Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

NÁVRH A TVORBA INTERAKTIVNÍ EXHIBICE S VYUŽITÍM GEOINFORMAČNÍCH TECHNOLOGIÍ

Diplomová práce

Bc. Petr MUŽÍČEK

Vedoucí práce RNDr. Jan BRUS, Ph.D.

Olomouc 2021
Geoinformatika

ANOTACE

Práce se zabývá analýzou potenciálu a aplikováním geoinformačních technologií do tvorby interaktivní exhibice historických objektů. Jejím cílem je návrh a implementace řešení využívající primárně technologie rozšířené reality. Dílčím cílem práce je optimalizace historického modelu území pro následné využití pro interaktivní prezentaci v prostředí rozšířené reality. Je detailně popsán postup práce v prostředí programů Blender a Unity.

Optimalizace modelů proběhla v prostředí Blender. Optimalizace proběhla ve smyslu geometrických úprav modelů, texturování a změny jejich rotace. Výsledné modely byly přeneseny do Unity. Zde byla také vytvořena mobilní aplikace, která je pilotním výsledkem a výstupem diplomové práce. Aplikace obsahuje logickou strukturu v podobě scén vytvořených v Unity. Praktické použití aplikace je v lokalitě Svatováclavského návrší v Olomouci, zejména její exteriérová část. Interiérová část umožňuje zobrazit návrší kdekoliv.

Hlavním výsledkem je tedy mobilní aplikace dostupná na mobilní zařízení využívající operační systém Android verze 10.0 a výše. Dalšími výsledky jsou webové stránky, informační poster a grafické markery.

KLÍČOVÁ SLOVA

rozšířená realita; plocha; marker; Android

Počet normostran práce: 52

Počet příloh: 22 (z toho 2 volné a 1 elektronická)

ANOTATION

The diploma thesis focuses on analysis of potential and application of geoinformation technologies in the creation of interactive exhibitions of historical objects. Its goal is the design and implementation of solutions using primarily augmented reality technology. A partial goal of the work is to optimize the historical model of the area for subsequent use for interactive presentation in the augmented reality environment. The procedure of work was done in the environment of Blender and Unity programs and is described in detail.

Model optimization took place in the Blender environment. The optimization took place in terms of geometric modifications of models, texturing and changes in their rotation. The resulting models were transferred to Unity. A mobile application was also created here, which is a pilot result and output of the diploma thesis. The application contains a logical structure in the form of scenes created in Unity. The practical use of the application is in the locality of St. Wenceslas Hill in Olomouc, especially its exterior part. The interior allows you to view the hill anywhere.

The main result is a mobile application available on mobile devices using the Android operating system version 10.0 and higher. Other results are a website, an information poster and graphic markers.

KEYWORDS

augmented reality; plane; marker; Android

Number of pages: 52

Number of appendixes: 23

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, aby jeden výtisk diplomové práce byl uložen v Knihovně UP k prezenčnímu nahlédnutí,

- 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).

V Olomouci dne

Bc. Petr Mužíček

Děkuji vedoucímu práce RNDr. Janu Brusovi, Ph.D. za nekonečnou trpělivost, konstruktivní podněty a připomínky při vypracování práce. Dále děkuji svým spolužákům Tereze Novákové, Elišce Regentové a Martinu Sadílkovi za neutuchající podporu a společné roky studia. Poděkování patří také skvělým kamarádkám Kláře Košťálové a Tereze Klodnerové, které mi byly zejména psychickou oporou ve chvílích dobrých i zlých. Největší díky ovšem patří mé rodině, bez jejíž podpory bych se na během studia nikdy neobešel.

Velký dík patří Vlastivědnému muzeu v Olomouci za poskytnutí modelů návrší, také studentům geoinformatiky a Mgr. Radku Barvířovi za vytvoření digitálních modelů návrší, bez nichž by vypracování diplomové práce nebylo možné.

UNIVERZITA PALACKÉHO V OLOMOUCI

Přírodovědecká fakulta

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Petr MUŽÍČEK**
Osobní číslo:	**R190013**
Studijní program:	**N1301 Geografie**
Studijní obor:	**Geoinformatika**
Téma práce:	**Návrh a tvorba interaktivní exhibice s využitím geoinformačních technologií**
Zadávající katedra:	**Katedra geoinformatiky**

Zásady pro vypracování

Cílem diplomové práce je analýza potenciálu a aplikování geoinformačních technologií do tvorby interaktivní exhibice hostorických objektů. Student nejprve ověří stávající řešení a následně v případové studii narvhrne a implemetuje řešení využívající primárně technologii rozšířené reality. Dílčím cílem práce bude optimalizace historického modelu území pro následné využití pro interaktivní prezentaci v prostředí rozšířené reality. 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:

ARISO, J. M. Augmented reality. Berlin Studies in knowledge Research. Berlin, 2017. ISBN 978-3-11-049700-7.

