit program SketchUp od firmy Tribmle ve verzi 18. SketchUp lze využít zejména pro tvorbu architektonických staveb či návrh interiérů budov. Pomocí tohoto softwaru byly vymodelovány zejména plochy na umístění markerů, sloužící jako esenciální část každého vytištěného modelu.

Pro přípravu modelů na 3D tisk byl také využit program Ultimaker Cura 5.2.2. Program posloužil rozličnému nastavení vlastností před tiskem, včetně výšky vrstvy či vnitřní výplně. Úkolem programu je také provést tzv. slicing, tedy rozdělení daného 3D modelu na vrstvy tak, jak se výsledně tisknou na 3D tiskárně.

3.4 Postup zpracování



Obr. 3.1 Postup zpracování.

4 PŘÍPRAVA DAT

První úkolem v praktické části práce byla příprava dat. Samotný proces přípravy byl stěžejní součástí práce, jelikož bylo nutné data zpracovat do vhodné podoby pro další využití. V této kapitole upravená či vytvořená data později sloužila jako modely zobrazované v jednotlivých scénách mobilní aplikace. Následující odstavce popisují kroky při přípravě dat a jsou rozděleny podle jednotlivých studií. Příprava dat probíhala primárně v programech ArcGIS Pro a Blender.

4.1 Hoover Dam

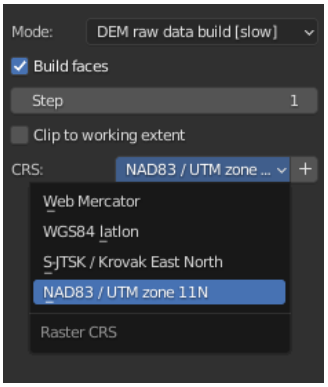
Nejprve bylo nutné získat data digitálního modelu reliéfu vybrané oblasti, a to ze zdroje uvedeného v podkapitole 3.2. Data byla stažena ve formátu TIF. Souřadnicový systém zdrojových dat byl NAD 1983 UTM Zone 11N (EPSG: 26911). Původní rozměr rastru byl 10 000 × 10 000 pixelů, z čehož byla pomocí operace *Extract by Mask* v prostředí ArcGIS Pro vybrána podmnožina 2001 × 2001 pixelů. Z důvodu rozlišení zdrojového DMR se pixely rovnaly metrům (tedy 2000 pixelů odpovídá 2 km). Oblast byla vymezena pomocí polygonového prvku určujícího hranice oblasti zájmu. Dalším krokem bylo zjištění souřadnic lomových bodů vybrané oblasti v souřadnicovém systému WGS 1984 (EPSG: 4326). Pro tento účel posloužila funkce *Feature Vertices to Points*, jejímž výsledkem byla bodová reprezentace vrcholů hranice oblasti. Oněm bodům byly poté přidány atributy X a Y, následně dopočítány pomocí funkce *Calculate Geometry Attributes*. Takto připravený rastr byl poté uložen ve formátu TIF a připraven na import do programu Blender.

V prostředí ArcGIS Pro byly také vypočítány primární topografické atributy, jejichž výsledky později posloužily jako textury. Z důvodu rozepsaného níže bylo před výpočtem analýz nutné upravit vybranou oblast, specificky zvětšit z původního rozměru 2001 × 2001 pixelů na rozměr 2303 × 2303 pixelů. Za tímto účelem byla využita funkce *Buffer,* pomocí které byla vytvořena obalová zóna na vnější části polygonu vymezujícího hranice oblasti, o rozměru 302 m. Poté byla využita funkce *Minimum Bounding Geometry* pro vytvoření pravoúhlého polygonu, který vymezoval oblast zvětšenou právě o 302 pixelů. Následně byla opět provedena funkce *Extract by Mask*. S takto vymezenou oblastí přišly na řadu prostorové analýzy, pro případ první studie byla využita funkce *Slope* pro určení sklonu svahu. Dále, jelikož obsahoval informace o výšce, byl rastr vybrané oblasti klasifikován do sedmi kategorií podle nadmořské výšky. Vznikl tak rastr zobrazující barevnou hypsometrii oblasti. Dva zmíněné rastry byly poté uloženy a vyexportovány ve formátu PNG, s použitím barevného prostoru RGB, z důvodu zachování barev.

Dalším krokem byl import rastru vybrané oblasti (2001 × 2001 pixelů) do programu Blender. Import byl docílen zásuvným modelem (tzv. plugin) BlenderGIS, který je dostupný ke stažení na webu GitHub (domlysz, 2022). Po instalaci pluginu se základní funkcionalita softwaru Blender rozšířila o kategorii GIS, s možností importu rastru ve formátu TIF. Při importu rastru je nutné použít korektní nastavení. Možnost importu disponuje několika možnostmi, přičemž musí být zvolena možnost *DEM Raw Data Build [slow]*, která naimportuje data ve formě bodového mračna a vytvoří tak mesh, tedy síť vrcholů tvořících plochy, které tvoří souvislý 3D objekt. Dalším důležitým nastavením je také zvolení správného souřadnicového systému, které, v případě, že souhlasí se souřadnicovým systémem původních dat (v tomto případě NAD 1983 UTM Zone 11N) vyústí v přesnější umístění textury na model, což je potřeba později v postupu práce. Obr. 4.1 ukazuje nastavení při importu. Výsledek po prvotním importu dat ukazuje Obr. 4.2.

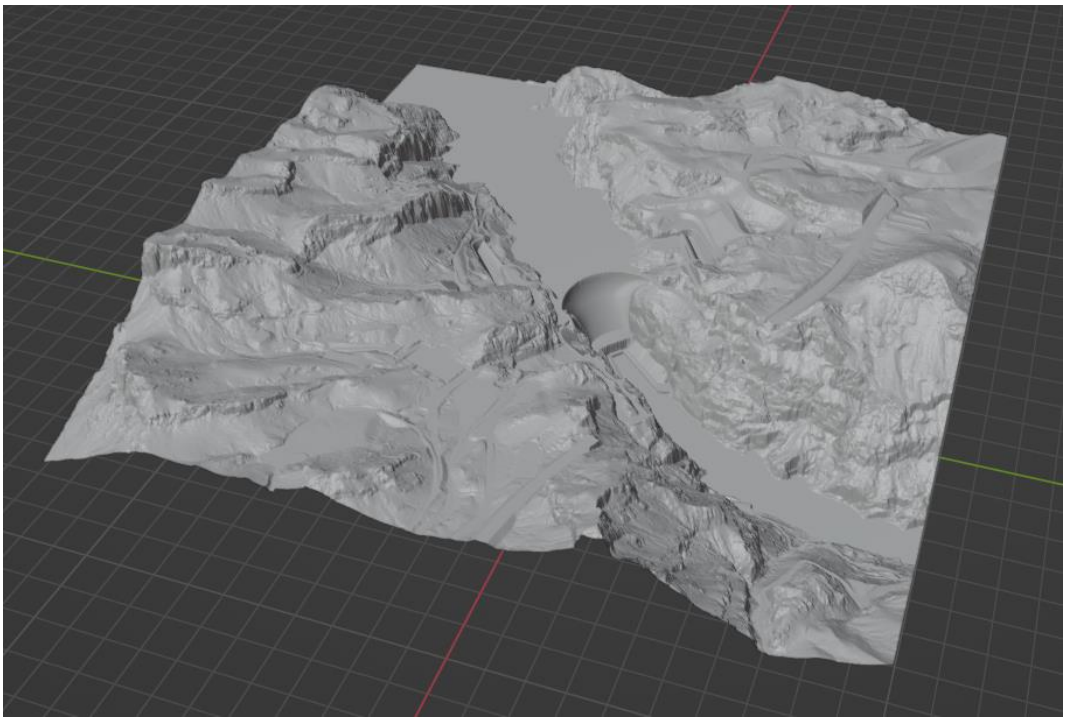
Čistě z teoretického hlediska by dalším krokem bylo přiřazení textury 3D modelu, následný export ve formátu FBX a import do programu Unity. Jelikož však zdrojový rastr

byl velmi podrobný (130 MB celková velikost), stejně tak i mesh vytvořená importem v přechozím kroku byla příliš detailní, soubor měl ve formátu FBX velikost přibližně 230 MB, což je příliš velké (zejména z hlediska počtu trojúhelníků) a náročné na vykreslení pro mobilní telefon, což bylo i testováno. Aplikace, běžící na testovacím zařízení, z důvodu příliš velké zátěži vykreslovala obsah v jednotkách snímků za sekundu. Bylo tedy zjevné, že je potřeba model zjednodušit. Za tímto účelem byl soubor exportován ve formátu STL, načež následovalo zjednodušení v programu Netfabb,



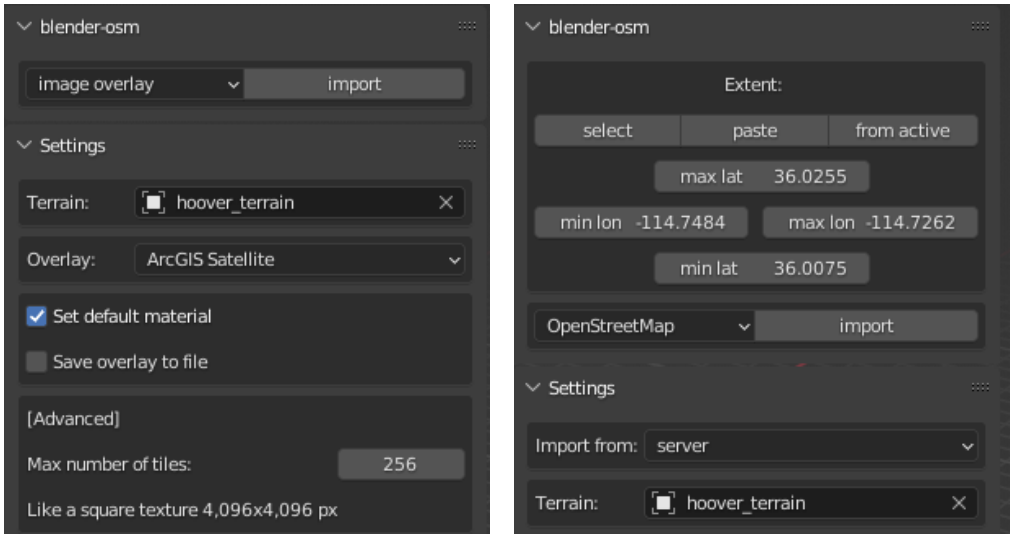
Obr. 4.1 Nastavení importu TIF souboru pomocí BlenderGIS (zdroj: autor).

V prostředí programu Netfabb se vždy při importu souboru zobrazují základní vlastnosti jako je počet schránek modelu či právě celkový počet trojúhelníků. Původní počet trojúhelníků modelu byl 8 000 000, bylo tak využito funkce v rámci oprav modelů, které Netfabb nabízí a počet byl snížen na přibližně 361 000 trojúhelníků, čímž se velikost STL souboru zmenšila z původních 381 MB na 17,2 MB. Zjednodušený model si, i přes nižší počet trojúhelníků, zachoval důležité charakteristiky terénu. Pro tisk v pozdější části práce byl použit původní TIF soubor a tedy nezjednodušený STL soubor.



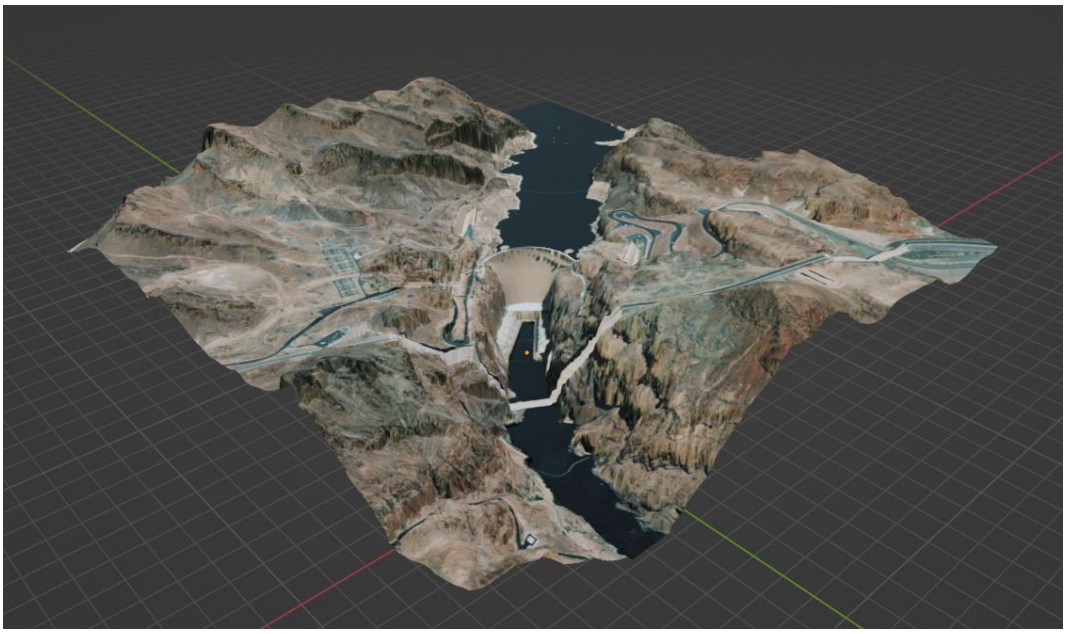
Obr. 4.2 Prvotní výsledek importu dat (zdroj: autor).

Po zjednodušení byl model vyexportován ve formátu STL a znovu nahrán do prostředí programu Blender. S výrazně zjednodušeným modelem přišel na řadu další krok, a sice texturování modelu. Jako základní textura modelu byl využit satelitní snímek, zprostředkovaný prostřednictvím pluginu BlenderOSM (vvoovv, 2021). Tento plugin nabízí jednoduchý import satelitních dat přes jednoduché menu v programu Blender. Nabízí i výběr zdroje satelitních dat, mezi které patří např. ArcGIS Satelite, MapBox Satelite či OSM Mapnik. Pro potřeby práce byl využit ArcGIS Satelite, pro jeho použití musel být vygenerován ArcGIS API klíč. Po vygenerování klíče byla z menu pluginu vybrána možnost *image overlay* a jako terén určen importovaný model Hooverovy přehrady. Možnosti importu také zahrnovaly možnost zadání rozsahu ve WGS souřadnicích. Právě zde byly využity hodnoty vrcholů hranice oblasti, které byly vypočítány v předešlém postupu. Nastavení importu zachycuje Obr. 4.3. Po zadání souřadnic byla naimportována textura satelitního snímku odpovídající oblasti a automaticky přiřazena modelu i jako materiál. Importem textury se také vykonala operace rozložení modelu na 2D síť a tím byla vytvořena tzv. UV mapa (U a V odpovídá osám ve 2D prostoru, jelikož X a Y jsou již použity v rámci 3D prostoru).



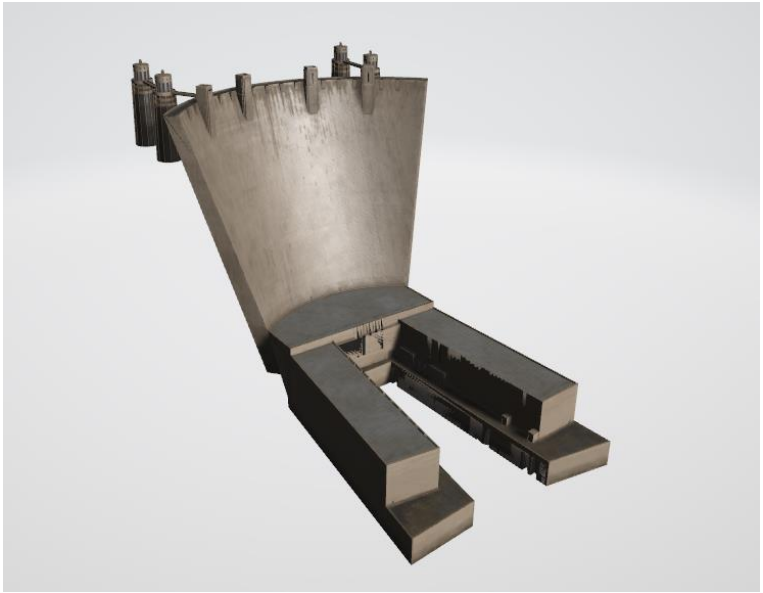
Obr. 4.3 Nastavení importu textury satelitního snímku (vlevo). Souřadnice pro import se nastavují na samostatné kartě (vpravo) (zdroj: autor).

Jelikož BlenderOSM importuje satelitní snímky na bázi dlaždic určité velikosti, použití pluginu bylo problematické. Bylo zjištěno, že importovaná textura byla asi o 300 pixelů větší než vybrané území (2001 × 2001 pixelů). UV mapa reprezentující mesh byla tak menší a v některých místech neodpovídala zobrazená textura charakteru terénu. Bylo tak nutné UV mapu manuálně posunout v sekci *UV Editing* v prostředí programu Blender. Zmíněný problém je taktéž důvodem, proč byl v předešlém postupu práce vytvořen rastr vybraného území, který byl o 302 pixelů větší. Bylo to právě z důvodu přizpůsobení textur použité UV mapě, pro další textury totiž byla využita identická UV mapa, tedy ta vytvořena importem satelitního snímku pomocí BlenderOSM. Kdyby měl rastr původní rozměry, pak by UV mapa velikostně neodpovídala. I tak bylo nutné UV mapu mírně zmenšit. Experimentováno bylo i s ručním vytvořením UV mapy, které však nepřineslo zamýšlený výsledek, byl tak použit představený postup. Výslednou mesh s importovanou texturou zachycuje Obr. 4.4.



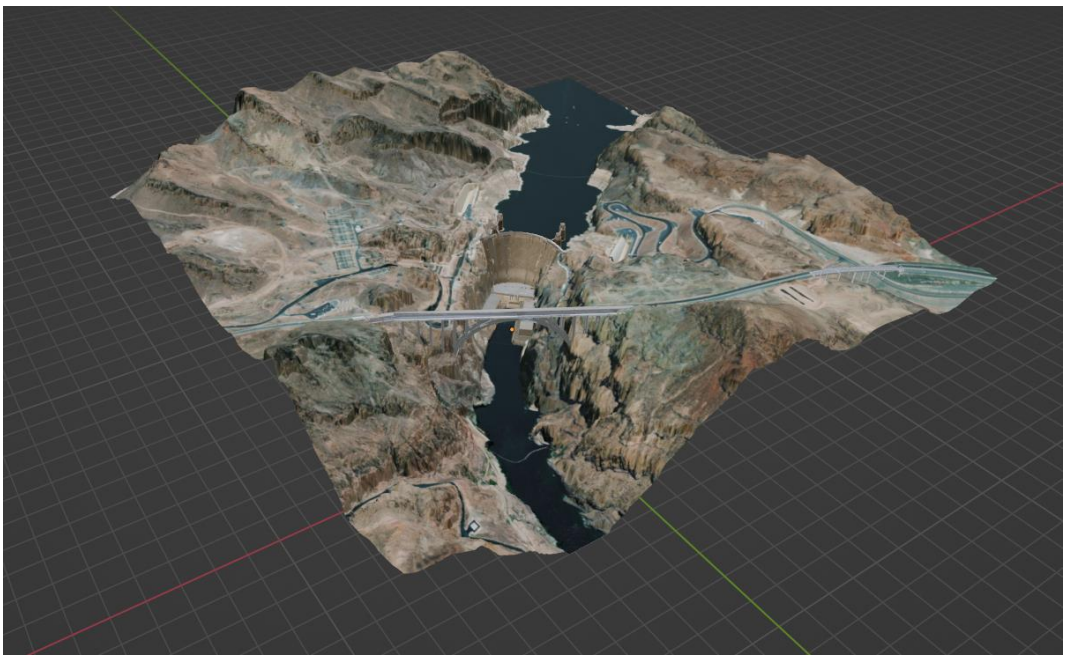
Obr. 4.4 Mesh oblasti Hoover Dam po zjednodušení modelu a aplikování textury satelitního snímku (zdroj: autor).

Připravený model bylo následně možné vyexportovat jako FBX a importovat do softwaru Unity, ve kterém probíhala další práce. V programu Blender však byly připraveny také další varianty modelu obsahující jiné textury (předtím vytvořeny v ArcGIS Pro) či sekundární data. V případě vybraného území představovala vedlejší data 3D modely Hooverovy přehrady a také dvou mostů, které se v oblasti nachází. Jak je již uvedeno v kapitole 3.2, 3D model Hooverovy přehrady (Obr. 4.5) byl zakoupen, přičemž byl dodán s UV mapou a odpovídajícími texturami.



Obr. 4.5 3D model Hooverovy přehrady (zdroj: autor).

Model přehrady měl v základu velikost téměř 83 MB, a to z důvodu použitých textur, které byly o rozlišení 4096 × 4096 pixelů. Textury tak byly zmenšeny pomocí *UV Editing,* specificky možností *Resize* v nabídce *Image*. Velikost textur (včetně tzv. normal textur a roughness textur) byla zmenšena na 1024 × 1024 pixelů, což bylo pro potřeby aplikace dostačující a velikost výsledného souboru se tím zmenšila na 6,73 MB. Bylo také nutné mírně upravit rozměry modelu, aby se co nejlépe přizpůsobil okolnímu terénu. Dalším krokem byla příprava mostu nacházejícího se před přehradou. Model mostu byl obstarán ze zdroje uvedeného v kapitole 3.2 a upraven. V základu byl soubor stažen ve formátu SKP a pomocí programu SketchUp vyexportován jako FBX, avšak s tím problémem, že neobsahoval některé textury. Ty tak byly doplněny až později v programu Unity. Následovala příprava druhého mostu nacházejícího se ve vybrané oblasti. 3D model zmíněného mostu, značně menších rozměrů a jednodušší po stavební stránce, byl vytvořen autorem práce a později mu taktéž byly přiřazeny textury. Většina textur byla aplikována už v programu Blender, avšak při exportu do formátu FBX, i přes korektní nastavení exportu, se textury nesprávně přenesly do softwaru Unity. Proto bylo nutné modelům znovu přiřadit texturu až právě v programu Unity. Za tímto účelem byly všechny použité textury uloženy do formátu PNG pro pozdější využití. Pro potřeby zobrazení mostů na modelu byla v grafickém programu upravena původní textura satelitního snímku tak, aby na místě mostů byla voda či odpovídající terén.

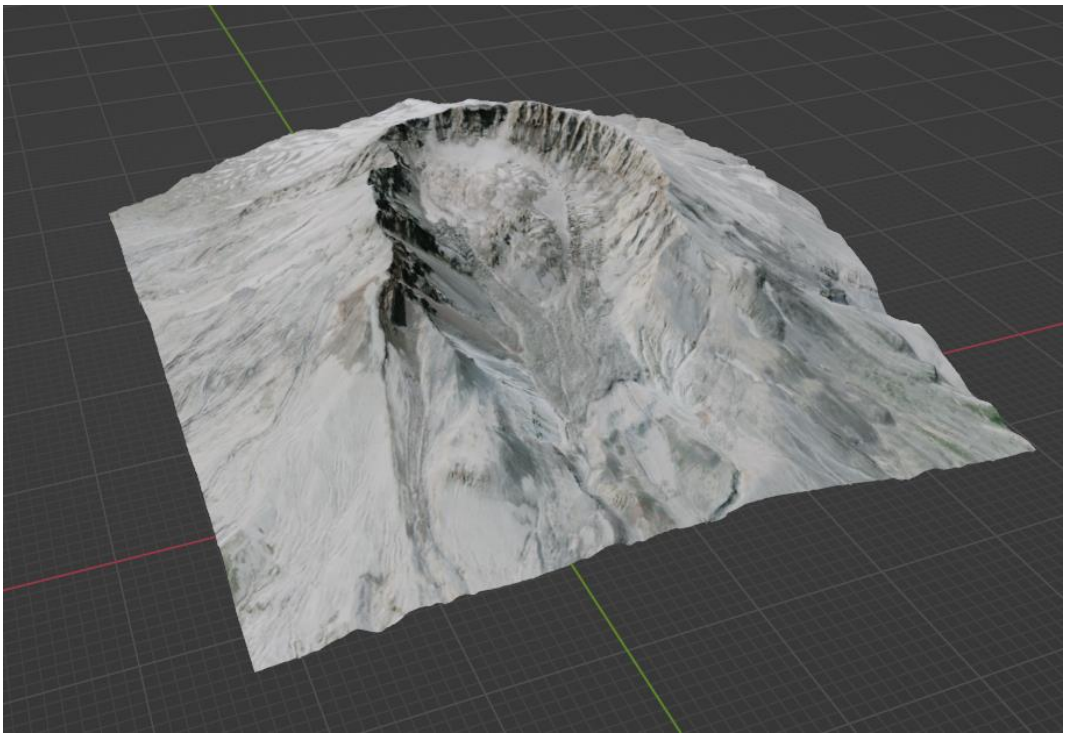


Obr. 4.6 Mesh oblasti Hoover Dam obsahující upravenou texturu satelitního snímku doplněn o 3D modely přehrady a dvou mostů (zdroj: autor).

4.2 Mount St. Helens

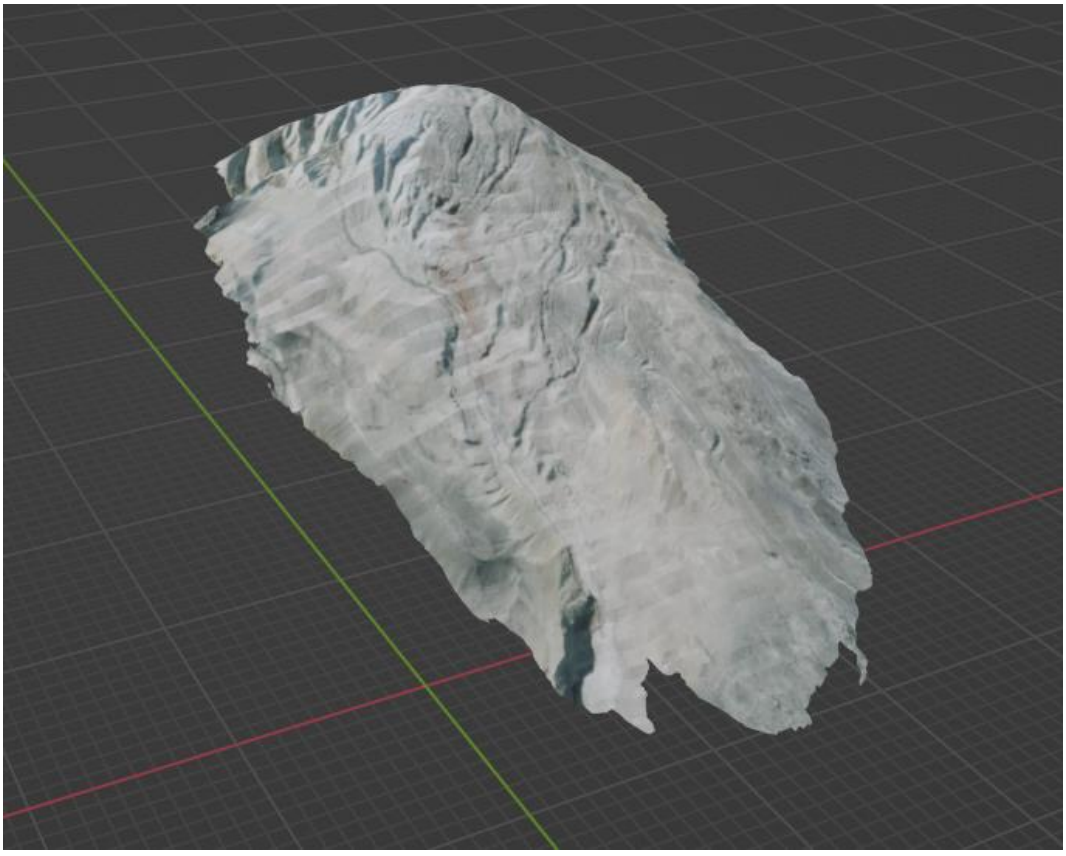
Postup byl v mnoha bodech analogický postupu uvedenému v podkapitole 4.1, s několika významnými rozdíly. Stejně jako v předešlém postupu, nejprve bylo nutné získat data. Na rozdíl od území Hooverovy přehrady, data pokrývající vybrané území Mount St. Helens se skládala ze čtyř TIF souborů. V tomto případě byla data v souřadnicovém systému NAD 1983 UTM Zone 10N a každý ze souborů byl o rozsahu 10 012 × 10 012 pixelů. Dalším krokem po získání dat tak bylo spojení více rastrů v jeden, čehož bylo docíleno v prostředí ArcGIS Pro funkcí *Mosaic To New Raster.* Výsledný rastr měl rozsah 20 012 × 20 012 pixelů, z čehož byla pomocí funkce *Extract by Mask* a polygonového prvku vymezujícího oblast zájmu vybrána podmnožina 5001 × 5001 pixelů. Další postup se shodoval, došlo k vytvoření vrcholů hranice oblasti a vypočítání odpovídajících souřadnic.

Dále byl pro vymezenou oblast vytvořen rastr sklonu svahu a také upraven původní rastr do osmi kategorií pro zobrazení hypsometrie. Stejně tak byl opět vytvořen rastr zabírající větší území (5657 × 5657) z důvodu uvedeného v podkapitole 4.1. Vytvořené rastry poté byly exportovány v odpovídajícím formátu. Následoval import do programu Blender, bylo využito shodné nastavení, s úpravou souřadnicového systému na NAD 1983 UTM Zone 11N. Dalším krokem bylo zjednodušení modelu pomocí programu Netfabb. Původně byl počet trojúhelníků modelu přibližně 50 000 000, tento počet byl zredukován na 1 033 112. Kromě redukce počtu bylo potřeba provést opravu zdegenerovaných stěn, které vznikly jako následek snížení počtu trojúhelníků. Následoval export modelu ve formátu STL, import do programu Blender. Zde proběhlo texturování pomocí BlenderOSM a opět také manuální úprava UV mapy. Poté byl model uložen jako FBX.



Obr. 4.7 Mesh oblasti Mount St. Helens po zjednodušení a aplikování textury satelitního snímku (zdroj: autor)

Stejně jako v případě první studie byly ve formátu FBX připraveny i další textury a také sekundární data. Těmi byl v případě Mount St. Helens DMR zobrazující reliéf před erupcí v roce 1980, která významně upravila charakter reliéfu. Nejprve bylo zapotřebí využít program QGIS a pomocí pluginu DEMto3D vygenerovat odpovídající 3D model s využitím zmíněného DMR. 3D model byl vytvořen také pro primární data. Výsledkem tak byly dva 3D modely, které byly následně importovány do programu Netfabb. Zde proběhlo logické odečtení (Boolean Subtraction) zmíněných modelů, výsledkem operace byl 3D model zobrazující rozdíl mezi současným a předešlým stavem reliéfu. Vytvořený model byl poté dále upraven využitím automatických oprav nabízených v rámci programu, jelikož po odečtení částí obsahoval množství chyb. Jelikož se modely přesně nepřekrývaly, bylo zapotřebí některé části výsledku logického rozdílu manuálně odstranit. Toho bylo docíleno v programu Blender, kde byla modelu také přiřazena textura. Byla využita shodná textura ze satelitního snímkování jako v případě primárních dat, s tím rozdílem, že UV mapa modelu byla upravena tak, aby pokrývala pouze část textury. Teoreticky se dalo využit textur např. z družice Landsat, pokrývající oblast před rokem 1980. Rozlišení snímků však bylo příliš nízké pro potřeby vizualizace. Výsledek popsané práce nabízí Obr. 4.8. Další úprava dat zobrazených ve scéně již probíhala s použitím softwaru Unity (přidání částicových efektů simulující kouř).

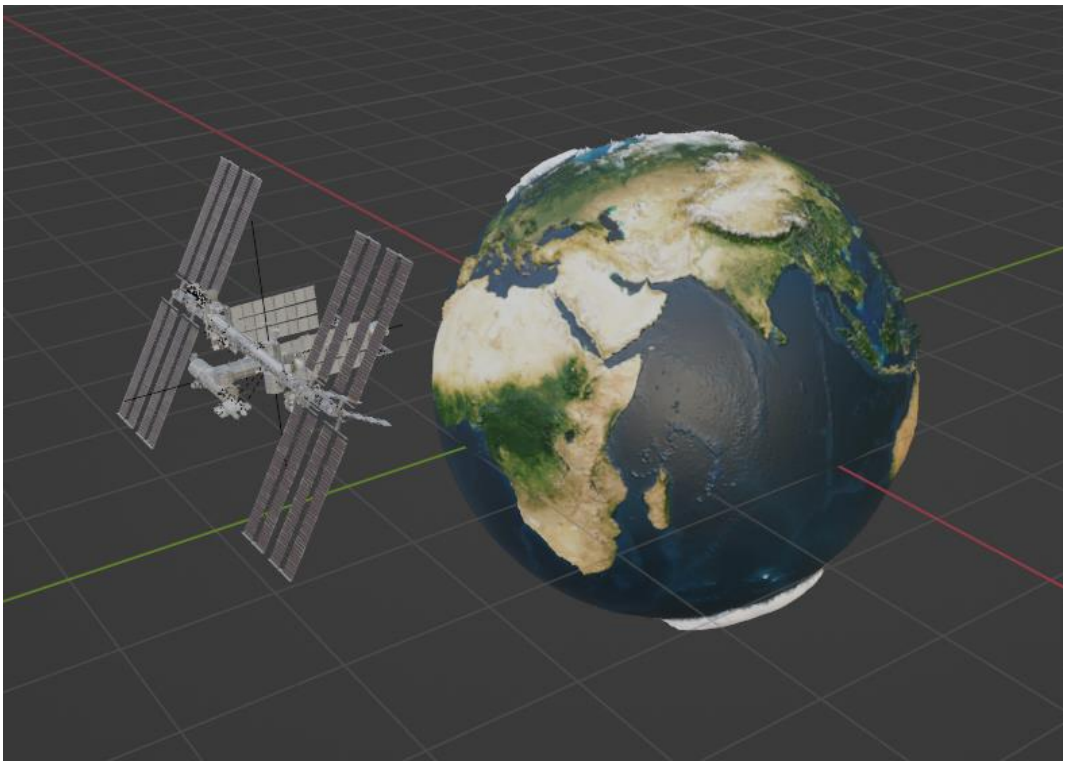


Obr. 4.8 Mesh vzniklá logickým rozdílem dvou DMR, zobrazující původní charakter reliéfu před sopečnou erupcí (zdroj: autor).

4.3 Globus

V případě třetí studie byl postup práce značně odlišný, zejména z toho důvodu, že nebylo nutné zdrojová data upravit před nahráním do softwaru Blender. Zakoupený model byl dodán jak v nativním formátu BLEND, tak ve formátu FBX. Model taktéž měl nachystanou korektní UV mapu a texturu v podobě satelitního snímku povrchu planety. Z toho důvodu bylo možné ihned přistoupit ke tvorbě modelů pro jednotlivé scény mobilní aplikace, vycházejících právě z dat zdrojových. Za tímto účelem byla využita sekundární data uvedená v podkapitole 3.2.

Nejprve bylo nutné obstarat model ISS. Po získání modelu byly v prostředí programu Blender uloženy všechny textury modelu pro další práci. Po exportu modelu ve formátu FBX totiž, i přes správné nastavení, neměl model textury. Bylo tak nutné importovat textury do programu Unity, načež byl importován samotný model a správně se propojil s odpovídajícími texturami. Ekvivalentní postup platil pro všechny modely, i v přechozích případech. V prostředí programu Blender byl nachystán model planety Země doprovozen o model ISS (Obr. 4.9). Model byl následně exportován jako FBX.



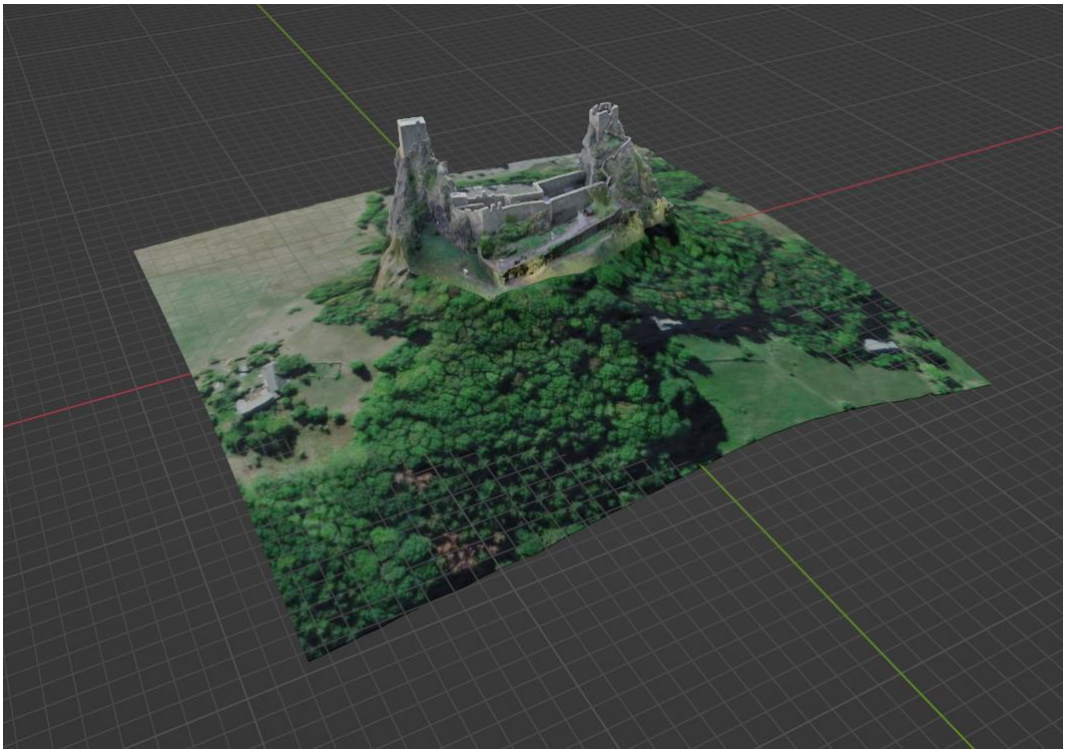
Obr. 4.9 3D model Země doplněn o 3D model ISS (zdroj: autor).

Následovalo vytvoření UV sféry v programu Blender. Vytvořený objekt posloužil jako objekt představující Měsíc. Objektu byly přiřazeny textury odpovídající povrchu Měsíce. Stejných sfér bylo ve výsledku vytvořeno více, každá z nich představující dané planetární těleso (terestriální a jiné planety) či odpovídající přirozené družice. Všem vytvořeným modelům byla přiřazena odpovídající textura a následně byly exportovány jako FBX.

4.4 Trosky

Postup pro zpracování dat v případě poslední studie byl v mnohém analogický první a druhé studii. Klíčovým rozdílem byla velikost území a také prostorové rozlišení dat. Vzhledem k dostupným datům bylo vybráno zájmové území 350 × 350 m. Prvním krokem tedy bylo vytvoření podmnožiny z datové sady DMR 5G, a to pomocí funkce *Extract by Mask*. Vytvořená podmnožina měla rozlišení 176 × 176 pixelů. Následovalo vytvoření vrcholů oblasti, která vymezovala zájmové území a vypočítání souřadnic.

Pro zájmovou oblast byly dále vytvořeny rastry sklonu a hypsometrie. Byla také využita datová sada DMP (digitální model povrchu) pro vytvoření rastru digitálního modelu povrchu oblasti. Specificky byl využit v ArcGIS Pro dostupný DMP 1G. Vytvořené rastry byly exportovány v odpovídajícím formátu. Na rozdíl od první a druhé studie neproběhlo vytvoření většího rastru pro potřeby UV mapy. Následoval import do programu Blender, zdrojový rastr byl importován se souřadnicovým systémem S-JTSK Krovak EastNorth (EPSG: 5514). Vzhledem k nižšímu rozlišení nebylo nutné model upravovat, dalším krokem tedy byla aplikace textury s využitím BlenderOSM. Vzniklou UV mapu nebylo zapotřebí upravit. Pro potřeby dalších textur (sklon, hypsometrie) byla vytvořena alternativní UV mapa pomocí funkce v programu Blender (UV → Project from View). Významným rozdílem oproti ostatním modelům je nemožnost exportu modelu vzniklého ze zdrojových dat DMP 1G ve formátu FBX. Proběhl tak export do formátu OBJ. Všechny ostatní modely byly exportovány ve formátu FBX.

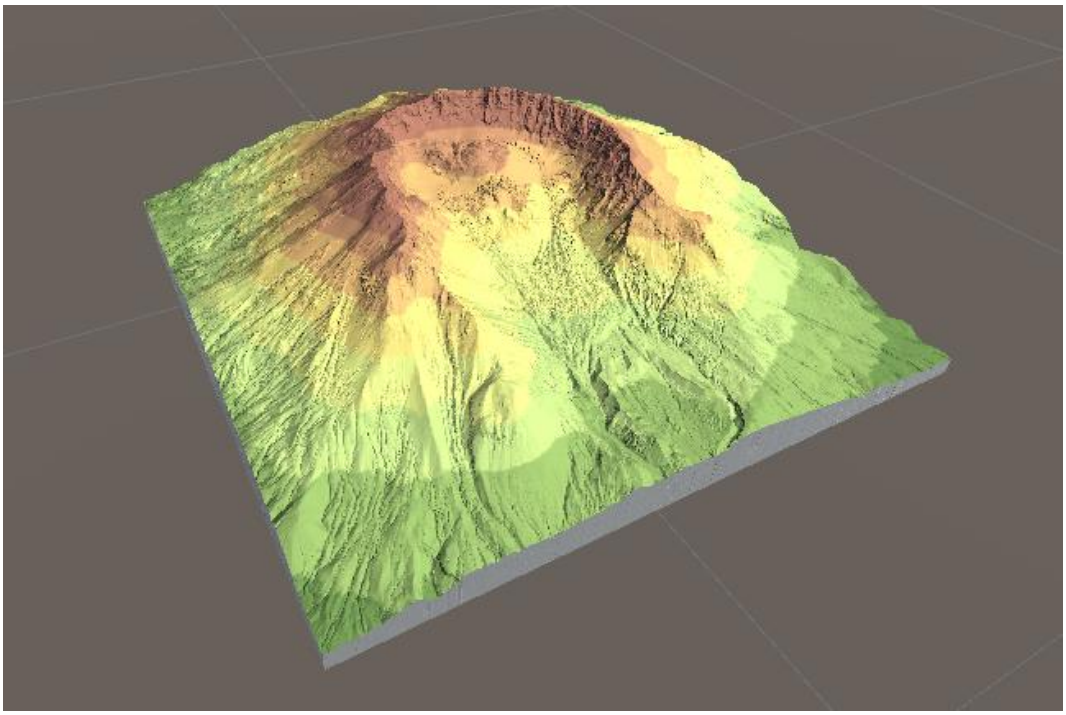


Obr. 4.10 Mesh oblasti hradu Trosky doplněn texturou satelitního snímku a 3D modelem hradu (zdroj: autor).

4.5 Úprava dat pro potřeby Plane Detection

Vyvíjená aplikace obsahovala dva módy, jeden pracující na bázi detekce referenční značky (markeru) a zobrazení obsahu na základě této detekce (Image Tracking). Přípravě dat pro tento mód se věnoval dosavadní obsah kapitoly 4. Pro druhý z módů (Plane Detection), pracující na bázi detekce rovin v prostoru a umístění objektu na danou plochu po dotknutí se obrazovky, bylo potřeba již nachystaná data ještě mírně poupravit, a to zejména pro případ prvních dvou studií (Hoover Dam a Mount St. Helens). Původně upravená data byla reprezentována pouze meshi o nulovém objemu, což bylo pro zobrazení na základě markeru dostačující, ne však pro zobrazení na rovině, kdy bylo žádané, aby zobrazovaný model měl objem a tedy se jevil jako korektní 3D objekt. Úpravy byly provedeny v prostředí programu Blender.

První úpravou bylo, po přepnutí do módu editace, přidání objemu modelům. Toho bylo docíleno funkcí *Solidify Faces* (Face → Solidify Faces). Poté bylo možno model extrudovat, což ve své podstatě znamenalo vytvoření podstavy. Při extruzi však došlo u modelů k tomu, že se zrcadlově obrátily a byla tak vytvořena v podstatě jejich kopie. Z toho důvodu bylo nutné spodní část modelu odstranit. Poté bylo nutné zarovnat podstavu do roviny. Za tímto účelem byla vytvořena rovina (Add → Plane) mírně přesahující rozměry daného modelu. Následně byl modelu přidán booleovský modifikátor, pomocí kterého byl učiněn logický rozdíl mezi daným modelem a vytvořenou rovinou. Došlo tak k zarovnání podstavy. Dalším krokem bylo rozdělení vrchní a boční části modelu. Za tímto účelem bylo potřeba v módu editace učinit manuální selekci všech stěn tvořících boční část modelu a následně provést oddělení vybraných částí (P → Separate → Selection). Kdyby nebyl učiněn tento krok, při aplikaci textury na daný model by byl stejnou texturou kromě vrchní části modelu texturován i zbytek modelu, což nebylo žádoucí. Po zmíněných operacích zbývalo aplikovat odpovídající texturu dle scény (satelitní snímek, sklon atd.). Následoval export ve formátu FBX a import do programu Unity.



Obr. 4.11 Příklad výsledku úpravy dat pro potřeby Plane Detection (zdroj: autor).

5 MOBILNÍ APLIKACE

Stěžejním krokem v rámci diplomové práce bylo vytvoření mobilní aplikace, schopné naplit hlavní cíl práce, a sice vytvoření technického řešení pro integraci rozšířené reality a 3D modelů reliéfu. Aplikace byla tvořena pro operační systém Android a postup byl rozdělen na dvě klíčové části. V první části probíhalo vytvoření prvotních prototypů aplikace, na kterých byla testována požadovaná funkcionalita. Druhá část pak zahrnuje samotný vývoj aplikace. Při popisu aplikace jsou nejdříve vysvětleny dvě hlavní scény (Image Tracking a Plane Detection) a jejich funkcionalita, poté menu aplikace a s tím související přepínaní scén, jakož i další funkce obsažené v aplikaci.

5.1 Prototypy aplikace

Než mohlo začít vytváření prototypů aplikace, bylo zapotřebí správně nastavit prostředí Unity, jelikož vývoj AR měl svá specifika. Nejprve bylo nutné zajistit podporu sestavování aplikace pro systém Android. Za tímto účelem byl doinstalován balíček přidávající požadovanou funkci. Další důležitá nastavení zahrnovala změnu API pro vykreslení obsahu na OpenGLES3 (OpenGL for Embedded Systems) a změnu architektury procesoru na ARM64. Dalším nastavením byla také změna skriptovacího backendu z Mono na IL2CPP (Intermediate Language to C++). V neposlední řadě bylo potřeba v nastavení projektu v záložce XR Plug-in Management vybrat možnost ARCore. Všechna zmíněná nastavení byla využita i při ostrém vývoji aplikace. Aby nebylo vynecháno žádné důležité nastavení, byla použita šablona pro AR aplikace, kterou Unity nabízí.

Po prvotním nastavení projektu byly přidány vyžadované objekty do scény a přidána komponenta Reference Image Library (podrobněji popsáno v podkapitole 5.2). Jelikož Unity pracuje na principu objektově orientovaného programování, je možné s každým objektem ve scéně pracovat a určovat mu chování, které rozšiřuje funkcionalitu dále, než co dovolují základní skripty připojeny k danému objektu (pozn.: Ne každý objekt má v základu svůj vlastní skript, ale manažery jako AR Tracked Image Manager ano).

Vytváření prototypů se zaměřovalo zejména na korektní inicializace a zobrazení obsahu ve scéně. Aby se objekt charakteru rozšířené reality zobrazil ve scéně, bylo nutné jej inicializovat, respektive instanciovat, což v podstatě znamenalo vytvoření jeho kopie. Takto vytvořené kopie pak dědí vlastnosti objektů, ze kterých vycházejí, přičemž hodnoty jako pozice, rotace a měřítko (dohromady tvořící transformace objektů) mohou být upraveny v prostředí editoru, ale i přímo v kódu rozšiřujícího skriptu. Původně byla aplikace zamýšlena tak, že v rámci jedné scény se zobrazí více obsahu najednou, případně se zrovna nepoužívaný obsah (respektive marker ve scéně, který není zrovna čten) učiní neaktivním. Z toho důvodu bylo experimentováno s datovými strukturami pro uložení obsahu tak, aby na každý jeden marker mohl být navázán vlastní obsah. Jednou z datových struktur využitých pro prototyp aplikace byl slovník, fungující na principu klíč – hodnota. Klíčem v tomto případě bylo jméno inicializovaného objektu, hodnotou herní objekt (v terminologii Unity tzv. GameObject) odpovídající jménu. Pro tento přístup bylo mimo slovníku využito také pole, o základní inicializaci objektů se staral následující kód:

```
[SerializeField]
private GameObject[] placablePrefabs;
private Dictionary<string, GameObject> spawnedPrefabs = new
Dictionary<string, GameObject>();
private ARTrackedImageManager trackedImageManager;
```

Příznak SerializeField zapříčiňuje, že se daný objekt objeví jako prvek v editoru, se kterým je možno dále pracovat, v tomto případě do něj napřímo přidat data. Deklarace na posledním řádku je z důvodu reference na manažera AR obsahu. O inicializaci odpovídajícího obsahu se pak staral následující cyklus:

```
foreach (GameObject prefab in placablePrefabs)
{
      GameObject newPrefab = Instantiate(prefab, Vector3.zero,
      Quaternion.identity);
   newPrefab.name = prefab.name;
   spawnedPrefabs.Add(prefab.name, newPrefab);
   prefab.SetActive(false);
}
```

Ideou cyklu bylo, že postupně procházel pole objektů a každý z objektů inicializoval na pozici (0, 0, 0), bez rotace. Následně došlo k přiřazení správného názvu, přidání daného objektu do slovníku a jeho deaktivaci. Za předpokladu shodujícího se názvu do slovníku přidaného objektu a obrázku reprezentujícího marker došlo k zobrazení objektu ve scéně. Popsaný přístup se ve výsledku ukázal jako nevhodný, hlavně z důvodu problematické deaktivace již aktivovaných objektů v případě více objektů ve scéně. Pro ostrou aplikaci tak nebyl slovníkový přístup použit (pozn.: ve výsledné aplikaci není zobrazováno více objektů v jedné scéně, to však ve fázi prototypování nebylo pevně dáno).

Experimentováno bylo také se změnou hodnot transformace objektu, což bylo pro aplikaci stěžejní. Jak bylo zmíněno výše v textu, existují dva základní přístupy. Hodnotu je možné přiřadit v editoru každému jednotlivému objektu. V takovém případě se při inicializaci objektu zadaná hodnota použije pro určení transformace. Druhým přístupem je zobecnění hodnot a jejich přiřazení v kódu. Při tomto přístupu dojde k přepsání hodnot v editoru nově zadanými hodnotami. Příkladem budiž iniciálně nastavené měřítko objektu na hodnotu 1 a přiřazení hodnoty 0,25 do příslušné proměnné. Po vykonání kódu se měřítko očekávatelně zmenší na čtvrtinu hodnoty. Změna transformace objektu pomocí kódu mohla vypadat následovně:

```
Vector3 position = trackedImage.transform.position;
Quaternion rotation = Quaternion.Euler(45f, 0f, 0f);
Vector3 scale = new Vector3(0.25f, 0.25f, 0.25f);
Vector3 offset = new Vector3(0f, -0.4f, 0.16f);

prefab.transform.position = position + offset;
prefab.transform.rotation = rotation;
prefab.transform.localScale = scale;
```

Právě přičítání offsetu k danému objektu bylo pro práci klíčové, za normálních okolností se totiž AR obsah po přečtení markeru zobrazil přímo na daném markeru. Zadaná hodnota offsetu tak měla za úkol zobrazený obsah posunout tak, aby se zobrazil nad vytištěným 3D modelem. Při ostrém vývoji aplikace byl nakonec využit postup zadávání hodnot přímo v editoru Unity. Důsledkem je větší přehlednost za cenu složitější správy objektů, jelikož pozici u všech objektů bylo potřeba měnit manuálně.

Druhou datovou strukturou pro uložení dat testovanou v rámci prototypování aplikace bylo pole objektů, které se při zapnutí aplikace do pole načetly a poté mohly být zavolány na základě shody jména. Pole bylo využito z důvodu potřeby uchovat více objektů v jeden moment. Proces přiřazení objektů do pole vypadal následovně:

```
private List<GameObject> prefabArray = new List<GameObject>();

private void Awake()
{
      prefabArray.Add(LoadPrefab("hoover_test"));
      prefabArray.Add(LoadPrefab("helens_test"));
}
```

Uvedený postup opět pracoval s předpokladem zobrazení více modelů v rámci jedné scény při běhu aplikace, od čehož bylo později upuštěno. Načítání objektů do pole by fungovalo i v případě zobrazení pouze jednoho objektu ve scéně, z důvodu výkonu to však bylo ve finále vyhodnoceno jako nevyhovující. Posledním důležitým problémem bylo zobrazení odpovídajícího obsahu pro více stran. Model v prostředí Blender měl vždy jednu nativní stranu a tedy i nativní rotaci v ose Y (0°). Původním záměrem v práci byla schopnost zobrazit AR obsah z více stran. V případě, že by uživatel naskenoval marker z nativního pohledu, objekt by se zobrazil s rotací 0°, v případě přečtení z jiné strany by se zobrazil s rotací např. 90° (analogicky pro všechny ostatní strany). Ve výsledku by tak každý objekt existoval ve variantě 0°, 90°, 180°, 270°. Toho mohlo být docíleno v prostředí editoru tím, že by se pro každý z objektů ve scéně vytvořila jeho kopie, modifikující pouze rotaci objektu v ose Y. Druhým přístupem bylo zachovat si pouze jeden objekt pro jednu specifickou scénu, ale měnit rotaci dynamicky v kódu, což vypadalo následovně:

```
switch (imageName)
{
      case "HooverDam":
            xORzPosition = 0.16f;
            yPosition = -0.4f;
            xORzRotation = 45f;
            scale = 0.26f;
            break;
      case "StHelens":
            xORzPosition = 0.25f;
            yPosition = -0.6f;
            xORzRotation = 30f;
            scale = 0.14f;
            break;
      default:
            break;
}
```

Určování pozice a rotace pro proměnné xORzPosition a xORzRotation probíhalo v dalším segmentu kódu. Daný přístup byl zvolen z důvodu orientace os X, Y, Z v prostoru vůči kameře. Pozice a rotace se tak musela správně měnit, a to na základě úhlu rotace osy Y, který byl obsažen v názvu referenčního obrázku v editoru Unity a získán pomocí rozdělení odpovídajícího textového řetězce (př. název obrázku byl HooverDam90, po rozdělení řetězce vznikl řetězec „HooverDam“ a řetězec „90“, což odpovídá jednotlivým případům v příkazu switch). Proměnné se tak měnily na základě rotace osy Y získané jako část textového řetězce, po přetypování na desetinné číslo (float).

```
switch (yRotation)
{
      case 0:
        zPosition = xORzPosition;
        xRotation = xORzRotation;
        break;
      case 90:
        xPosition = xORzPosition;
        zRotation = xORzRotation;
        break;
      case 180:
        zPosition = -xORzPosition;
        xRotation = -xORzRotation;
        break;
      case 270:
        xPosition = -xORzPosition;
        zRotation = -xORzRotation;
        break;
}
```

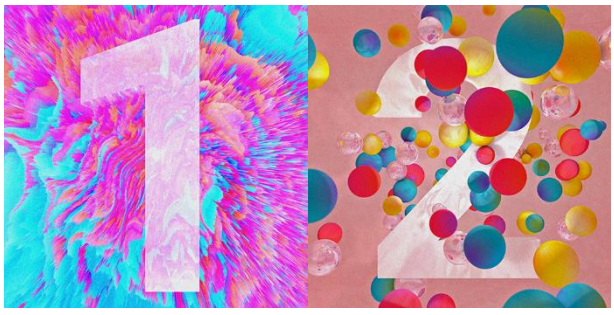
Po přiřazení odpovídajících proměnných následně byly ony proměnné předány jako parametry pro změnu transformace objektu a objekt aktivován. Při testování se ukázalo, že popsaný postup s sebou nese jistá úskalí a bylo by nutné hodnotu offsetu určit pro každou stranu modelu (každý marker) zvlášť. Zmíněné problémy jsou blíže popsány v diskusi práce. Ve výsledku bylo rozhodnuto, že každý z modelů měl pouze jednu primární stranu, ze které lze marker číst.

Poslední záležitostí při prototypování aplikace bylo její pojmenování. V programovacím jazyku C# se u každé funkce deklaruje její návratový typ (např. pokud má funkce vrátit herní objekt, je deklarována s typem GameObject). Jestliže funkce sama nic nevrací, deklaruje se s typem void (znamenající prázdno, nic). Návratovým typem takové funkce je pak také void. Ještě před samotným ostrým vývojem aplikace existovalo v rámci jmenných prostorů jednotlivých skriptů několik funkcí s návratovým typem void, které zůstaly přítomny i ve finální verzi. Z toho důvodu tak aplikace byla nazvána Project VOID.

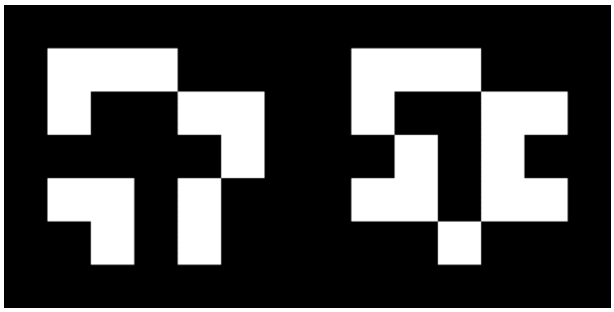
5.2 Optimalizace tvaru a struktury markerů

Vedlejším cílem práce bylo vytvoření a optimalizace nejvhodnější struktury markerů. Během prototypování aplikace i ostrého vývoje bylo vyzkoušeno množství markerů, různých struktur a velikostí. Při čtení markeru aplikace (resp. aplikace prostřednictvím kamery) hledá unikátní vzory (tzv. patterns), které nabývají na čitelnosti se zvětšující se velikostí markeru. Z toho důvodu byl pro výsledné modely využit marker velkých rozměrů. V prvotním testování byl použit obyčejný QR kód (Obr. 5.1), který se však pro potřeby práce ukázal být nevhodným, zejména z důvodu malé velikosti a příliš necharakteristického vzorce. V dalším kroku byly vyzkoušeny referenční markery, které nabízí Unity v rámci AR šablony (Obr. 5.2). Tyto markery obsahovaly dostatečně charakteristický vzorec, avšak ve finále byly taktéž vyhodnoceny jako nevhodné, zejména z důvodu přílišné pestrosti. Z důvodu pestrosti markerů byly testovány i v odstínech šedi, to však znatelně zhoršilo jejich čitelnost. V dalším kroku byly testovány automaticky vygenerované markery pomocí skriptu v jazyce Python (walchko, 2018). Skript byl spuštěn v lokálním prostředí, výsledkem byly celkem čtyři vytvořené markery. Tímto způsobem vygenerované markery (Obr. 5.3) se v prostředí Unity ukázaly jako nedostatečně charakteristické, jelikož neobsahovaly dostatek referenčních bodů (tzv. key points). To mohlo být ovlivněno malým rozlišením nebo příliš jednoduchým vzorem, který by kamera nedokázala rozpoznat (pozn.: Se zmíněnými markery se aplikace nezkompilovala, nebylo tak možné je vyzkoušet přímo).

Obr. 5.1 První z testovaných markerů, v podobě jednoduchého QR kódu (zdroj: autor).



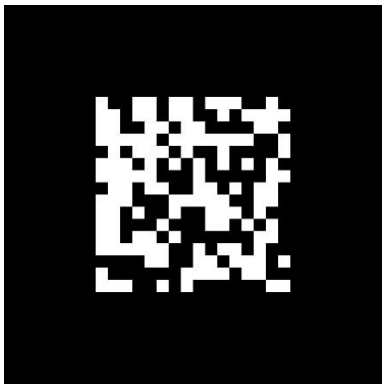
Obr. 5.2 Referenční markery nabízené AR šablonou v softwaru Unity (zdroj: autor).



Obr. 5.3 Příklad markerů vygenerovaných pomocí skriptu v jazyce Python (zdroj: autor).

Další alternativou markerů byly automaticky vygenerované markery, specificky mířené na knihovnu Vuforia či ARToolKit (Shawn Lehner, 2017). Jednalo se o skript v jazyce C#, který generoval markery s dobrou viditelností a dostatkem referenčních bodů pro sledování, respektive čtení markerů. Testovány byly obě varianty, tedy marker optimalizovaný pro knihovnu Vuforia (Obr. 5.4) a marker optimalizovaný pro knihovnu ARToolKit (Obr. 5.5). Ve finále se jako vhodnější ukázala druhá varianta, jelikož vygenerované vzory jsou dostatečně detailní a navíc marker obsahuje pevné vymezení v podobě černého rámce. Ten byl sice obsažen i u první varianty, avšak rozpoznání bylo za některých podmínek poměrně problematické. V průběhu vývoje aplikace se vyskytl problém, který zapříčinil, že při přepínání scén selhalo obnovení automatického zaostřování kamery, a to i v případě explicitního obnovení pomocí skriptu v programu Unity. Důsledkem byla rozostřená kamera telefonu, která dělala detekci markerů značně složitější. Při vývoji aplikace byly testovány dva telefony. Prvním z testovaných strojů byl Samsung Galaxy M31s, s operačním systémem Android 12. Druhým testovaným pak Huawei Nova 3 s operačním systémem Android 9. Při použití první varianty a přítomnosti rozostřené kamery nebylo možné s prvním uvedeným zařízením marker korektně přečíst a tedy zobrazit obsah. Bylo tak zamezeno důležité funkcionalitě. Druhý z telefonů zvládl rozpoznat první marker i přes přítomnost rozostření, s občasně špatným umístěním. Druhá varianta fungovala na obou strojích i v případě rozostření, bylo tak definitivně rozhodnuto pro použití právě těchto markerů. Problém s rozostřením kamery byl ve finále vyřešen, poté byly telefony schopné číst daný marker i z poměrně větší vzdálenosti (přibližně 50 cm). Hodí se zmínit, že rozpoznání markerů bylo do velké míry ovlivněno světelnými podmínkami, což platí pro technologii AR obecně, jelikož využívá senzory telefonu. Pro co nejlepší výsledky při čtení markerů je tak zapotřebí nacházet se v dobře osvětlené místnosti.

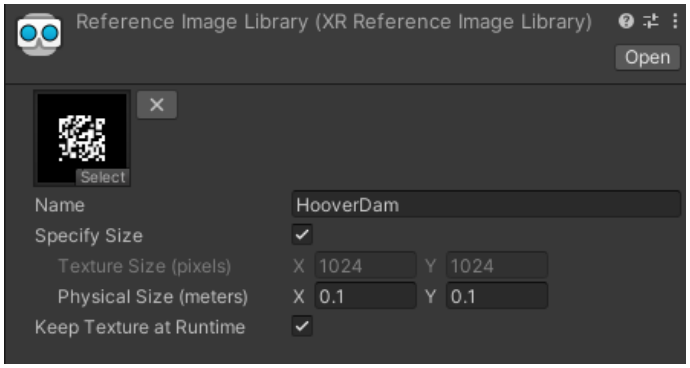
Obr. 5.4 Marker vytvořený pomocí C# skriptu, ve variantě pro knihovnu Vuforia (zdroj: autor).



Obr. 5.5 Příklad markeru vytvořeného pomocí C# skriptu, ve variantě pro knihovnu ARToolKit. Tyto markery byly ve finále využity pro zobrazení AR obsahu (zdroj: autor).

5.3 Vývoj aplikace

Prvním krokem při vývoji aplikace bylo vytvoření primární scény pro zobrazení obsahu ve formě rozšířené reality. Za tímto účelem musel nejdříve být nastaven projekt v prostředí Unity, k čemuž byla využita AR šablona. Zmíněná šablona dělá důležité změny v nastavení projektu. V rámci šablony jsou také již nainstalovány důležité balíčky pro AR (AR Foundation a ARCore XR Plugin). Nejdůležitější změny byly popsány v úvodu podkapitoly 5.1. Přestože byla využita šablona, některá nastavení musela být stále změněna manuálně. Specificky se jednalo o změnu cílové platformy z *Windows, Mac, Linux* na *Android*. Toho bylo docíleno v nastavení pro sestavování aplikací (File → Build Settings). Dále bylo nutno nastavit podporu pro knihovnu ARCore (Edit → Project Settings → XR Plug-in Management). Po úvodním nastavení projektu přišlo na řadu vytvoření knihovny referenčních obrázků (Reference Image Library). Toho bylo docíleno přidáním nového objektu do složky Assets v prostředí Unity (Create → XR → Reference Image Library). Jedná o objekt, do kterého lze přidat obrázky, které reprezentovaly markery pro zobrazení obsahu. Po sestavení aplikace pak aplikace hledá korespondující marker a po přečtení zobrazí daný obsah. Pro jednotlivý marker lze nastavit jeho skutečnou velikost, což je vhodné pro snadnější rozpoznání na marker napojeného obsahu.



Obr. 5.6 Uložení referenčního obrázku do Reference Image Library (zdroj: autor).

Celkem byly do referenční knihovny umístěny čtyři markery, každý pro jeden vytištěný 3D model. Dalším krokem byla úprava importovaných modelů, jejichž vytvoření bylo popsáno v kapitole 4. Předně, data bylo nutné importovat ve správném měřítku, aby výchozí měřítko 1 bylo prostorově rozumně velké. Z toho důvodu byly modely již exportovány v měřítku setině své původní velikosti (0,01) a poté dle potřeby zmenšeny v prostředí programu Unity. Tento krok byl zapotřebí z důvodu jiných jednotek mezi softwary Blender a Unity. Následně bylo u každého modelu nutné změnit rotaci o 45° (ve směru nebo proti směru hodinových ručiček podle nativní orientace objektu). Důvod pro rotaci je rozepsán později v textu. Dalším krokem byla mírná vizuální úprava některých modelů, které sloužily jako jednotlivé geovizualizace (scény) v rámci aplikace. Příkladem budiž model představující první scénu pro model Hoover Dam. Z důvodu nekompletních textur při exportu bylo nutné mosty obsažené v modelu manuálně texturovat. Za tímto účelem byly využity textury betonu, zdarma dostupné v rámci Unity Asset Store (Yughues 2015). Po přiřazení textur byl z modelu vytvořen prefabrikovaný objekt (v Unity frekventován pod pojmem prefab). Odpovídající prefab byl postupně vytvořen pro každý model reprezentující danou scénu. Textury byly přiřazeny také všem objektům, u kterých nedošlo k automatickému přiřazení v rámci importu. Mezi významnější úpravy modelů spadalo také přidání částicových efektů v případě modelů spadajících pod druhou studii (Mount St. Helens). Jedna ze scén byla nachystána se záměrem zobrazení kouře stoupajícího z kaldery. Unity nativně nabízí systém pro správu částicových efektů, vytvořený efekt však nenaplňoval představy. Prostřednictvím Unity Asset Store tak byl zakoupen soubor objektů (tzv. asset) reprezentujících kouřové efekty (Peek 2021). V rámci stejné scény bylo také využito částicových efektů exploze (Unity Technologies 2023c). Poslední významnou úpravou modelů bylo přiřazení skriptu pro rotaci objektu kolem jiného objektu (Obr. 5.7), což bylo využito pro modely v rámci třetí studie (globus). Pomocí skriptu došlo např. k vizualizaci Měsíce obíhajícího kolem Země, s využitím zabudované funkce RotateAround.

```
Unity Script (17 asset references) | 0 references
3  public class RotateObject : MonoBehaviour
4  {
5      // An object to be rotated around.
6      public GameObject sphere;
7      // Another object to be rotated around.
8      public GameObject sphere2;
9      // Speed of the rotation.
10     public float angular_speed;
11
       Unity Message | 0 references
12     void Update()
13     {
14         // If a GameObject (set inside the Unity editor) is not null, executes the function.
15         if (sphere != null)
16         {
17             transform.RotateAround(sphere.transform.position, Vector3.up, angular_speed * Time.deltaTime);
18         }
19         if (sphere2 != null)
20         {
21             transform.RotateAround(sphere2.transform.position, Vector3.up, angular_speed * Time.deltaTime);
22         }
23     }
24 }
```

Obr. 5.7 Skript sloužící pro rotaci objektu kolem jiného objektu (zdroj: autor).

Funkcionalita skriptu spočívá v tom, že po napojení na daný objekt je možné objektu přiřadit až dva další objekty (řádky 6 a 8), kolem kterých obíhá. Lze tak simulovat např. oběh planet kolem Slunce či, v případě využití v aplikaci, oběh Měsíce či jiného tělesa kolem Země. Jelikož proměnné určeny v rámci jmenného prostoru třídy byly nastaveny na typ public, jsou dostupné v rámci editoru Unity. Objektům lze nastavit i rychlost rotace (řádek 10). Následným krokem bylo všechny vytvořené prefabrikované objekty umístit do složky

Resources, aby mohly být později načteny do scény. Následovalo zprovoznění hlavní funkcionality, a sice napsání skriptu ImageTracking, který inicializoval daný AR obsah na základě detekce markeru a umístil tento obsah nad model.

5.3.1 Image Tracking

V rámci ImageTracking třídy se děje několik klíčových operací. Nejdříve dojde k načtení dat umístěných ve složce Resources, a to pomocí funkce LoadPrefab (Obr. 5.8), ve formě herního objektu (GameObject). Následně je pro objekt provedeno vytvoření instance, tedy v zásadě vytvoření kopie objektu. Jelikož Unity se vyznačuje objektově orientovaným přístupem, většina objektů má určitou množinu vlastností a metod, ke kterým lze přistoupit pomocí tečkové notace. Jedním takovým parametrem je prefab.name. Řádek 95 tak přiřadí parametru prefab.name hodnotu argumentu funkce, což je důležité z hlediska další hierarchie třídy ImageTracking. Řádek 96 poté nastaví každý daný objekt na hodnotu false, znamenající jeho vypnutí. Funkce vrací objekt typu GameObject ve formě iniciovaného prefabu. Dalším úkolem třídy ImageTracking je inicializace správného prefabu ve scéně. Jelikož funkce LoadPrefab bere jako parametr řetězec, je do funkce předán název vybraného modelu spolu s indexem. Přiřazení správných hodnot se děje v rámci funkce SetSelectedPrefab, který má dva parametry. Prvním parametrem je jméno vybraného modelu (např. „Hoover Dam“). Druhým parametrem je index modelu, který se pohybuje v rozmezí 01–04, v závislosti na vybrané scéně. Z toho důvodu musely být všechny objekty nacházející se ve složce Resources správně pojmenovány („HooverDam01“, „HooverDam02“, analogicky pro ostatní), jinak by při předání parametru funkci LoadPrefab nedošlo k načtení objektu, respektive objekt by byl hodnoty null, tedy neobsahující žádnou hodnotu. V případě správného pojmenování objektů poté funkce SetSelectedPrefab inicializuje odpovídající objekt, což se v rámci hierarchie aplikace děje ve vedlejším menu aplikace, odpovídající vybranému modelu (vysvětleno později v textu).

```
1 reference
80  private GameObject LoadPrefab(string prefabName)
81  {
82      GameObject go = Resources.Load<GameObject>(prefabName);
83      GameObject prefab = Instantiate(go);
84
85      prefab.name = prefabName;
86      prefab.SetActive(false);
87
88      return prefab;
89  }
```

Obr. 5.8 Funkce LoadPrefab, sloužící pro inicializaci daného objektu (zdroj: autor).

```
18  /// <summary>
19  /// Sets a prefab (to be loaded) accordingly based on the function parameters.
20  /// </summary>
21  /// <param name="name">Name of the selexcted model (e.g. "HooverDam").</param>
22  /// <param name="index">Index of the selected model (e.g. "01").</param>
    12 references
23  static public void SetSelectedPrefab(string name, string index)
24  {
25      selectedModel = name;
26      selectedPrefabIndex = index;
27  }
```

Obr. 5.9 Funkce SetSelectedPrefab, sloužící načtení daného objektu (zdroj: autor).

Po inicializaci správného objektu se v metodě Awake objekt načte. Metoda Awake se děje po každém spuštění scény. Na počátku třídy je definovaná proměnná selectedPrefab, typu GameObject. Tato proměnná dokáže držet referenci pouze na jeden objekt (prefab), což je z hlediska aplikace dostačující, ve scéně se totiž nikdy nevyskytuje více, než jeden objekt. Odpovídající objekt se do proměnné přiřadí pomocí funkce LoadPrefab, parametry funkce jsou jméno vybraného modelu a index vybraného modelu. Následuje vykonání funkce OnTrackedImagesChanged. Parametrem zmíněné funkce je struktura polí retenčních obrázků (tzv. tracked images). Funkce drží tři základní stavy, ve kterých se mohou proměnné v parametru vyskytnout. Zmíněnými stavy jsou přidání, aktualizace a odebrání. Z pohledu aplikace je důležitý pouze stav přidání. V ostatních dvou stavech se v rámci kódu nic neděje. Pro každý přidaný objekt (v tomto případě detekovaný marker) se vykoná funkce OnAddedTrackedImage, která pracuje s parametrem trackedImage, představující v dané chvíli detekovaný marker. V rámci funkce se dějí dvě klíčové věci. Detekovaný marker se z pohledu terminologie OOP stane rodičem k markeru přiřazenému objektu, tedy předává danému objektu svoje prostorové charakteristiky (pozice, rotace, měřítko). Z toho důvodu bylo nutné každý z objektů rotovat o 45°. Markery jsou totiž umístěny na rovině, která je nakloněna o 45° a jelikož marker je rodičem přiřazeného objektu, daný objekt se implicitně inicializuje s rotací 45° v ose Y (ačkoliv v editoru by v tu chvíli byla stále 0). Rotace objektu o 45° ve správném směru (dle nativní pozice) tak kompenzuje iniciální rotaci. Druhou klíčovou záležitostí je aktivace daného objektu. Aby se po přečtení markeru objekt inicializoval na správné pozici, bylo v editoru Unity nutné nastavit odpovídající offset pozice. Z teoretického hlediska se nejedná o složitou problematiku, ukázalo se však, že zobrazovaný obsah je na zadávané hodnoty offsetu velmi citlivý, nikdy tak nebylo dosaženo zcela ideálního překrytí. Výsledný programový kód třídy ImageTracking byl v editoru napojen na objekt ARSessionOrigin, což, jak název vypovídá, je objekt, který slouží jako počátek veškerého AR obsahu, včetně AR kamery a obsahu inicializovaného ve scéně. Výsledkem byla scéna ImageTracking, schopná detekovat daný marker a zobrazit obsah. Scéna obsahovala doplňkovou funkcionalitu, kterou naznačují UI prvky (Obr. 5.11).

```
100     /// <summary>
101     /// For selected tracked image, sets the tracked image as prefab parent and enables it.
102     /// </summary>
103     /// <param name="trackedImage">A tracked image from the Reference Image Library.</param>
        1 reference
104     private void OnAddedTrackedImage(ARTrackedImage trackedImage)
105     {
106         // If tracked image name is not equal to selected model name, return.
107         if (trackedImage.referenceImage.name != selectedModel)
108         {
109             Debug.Log($"Debug: OnAddedTrackedImage failed to match tracked image \"{trackedImage.referenceImage.name}\" with selected model \"{selectedModel}\"");
110             return;
111         }
112
113         selectedPrefab.transform.parent = trackedImage.transform;
114         selectedPrefab.SetActive(true);
```

Obr. 5.10 Funkce OnAddedTrackedImage, sloužící pro určení rodiče a aktivaci (zdroj: autor).

Obr. 5.11 Náhled na ImageTracking scénu a UI prvky, které obsahuje (zdroj: autor).

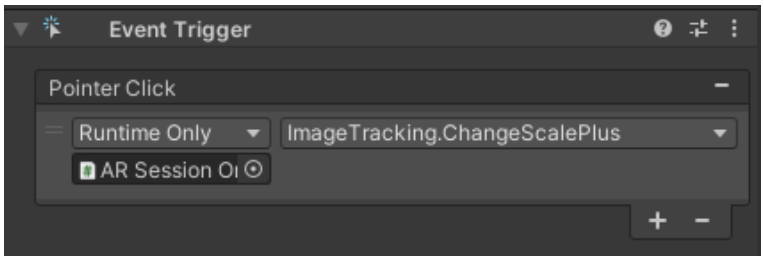
Jako první si lze povšimnout dvou tlačítek v dolní části obrazovky. Tlačítko napravo slouží pro vypnutí aplikace, tlačítko vlevo potom pro návrat do odpovídajícího vedlejšího menu, na základě zvoleného modelu. Každé vedlejší menu je reprezentováno vlastní scénou, přechody se dějí na základě názvu scény. V horní části obrazovky lze v levé části naleznout dvě tlačítka. Tlačítko ADV slouží pro zobrazení tzv. pokročilého módu. Funkcí tohoto módu je možnost si přizpůsobit měřítko a pozici (individuálně pro každou z os X, Y, Z) o pevně danou hodnotu. Specifická hodnota je 0,1 (v kladném či záporném směru). Z matematického hlediska se jedná o přičítání či odčítání vektoru o hodnotě (0,1, 0, 0) pro případ osy X, obdobně pro ostatní osy, se změněnou odpovídající souřadnicí. Při změně měřítka dochází k přičtení či odečtení vektoru (0,1, 0,1, 0,1). Tlačítko zcela vpravo nahoře poté slouží pro zobrazení informačního textu, který koresponduje s vybraným modelem. To znamená, že pokud si uživatel zvolí model HooverDam, potom se po kliknutí na tlačítko, zobrazí informační box s textem obsahujícím základní informace o Hooverově přehradě (analogicky pro ostatní modely). Informace se doplňují dynamicky na základě vybraného modelu, a to prostřednictvím samostatného skriptu, který má na starost práci s textem. Funkce pro změnu pozice a měřítka je obsažena v rámci skriptu ImageTracking, jelikož modifikuje hodnoty objektu existujícího v rámci jmenného prostoru onoho skriptu. Funkce pro ostatní tlačítka jsou definovány v samostatném skriptu InGameMenu. Poslední důležitou funkcionalitou třídy ImageTracking je deinicializace třídy XRLoader, což je základní třída pro všechny AR implementace, mezi něž patří i v práci využitý framework AR Foundation. Právě deinicializace a poté opětovná inicializace třídy při načítání dat řešila problém s rozostřením kamery, který je uveden v podkapitole 5.2.

```
0 references
133  public void ChangePositionXPlus()
134  {
135      Vector3 vec = selectedPrefab.transform.position;
136      selectedPrefab.transform.position = new Vector3(vec.x + 0.1f, vec.y, vec.z);
137  }
138
```

Obr. 5.12 Funkce ChangePositionXPlus, sloužící pro změnu pozice objektu v ose X o danou hodnotu (zdroj: autor).

Funkce, které mají příznak public lze využít v prostředí editoru Unity, čehož bylo využito pro napojení daných funkcí na odpovídající tlačítka. Každému objektu ve scéně jde v editoru přiřadit komponenta Event Trigger. Jedná se o komponentu, která zpracovává

události na objektu po vykonání určité akce. Mezi akce patří např. Drag, Drop, Select nebo PointerClick. Právě PointerClick, v mobilním prostředí znamenající dotknutí se obrazovky, byl využit při vývoji aplikace. Každému tlačítku ve scéně, které mělo vykonávat nějakou akci (funkci) po dotknutí se obrazovky tak byla přiřazena komponenta Event Trigger s akcí PointerClick. Mimo volání programátorem definovaných funkcí může být vykonanou událostí např. vypnutí určité části UI. Jelikož všechny komponenty v programu Unity jsou ve své podstatě objekty, lze s nimi i tak pracovat a daný objekt jednoduše učinit neaktivním (GameObject.SetActive(false)) nebo aktivním (GameObject.SetActive(true)). Funkcionality bylo značně využito zejména při tvorbě menu aplikace.



Obr. 5.13 Komponenta Event Trigger a na ni napojena událost v editoru Unity (zdroj: autor).

5.3.2 Plane Detection

Kromě ImageTracking obsahuje aplikace i druhou scénu, která pracuje na principu detekce ploch v prostoru a následném umístění obsahu po kliknutí na obrazovku, v případě, že se místo dotyku protíná s rovinou. Pro tyto účely byla definována třída TapToPlace. Podstatná funkcionalita skriptu se vykonává v metodě Update, což znamená, že hodnoty či informace se aktualizují každý snímek. V úvodu třídy dojde k deklaraci důležitých proměnných a poté již ve zmíněné metodě Update dojde k posloupnosti příkazů, které zabezpečují zobrazení obsahu v tom momentě, kdy se uživatel dotkne obrazovky.

```
9       // GameObject to be spawned.
10      private GameObject spawnedObject;
11      // Position on the screen that the user touches.
12      private Vector2 touchPosition;
13      // Reference to AR Raycast Manager in Unity.
14      private ARRaycastManager arRaycastManager;
15      // Reference to AR Plane Manager in Unity.
16      private ARPlaneManager arPlaneManager;
17      // List that stores raycast hits (gets information back from a raycast).
18      private List<ARRaycastHit> hits = new List<ARRaycastHit>();
```

Obr. 5.14 Deklarace používaných proměnných v rámci třídy TapToPlace s odpovídajícím komentářem (zdroj: autor).

Posloupnost akcí je pak přibližně následující: Jestliže se uživatel dotknul obrazovky, získá se pozice tohoto dotyku. Poté, za předpokladu, že proměnná spawnedObject obsahuje hodnotu null, dojde k načtení odpovídajícího modelu. Modely jsou, shodně s ImageTracking scénou, umístěny ve složce Resources v editoru Unity. Proces přiřazení správného modelu je téměř identický, pracuje se s názvem modelu (např. „HooverDam“) a indexem modelu (např. „01“). Rozdílem je, že v případě TapToPlace třídy se mezi dva zmíněné řetězce přidá řetězec „Plane“. Celý název daného modelu je tak např. „HooverDamPlane01“. Jestliže je objekt s daným názvem nalezen, pak proběhne vytvoření

instance (kopie objektu). Poté, za předpokladu, že je aktivní manažer detekovaných rovin ve scéně, dochází k aktualizaci pozice na základě dotyku uživatele. Tedy, při prvním dotyku dojde k přidání modelu do scény, při opakovaných dotycích je pozice měněna v souladu s tím, zda se uživatel dotknul místa, ve kterém existuje detekovaná rovina. Výsledný kód třídy TapToPlace je v editoru Unity napojen na objekt AR Session Origin. Výsledkem byla scéna PlaneDetection, schopná v reálném prostředí detekovat virtuální roviny a na ně umístit obsah. Stejně jako scéna ImageTracking, i scéna PlaneDetection obsahovala doplňkovou funckionalitu, naznačenou UI prvky (Obr. 5.15).

Stejně jako v předešlém případě, i scéna PlaneDetection obsahuje v dolní části obrazovky tlačítko pro vypnutí aplikace, jakož i tlačítko pro návrat do odpovídajícího vedlejšího menu. Shodný je také informační box v horní pravé části obrazovky. V dolní části nad tlačítkem pro vypnutí aplikace se nachází dvě tlačítka ROT+ a ROT-. Tato tlačítka slouží pro změnu rotace ve scéně umístěného objektu. Rotace probíhá pouze v ose Y, a to o hodnotu 15°. Z pohledu matematiky jde opět o přičtení či odečtení vektoru, specificky vektoru (0, 15, 0). Jelikož rotace v Unity je nativně typu Quaternion (systém rotace využívající čtyři souřadnice namísto tří), bylo nutné v rámci funkce pro změnu rotace provést konverzi na eulerovské úhly. Posledním UI elementem je tlačítko sloužící pro vypnutí, respektive zapnutí detekovaných ploch. Detekovaným plochám byl v editoru Unity přiřazen materiál, aby uživatel dostával vizuální zpětnou vazbu o tom, zda se v daném prostoru podařilo detekovat plochu či nikoliv (může trvat až desítky sekund v závislosti na světelných podmínkách).

Obr. 5.15 Náhled na PlaneDetection scénu a UI prkvy, které obsahuje (zdroj: autor).

Poté, v případě, že uživatel nevyžaduje změnu pozice, lze tlačítkem vypnout všechny v dané době detekované plochy, což zamezí detekci dalších ploch i schopnosti aktualizovat pozici. Funkce funguje na principu vypnutí či zapnutí komponenty AR Plane Manager a rotaci booleovských hodnot. Právě po vypnutí detekovaných ploch je následně vhodné rotovat s modelem, jelikož jinak může nastat situace, že po stisknutí tlačítka pro rotaci se změní i pozice objektu, jelikož se tlačítko kryje s detekovanou plochou. Všechny pro scénu unikátní funkce sdílí jmenný prostor s třídou TapToPlace.

```
88      /// <summary>
89      /// Enables/disables tracking planes based on the previous state.
90      /// </summary>
        0 references
91      public void ToggleTracking()
92      {
93          bool newState = !arPlaneManager.enabled;
94          foreach (var plane in arPlaneManager.trackables)
95          {
96              plane.gameObject.SetActive(newState);
97          }
98          arPlaneManager.enabled = newState;
99      }
100 }
```

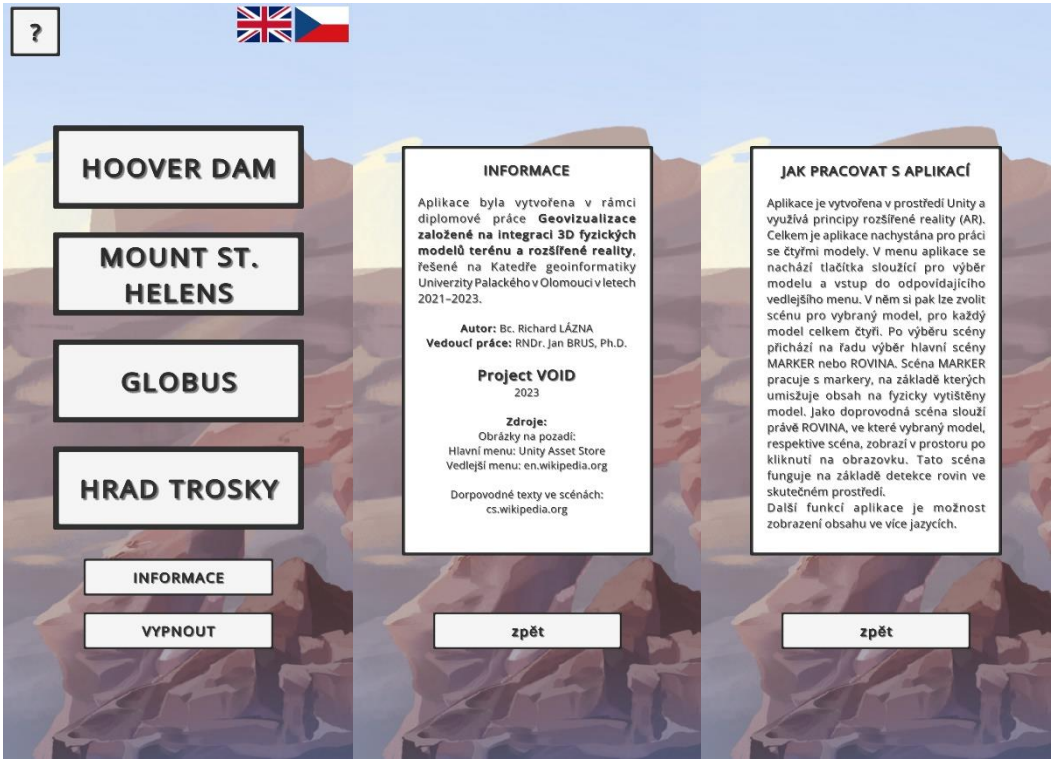
Obr. 5.16 Funkce ToogleTracking, sloužící pro vypnutí a zapnutí detekovaných ploch (zdroj: autor).

5.3.3 Hlavní menu aplikace

Další součástí aplikace je hlavní menu, které slouží zejména pro další navigaci. Menu existuje v rámci vlastní scény a zobrazí se po zapnutí aplikace. Obsahuje několik tlačítek, které odkazují na další elementy v rámci scény nebo další scény aplikace. Funckionalita hlavního menu je ovládána pomocí Event Trigger komponent, které byly popsány v podkapitole 5.3.1. Funkce, které jsou v rámci komponent volány, existují ve vlastní třídě (skriptu) s názvem MainMenu. V rámci třídy je definováno několik stěžejních funkcí, především obstarávající management scén v aplikaci. Za tímto účelem je využita knihovna SceneManagement, kterou nabízí program Unity. Referenci na jednotlivé scény lze získat indexem scény nebo zadáním názvu v podobě textového řetězce. Při vývoji aplikace bylo využito druhé možnosti. Jmenný prostor třídy MainMenu sdílí i scény jednotlivých vedlejších menu, z toho důvodu obsahuje i funkce, které se volají až v rámci některého ze čtyř vedlejších menu.

Předně, obsaženy jsou funkce pro přepnutí do odpovídajícího vedlejšího menu na základě stisknutí korespondujícího tlačítka a pro také pro vypnutí aplikace. Ostatní funkce slouží až funkcionalitě vedlejších menu. Některá logika menu aplikace je řešena přes Event Trigger vypnutím/zapnutím některého ze zvolených elementů scény. Podobu menu, včetně doplňujících informací ve formě textových polí, lze vidět na Obr. 5.17. Obrázek využitý na pozadí menu byl získán z Unity Asset Store (suhyun, 2019).

Doplňující obsah je dostupný po kliknutí na korespondující tlačítko. Pokud chce uživatel zobrazit informace o aplikaci, slouží k tomu tlačítko INFORMACE v dolní části menu. Jestliže chce zobrazit návod k aplikaci, stiskne tlačítko „?“. Všechen doplňující obsah je zobrazován v rámci stále stejné scény, pomocí komponenty Event Trigger je vypnut nebo zapnut odpovídající obsah.



Obr. 5.17 Hlavní menu aplikace (vlevo), včetně doplňujících informací o aplikaci (uprostřed) a návodu k aplikaci (vpravo) (zdroj: autor).

V menu lze spatřit ještě jednu doplňující funkcionalitu, kterou představují česká a britská vlajka v pravém horním rohu menu. Z pohledu UI se jedná o tlačítka. Tato tlačítka slouží pro přepnutí jazyka aplikace do anglického, respektive do českého jazyka. Pro danou funkcionalitu byl napsán skript TextHandler. Daný skript byl poté připojen ke každému textovému objektu napříč scénami. V prostředí editoru je každému takovému objektu přiřazeno unikátní ID, na základě kterého je přiřazen text. V rámci skriptu se získá reference na daný textový objekt a poté se na základě podmínek vyhodnocujících dané ID textu a aktivní jazyk přiřadí odpovídající text. Kratší texty jsou přiřazeny přímo, delší texty jsou uloženy do proměnných, které jsou poté použity u korespondujících ID. Změna jazyka je perzistentní mezi scénami díky využití statické proměnné, která určuje v dané chvíli aktivní jazyk. V rámci jednotlivých selektorů je také nastavena velikost fontu. Obr. 5.18 a Obr. 5.19 demonstrují část programového kódu třídy TextHandler. Pro přepnutí jazyka slouží na tlačítka napojené funkce přes Event Trigger, které slouží pro změnu logické hodnoty. Pravdivá hodnota proměnné (lang = true) koresponduje s českým jazykem, nepravdivá hodnota (lang = false) s anglickým jazykem. Skript TextHandler se stará také o zobrazení korespondujícího textu po kliknutí na informační tlačítko ve scénách ImageTracking a PlaneDetection. Za tímto účelem pracuje s hodnotou selectedModel z třídy ImageTracking, která je nastavena na hodnotu static, což znamená globální proměnnou, proto je možné k ní přistoupit z jiného jmenného prostoru.

```
423         if (textID == "exit_button")
424         {
425             if (lang)
426             {
427                 displayedText.text = "VYPNOUT";
428             }
429             else
430             {
431                 displayedText.text = "EXIT";
432             }
433             displayedText.fontSize = 50;
434         }
435     }
```

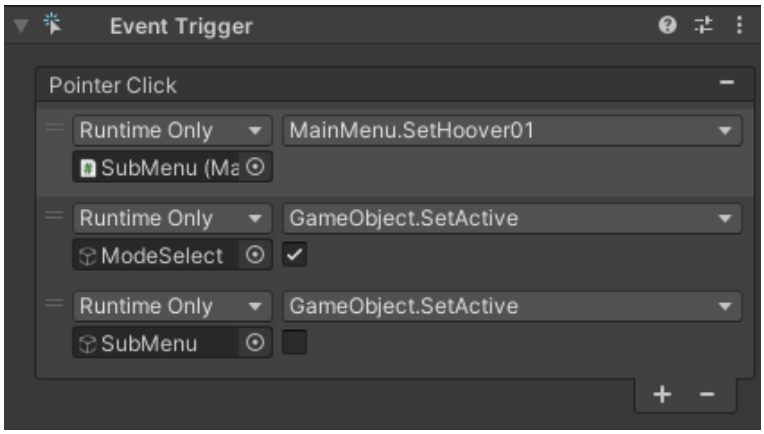
Obr. 5.18 Část třídy TextHandler, sloužící pro přepínání jazyka podle ID textu (zdroj: autor).

```
372         if (textID == "info_box_text")
373         {
374             if (ImageTracking.selectedModel == "HooverDam")
375             {
376                 if (lang)
377                 {
378                     displayedText.text = hooverText;
379                 }
380                 else
381                 {
382                     displayedText.text = hooverText_EN;
383                 }
384                 displayedText.fontSize = 30;
385             }
```

Obr. 5.19 Část třídy TextHandler, sloužící pro zobrazení korespondujícího textu dle vybraného modelu (zdroj: autor).

5.3.4 Vedlejší menu aplikace

Po výběru některého z modelů (kliknutím na odpovídající tlačítko) se uživatel dostane do jednoho ze čtyř vedlejších menu. I vedlejší menu, stejně jako hlavní, z velké části využívá schopností komponenty Event Trigger pro volání funkcí nebo změnu stavu některého z prvků UI. V každém vedlejším menu se nacházejí zejména čtyři tlačítka, každé odkazující na jednu z nachystaných geovizualizací pro vybraný model. Pro korektní inicializaci dané scény jsou využity funkce implementovány v rámci třídy MainMenu, které využívají funkci SetSelectedPrefab z třídy ImageTracking. Jak již bylo uvedeno, funkce SetSelectedPrefab očekává dva parametry. Jedním parametrem je jméno modelu a druhým index modelu. Index modelu odpovídá indexu daného scénáře ve vedlejším menu. Parametrů ve funkci SetSelectedPrefab dále využívá funkce LoadPrefab, starající se o vytvoření instance objektu ve scéně. Z toho důvodu je v rámci třídy MainMenu pro každý scénář (každou geovizualizaci) definována funkce, v jejímž těle se vykoná funkce SetSelectedPrefab s odpovídajícími parametry (např. „HooverDam“ a „01“). Z toho důvodu bylo důležité správně pojmenovat všechny objekty ve složce Resources v editoru Unity, jinak nikdy nedojde k načtení daného modelu a tedy ani k zobrazení při přečtení markeru. Jelikož vždy existuje pouze jeden aktivní model, může existovat pouze jedna aktivní scéna, tedy scéna ImageTracking, ve které se inicializuje odpovídající objekt. Takto inicializovaný objekt zaniká po tom, co uživatel scénu opustí (tlačítko zpět, vypnutí aplikace).



Obr. 5.20 Příklad inicializace objektu ve scéně pomocí komponenty Event Trigger (zdroj: autor).

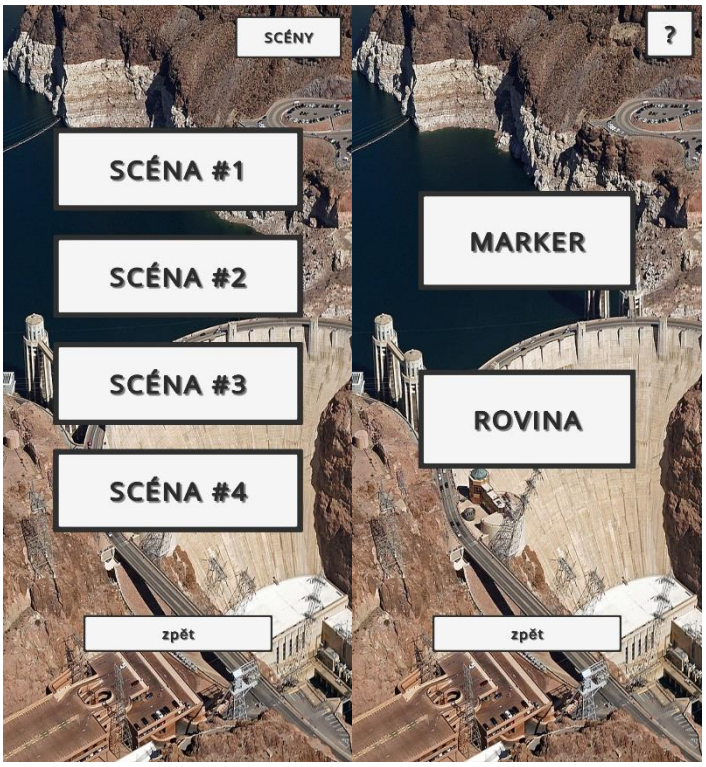
```
0 references
public void SetHoover01()
{
    ImageTracking.SetSelectedPrefab("HooverDam", "01");
    LoaderUtility.Initialize();
}
```

Obr. 5.21 Funkce pro inicializaci objektu a XRLoader komponenty (zdroj: autor).

Lze si povšimnout, že v rámci funkcí pro inicializaci objektů probíhá také inicializace XRLoader komponenty, která byla zmíněna v podkapitole 5.3.1. Funkce analogické k funkci SetHoover01 jsou vykonány pro každou scénu daného modelu, celkem se tedy jedná o 16 funkcí (čtyři pro každý model). V rámci každého vedlejšího menu nastává po výběru daného scénáře také výběr módu. Uživatel volí mezi dvěma módy, které odpovídají funkcionalitám popsaným v podkapitolách Image Tracking a Plane Detection.

Podrobně jedno z vedlejších menu zachycuje Obr. 5.22. V části menu pro výběr scény si lze všimnout v pravém horním rohu tlačítka SCÉNY. Po kliknutí na toto tlačítko se uživateli zobrazí informace o dostupných scénách formou informačního boxu. Zobrazení funguje pomocí komponenty Event Trigger, která po kliknutí na tlačítko učiní ostatní obsah neaktivním (s výjimkou obrázku na pozadí a tlačítka zpět) a zobrazí pouze informační box, jehož obsah se mění v závislosti na konkrétním vedlejším menu. Při výběru módu pak po kliknutí na tlačítko MARKER nebo tlačítko ROVINA dojde k inicializaci odpovídající scény (ImageTracking nebo PlaneDetection). Za tímto účelem jsou využity funkce, které jsou definovány v rámci třídy MainMenu.

Výběr módu také nabízí doplňkovou funckionalitu. V pravém horním rohu se nachází tlačítko „?“, které po kliknutí uživateli zobrazí informační box s náhledem na v dané chvíli vybranou scénu, ve formě obrázku. Obrázek je do informačního boxu doplňován dynamicky pomocí samostatného skriptu ImageHandler. Z interního pohledu aplikace je informační box tlačítkem, skript tak funguje na principu získání reference na tlačítko a poté načtení odpovídajícího obrázku, načteného pomocí funkce Resources<Load>. Stejnou funkcí probíhá i načítání modelů do scény. Pro správné načtení byla vytvořena ve složce Resources podsložka Images, obsahující všechny obrázky určené pro načtení. Skriptu je v metodě OnEnable (při zapnutí) přiřazen konkrétní obrázek na základě proměnných selectedModel (jméno modelu, např. „HooverDam“) a selectedPrefabIndex (index modelu, např. „01“).



Obr. 5.22 Jedno z vedlejších menu aplikace, ukazující výběr scény (vlevo) a výběr módu (vpravo) (zdroj: autor).

```
27      0 references
        public void GetToMainScene()
28      {
29          SceneManager.LoadScene("ImageTracking");
30      }
31
32      0 references
        public void GetToPlaneScene()
33      {
34          SceneManager.LoadScene("PlaneDetection");
35      }
36
```

Obr. 5.23 Funkce pro inicializaci zvolené scény v rámci vedlejšího menu (zdroj: autor).

```
13      private void OnEnable()
14      {
15          var newSprite =
16              Resources.Load<Sprite>
17              ($"Images/{ImageTracking.selectedModel + ImageTracking.selectedPrefabIndex}");
18
19          sceneButton.image.sprite = newSprite;
20      }
```

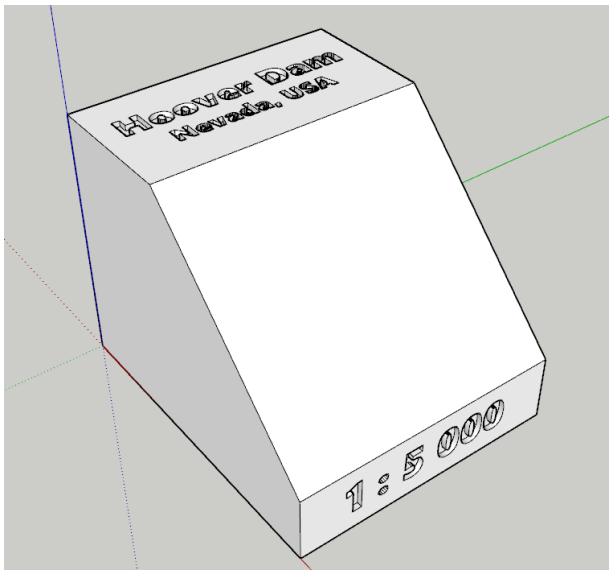
Obr. 5.24 Dynamické načítání obrázků jednotlivých scén (zdroj: autor).

6 FINALIZACE PRÁCE

Posledním krokem vlastního řešení byla finalizace práce, která obnášela zejména přípravu modelů na 3D tisk pomocí programu QGIS a také optimalizace aplikace, zahrnující optimalizaci velikosti výsledného APK souboru. Byly také vytvořeny plochy na umístění markerů. Mezi další záležitosti při finalizaci práce patří také úprava offsetu pro zobrazené objekty v případě finálně vytištěných modelů.

6.1 Příprava modelů na 3D tisk

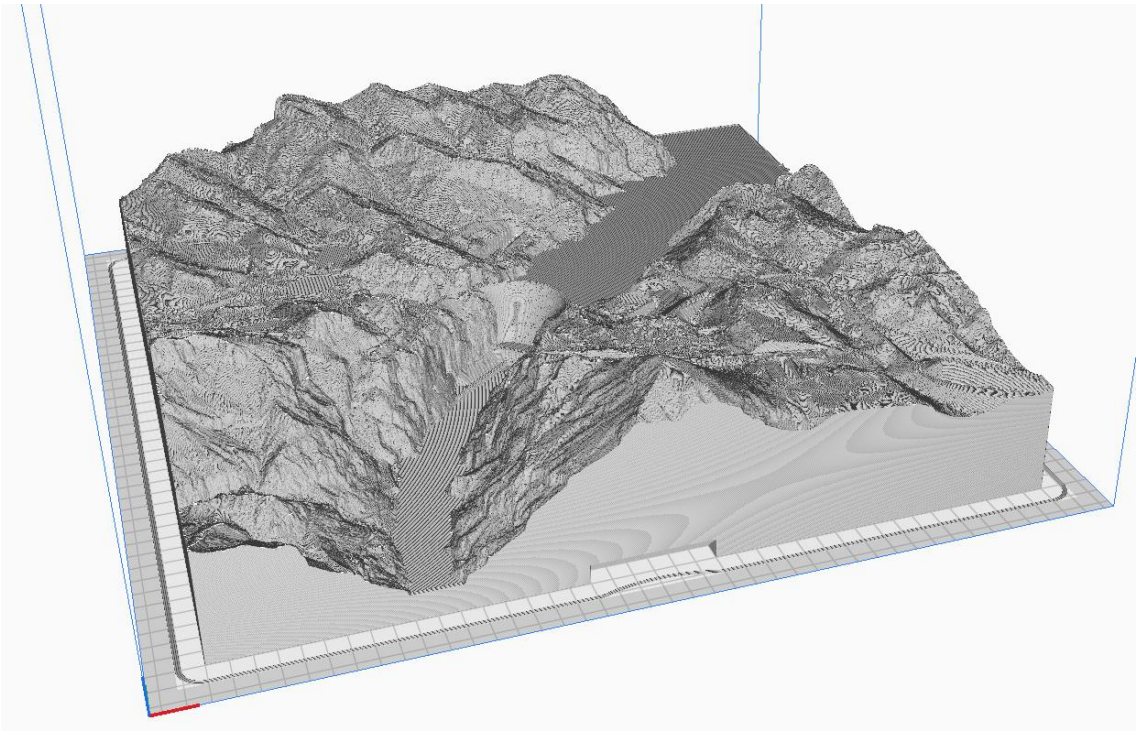
Nejprve bylo nutné v prostředí programu QGIS vytvořit STL soubory ze zdrojových dat ve formátu TIF. Za tímto účelem byl využit plugin DEMto3D, dostupný přes nabídku Raster. V případě první studie bylo v nastavení pluginu vybráno měřítko 1 : 5000. Rozměry vygenerovaného STL tak činily 400,2 × 400,2 mm (přibližně 40 × 40 cm). V případě druhé studie bylo vybráno měřítko 1 : 12 500, rozměry modelu tak činily 400,8 × 400,8 mm. Po vytvoření STL souborů bylo nutné provést několik úprav. Předně, v programu SketchUp došlo k vytvoření podstavy modelů. Podstava měla shodné rozměry jako vygenerované modely, tedy 40 × 40 cm. Důležitou součástí podstavy bylo vytvoření otvoru na jedné straně podstavy, o rozměrech 6 × 1 cm a sklonem 45°. Takto vytvořený otvor posloužil pro napojení dílu, na kterém byla umístěna plocha pro umístění markeru. Rozměr dílu byl dimenzován tak, aby bylo možné využít 10 × 10 cm marker, tedy samotná plocha, nakloněna o 45°, musela být shodných rozměrů. Plocha byla nakloněna o 45° z důvodu uživatelsky přívětivějšího čtení markeru, než v případě, kdy by naklonění provedeno nebylo, nebo by díl zcela chyběl a marker by byl umístěn přímo na modelu. Díl pro umístění markeru obsahuje výstupek o rozměrech 6 × 1 cm a sklonem 45° v opačném směru, tvoří tak protidíl k podstavě modelu. Dalším krokem bylo spojení daného modelu, vygenerovaného ze zdrojového TIF souboru, s vytvořenou podstavou. Spojení proběhlo pomocí programu Blender, spojený model byl následně exportován ve formátu STL. Takto vyexportovaný model byl připraven na import do programu Ultimaker Cura.



Obr. 6.1 Vytvořený díl pro umístění markeru (zdroj: autor).

V prostředí programu Ultimaker Cura bylo zapotřebí učinit několik klíčových nastavení a poté model tzv. naslicovat (rozdělit na jednotlivé vrstvy. Důležitým nastavením byla výška vrstvy, která byla, vzhledem k velikosti modelu, zvolena na 0,2 mm. Bylo by možné zvolit i např. 0,1 mm pro větší detail, to by však významně zvýšilo čas tisku. Dalším důležitým nastavením bylo zvolení výplně 3D modelu. Vybrána byla výplň Lightning, která oproti klasickým výplním funguje na principu vystavění výplně postupně od zdola nahoru, přičemž se rozpíná jako větve stromu či právě blesky (odtud také název Lightning). Výhodou této výplně je významná úspora materiálu. Následně byl proveden slicing, jehož výstupem byl G-code, tedy soubor pokynů pro tiskárnu. Se stejným nastavením byly vytištěny i díly s plochami pro umístění markeru. Pro tisk byla využita tiskárna Creality CR-10 Max, jejíž hlavní devízou je velký tiskový prostor (450 × 450 × 470 mm). Tiskárna byla vybrána právě z důvodu velkého tiskového prostoru, aby bylo možné vytisknout dané 3D modely bez nutnosti je dělit na více částí.

Příprava na tisk v případě modelu globu Země byla poněkud odlišná. Jelikož se jednalo už hotový 3D model, nebylo zapotřebí vytvářet STL soubor z výškových dat. Bylo však nutné model upravit pro potřeby 3D tisku. Původní model využitý pro tisk globu byl rozdělen na dvě poloviny, muselo tak dojít ke spojení a manuální úpravě vnitřní části modelu tak, aby se jevil jako jedna souvislá část. Také bylo nutné model upravit v programu Netfabb a eliminovat tak chyby, které by mohly způsobit problémy v procesu tisku. Dále byla vytvořena podstava kruhového tvaru, na které byl globus umístěn. Tato podstava shodně obsahovala otvor (v tomto případě menších rozměrů, z důvodů rozměru podstavy) k napojení dílu s plochou pro umístění markeru. Pro případ posledního modelu byla z důvodu nízkého rozlišení dat vytištěna pouze verze v menším měřítku, s rozměry přibližně 160 × 160 mm. Postup byl analogický jako v případě první a druhé studie, s tím rozdílem, že rozměry plochy pro umístění markeru byly upraveny na 5 × 5 cm.



Obr. 6.2 Náhled na 3D model Hooverovy přehrady, připravený na tisk (zdroj: autor).

6.2 Úprava offsetu a optimalizace aplikace

S vytištěnými modely byla dalším krokem úprava offsetu modelu při detekci markeru. Implicitní chování Marker-Based AR je takové, že po přečtení markeru, který slouží jako referenční bod v prostoru, se konkrétní obsah zobrazí přímo na daném markeru. Popsané chování bylo z pohledu cíle práce nedostačující, bylo tak potřeba upravit pozici zobrazeného obsahu, čehož bylo docíleno v editoru programu Unity zadáním hodnot transformace pozice v osách Y (nahoru a dolů) a Z (dopředu, dozadu). Hodnoty byly notně jiné, než v případě modelů vytištěných pro potřeby testování. Bylo tak nutné provést úpravu hodnot tak, aby zobrazený obsah co nejlépe překryl vytištěný 3D model.

Dalším krokem byla optimalizace aplikace. V rámci optimalizace proběhlo zmenmšení velikosti některých pro modely využitých textur. Během vývoje také existovalo množství testovacích modelů. Bylo tak zapotřebí z projektu odstranit veškeré nepotřebné objekty pro zmenšení celkové velikosti aplikace. Posledním bodem byla optimalizace z pohledu výpočetního výkonu. V prostředí Unity lze po přiřazení textury modelu vybrat tzv. shader (instrukce, které určují, jakým způsobem bude určitý grafický objekt vykreslen, včetně nasvícení). Při testování aplikace bylo zjištěno, že některé shadery jsou z pohledu mobilní aplikace příliš náročné a vedou k významnému poklesu snímků za sekundu. Z tohoto důvodu musely být některé modely (reprezentující jednotlivé scény) upraveny změnou využitého shaderu. Další optimalizací z hlediska výpočetního systému bylo také přesunutí některé funkcionality mimo metodu Update. Jelikož metoda Update se děje každý jednotlivý snímek, může (podle náročnosti vykonané operace) představovat nechtěné břímě z hlediska výkonu.

6.2.1 Úprava aplikace pro modely v menším měřítku

Z důvodu technických limitací byla výsledná mobilní aplikace upravena tak, aby již vytvořené vizualizace mohly být využity pro 3D modely s menším měřítkem. Z toho důvodu byl skript ImageTracking rozšířen o třídu OffsetData, která na základě jména modelu přiřazovala hodnoty transformace (pozice a měřítka, rotace zůstala shodná). Další úpravou v ImageTracking skriptu bylo také přidání schopnosti rozeznat marker pro model v menším měřítku, a to na základě jména markeru. Za tímto účelem byla rozšířena Reference Image Library o několik markerů. Modely v menším měřítku byly vytištěny na tiskárně Prusa i3 MK3S a po vytištění vybaveny odpovídajícím markerem.

```
2 references
203  static public class OffsetData
204  {
         1 reference
205      static public Vector3 GetSmallPosition(string modelName)
206      {
207          switch (modelName)
208          {
209              case "HooverDam": return new Vector3(0f, -0.4f, 0.16f);
210              default: return Vector3.zero;
211          }
```

Obr. 6.3 Třída OffsetData a jedna z funkcí na změnu transformace objektu (zdroj: autor).

7 VÝSLEDKY

Hlavním výsledkem diplomové práce je vytvořená mobilní aplikace pro operační systém Android ve verzi 8.1 a novější. Aplikace využívá technologii rozšířené reality, která je implementována s využitím herního enginu Unity. V rámci Unity byla pro tvorbu využita knihovna AR Foundation, která slouží jako prostředník mezi knihovnou ARCore, která dovoluje AR funkcionalitu v prostředí systému Android, a enginem Unity. Aplikace obsahuje hlavní menu, ve kterém si uživatel zvolí jeden ze čtyř modelů. Další funkcionalitou menu jsou doplňující informace k aplikaci, návod na práci s aplikací či možnost přepnout jazyk aplikace. Pro každý ze čtyř modelů aplikace obsahuje jedno vedlejší menu s výběrem čtyř geovizualizací. Celkem tak aplikace obsahuje 16 geovizualizací, s širokým záběrem obsahu, od jednoduchých textur po simulaci kouře nebo simulaci planetárního oběhu. Přehled vytvořených geovizualizací ukazuje Obr. 7.1. Ve vedlejším menu si lze zobrazit seznam dostupných scén, s krátkým popisem toho, co uživatel uvidí v rámci vybraného scénáře. Po výběru daného scénáře uživatelem si uživatel zvolí režim zobrazení obsahu. Dostupnými režimy jsou MARKER a ROVINA. Před výběrem režimu si uživatel může zobrazit přibližnou vizuální podobu toho, co v rámci scény uvidí. Po výběru režimu je následně uživatel odkázán korespondující hlavní scény (Image Tracking nebo Plane Detection).

Scéna Image Tracking využívá z pohledu rozšířené reality Marker-Based AR, která v rámci aplikace funguje na bázi detekce referenční značky (markeru) a umístění obsahu na daný marker. V rámci scény Image Tracking uživatel kamerou telefonu či jiného zařízení s operačním systém Android sejme daný marker reprezentující obsah ve formě rozšířené reality. Pro každý model existuje jeden unikátní marker, celkem tedy čtyři markery. Po detekci daného markeru je nad vytištěným 3D modelem reliéfu zobrazen virtuální obsah v podobě textur, animací či doplňujících 3D objektů. Obsah je umístěn nad model pomocí offsetu od pevně umístěného markeru. Offset vytvořeného objektu lze uživatelsky upravit o pevně danou hodnotu, stejně tak měřítko. Lze také zobrazit doplňující informace o oblasti související s daným vytištěným modelem (např. pro model oblasti Hoover Dam se zobrazí informace o Hooverově přehradě). Vedlejším výstupem práce je nalezení nástroje pro generování unikátních markerů, které se vyznačují žádoucími vlastnostmi dle provedeného testování markerů autorem práce. Jelikož AR v prvé řadě detekuje na daném markeru charakteristické vzory, které jsou s využitím nalezeného generátoru dostatečně unikátní, jedná se o obecně využitelný nástroj pro další AR aplikace.

Scéna Plane Detection využívá z pohledu rozšířené reality Markerless AR, která v rámci aplikace funguje na bázi detekce horizontálně umístěných virtuálních rovin v reálném prostředí. Po detekci plochy ve scéně se lze plochy dotknout a vyvolat objekt odpovídající vybranému scénáři ve vedlejším menu. Objekty představující jednotlivé geovizualizace byly pro potřeby Plane Detection upraveny tak, aby se jevily jako validní, uzavřené 3D objekty s nenulovým objemem. Po umístění objektu do scény lze kameru telefonu nebo jiného zařízení umístit blíže k objektu a prozkoumat tak množství detailů. Detekovanou rovinu lze stisknutím odpovídajícího tlačítka skrýt. Uživatel také může modelem rotovat v ose Y o pevně daných 15° v kladném či záporném směru.

Z interního pohledu logiky aplikace se aplikace skládá celkem ze sedmi scén. První scéna je reprezentována hlavním menu. Poté následují čtyři scény, každá pro jedno vedlejší menu. Hlavní obsah aplikace tvoří scény 6 a 7 (z pohledu indexace 5 a 6), tedy scény Image Tracking, respektive Plane Detection. Samotné scény pro každou vytvořenou geovizualizaci nejsou zapotřebí, jelikož aplikace využívá dynamického načítání dat do hlavní scény až podle toho, co si uživatel vybere v hlavním, respektive zejména ve vedlejším menu. Až po

této uživatelské volbě dojde k načtení odpovídajících dat, která následně zanikají při vrácení se do vedlejšího či hlavního menu, nebo vypnutí aplikace. Vždy se tak uživateli zobrazí pouze jeden objekt, korespondující s výběrem. V případě potřeby zobrazení jiné geovizualizace se při návratu do vedlejšího nebo hlavního menu daný objekt deinicializuje, aby se mohl výběrem scény inicializovat jiný. Inicializace probíhá již při výběru dané geovizualizace, nikoliv až při výběru režimu.

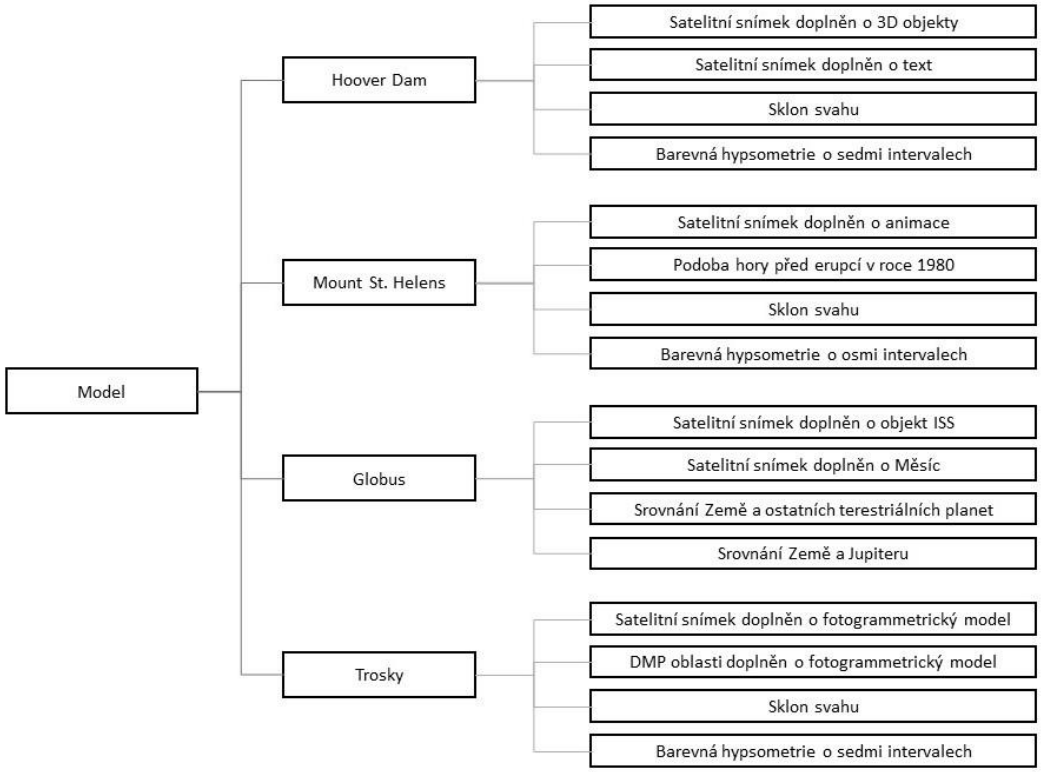
Hodnotným dílčím výsledkem aplikace jsou i vytvořené skripty v jazyce C#, které mají na starost funkcionalitu aplikace. Celkem bylo vytvořeno sedm C# (.cs) skriptů. Skript ImageTracking.cs zabezpečuje funkce dostupné v rámci Image Tracking scény. Mezi hlavní úkoly skriptu patří vytvoření instancí objektů a předání odpovídajících proměnných pro inicializaci dat. Stěžejním úkolem skriptu je vykonání logiky po přečtení markeru. Sem spadá zapnutí objektu a také nastavení vztahu rodič-dítě (z pohledu OOP) mezi markerem a daným objektem ve scéně. Skript zahrnuje doplňkovou funckionalitu, včetně uživatelsky přístupné změny měřítka či pozice modelů. Skript PlaneDetection.cs zabezpečuje funkce pro scénu PlaneDetection. Mezi stěžejní operace skriptu patří detekce dotknutí se obrazovky a vyvolání odpovídajícího objektu. Doplněn je pak o funkce rotace daného objektu a také schopnost uživatelsky vypnout či zapnout detekovanou rovinu. Dalším skriptem je MainMenu.cs, starající se o přepínání scén v aplikaci a také inicializaci správných dat na základě vybraného modelu a geovizualizace. Důležitým skriptem je také InGameMenu.cs, jež obsahuje funkce, které vykonávají operace jako vypnutí aplikace či návrat do odpovídajícího vedlejšího menu, využitelné v prostředí hlavní scény. Dalším vytvořeným skriptem je TextHandler.cs, který má na starost práci s textovými řetězci pro případ změny jazyka aplikace a také dynamické zobrazení odpovídajícího textu pro daný model. Skript ImageHandler.cs řeší zobrazení odpovídajícího obrázku reprezentujícího scénu při výběru režimu ve vedlejším menu. V neposlední řadě, skript RotateObject.cs zabezpečuje rotaci objektů, která je využita v rámci několika geovizualizací v aplikaci. Všechny napsané skripty jsou obsaženy v přílohách práce a mohou být využity pro další AR aplikace.

Výsledná logika aplikace funguje následovně: Po spuštění aplikace se uživateli zobrazí hlavní menu, ve kterém si může vybrat jeden ze čtyř modelů (Hoover Dam, Mount St. Helens, Globus, Trosky). Mimo to lze v hlavním menu změnit jazyk aplikace, zobrazit další informace o aplikaci a návod na práci s aplikací, či aplikaci vypnout. Poté, co si uživatel vybere jeden ze čtyř modelů, je přenesen do odpovídajícího vedlejšího menu (pro každý model jedno). Ve vedlejším menu si následně vybere jednu ze čtyř scén. Seznam dostupných scén si lze zobrazit přes odpovídající tlačítko. Po výběru scény se do paměti aplikace načte vybraná scéna, aby mohla být později zobrazena. Následně přichází na řadu výběr módu. Na výběr je mezi dvěma režimy. První z režimů funguje na bázi detekce markerů a zobrazení odpovídajícího obsahu. Tento mód je značen tlačítkem obsahujícím text MARKER. Druhý režim funguje na bázi detekce ploch v reálném prostředí a následném umístění obsahu na danou plochu po dotknutí se obrazovky. Tento mód je značen tlačítkem s textem ROVINA, či PLANE v anglické jazykové mutaci.

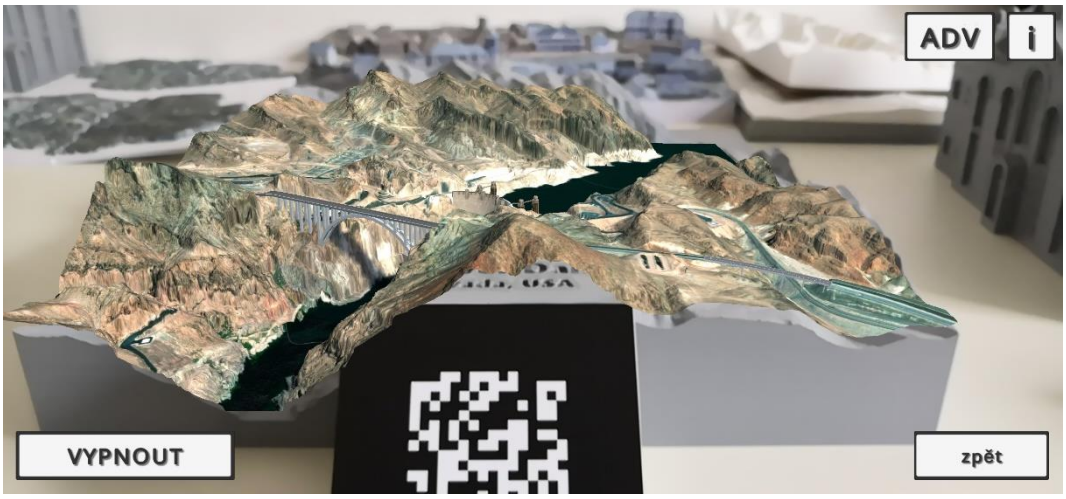
Po výběru módu je uživatel přenesen do hlavní scény. V případě výběru režimu MARKER poté uživatel pomocí kamery mobilního telefonu či tabletu detekuje odpovídající marker. Detekovaný marker musí odpovídat vybranému modelu, jelikož pro každý model existuje jeden marker. Zobrazený obsah poté odpovídá vybrané scéně. Se zobrazeným obsahem lze v případě potřeby manipulovat změnou pozice a měřítka. V případě výběru režimu ROVINA se aplikace snaží detekovat roviny v reálném prostředí. Tato detekce je velmi náchylná na světelné podmínky a do jisté míry také charakter povrchu. Po detekci roviny se poté uživatel může dané roviny dotknout (resp. dotknout se obrazovky v místě,

kde je detekována rovina), načež se zobrazí obsah odpovídající vybrané scéně. Po zobrazení obsahu lze vypnout detekci rovin. V rámci obou hlavních scén se lze vrátit do odpovídajícího vedlejšího menu tlačítkem ZPĚT nebo vypnout aplikaci. Po návratu do vedlejšího nebo hlavního menu proběhne deinicializace v dané době do paměti načteného objektu. Při výběru jiné scény tak dojde na načtení nového objektu. Další logika se chová analogicky k uvedeným informacím.

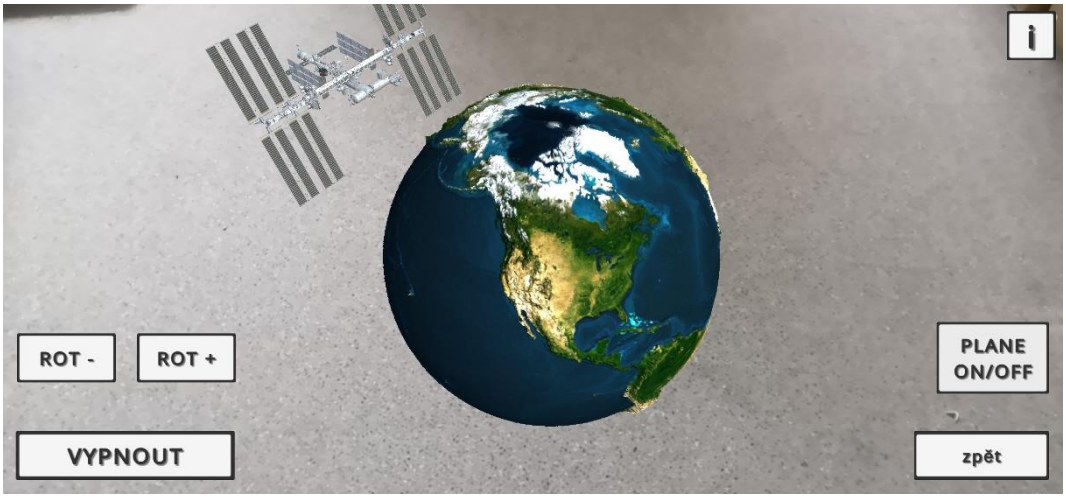
Dílčím výsledkem práce jsou také vytištěné 3D modely. Vytištěných modelů existuje celkem šest, což ale nemění logiku aplikace, jelikož jde stále o stejné čtyři studie, které byly představeny v kapitole 3. Důvodem pro větší počet modelů bylo rozhodnutí vytisknout některé modely v menším měřítku, což bylo ovlivněno technickými obtížemi, ať už ze strany funkcionality AR či ze strany 3D tisku. Pro první studii (Hoover Dam) tak existuje model s rozměry 400,2 × 400,2 mm, který danou oblast zobrazuje v měřítku 1 : 5 000. Dále však existuje model v menším měřítku, s rozměry 160,8 × 160,8 mm. Tento model zobrazuje oblast v měřítku 1 : 12 500. V případě druhé studie taktéž existují dva modely. První z nich má rozměry 400,8 × 400,8 mm a zobrazuje danou oblast v měřítku 1 : 12 500. Druhý model, s rozměry 166,7 × 166,7 mm, je v měřítku 1 : 30 000. Pro případ třetí studie byl vytištěn pouze jeden model z důvodů technických limitací při 3D tisku. Výsledný model zobrazuje planetu Zemi v měřítku 1 : 60 000 000 (bez stojanu). Pro poslední studii byl taktéž vytištěn pouze jeden model, a to z důvodu malé podrobnosti zdrojového DMR. Model má rozměry 160 × 160 mm a zobrazuje danou oblast v měřítku 1 : 2 200. Všechny vytištěné 3D modely ukazují reliéf dané oblasti a zprostředkovávají uživateli haptický vjem.



Obr. 7.1 Geovizualizace vytvořené v rámci práce (zdroj: autor).



Obr. 7.2 Jedna z vytvořených geovizualizací zobrazená v režimu MARKER (zdroj: autor).



Obr. 7.3 Jedna z vytvořených geovizualizací zobrazená v režimu ROVINA (zdroj: autor).

Obr. 7.4 Ukázka vytištěných 3D modelů s odpovídajícími markery (zdroj: autor).

8 DISKUZE

Diplomová práce se zabývala vývojem mobilní aplikace pro systém Android s cílem integrace vytištěných 3D modelů reliéfu a rozšířené reality. Za tímto účelem bylo učiněno množství kroků. Prvním významným krokem byla akvizice dat pro jednotlivé modely. Zvolené studie (Hoover Dam, Mount St. Helens, Globus, Trosky) byly vybrány z toho důvodu, že vhodným způsobem demonstrují zamyšlenou integraci rozšířené reality a vytištěných 3D modelů. Z pohledu aplikace jsou sice data demonstrována stejnou formou, avšak každý model obsahuje charakteristické vizualizace. V případě Hoover Dam je možné ukázat digitální model reliéfu, doplněný o 3D objekty, které jsou pro vybranou oblast velmi charakteristické, nebo přidat popisné texty, a tím učinit 3D model zajímavějším. V případě Mount St. Helens je možné do jisté míry nasimulovat přírodní jev a také ukázat podobu hory před výrazným zásahem do charakteru jejího reliéfu. Globus je zajímavý tím, že dovoluje zobrazit objekty mimo planetu a zprostředkovat tak i představu o měřítku planetárních těles, či demonstrovat planetární jevy, např. demonstrací oběhu Měsíce kolem Země, což je zajímavé i z edukativního hlediska. Poslední model pak ukazuje zejména rozdíl mezi DMR a plně texturovaným modelem povrchu. Důvod pro výběr právě čtyř modelů byl pak primárně ten, že každý z nich může demonstrovat něco unikátního. Ačkoliv se některé geovizualizace opakují (zejména ty založené na GIS analýzách jako sklon a hypsometrie), i ty jsou vždy jistým způsobem charakteristické pro danou oblast. Čtyři modely, respektive čtyři geovizualizace pro každý model, souvisí také s návrhem UI aplikace. Teoreticky vzato by modelů mohlo být více. Rozšíření by však vyžadovalo úpravy aplikace, zejména co se týče rozložení UI prvků, což by mohlo učinit aplikaci méně přehlednou a čitelnou na některých zařízeních. Vnitřní logika aplikace je však navržena tak, že z pohledu rozšíření kódu by se nejednalo o nijak složitý problém. Jinou problematikou je však vytvoření nových scén, které by zahrnovalo náročnou úpravu dat.

Právě úprava dat byla v rámci diplomové práce časově velmi náročnou činností, se kterou se pojilo množství problémů. Předně, jelikož nebylo možno využít příliš podrobná data pro virtuální zobrazení, musela být data zjednodušena a tím ztratila část detailu. Tohoto faktu si však uživatel běžně nevšimne. Významnější problém nastal při texturování dat. Jelikož pro objekty vytvořené UV mapy a některé textury byly vytvořeny pomocí pluginu BlenderOSM, který pracuje se systémem dlaždic, nemohlo dojít k přesnému umístění textury. Z toho důvodu musela být učiněna manuální úprava, která však nebyla provedena zcela přesně. U textur si tak lze všimnout jistých drobných nepřesností, které vyplývají právě z manuálního umístění. Dále, příprava dalších textur v prostředí GIS a jejich přenesení do programu Blender bylo zatíženo příliš složitým a těžko opakovatelným postupem. Jelikož tyto textury využívaly shodnou UV mapu, i zde muselo dojít k manuálním úpravám a i zde lze spatřit některé nepřesnosti. Stejně tak umístění některých objektů v rámci některých modelů je poznamenáno určitou nepřesností či úpravou vůči realitě. Toho si nejvíce lze všimnout v případě, kdy daný 3D objekt překrývá vytvořenou 3D mesh. Příkladem je umístění 3D modelu přehrady a mostů do první scény modelu Hoover Dam. Jelikož zdrojová data zachycují i některé antropogenní prvky (charakter přehrady), bylo poměrně problematické překrýt část modelu tak, aby výsledek působil věrohodně. U obou mostů se zase vyskytuje problém se zásahem do terénu, jelikož nejsou upraveny na přesné rozměry, v některých místech se tak propadají dovnitř vytvořené meshe. Podobného problému si lze všimnout i ve druhé scéně pro Mount St. Helens, ukazující část hory před rokem 1980. Jelikož provedený logický rozdíl nebyl zcela přesný, lze si na modelu všimnout určitých nepřesností, zejména pak v režimu ROVINA, kde je model zobrazen jako celek i s původní spodní částí.

Samotný vývoj aplikace se neobešel bez problémů zejména z důvodu použité technologie. AR Foundation je velmi schopný nástroj, ale technologie AR je velmi náchylná na určité způsoby manipulace a na světelné podmínky. Předně, je potřeba zmínit, že v prostředí Unity existují pro vývoj AR aplikací dva základní objekty. Těmito objekty jsou AR Session a AR Session Origin. První ze jmenovaných má na starost celý životní cyklus AR obsahu a z hlediska programové logiky se jedná o globální objekt, tedy existující napříč všemi jmennými prostory. To ovlivnilo některá rozhodnutí při vývoji aplikace, zejména spojené s problémem opětovného ostření kamery, který byl zmíněn v podkapitole 5.2. Právě z důvodu globální povahy celého životního cyklu bylo náročné se s problémem vypořádat a je ve finále vyřešen z pohledu AR poměrně invazivní akcí, a sice deinicializací třídy XRLoader, pod kterou spadá veškerý AR obsah. Velmi pravděpodobně existuje lepší řešení, bylo experimentováno s restartem pouze AR Session objektu, nikoliv celé třídy XRLoader. To se však ukázalo jako nedostačující, zjevně z důvodu chyby v implementaci knihovny, jelikož se problém vyskytoval napříč několika zařízeními s různými verzemi operačního systému Android. Dílčími problémy je také jistá neoptimálnost řešení některých problémů přes právě globální proměnné, tak, aby bylo možné k nim přistupovat napříč jmennými prostory. V projektu tohoto rozsahu nejsou problémy s globálními proměnnými ihned patrné, avšak obecně se jedná o špatný programátorský návyk. Zde k němu však bylo přistoupeno z důvodu zjednodušení logiky aplikace a zamezení redundance funkcí.

Původní návrh v rámci diplomové práce byl ten, že každý z modelů nebude mít pouze jeden marker, ale čtyři, tedy jeden z každé strany. Z tohoto návrhu muselo být upuštěno, což souvisí s druhým základním objektem AR aplikací v Unity, a sice AR Session Origin. AR Session Origin, jak už název napovídá, představuje něco jako souřadnice (0, 0, 0) ve scéně, tedy původ celé AR Session. Z pohledu OOP pak AR Session Origin je rodičem objektu AR Camera, který představuje v reálné scéně kameru telefonu. Díky tomuto vztahu spolu objekty sdílí transformaci, tedy pozici, rotaci a měřítko, což efektivně znamená, že kamera ve scéně je vždy na souřadnicích (0, 0, 0) a pozice ostatních objektů se odvíjí právě od pozice kamery. Pro jeden marker z každé strany existoval v jednom momentě vývoje aplikace i funkční kód (jak dokládá podkapitola 5.1), nakonec se však právě z důvodu chování AR Session Origin ukázala rotace objektu ve scéně jako velmi problematická.

3D prostor pracuje se třemi osami X, Y, Z. Jelikož AR kamera z jistého úhlu pohledu udává střed 3D prostoru (v kontextu AR), vše se odvíjí právě od pozice kamery. To znamená, že při detekci markeru a vyvolání obsahu se onen obsah umístí do scény, s pevně danými osami X, Y, Z (v tento moment X značí směr doleva nebo doprava, Y nahoru a dolů a Z dopředu či dozadu). V momentě, kdy dojde k přečtení markeru z jiné strany, by se dalo očekávat, že se osy v prostoru změní podle nového směru. AR Session Origin, stejně jako AR Session je globálně existujícím objektem. V momentě detekce markeru se do interních proměnných tohoto objektu zapíše prostorové určení scény. Po přečtení markeru z o 90° otočené strany pak následuje to, že kde byla předtím osa X, je nyní osa Z. Další rotace probíhá shodně, tedy při přečtení markeru z o 180° otočené strany se Z rovná -Z a X se rovná –X. Po dalším otočení o 90° se Z rovná –X a X se rovná –Z. Tato fixace os v prostoru pak zcela logicky ovlivňuje i pozici a tedy i offset. Kromě fixace os v prostoru je zde dalším problémem také kvalita sensorů. AR pro vykonání své funkcionality spoléhá na sensory telefonu (či jiného zařízení) a tedy ačkoliv by se hodnota offsetu měla ze všech stran rovnat (po přehození os), nebylo tomu tak a bylo by tak nutné jeden offset odhadovat celkem čtyřikrát pro každý model. Jak bylo uvedeno v podkapitole 5.1, hodnoty offsetu jsou velmi náchylné i na nejmenší změny a bylo by tak extrémně náročné určit hodnoty pro všechny čtyři strany. Z toho důvodu bylo od čtyř stran upuštěno a místo toho byl, jako náhrada za

chybějící funkcionalitu, vytvořen režim Plane Detection, ve kterém si uživatel může model zobrazit a volně prohlédnout ze všech stran.

Specifické chování AR kamery má za následek také další, a zcela nejvýznamnější, nevýhodu práce. Umístění obsahu po detekci markeru je velmi ovlivněno úhlem kamery, respektive úhlem, pod jakým je kamerou sejmut daný marker. Tento úhel totiž ovlivní zadaný offset a stane se tak, že při určitých úhlech je virtuální obsah umístěn chybně, tedy nepřekrývá model. Jelikož se i měřítko odvíjí od kamery (původ 3D scény se přibližuje či oddaluje 3D objektu, což má logicky za následek zvětšení či zmenšení), pojí se s detekcí obsahu další problém. Pokud je kamera velmi blízko markeru, dojde k přečtení, ale vyvolaný obsah se zobrazí v malém měřítku, tedy nepřekrývá model. Je potřeba kamerou snímat z určité vzdálenosti (přibližně 50 cm) a ze správného úhlu, aby došlo k co nejsprávnějšímu umístění obsahu. Představený problém zřejmě souvisí s nesprávným přičtením offsetu. Marker-Based AR ve svém základu dělá pouze to, že na daný marker zobrazí obsah, přičemž marker plní roli jakési prostorové kotvy. Jestliže je cílem měnit pozici obsahu, pak se technologie nezachová zcela podle představ. Je velmi složité určit, zda je tento problém způsoben konkrétní implementaci AR Foundation, knihovnou AR Core, sensory telefonu či nesprávnou metodou manipulace s offsetem. Z pohledu naplnění cíle práce se jedná o zcela nejvýznamnější problém a ovlivňuje kvalitu výsledku.

Dalším problémem aplikace je nepochybně její náročnost na výpočetní výkon. Telefony mají obecně oproti osobním počítačům velmi omezený výkon a je zapotřebí specifických optimalizací. I když některé optimalizace byly v práci učiněny (zjednodušení modelu) a engine Unity provádí některé další optimalizace (např. textur, ale i další zjednodušení 3D dat při jejich importu) automaticky, ale ani to nevyřešilo velkou náročnost aplikace. Jak již bylo zmíněno, aplikace byla v rámci vývoje testována na dvou mobilních telefonech (Samsung M31s a Huawei Nova 3), které byly z hlediska komponentů téměř shodné, s jedním klíčovým rozdílem. Tím rozdílem byl grafický chip, který byl výkonnější na druhém uvedeném telefonu. Rozdíl byl velmi znát ve výsledných snímcích za sekundu v aplikaci. Vykreslování obsahu je obecně velmi závislé na grafické kartě (nebo grafickém chipu v případě telefonu) a čím výkonnější je grafická výpočetní jednotka, tím jednodušší operací je vykreslení obsahu (velmi zjednodušeně). Problémy s výkonem mohly být do jisté míry ovlivněny použitím příliš velkých modelů nebo náročných shaderů. Z části se však jedná o prostý nedostatek výkonu mobilních telefonů. Klíčovým poznatkem je, že v případě běhu aplikace na méně výkonném stroji může uživatel očekávat snížený uživatelský zážitek.

Významnější nevýhodou může být z pohledu zmíněného uživatelského zážitku také návrh aplikace. Některá funkcionalita je dostupná až po několika kliknutích a může tak působit neintuitivně. Návrh aplikace je však jednoduchý a navigačních prvků není ve výsledku tolik, aby to bylo pro uživatele nepochopitelné. Z určitého pohledu se může i stylizace (barevné provedení) aplikace zdát nedostatečnou, avšak jednoduchý design s sebou nese jisté výhody, zejména pak přehlednost jednotlivých možností dostupných v aplikaci.

Problémy se vyskytují i v případě vytištěných 3D modelů. Prvním problémem je kvalita dat v případě čtvrté studie. Jelikož se jedná o poměrně malou oblast, je na výsledném 3D modelu patrná nedostatečná míra detailu. Problém se vyskytl také v případě globu. Vzhledem k charakteru objektu probíhal tisk s využitím podpor, které se ve finále nepodařilo zcela odstranit a kvalita 3D modelu je tímto faktem značně ovlivněna.

9 ZÁVĚR

Cílem diplomové práce bylo vytvoření technického řešení schopného integrovat rozšířenou realitu a vytištěné 3D modely reliéfu, a to doplněním modelů o textury, animace a další prostorové objekty. Za tímto účelem byla vytvořena mobilní aplikace pro operační systém Android, využívající principy rozšířené reality (AR). Pro vývoj aplikace byl využit program Unity. Funckionalita AR byla implementována za pomocí knihovny AR Foundation. Vedlejším cílem práce bylo také vytvoření a optimalizace nejvhodnějšího tvaru a struktury referenční značky pro zobrazení obsahu (tzv. marker).

Práce započala akvizicí dat, která byla následně připravena v prostředí programu Blender, kde byly pro každý model vytvořeny scény, které později byly zobrazeny v rámci mobilní aplikace. Následoval proces tvorby mobilní aplikace, který byl rozdělen na vytvoření prototypů a ostrý vývoj. V rámci vytvořených prototypů aplikace bylo otestováno množství funckionality, z které některá byla využita při ostrém vývoji aplikace. Během prototypování byly testovány rozličné formy markerů. Ve finále byl nalezen nástroj pro generování markerů, které obsahovaly dostatečně charakteristický vzor a byly snadno detekovatelné i za horších podmínek. Takto vygenerované unikátní markery lze využít pro širokou škálu AR aplikací.

Následoval vývoj aplikace, který byl rozdělen na několik menších úkolů. Nejdříve proběhlo vytvoření scény ImageTracking, která měla na starosti logiku kolem detekce markerů a zobrazení správného obsahu. Dále byla implementována scéna PlaneDetection, jejímž úkolem bylo v reálném prostředí detekovat virtuální roviny a po dotknutí se obrazovky umístit obsah. Následovalo navržení a implementace hlavního menu aplikace a také vedlejších menu aplikace. Jako poslední byla implementována doplňková funkcionalita, mezi kterou patří možnost přepnutí aplikace do anglického jazyka či zobrazení rozšiřujících informací ve scéně.

Výsledkem diplomové práce je zejména vytvořená mobilní aplikace pro operační systém Android, fungující na principu AR. Aplikace obsahuje dvě hlavní scény, starající se o hlavní funkcionalitu. První scénou je ImageTracking, fungující na principu detekce markeru a zobrazení daného obsahu na základě specifického markeru. Zobrazený obsah je pomocí změny pozice zobrazen nad vytištěným 3D modelem reliéfu. Druhou scénou je pak PlaneDetection, fungující na principu detekce virtuální roviny a umístění obsahu na danou rovinu po dotknutí se obrazovky.

Výsledná aplikace je navržena na práci se čtyřmi modely, pro každý model obsahuje čtyři vytvořené geovizualizace, celkem tak 16 geovizualizací. Dané čtyři modely byly vytištěny na 3D tiskárně a slouží jako stěžejní prvek fungování aplikace. Aplikace díky širokému záběru různých geovizualizací může sloužit jako nástroj pro edukační účely interaktivní formou, či jako demonstrace možností rozšířené reality.

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

ADEDOKUN-SHITTU, Nafisat A., Adedeji Hammed AJANI, Kehinde Muritala NUHU a Abdul Jaleel Kehinde SHITTU, 2020. Augmented reality instructional tool in enhancing geography learners academic performance and retention in Osun state Nigeria. In: *Education and Information Technologies* [online]. s. 3021–3033. ISBN 1063902010. Dostupné z: doi:10.1007/s10639-020-10099-2

AIRCARDS, 2021. *Markerless vs. Marker-based AR with Examples – Aircards* [online] [vid. 2022-08-23]. Dostupné z: https://www.aircards.co/blog/markerless-vs-marker-based-ar-with-examples

APPLE, 2023. *ARKit 6 - Augmented Reality - Apple Developer* [online] [vid. 2023-01-30]. Dostupné z: https://developer.apple.com/augmented-reality/arkit/

ARTH, Clemens, Raphael GRASSET, Lukas GRUBER, Tobias LANGLOTZ, Alessandro MULLONI a Daniel WAGNER, 2015. The History of Mobile Augmented Reality. In: *Inst. for Computer Graphics and Vision Graz University of Technology, Austria* [online]. Dostupné z: http://arxiv.org/abs/1505.01319

AZUMA, Ronald T., 1997. A Survey of Augmented Reality. In: *Teleoperators and Virtual Environments* [online]. s. 355–385. ISSN 15313263. Dostupné z: https://doi.org/10.1162/pres.1997.6.4.355

CARMIGNIANI, Julie, Borko FURHT, Marco ANISETTI, Paolo CERAVOLO, Ernesto DAMIANI a Misa IVKOVIC, 2011. Augmented reality technologies, systems and applications. In: *Multimedia Tools and Applications* [online]. s. 341–377. ISSN 13807501. Dostupné z: doi:10.1007/s11042-010-0660-6

CAUDELL, T.P. a D.W. MIZELL, 1992. Augmented reality: an application of heads-up display technology to manual manufacturing processes. In: *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences* [online]. s. 659–669 roč.2. Dostupné z: doi:10.1109/hicss.1992.183317

DESERTSAGE, 2022. *Hoover Dam with 8k and 4k Textures Low-poly - Buy Royalty Free 3D model by Desertsage (@Desertsage) [8111269]* [online] [vid. 2023-03-29]. Dostupné z: https://sketchfab.com/3d-models/hoover-dam-with-8k-and-4k-textures-low-poly-8111269ca44e49f28f0c41af8df404f0

DEVAUX, A., C. HOARAU, M. BRÉDIF a S. CHRISTOPHE, 2018. 3D urban geovisualization: In situ augmented and mixed reality experiments. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. s. 41–48. ISSN 21949050. Dostupné z: doi:10.5194/isprs-annals-IV-4-41-2018

DOMLYSZ, 2022. *GitHub - domlysz/BlenderGIS: Blender addons to make the bridge between Blender and geographic data* [online] [vid. 2023-03-31]. Dostupné z: https://github.com/domlysz/BlenderGIS

ELMQADDEM, Noureddine, 2019. Augmented Reality and Virtual Reality in education. Myth or reality? In: *International Journal of Emerging Technologies in Learning* [online]. s. 234–242. ISSN 18630383. Dostupné z: doi:10.3991/ijet.v14i03.9289

EPIC GAMES, 2022a. *Augmented Reality Overview | Unreal Engine Documentation* [online] [vid. 2023-01-31]. Dostupné z: https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/AR/HandheldAR/AROverview/

EPIC GAMES, 2022b. *Blueprints Visual Scripting in Unreal Engine | Unreal Engine 5.0 Documentation* [online] [vid. 2023-01-31]. Dostupné z: https://docs.unrealengine.com/5.0/en-US/blueprints-visual-scripting-in-unreal-engine/

FEINER, Steven, Blair MACINTYRE, Tobias HÖLLERER a Anthony WEBSTER, 1997. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. In: *Personal Technologies* [online]. s. 208–217. ISSN 16174909. Dostupné z: doi:10.1007/BF01682023

GHERGHINA, Alexandru, Alexandru Corneliu OLTEANU a Nicolae TAPUS, 2013. A marker-based augmented reality system for mobile devices. In: *2013 11th RoEduNet International Conference* [online]. B.m.: IEEE. ISBN 9781467361149. Dostupné z: doi:10.1109/RoEduNet.2013.6511731

GOOGLE, 2022. *Overview of ARCore and supported development environments | Google Developers* [online] [vid. 2022-09-09]. Dostupné z: https://developers.google.com/ar/develop

GREENBERG, Harvey M., 2013. *Mount Saint Helens DEMs* [online] [vid. 2023-04-01]. Dostupné z: http://gis.ess.washington.edu/data/raster/thirtymeter/mtsthelens/index.html

HARTLOFF, Matthew, 2018. *Bathymetric Earth Globe by mhartloff - Thingiverse* [online] [vid. 2023-04-23]. Dostupné z: https://www.thingiverse.com/thing:3229218

HÖLLERER, Tobias a Steven K. FEINER, 2004. Mobile Augmented Reality. Telegeoinformatics: Location-based computing and services ISSN 2375-0529.

HÖLLERER, Tobias a Dieter SCHMALSTIEG, 2016. A Brief History of Augmented Reality: Introduction to Augmented Reality. *Pearson Education* [online] [vid. 2022-07-29]. Dostupné z: http://www.informit.com/articles/article.aspx?p=2516729&seqNum=2

HONG, Ong Ace, Noor Dayana Abd HALIM, Nurul Nadwa ZULKIFLI, Nurul Farhana JUMAAT, Norasykin Mohd ZAID a Mahani MOKHTAR, 2022. Designing Game-Based Learning Kit with Integration of Augmented Reality for Learning Geography.

In: *International Journal of Interactive Mobile Technologies* [online]. s. 4–16. ISSN 18657923. Dostupné z: doi:10.3991/ijim.v16i02.27377

CHACKO, Phill, 2020. *Introducing Unity Mars – a first-of-its-kind solution for intelligent AR | WYSIWYG Authoring | Unity Blog* [online] [vid. 2023-01-31]. Dostupné z: https://blog.unity.com/technology/introducing-unity-mars-a-first-of-its-kind-solution-for-intelligent-ar

CHATZOPOULOS, DImitris, Carlos BERMEJO, Zhanpeng HUANG a Pan HUI, 2017. Mobile Augmented Reality Survey: From Where We Are to Where We Go. In: *IEEE Access* [online]. B.m.: IEEE, s. 6917–6950. ISSN 21693536. Dostupné z: doi:10.1109/ACCESS.2017.2698164

CHEN, Yunqiang, Qing WANG, Hong CHEN, Xiaoyu SONG, Hui TANG a Mengxiao TIAN, 2019. An overview of augmented reality technology. In: *Journal of Physics: Conference Series* [online]. ISSN 17426596. Dostupné z: doi:10.1088/1742-6596/1237/2/022082

CHITANIUC, Mirela a Adrian IFTENE, 2018. GeoAR-An Augmented Reality Application to Learn Geography. In: *Revista Romana de Interactiune Om-Calculator* [online]. s. 93–108. Dostupné z: http://rochi.utcluj.ro/rrioc/articole/RRIOC-11-2-Chitaniuc.pdf

INOVE, 2019. *Solar Textures | Solar System Scope* [online] [vid. 2023-03-29]. Dostupné z: https://www.solarsystemscope.com/textures/

JJASPER123, 2014. *Mike O'Callaghan – Pat Tillman Memorial Bridge (Hoover Dam Bypass)* [online] [vid. 2023-03-29]. Dostupné z: https://3dwarehouse.sketchup.com/model/46d90ab693e84fe93783eea36adc828e/Mike-OCallaghan-Pat-Tillman-Memorial-Bridge-Hoover-Dam-Bypass

KLAVINS, Ainars, 2021. *How to create your own augmented reality marker? - Overlyapp* [online] [vid. 2023-04-09]. Dostupné z: https://overlyapp.com/blog/how-to-create-an-augmented-reality-marker

LEE, Kangdon, 2012. Augmented Reality in Education and Training. In: *TechTrends* [online]. s. 13–21. ISBN 1152801205593. Dostupné z: doi:10.1007/s11528-012-0559-3

LOBO, M. J. a S. CHRISTOPHE, 2020. Opportunities and challenges for augmented reality situated geographical visualization. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. s. 163–170. ISSN 21949050. Dostupné z: doi:10.5194/isprs-Annals-V-4-2020-163-2020

MILGRAM, Paul a Fumio KISHINO, 1994. A Taxonomy of Mixed Reality Visual Displays. In: *IEICE Transactions on Information and Systems*.

MUŽÍČEK, Petr, 2021. *Návrh a tvorba interaktivní exhibice s využitím geoinformačních technologií* [online]. B.m. Univerzita Palackého v Olomouci. Dostupné z: https://theses.cz/id/8dll08/

NASA, 2019a. *Deimos 3D Model | NASA Solar System Exploration* [online] [vid. 2023-03-29]. Dostupné z: https://solarsystem.nasa.gov/resources/2434/deimos-3d-model/

NASA, 2019b. *International Space Station 3D Model | NASA Solar System Exploration* [online] [vid. 2023-03-29]. Dostupné z: https://solarsystem.nasa.gov/resources/2378/international-space-station-3d-model/

NASA, 2019c. *Phobos 3D Model | NASA Solar System Exploration* [online] [vid. 2023-03-29]. Dostupné z: https://solarsystem.nasa.gov/resources/2358/phobos-3d-model/

NGUYEN, Vinh T. a Tommy DANG, 2017. Setting up Virtual Reality and Augmented Reality Learning Environment in Unity. In: *Adjunct Proceedings of the 2017 IEEE International Symposium on Mixed and Augmented Reality, ISMAR-Adjunct 2017* [online]. s. 315–320. ISBN 9780769563275. Dostupné z: doi:10.1109/ISMAR-Adjunct.2017.97

OPENCV, 2023. OpenCV: Detection of ArUco Markers. *OpenCV* [online] [vid. 2023-04-09]. Dostupné z: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

PAPADOPOULOU, Ermioni Eirini, Vlasios KASAPAKIS, Christos VASILAKOS, Apostolos PAPAKONSTANTINOU, Nikolaos ZOUROS, Athanasia CHRONI a Nikolaos SOULAKELLIS, 2020. Geovisualization of the excavation process in the Lesvos petrified forest, Greece using augmented reality. In: *ISPRS International Journal of Geo-Information* [online]. ISSN 22209964. Dostupné z: doi:10.3390/ijgi9060374

PEEK, Luke, 2021. *Realistic Smoke VFX | Fire & Explosions | Unity Asset Store* [online] [vid. 2023-04-04]. Dostupné z: https://assetstore.unity.com/packages/vfx/particles/fire-explosions/realistic-smoke-vfx-58504

QIAO, Xiuquan, Pei REN, Schahram DUSTDAR, Ling LIU, Huadong MA a Junliang CHEN, 2019. Web AR: A Promising Future for Mobile Augmented Reality-State of the Art, Challenges, and Insights. In: *Proceedings of the IEEE* [online]. s. 651–666. ISSN 15582256. Dostupné z: doi:10.1109/JPROC.2019.2895105

RAO, Jinmeng, Yanjun QIAO, Fu REN, Junxing WANG a Qingyun DU, 2017. A mobile outdoor augmented reality method combining deep learning object detection and spatial relationships for geovisualization. In: *Sensors (Switzerland)* [online]. ISSN 14248220. Dostupné z: doi:10.3390/s17091951

ROCK PAPER REALITY, 2021. *How Does Web-based Augmented Reality Work?* [online] [vid. 2022-08-26]. Dostupné z: https://rockpaperreality.com/insights/web-ar/how-does-web-based-augmented-reality-work/

SAIDIN, Nor Farhah, Noor Dayana Abd HALIM a Noraffandy YAHAYA, 2015. A review of research on augmented reality in education: Advantages and applications. In: *International Education Studies* [online]. s. 1–8. ISSN 19139039. Dostupné z: doi:10.5539/ies.v8n13p1

SHAWN LEHNER, 2017. *ARMarker - Augmented Reality Marker Generator* [online] [vid. 2023-04-04]. Dostupné z: https://shawnlehner.github.io/ARMaker/

SINHA, Disha, 2021. An Overview: Understanding Different Types of Augmented Reality. *Analytics Insight* [online] [vid. 2022-08-23]. Dostupné z: https://www.analyticsinsight.net/an-overview-understanding-different-types-of-augmented-reality/

SOFTTEK, 2021. *What are the different types of Augmented Reality?* [online] [vid. 2022-08-22]. Dostupné z: https://blog.softtek.com/en/what-are-the-different-types-of-augmented-reality

STUDIO OCHI, 2018. *Sculpted 3D Earth Globe - Buy Royalty Free 3D model by Studio Ochi (@studioochi) [9886359]* [online] [vid. 2023-03-29]. Dostupné z: https://sketchfab.com/3d-models/sculpted-3d-earth-globe-9886359dca8e40a593bc0820e295e63f

SUHYUN, 2019. *Red cliff background | 2D Textures & Materials | Unity Asset Store* [online] [vid. 2023-04-05]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/red-cliff-background-147959

THOMAS, Bruce, Ben CLOSE, John DONOGHUE, John SQUIRES, Phillip DE BONDI, Michael MORRIS a Wayne PIEKARSKI, 2000. ARQuake: an outdoor/indoor augmented reality first person application. In: *International Symposium on Wearable Computers, Digest of Papers* [online]. s. 139–146. ISBN 0769507956. Dostupné z: doi:10.1109/iswc.2000.888480

THOMAS, Bruce, Victor DEMCZUK, Wayne PIEKARSKI, David HEPWORTH a Bernard GUNTHER, 1998. A wearable computer system with augmented reality to support terrestrial navigation. In: *International Symposium on Wearable Computers, Digest of Papers* [online]. s. 168–171. ISBN 0818690747. Dostupné z: doi:10.1109/ISWC.1998.729549

UFKES, Alex a Mark FIALA, 2013. A markerless augmented reality system for mobile devices. In: *Proceedings - 2013 International Conference on Computer and Robot Vision, CRV 2013* [online]. B.m.: IEEE, s. 226–233. ISBN 9780769549835. Dostupné z: doi:10.1109/CRV.2013.51

UNITY TECHNOLOGIES, 2017. Unity - Manual: Vuforia SDK overview. *Unity Technologies* [online] [vid. 2023-01-31]. Dostupné z: https://docs.unity3d.com/2017.2/Documentation/Manual/vuforia-sdk-overview.html

UNITY TECHNOLOGIES, 2022. *AR Foundation | AR Foundation | 5.0.3* [online] [vid. 2023-01-31]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html

UNITY TECHNOLOGIES, 2023a. *Advanced workflows for AR developers | Unity Mars* [online] [vid. 2023-01-31]. Dostupné z: https://unity.com/products/unity-mars

UNITY TECHNOLOGIES, 2023b. *Unity's AR Foundation Framework | Cross platform augmented reality development software | Unity* [online] [vid. 2023-01-31]. Dostupné z: https://unity.com/unity/features/arfoundation

UNITY TECHNOLOGIES, 2023c. *Unity Particle Pack - Asset Store* [online] [vid. 2023-04-04]. Dostupné z: https://assetstore.unity.com/packages/essentials/tutorial-projects/unity-particle-pack-127325

UNITY TECHNOLOGIES, 2023d. *Unity QA - LTS Releases - Unity* [online] [vid. 2023-02-01]. Dostupné z: https://unity3d.com/unity/qa/lts-releases

USGS, 2023. *TNM Download v2* [online] [vid. 2023-03-29]. Dostupné z: https://apps.nationalmap.gov/downloader/

VUFORIA DEVELOPER LIBRARY, 2021. *Vuforia Engine Overview | VuforiaLibrary* [online] [vid. 2023-01-31]. Dostupné z: https://library.vuforia.com/getting-started/vuforia-features

VVOOVV, 2021. *GitHub - vvoovv/blender-osm: One click download and import of OpenStreetMap and terrain for Blender! Global coverage! Source code is in the branch „release"* [online] [vid. 2023-03-31]. Dostupné z: https://github.com/vvoovv/blender-osm

WAGNER, Daniel a Dieter SCHMALSTIEG, 2003. First steps towards handheld augmented reality. In: *Proceedings - International Symposium on Wearable Computers, ISWC* [online]. s. 127–137. ISBN 0769520340. Dostupné z: doi:10.1109/iswc.2003.1241402

WAGNER, Daniel a Dieter SCHMALSTIEG, 2009. History and future of tracking for mobile phone augmented reality. *Proceedings - 2009 International Symposium on Ubiquitous Virtual Reality, ISUVR 2009* [online]. (December 2013), 7–10. Dostupné z: doi:10.1109/ISUVR.2009.11

WALCHKO, 2018. *GitHub - MomsFriendlyRobotCompany/ar_markers: Simple python AR marker generator and detector* [online] [vid. 2023-04-04]. Dostupné z: https://github.com/MomsFriendlyRobotCompany/ar_markers

WRIGHT, Ernie, 2019. *SVS: CGI Moon Kit* [online] [vid. 2023-03-29]. Dostupné z: https://svs.gsfc.nasa.gov/cgi-bin/details.cgi?aid=4720

WU, Hsin Kai, Silvia Wen Yu LEE, Hsin Yi CHANG a Jyh Chong LIANG, 2013. Current status, opportunities and challenges of augmented reality in education. In: *Computers and Education* [online]. B.m.: Elsevier Ltd, s. 41–49. ISSN 03601315. Dostupné z: doi:10.1016/j.compedu.2012.10.024

YUGHUES, 2015. *Yughues Free Concrete Materials | 2D Concrete | Unity Asset Store* [online] [vid. 2023-04-04]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/concrete/yughues-free-concrete-materials-12951

ZHANG, Guoyong, Jianhua GONG, Yi LI, Jun SUN, Bingli XU, Dong ZHANG, Jieping ZHOU, Ling GUO, Shen SHEN a Bingxiao YIN, 2020. An efficient flood dynamic visualization approach based on 3D printing and augmented reality. In: *International Journal of Digital Earth* [online]. s. 1302–1320. ISSN 17538955. Dostupné z: doi:10.1080/17538947.2019.1711210

PŘÍLOHY

SEZNAM PŘÍLOH

Elektronické přílohy:

Příloha 1 Skript ImageTracking.cs
Příloha 2 Skript TapToPlace.cs
Příloha 3 Skript MainMenu.cs
Příloha 4 InGameMenu.cs
Příloha 5 Skirpt TextHandler.cs
Příloha 6 Skript ImageHandler.cs
Příloha 7 Skript RotateObject.cs
Příloha 8 Markery

Volné přílohy

Příloha 9 Hoover Dam 1 : 5 000
Příloha 10 Mount St. Helens 1 : 12 500
Příloha 11 Globus 1 : 60 000 000
Příloha 12 Hrad Trosky 1 : 2 200
Příloha 13 Hoover Dam 1 : 12 500
Příloha 14 Mount St. Helens 1 : 30 000
Příloha 15 Poster

Popis struktury odevzdávaných digitálních dat na datové úložiště katedry

Adresáře:

Aplikace: Aplikace vytvořená v rámci diplomové práce.
Poster: Poster ve formátu PDF.
Text_Prace: Plný text práce ve formátech DOCX a PDF.
Unity_Projekt: Soubory Unity projektu včetně všech skriptů.
Vstupni_Data: Vstupní data využítá pro zpracování práce.
Vystupni_Data: Výstupní data ve formátech FBX a STL, vytvořené textury.
WEB: Webová stránka vytvořená pro diplomovou práci

```
using UnityEngine;
using UnityEngine.XR.ARFoundation;

[RequireComponent(typeof(ARTrackedImageManager))]
public class ImageTracking : MonoBehaviour
{
    /// Prefab which is loaded in the Awake method.
    private GameObject selectedPrefab = null;

    /// Strings holding name and index of a chosen prefab.
    static public string selectedModel = string.Empty;
    static public string selectedPrefabIndex = string.Empty;
    private bool addedCalled = false;

    /// Reference to Tracked Image Manager compotent.
    private ARTrackedImageManager trackedImageManager;

    /// <summary>
    /// Sets a prefab (to be loaded) accordingly based on the function parameters.
    /// </summary>
    /// <param name="name">Name of the selexcted model (e.g. "HooverDam").</param>
    /// <param name="index">Index of the selected model (e.g. "01").</param>
    static public void SetSelectedPrefab(string name, string index)
    {
        selectedModel = name;
        selectedPrefabIndex = index;
    }

    private void Awake()
    {
        // Tracked Image Manager component definition.
        trackedImageManager = GetComponent<ARTrackedImageManager>();

        selectedPrefab = LoadPrefab(selectedModel + selectedPrefabIndex);
    }

    private void Start()
    {
        // Sets sceen orientation to be both landscape and portrait.
        Screen.orientation = ScreenOrientation.AutoRotation;
    }

    /// Every time a scene loads, the OnEnable event is called.
    private void OnEnable()
    {
        trackedImageManager.trackedImagesChanged += OnTrackedImagesChanged;
    }

    /// Every time a scene unloads, the OnDisable event is called.
    private void OnDisable()
    {
        trackedImageManager.trackedImagesChanged -= OnTrackedImagesChanged;
    }

    private void OnDestroy()
    {
        // Deinitializes the XRLoader class (base class for all XR implementations, i.e. AR
Foundation).
        LoaderUtility.Deinitialize();
    }

    /// <summary>
    /// Reacts to the changing state of tracked images, notifications are received from the
tracked image manager.
    /// </summary>
    /// <param name="eventArgs">A struct containing lists of tracked images, which were
added, updated and removed.</param>
    private void OnTrackedImagesChanged(ARTrackedImagesChangedEventArgs eventArgs)
    {
        // For each added tracked image, finds the corresponding prefab and sets it active.
        foreach (ARTrackedImage trackedImage in eventArgs.added)
        {
            OnAddedTrackedImage(trackedImage);
        }
    }
```

```
    /// <summary>
    /// Loads and instantiates every prefab found in the Resources folder, according to the
prefab name.
    /// </summary>
    /// <param name="prefabName">The name of the prefab.</param>
    /// <returns></returns>
    private GameObject LoadPrefab(string prefabName)
    {
        GameObject go = Resources.Load<GameObject>(prefabName);
        GameObject prefab = Instantiate(go);
        prefab.name = prefabName;
        prefab.SetActive(false);
        return prefab;
    }

    /// <summary>
    /// For selected tracked image, sets the tracked image as prefab parent and enables it.
    /// </summary>
    /// <param name="trackedImage">A tracked image from the Reference Image Library.</param>
    private void OnAddedTrackedImage(ARTrackedImage trackedImage)
    {
        string modelNameFromImage;
        if (trackedImage.referenceImage.name.EndsWith("Small"))
        {
            modelNameFromImage = trackedImage.referenceImage.name[..^5];
        }
        else
        {
            modelNameFromImage = trackedImage.referenceImage.name;
        }

        // If tracked image name is not equal to selected model name, return.
        if (modelNameFromImage != selectedModel)
        {
            return;
        }

        // If added event already happened, return.
        if (addedCalled)
        {
            return;
        }
        addedCalled = true;

        // If a marker for small version of a model is read, set position and scale
accordingly.
        if (trackedImage.referenceImage.name.EndsWith("Small"))
        {
            selectedPrefab.transform.position =
OffsetData.GetSmallPosition(modelNameFromImage);
            selectedPrefab.transform.localScale =
OffsetData.GetSmallScale(modelNameFromImage);
        }
        selectedPrefab.transform.parent = trackedImage.transform;
        selectedPrefab.SetActive(true);
    }

    // Functions for changing the Transform of the selected prefab.

    public void ChangePositionZPlus()
    {
        Vector3 vec = selectedPrefab.transform.position;
        selectedPrefab.transform.position = new Vector3(vec.x, vec.y, vec.z + 0.1f);
    }

    public void ChangePositionYPlus()
    {
        Vector3 vec = selectedPrefab.transform.position;
        selectedPrefab.transform.position = new Vector3(vec.x, vec.y + 0.1f, vec.z);
    }

    public void ChangePositionXPlus()
    {
        Vector3 vec = selectedPrefab.transform.position;
```

```
        selectedPrefab.transform.position = new Vector3(vec.x + 0.1f, vec.y, vec.z);
    }

    public void ChangePositionZMinus()
    {
        Vector3 vec = selectedPrefab.transform.position;
        selectedPrefab.transform.position = new Vector3(vec.x, vec.y, vec.z - 0.1f);
    }

    public void ChangePositionYMinus()
    {
        Vector3 vec = selectedPrefab.transform.position;
        selectedPrefab.transform.position = new Vector3(vec.x, vec.y - 0.1f, vec.z);
    }

    public void ChangePositionXMinus()
    {
        Vector3 vec = selectedPrefab.transform.position;
        selectedPrefab.transform.position = new Vector3(vec.x - 0.1f, vec.y, vec.z);
    }

    public void ChangeScalePlus()
    {
        Vector3 scale = selectedPrefab.transform.localScale;
        selectedPrefab.transform.localScale = scale + new Vector3(0.1f, 0.1f, 0.1f);
    }

    public void ChangeScaleMinus()
    {
        Vector3 scale = selectedPrefab.transform.localScale;
        selectedPrefab.transform.localScale = scale - new Vector3(0.1f, 0.1f, 0.1f);
    }

    public void RotatePrefab()
    {
        Vector3 vecEuler = selectedPrefab.transform.rotation.eulerAngles;
        selectedPrefab.transform.rotation = Quaternion.Euler(vecEuler.x, vecEuler.y + 90f,
vecEuler.z);
    }
}

static public class OffsetData
{
    static public Vector3 GetSmallPosition(string modelName)
    {
        switch (modelName)
        {
            case "HooverDam": return new Vector3(0f, -0.4f, 0.18f);
            case "StHelens": return new Vector3(0f, -0.9f, 0.25f);
            case "Earth": return new Vector3(0f, 0f, 0f);
            case "Trosky": return new Vector3(0f, 0f, 0f);
            default: return Vector3.zero;
        }
    }

    static public Vector3 GetSmallScale(string modelName)
    {
        switch (modelName)
        {
            case "HooverDam": return new Vector3(0.26f, 0.26f, 0.26f);
            case "StHelens": return new Vector3(0.15f, 0.15f, 0.15f);
            case "Earth": return new Vector3(0f, 0f, 0f);
            case "Trosky": return new Vector3(0f, 0f, 0f);
            default: return Vector3.zero;
        }
    }
}
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;

[RequireComponent(typeof(ARRaycastManager))]
public class TapToPlace : MonoBehaviour
{
    // GameObject to be spawned.
    private GameObject spawnedObject;
    // Position on the screen that the user touches.
    private Vector2 touchPosition;
    // Reference to AR Raycast Manager in Unity.
    private ARRaycastManager arRaycastManager;
    // Reference to AR Plane Manager in Unity.
    private ARPlaneManager arPlaneManager;
    // List that stores raycast hits (gets information back from a raycast).
    private List<ARRaycastHit> hits = new List<ARRaycastHit>();

    private void Awake()
    {
        // AR Raycast Manager component definition.
        arRaycastManager = GetComponent<ARRaycastManager>();
        arPlaneManager = GetComponent<ARPlaneManager>();
    }

    private void Start()
    {
        // Sets sceen orientation to be both landscape and portrait.
        Screen.orientation = ScreenOrientation.AutoRotation;
    }

    void Update()
    {
        // If the user touches the screen, get position of the touch.
        if (Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch(0);

            if (touch.phase == TouchPhase.Began)
            {
                touchPosition = touch.position;
            }

            if (arRaycastManager.Raycast(touchPosition, hits,
TrackableType.PlaneWithinPolygon))
            {
                Pose hitPose = hits[0].pose;

                // If no object is spawned, spawns the corresponding prefab from the
Resources folder in Unity.
                if (spawnedObject == null)
                {
                    GameObject prefabObject =
Resources.Load<GameObject>(ImageTracking.selectedModel + "Plane" +
ImageTracking.selectedPrefabIndex);

                    // Instantiates the object on the position where the user touched the
screen.
                    spawnedObject = Instantiate(prefabObject, hitPose.position,
hitPose.rotation);
                }

                if (arPlaneManager.enabled)
                {
                    // If an object is already spawned, updates its position.
                    spawnedObject.transform.position = hitPose.position;
                }
            }
        }
    }

    /// <summary>
    /// Changes the rotation of the spawned prefab by 15 degrees clockwise.
    /// </summary>
```

```
    public void ChangeYRotationPlus()
    {
        Vector3 vecEuler = spawnedObject.transform.rotation.eulerAngles;
        spawnedObject.transform.rotation = Quaternion.Euler(vecEuler.x, vecEuler.y + 15.0f,
vecEuler.z);
    }

    /// <summary>
    /// Changes the rotation of the spawned prefab by 15 degrees counter-clockwise.
    /// </summary>
    public void ChangeYRotationMinus()
    {
        Vector3 vecEuler = spawnedObject.transform.rotation.eulerAngles;
        spawnedObject.transform.rotation = Quaternion.Euler(vecEuler.x, vecEuler.y - 15.0f,
vecEuler.z);
    }

    /// <summary>
    /// Enables/disables tracking planes based on the previous state.
    /// </summary>
    public void ToggleTracking()
    {
        bool newState = !arPlaneManager.enabled;
        foreach (var plane in arPlaneManager.trackables)
        {
            plane.gameObject.SetActive(newState);
        }
        arPlaneManager.enabled = newState;
    }
}
```

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.XR.ARFoundation;

public class MainMenu : MonoBehaviour
{
    public void GetSubMenu()
    {
        SceneManager.LoadScene("SubMenu");
    }

    public void GetSubMenu02()
    {
        SceneManager.LoadScene("SubMenu02");
    }

    public void GetSubMenu03()
    {
        SceneManager.LoadScene("SubMenu03");
    }

    public void GetSubMenu04()
    {
        SceneManager.LoadScene("SubMenu04");
    }

    public void GetToMainScene()
    {
        SceneManager.LoadScene("ImageTracking");
    }

    public void GetToPlaneScene()
    {
        SceneManager.LoadScene("PlaneDetection");
    }

    public void SetHoover01()
    {
        ImageTracking.SetSelectedPrefab("HooverDam", "01");
        LoaderUtility.Initialize();
    }

    public void SetHoover02()
    {
        ImageTracking.SetSelectedPrefab("HooverDam", "02");
        LoaderUtility.Initialize();
    }

    public void SetHoover03()
    {
        ImageTracking.SetSelectedPrefab("HooverDam", "03");
        LoaderUtility.Initialize();
    }

    public void SetHoover04()
    {
        ImageTracking.SetSelectedPrefab("HooverDam", "04");
        LoaderUtility.Initialize();;
    }

    public void SetHelens01()
    {
        ImageTracking.SetSelectedPrefab("StHelens", "01");
        LoaderUtility.Initialize();
    }

    public void SetHelens02()
    {
        ImageTracking.SetSelectedPrefab("StHelens", "02");
        LoaderUtility.Initialize();
    }

    public void SetHelens03()
    {
        ImageTracking.SetSelectedPrefab("StHelens", "03");
```

```
        LoaderUtility.Initialize();
    }

    public void SetHelens04()
    {
        ImageTracking.SetSelectedPrefab("StHelens", "04");
        LoaderUtility.Initialize();
    }

    public void SetEarth01()
    {
        ImageTracking.SetSelectedPrefab("Earth", "01");
        LoaderUtility.Initialize();
    }

    public void SetEarth02()
    {
        ImageTracking.SetSelectedPrefab("Earth", "02");
        LoaderUtility.Initialize();
    }

    public void SetEarth03()
    {
        ImageTracking.SetSelectedPrefab("Earth", "03");
        LoaderUtility.Initialize();
    }

    public void SetEarth04()
    {
        ImageTracking.SetSelectedPrefab("Earth", "04");
        LoaderUtility.Initialize();
    }

    public void SetTrosky01()
    {
        ImageTracking.SetSelectedPrefab("Trosky", "01");
        LoaderUtility.Initialize();
    }

    public void SetTrosky02()
    {
        ImageTracking.SetSelectedPrefab("Trosky", "02");
        LoaderUtility.Initialize();
    }

    public void SetTrosky03()
    {
        ImageTracking.SetSelectedPrefab("Trosky", "03");
        LoaderUtility.Initialize();
    }

    public void SetTrosky04()
    {
        ImageTracking.SetSelectedPrefab("Trosky", "04");
        LoaderUtility.Initialize();
    }

    public void Quit()
    {
        Application.Quit();
    }

    public void PreviousScene()
    {
        SceneManager.LoadScene("MainMenu");
    }

    private void Start()
    {
        // Sets the screen orientation to be portrait only.
        Screen.orientation = ScreenOrientation.Portrait;
    }
}
```

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class InGameMenu : MonoBehaviour
{
    [SerializeField]
    public GameObject advMenu;

    public void ExitApplication()
    {
        Application.Quit();
    }

    public void PreviousScene()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void GetSubMenu()
    {
        if (ImageTracking.selectedModel == "HooverDam" || ImageTracking.selectedModel ==
"HooverDamPlane")
        {
            SceneManager.LoadScene("SubMenu");
        }
        else if (ImageTracking.selectedModel == "StHelens" || ImageTracking.selectedModel ==
"StHelensPlane")
        {
            SceneManager.LoadScene("SubMenu02");
        }
        else if (ImageTracking.selectedModel == "Earth" || ImageTracking.selectedModel ==
"EarthPlane")
        {
            SceneManager.LoadScene("SubMenu03");
        }
        else if (ImageTracking.selectedModel == "Trosky" || ImageTracking.selectedModel ==
"TroskyPlane")
        {
            SceneManager.LoadScene("SubMenu04");
        }
    }

    public void AdvancedMode()
    {
        if (advMenu != null)
        {
            if (!advMenu.activeSelf)
            {
                advMenu.SetActive(true);
                return;
            }
            advMenu.SetActive(false);
        }
    }
}
```

```
using UnityEngine;
using TMPro;

public class TexHandler : MonoBehaviour
{
    private TMP_Text displayedText;

    public string textID;
    static public bool lang = true;

    private static string hooverScenesText = "<b><size=130%>SEZNAM SCÉN PRO MODEL #1: HOOVER
DAM</b></size>\r\n\r\n<align=\"justified\"><size=110%>Celkem jsou nachystány" +
        " čtyři scény, každou z nich lze zobrazit kliknutím na odpovídající
tlačítko.\r\n\r\n1. Satelitní snímek doplněn o 3D objekty (přehrada a mosty).\r\n" +
        "2. Satelitní snímek doplněn o text popisující místa na modelu.\r\n3. Sklon svahu.
\r\n4. Hypsometrie, klasifikována celkem do sedmi intervalů.";

    private static string hooverScenesText_EN = "<b><size=130%>LIST OF SCENES FOR MODEL #1:
HOOVER DAM</b></size>\r\n\r\n<align=\"justified\"><size=110%>There are four" +
        " scenes in total, each of which can be displayed by selecting the corresponding
button.\r\n\r\n1. Satelite imagery accompanied by 3D objects (dam and bridges).\r\n" +
        "2. Satelite imagery accompanied by text describing areas of the model.\r\n3. Slope.
\r\n4. Hypsometry, classified into seven intervals.";

    private static string sthelensScenesText = "<b><size=130%>SEZNAM SCÉN PRO MODEL #2: MOUNT
ST. HELENS</b></size>\r\n\r\n<align=\"justified\"><size=110%>Celkem jsou" +
        " nachystány čtyři scény, každou z nich lze zobrazit kliknutím na odpovídající
tlačítko.\r\n\r\n1. Satelitní snímek doplněn o animace (exploze a stoupající kouř)." +
        "\r\n2. Podoba hory před erupcí v roce 1980.\r\n3. Sklon svahu. \r\n4. Hypsometrie,
klasifikována celkem do osmi intervalů.";

    private static string sthelensScenesText_EN = "<b><size=130%>LIST OF SCENES FOR MODEL #2:
MOUNT ST. HELENS</b></size>\r\n\r\n<align=\"justified\"><size=110%>There are" +
        " four models in total, each of which can be displayed by selecting the corresponding
button.\r\n\r\n1. Satelite imagery accompanied by animations (an explosion" +
        " and rising smoke).\r\n2. The appereance of the mountain before the 1980
erruption.\r\n3. Slope. \r\n4. Hypsometry, classified into eight intervals.";

    private static string earthScenesText = "<b><size=130%>SEZNAM SCÉN PRO MODEL #3:
GLOBUS</b></size>\r\n\r\n<align=\"justified\"><size=110%>Celkem jsou nachystány čtyři" +
        " scény, každou z nich lze zobrazit kliknutím na odpovídající tlačítko.\r\n\r\n1.
Satelitní snímek doplněn o rotující objekt (ISS).\r\n2. Satelitní snímek doplněn" +
        " o rotující objekt (Měsíc).\r\n3. Srovnání velikosti Země s ostatními terestriálnimi
planetami (Merkur, Venuše, Mars). \r\n4. Srovnání velikosti Země s planetou Jupiter.";

    private static string earthScenesText_EN = "<b><size=130%>LIST OF SCENES FOR MODEL #3:
GLOBE</b></size>\r\n\r\n<align=\"justified\"><size=110%>There are" +
        " four models in total, each of which can be displayed by selecting the corresponding
button.\r\n\r\n1. Satelite imagery accompanied by a rotating object (ISS).\r\n" +
        "2. Satelite imagery accompanied by a rotating object (the Moon).\r\n3. Comparison
between size of the Earth and other terrestrial planets" +
        " (Mercury, Venus, Mars) \r\n4. Comparison beteween size of the Earth and Jupiter.";

    private static string troskyScenesText = "<b><size=130%>SEZNAM SCÉN PRO MODEL #4: HRAD
TROSKY</b></size>\r\n\r\n<align=\"justified\"><size=110%>Celkem jsou" +
        " nachystány čtyři scény, každou z nich lze zobrazit kliknutím na odpovídající
tlačítko.\r\n\r\n1. Satelitní snímek oblasti doplněn o model z fotogrammetrického zpracování"
+
        " oblasti.\r\n2. Digitální model povrchu (DMP) oblasti.\r\n3. Sklon svahu.\r\n" + "4.
Hypsometrie, klasifikována do sedmi intervalů.";

    private static string troskyScenesText_EN = "<b><size=130%>LIST OF SCENES FOR MODEL #4:
TROSKY CASTLE</b></size>\r\n\r\n<align=\"justified\"><size=110%>There are" +
        " four models in total, each of which can be displayed by selecting the corresponding
button.\r\n\r\n1. Model of the area resulting from photogrammetric scanning.\r\n" +
        "2. Digital Surface Model (DSM) of the area.\r\n3. Slope.\r\n" + "4. Hypsometry,
classified into seven intervals.";

    private static string hooverText =
        "<b><align=center>INFORMACE</b></align>\n<align=justified>Hooverova přehrada
(anglicky Hoover Dam) je betonová klenbová přehrada na řece Colorado" +
        " v USA ležící na hranicích států Arizona a Nevada." +
        " Je vysoká 220 m a dlouhá 379 m. Přehradní nádrž Mead se táhne do vzdálenosti 185 km
a její hloubka dosahuje až 180 m." +
        " Je pojmenovaná po prezidentu Hooverovi a byla" +
```

```
        " postavena ve 30. letech 20. století, jako ve své době největší přehrada a
nejvýkonnější vodní elektrárna na světě.</align>";

    private static string sthelensText =
        "<b><align=center>INFORMACE</b></align>\n<align=justified>Mount St. Helens (česky
Hora St. Helens, v jazyce původních obyvatel z kmene Kauliců Lawetlat'la)" +
        " je aktivní sopka, nacházející se v severní části Kaskádového pohoří, ve státě
Washington, ve Spojených státech amerických." +
        " Jedná se o stratovulkán, tudíž je tvořena lávou a pyroklastiky, které ji svým
střídavým ukládáním postupně vybudovaly." +
        " Hora byla pojmenována po britském diplomatovi, nesoucí titul St Helens. Je známá
svým masivním sesuvem a silnou erupcí z roku 1980," +
        " jež během pár minut zničila území o ploše 600 km² a zabila 57 lidí.</align>";

    private static string earthText =
        "<b><align=center>INFORMACE</b></align>\n<align=justified>Země je třetí planeta
sluneční soustavy se střední vzdáleností od Slunce asi 1 au," +
        " zároveň největší terestrická planeta v soustavě a jediné planetární těleso, na němž
je dle současných vědeckých poznatků potvrzen život." +
        " Země vznikla před 4,6 miliardami let a krátce po svém vzniku získala svůj jediný
přirozený satelit – Měsíc. Země obíhá kolem Slunce po elipse" +
        " s velmi malou excentricitou dráhy. Země jako domovský svět lidstva má mnoho názvů v
závislosti na národu, mezi nejznámější patří název" +
        " latinského původu Terra, či řecký název Gaia.</align>";

    private static string hooverText_EN =
"<b><align=center>INFO</b></align>\n<align=justified>Hoover Dam is a concrete arch-gravity
dam in the Black Canyon" +
        " of the Colorado River, on the border between the U.S. states of Nevada and Arizona.
It was constructed between 1931 and 1936 during the Great Depression and" +
        " was dedicated on September 30, 1935, by President Franklin D. Roosevelt. Its
construction was the result of a massive effort involving" +
        " thousands of workers, and cost over one hundred lives. It was referred to as Hoover
Dam after President Herbert Hoover in bills passed" +
        " by Congress during its construction; it was named Boulder Dam by the Roosevelt
administration.</align>";

    private static string sthelensText_EN =
"<b><align=center>INFO</b></align>\n<align=justified>Mount St. Helens (known as Lawetlat'la
to the indigenous Cowlitz" +
        " people, and Loowit or Louwala-Clough to the Klickitat) is an active stratovolcano
located in Skamania County, Washington, in the Pacific Northwest region" +
        " of the United States. It lies 52 miles (83 km) northeast of Portland, Oregon, and
98 miles (158 km) south of Seattle. Mount St. Helens" +
        " takes its English name from that of the British diplomat Lord St Helens, a friend
of explorer George Vancouver who surveyed the area in the" +
        " late 18th century. The volcano is part of the Cascade Volcanic Arc, a segment of
the Pacific Ring of Fire.</align>";

    private static string earthText_EN =
"<b><align=center>INFO</b></align>\n<align=justified>Earth is the third planet from the Sun
and the only place known in the" +
        " universe where life has originated and found habitability. While Earth may not
contain the largest volumes of water in the Solar System, only Earth sustains liquid" +
        " surface water, extending over 70.8% of the Earth with its ocean, making Earth an
ocean world. Earth's polar regions currently retain most of all" +
        " other water with large sheets of ice covering ocean and land, dwarfing Earth's
groundwater, lakes, rivers and atmospheric water. Land, consisting" +
        " of continents and islands, extends over 29.2% of the Earth and is widely covered by
vegetation.</align>";

    private static string troskyText =
"<b><align=center>INFORMACE</b></align>\n<align=justified>Trosky jsou zřícenina hradu na
vrcholu stejnojmenného vrchu v katastrálním území obce Troskovice v okrese Semily v
Libereckém kraji." +
        " Nachází se na území Chráněné krajinné oblasti Český ráj a zároveň také Geoparku
Český ráj. Hrad je ve vlastnictví státu (správu zajišťuje Národní památkový ústav)" +
        " a je přístupný veřejnosti. Od roku 2002 je zařazen mezi národní kulturní
památky.</align>";

    private static string troskyText_EN =
"<b><align=center>INFO</b></align>\n<align=justified>Trosky Castle (Czech: Hrad Trosky) is a
castle ruin in the municipality of" +
        " Troskovice in the Liberec Region of the Czech Republic. It lies about 10 kilometres
(6 mi) south of Semily. It is located on the summits of two basalt volcanic plugs." +
```

```
        " On the lower peak, 47 metres (154 ft), is the two-storey structure called Baba
(Crone), and on the higher outcrop, 57 metres (187 ft), is the four-sided structure known" +
        " as Panna (Maiden). The castle is a landmark of the Bohemian Paradise
region.</align>";

private void Awake()
    {
        displayedText = GetComponent<TMP_Text>();
    }

    private void Update()
    {
        // MAIN MENU TEXT STRINGS

        if (textID == "hoover_dam_selector")
        {
            if (lang)
            {
                displayedText.text = "HOOVER DAM";
            }
            else
            {
                displayedText.text = "HOOVER DAM";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "st_helens_selector")
        {
            if (lang)
            {
                displayedText.text = "MOUNT ST. HELENS";
            }
            else
            {
                displayedText.text = "MOUNT ST. HELENS";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "globe_selector")
        {
            if (lang)
            {
                displayedText.text = "GLOBUS";
            }
            else
            {
                displayedText.text = "GLOBE";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "prague_selector")
        {
            if (lang)
            {
                displayedText.text = "HRAD TROSKY";
            }
            else
            {
                displayedText.text = "TROSKY CASTLE";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "info_selector")
        {
            if (lang)
            {
                displayedText.text = "INFORMACE";
            }
            else
            {
                displayedText.text = "INFO";
            }
            displayedText.fontSize = 40;
        }
        if (textID == "quit_selector")
```

```
        {
            if (lang)
            {
                displayedText.text = "VYPNOUT";
            }
            else
            {
                displayedText.text = "EXIT";
            }
        }
        if (textID == "info_box_menu")
        {
            if (lang)
            {
                displayedText.text =
"<b><size=130%>INFORMACE</b></size>\r\n\r\n<align=\"justified\"><size=110%>Aplikace byla
vytvořena v rámci diplomové práce" +
                    " <b>Geovizualizace založené na integraci 3D fyzických modelů terénu a
rozšířené reality</b>, řešené na Katedře geoinformatiky Univerzity" +
                    " Palackého v Olomouci v letech 2021-
2023.</align></size>\r\n\r\n<b>Autor:</b> Bc. Richard LÁZNA\r\n<b>Vedoucí práce:</b>" +
                    " RNDr. Jan BRUS, Ph.D.\r\n\r\n<size=140%><b>Project
VOID</b></size>\r\n2023\r\n\r\n<b>Zdroje:</b>\r\nObrázky na pozadí:\r\nHlavní menu:" +
                    " Unity Asset Store\r\nVedlejší menu: en.wikipedia.org\r\n\r\nDorpovodné
texty ve scénách:\r\ncs.wikipedia.org";
            }
            else
            {
                displayedText.text = "<b><size=130%>ABOUT THE
APPLICATION</b></size>\r\n\r\n<align=\"justified\"><size=110%>The application was created as
part" +
                    " of the diploma thesis <b>Geovisualisations based on integrating 3D
physical terrain models and augmented reality</b>, which was being worked" +
                    " on at the Department of Geoinformatics at Palacký University Olomouc
between 2021 and 2023. </align></size>\r\n\r\n<b>Author:</b>" +
                    " Bc. Richard LÁZNA\r\n<b>Supervisor:</b> RNDr. Jan BRUS,
Ph.D.\r\n\r\n<size=140%><b>Project VOID</b></size>\r\n2023\r\n\r\n<b>Sources:</b>\r" +
                    "\nBackgroud images:\r\nMain menu: Unity Asset Store\r\nSub menus:
en.wikipedia.org\r\n\r\nAccompanying text in scenes:\r\nen.wikipedia.org\r\n";
            }
            displayedText.fontSize = 30;
        }
        if (textID == "back_button")
        {
            if (lang)
            {
                displayedText.text = "zpět";
            }
            else
            {
                displayedText.text = "back";
            }
            displayedText.fontSize = 40;
        }
        if (textID == "help_box_menu")
        {
            if (lang)
            {
                displayedText.text = "<b><size=130%>JAK PRACOVAT S
APLIKACÍ</b></size>\r\n\r\n<align=\"justified\"><size=110%>Aplikace je vytvořena v prostředí
Unity3D" +
                    " a využívá principy rozšířené reality (AR). Celkem je aplikace
nachystána pro práci se čtyřmi modely. V menu aplikace se nachází tlačítka sloužící" +
                    " pro výběr modelu a vstup do odpovídajícího vedlejšího menu. V něm si
pak lze zvolit scénu pro vybraný model, pro každý model celkem čtyři. Po výběru" +
                    " scény přichází na řadu výběr hlavní scény IMAGE nebo PLANE. Scéna IMAGE
pracuje s markery, na základě kterých umisťuje obsah na fyzicky vytištěny" +
                    " model. Jako doprovodná scéna slouží právě PLANE, ve které vybraný
model, respektive scéna, zobrazí v prostoru po kliknutí na obrazovku. Tato scéna" +
                    " funguje na základě detekce rovin ve skutečném prostředí. \r\nDalší
funkcí aplikace je možnost zobrazení obsahu ve více jazycích.</align></size>";
            }
            else
            {
```

```
                displayedText.text = "<b><size=130%>HOW THE APP
WORKS</b></size>\r\n\r\n<align=\"justified\"><size=110%>The application is made in Unity3D
and uses" +
                    " augmented reality (AR). The application is prepared to work with with
four models in total. The buttons in the app menu serve to choose a model" +
                    " and enter a coresponding submenu. A scene can then be chosen in the
submenu, one out of four total for each model. After choosing a scene, a mode" +
                    " selection is then offered, being either IMAGE or PLANE. The IMAGE mode
uses markers to place augmented reality content on physically printed models." +
                    " The PLANE scene serves to show the selected scenario on the phone after
touching the screen. The scene uses Plane Detection, which detects planes in" +
                    " real enviroment. \r\nThe application can also be switched into
different languages. </align></size>";
            }
            displayedText.fontSize = 30;
        }
// SUBMENUS TEXT STRINGS

        if (textID == "scene1_selector")
        {
            if (lang)
            {
                displayedText.text = "SCÉNA #1";
            }
            else
            {
                displayedText.text = "SCENARIO #1";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "scene2_selector")
        {
            if (lang)
            {
                displayedText.text = "SCÉNA #2";
            }
            else
            {
                displayedText.text = "SCENARIO #2";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "scene3_selector")
        {
            if (lang)
            {
                displayedText.text = "SCÉNA #3";
            }
            else
            {
                displayedText.text = "SCENARIO #3";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "scene4_selector")
        {
            if (lang)
            {
                displayedText.text = "SCÉNA #4";
            }
            else
            {
                displayedText.text = "SCENARIO #4";
            }
            displayedText.fontSize = 74;
        }
        if (textID == "scene_info")
        {
            if (lang)
            {
                displayedText.text = "SCÉNY";
            }
            else
            {
                displayedText.text = "SCENES";
```

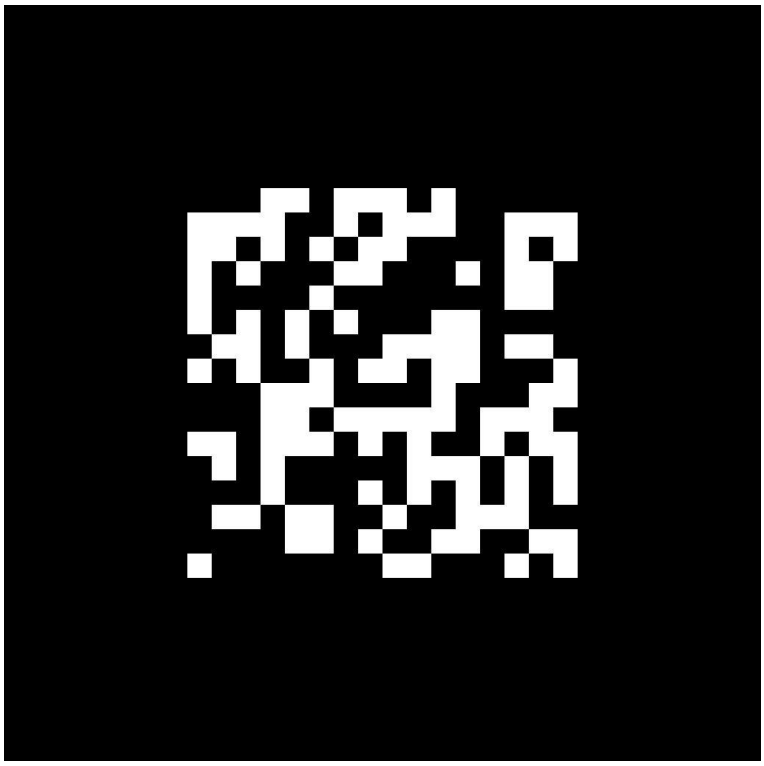
```
    }
    displayedText.fontSize = 40;
}
if (textID == "scene_mode_plane")
{
    if (lang)
    {
        displayedText.text = "ROVINA";
    }
    else
    {
        displayedText.text = "PLANE";
    }
    displayedText.fontSize = 74;
}
if (textID == "hoover_scenes_box_text")
{
    if (lang)
    {
        displayedText.text = hooverScenesText;
    }
    else
    {
        displayedText.text = hooverScenesText_EN;
    }
    displayedText.fontSize = 30;
}
if (textID == "st_helens_scenes_box_text")
{
    if (lang)
    {
        displayedText.text = sthelensScenesText;
    }
    else
    {
        displayedText.text = sthelensScenesText_EN;
    }
}
if (textID == "earth_scenes_box_text")
{
    if (lang)
    {
        displayedText.text = earthScenesText;
    }
    else
    {
        displayedText.text = earthScenesText_EN;
    }
}
if (textID == "trosky_scenes_box_text")
{
    if (lang)
    {
        displayedText.text = troskyScenesText;
    }
    else
    {
        displayedText.text = troskyScenesText_EN;
    }
}
if (textID == "scene_image_headline")
{
    if (ImageTracking.selectedPrefabIndex == "01")
    {
        if (lang)
        {
            displayedText.text = "SCÉNA #1";
        }
        else
        {
            displayedText.text = "SCENE #1";
        }
        displayedText.fontSize = 40;
    }
    if (ImageTracking.selectedPrefabIndex == "02")
```

```
            {
                if (lang)
                {
                    displayedText.text = "SCÉNA #2";
                }
                else
                {
                    displayedText.text = "SCENE #2";
                }
                displayedText.fontSize = 40;
            }
            if (ImageTracking.selectedPrefabIndex == "03")
            {
                if (lang)
                {
                    displayedText.text = "SCÉNA #3";
                }
                else
                {
                    displayedText.text = "SCENE #3";
                }
                displayedText.fontSize = 40;
            }
            if (ImageTracking.selectedPrefabIndex == "04")
            {
                if (lang)
                {
                    displayedText.text = "SCÉNA #4";
                }
                else
                {
                    displayedText.text = "SCENE #4";
                }
                displayedText.fontSize = 40;
            }
        }

// MAIN SCENE TEXT STRINGS

        if (textID == "info_box_text")
        {
            if (ImageTracking.selectedModel == "HooverDam")
            {
                if (lang)
                {
                    displayedText.text = hooverText;
                }
                else
                {
                    displayedText.text = hooverText_EN;
                }
                displayedText.fontSize = 30;
            }
            if (ImageTracking.selectedModel == "StHelens")
            {
                if (lang)
                {
                    displayedText.text = sthelensText;
                }
                else
                {
                    displayedText.text = sthelensText_EN;
                }
                displayedText.fontSize = 30;
            }
            if (ImageTracking.selectedModel == "Earth")
            {
                if (lang)
                {
                    displayedText.text = earthText;
                }
                else
                {
                    displayedText.text = earthText_EN;
                }
```

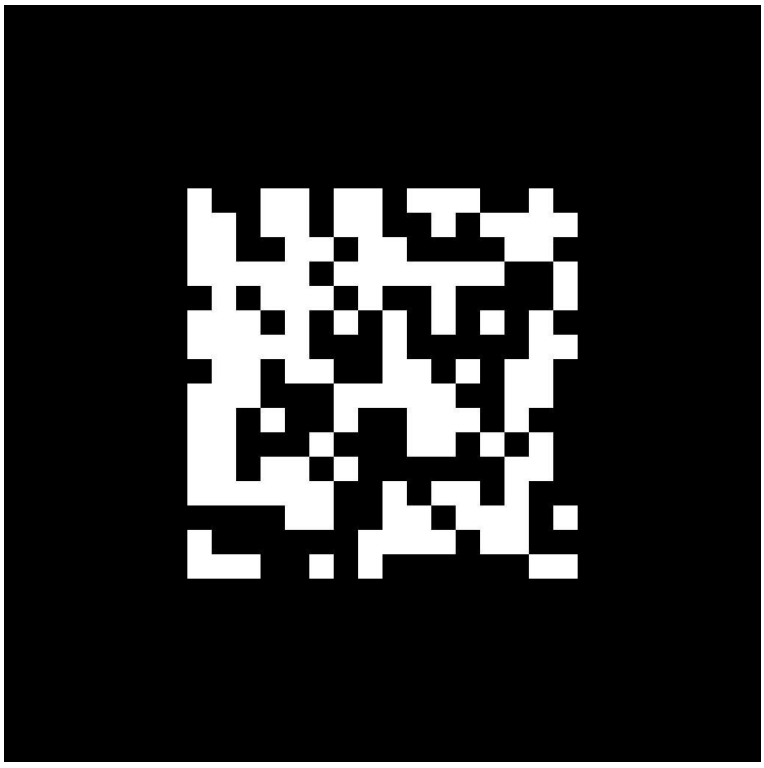
```
                displayedText.fontSize = 30;
            }
            if (ImageTracking.selectedModel == "Trosky")
            {
                if (lang)
                {
                    displayedText.text = troskyText;
                }
                else
                {
                    displayedText.text = troskyText_EN;
                }
                displayedText.fontSize = 30;
            }
        }
        if (textID == "close_button")
        {
            if (lang)
            {
                displayedText.text = "zavřít";
            }
            else
            {
                displayedText.text = "close";
            }
            displayedText.fontSize = 40;
        }
        if (textID == "exit_button")
        {
            if (lang)
            {
                displayedText.text = "VYPNOUT";
            }
            else
            {
                displayedText.text = "EXIT";
            }
            displayedText.fontSize = 50;
        }
    }

    public static void SetCZ()
    {
        lang = true;
    }

    public static void SetEN()
    {
        lang = false;
    }
}
```

```
using UnityEngine;
using UnityEngine.UI;

public class ImageHandler : MonoBehaviour
{
    public Button sceneButton;

    private void Awake()
    {
        sceneButton = GetComponent<Button>();
    }

    private void OnEnable()
    {
        var newSprite = Resources.Load<Sprite>($"Images/{ImageTracking.selectedModel +
ImageTracking.selectedPrefabIndex}");
        sceneButton.image.sprite = newSprite;
    }
}
```

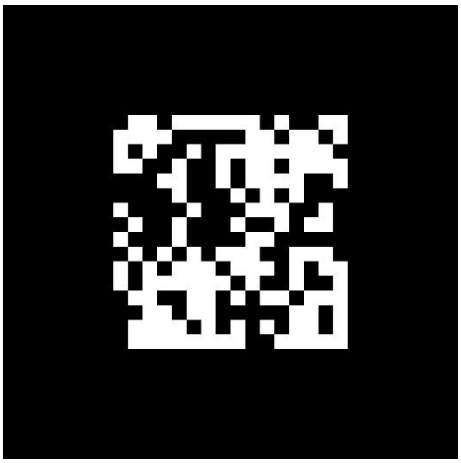
```
using UnityEngine;

public class RotateObject : MonoBehaviour
{
    // An object to be rotated around.
    public GameObject sphere;
    // Another object to be rotated around.
    public GameObject sphere2;
    // Speed of the rotation.
    public float angular_speed;

    void Update()
    {
        // If a GameObject (set inside the Unity editor) is not null, executes the function.
        if (sphere != null)
        {
            transform.RotateAround(sphere.transform.position, Vector3.up, angular_speed *
Time.deltaTime);
        }
        if (sphere2 != null)
        {
            transform.RotateAround(sphere2.transform.position, Vector3.up, angular_speed *
Time.deltaTime);
        }
    }
}
```



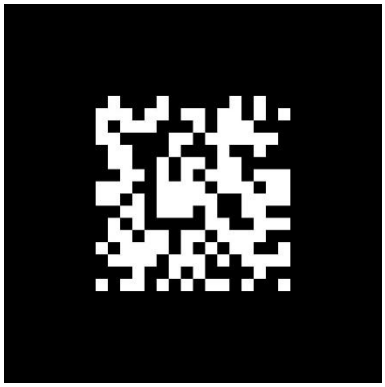
Marker k modelu Hoover Dam 1 : 5000.



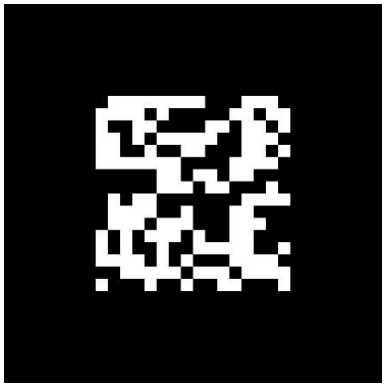
Marker k modelu Mount St. Helens 1 : 12 500.



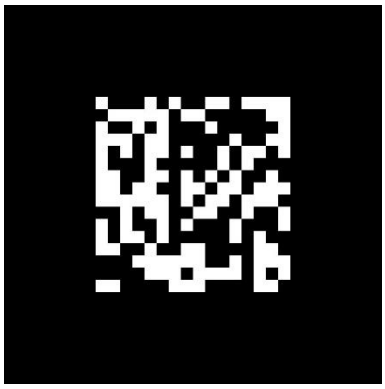
Marker k modelu Globus 1 : 60 000 000.



Marke k modelu Hrad Trosky 1 : 2 200.



Marker k modelu Hoover Dam 1 : 12 500.



Marker k modelu Mount St. Helens 1 : 30 000.