ONG, S., Beginning Windows Mixed Reality Programming: For HoloLens and Mixed Reality Headsets, Apress, 2017, ISBN: 978-1484227688.

MUÑOZ-SAAVEDRA, L., MIRÓ-AMARANTE, L., DOMÍNGUEZ-MORALES, M., Augmented and Virtual Reality Evolution and Future Tendency [online]. Architecture and Computer Technology Department, Universidad de Sevilla, 2020. Dostupné z: https://www.mdpi.com/2076-3417/10/1/322?.

ALKHAMISI, A. O. AND MONOWAR, M. M., Rise of Augmented Reality: Current and Future Application Areas. Department of Information Technology, Faculty of Computing and Information Technology, Jeddah, Saudi Arabia, 2013. http://dx.doi.org/10.4236/ijids.2013.14005.

VOŽENÍLEK, V. Diplomové práce z geoinformatiky. Edition ed. Olomouc: Vydavatelství Univerzity Palackého, 2002.

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

Datum zadání diplomové práce: **7. října 2019**
Termín odevzdání diplomové práce: **6. května 2021**

L.S.

doc. RNDr. Martin Kubala, Ph.D.
děkan

prof. RNDr. Vít Voženílek, CSc.
vedoucí katedry

V Olomouci dne 7. září 2020

OBSAH

SEZNAM POUŽITÝCH ZKRATEK

Zkratka	Význam
2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
AR	Augmented Reality
ARM	Augmented Reality Mirror
DMR	Digitální Model Reliéfu
MARS	Mixed and Augmented Reality Studio
MR	Mixed Reality
SDK	Software Development Kit
SW	SoftWare
UI	User Interface
VR	Virtual Reality

ÚVOD

Není to tak dávno, co byl světu představen první chytrý telefon nebo elektromobil. Ovšem lidstvo je předurčeno k překonávání limitů což se týká například dobývání vesmíru nebo vynalézáním nových technologií. Jsou to právě technologie, jejichž vývoj, objevování a zdokonalování postupuje neúprosným tempem směrem vpřed. Jednou z nich je i rozšířená realita (AR). Ta už dávno pronikla i do mobilních telefonů a její podpora dnes existuje prakticky na každém smartphonu nebo tabletu. Fenoménem jsou hry využívající technologie AR na zmíněná zařízení, například mobilní hra PokémonGO je celosvětově proslulá.

Rozšířená realita je druh smíšené reality, kdy zejména smartphone nebo tablet pomocí AR detekuje prostor zaznamenaný kamerou těchto zařízení. Společně se zaznamenanou scénou jsou zaregistrována i osvětlení, poloha a také orientace kamery v prostoru. Na základě těchto parametrů je pak do prostoru umístěn počítačem vytvořený objekt nebo grafika v podobě videa, animace nebo textu.

S tímto pokrokem se AR rychle začala využívat zejména v herním průmyslu, logistice a marketingu. Svojí interní AR aplikaci tak mají světové značky jako například LEGO nebo IKEA. Tímto způsobem se tak propagace produktů dostává do dříve nemyslitelných měřítek.

V České republice tato technologie a její využití není příliš rozšířené a známé, i přes zmíněný fenomén v podobě PokémonGO. Cílem diplomové práce je proto vytvořit mobilní aplikaci využívající technologii AR za účelem přiblížit veřejnosti její potenciál a možné využití, které tak bude demonstrováno na lokalitě Svatováclavského návrší v Olomouci, konkrétně v jeho venkovní a vnitřní exhibici. Návrší prošlo v průběhu staletí velkým vývojem, a proto prostřednictvím aplikace bude umožněno prohlédnout si návrší a jeho změny v průběhu století za pomoci vytvořené aplikace, která tak zvýší atraktivnost exhibice a její interaktivnost s návštěvníkem.

1 CÍLE PRÁCE

Cílem diplomové práce je analýza potenciálu a aplikování geoinformačních technologií do tvorby interaktivní exhibice historických objektů. Student nejprve ověří stávající řešení a následně v případové studii navrhne a implementuje řešení využívající primárně technologii rozšířené reality. Dílčím cílem práce bude optimalizace historického modelu území pro následné využití pro interaktivní prezentaci v prostředí rozšířené reality.

Hlavní cíl je rozdělen do dvou hlavních částí. První je optimalizace historických digitálních modelů pro jejich následné využití a zobrazení v prostředí rozšířené reality. Druhým pak je vytvoření mobilní aplikace pro platformu Android, díky které bude umožněna prezentace historických modelů v prostředí rozšířené reality.

Dílčími cíli bylo vytvoření logické struktury mobilní aplikace. Ta je rozdělena na dvě hlavní části. První je prezentace historických modelů v interiérové části exhibice Svatováclavského návrší v prostředí rozšířené reality. Zde byly vytvořeny dvě další části, z nichž každá má za úkol historický model prezentovat pomocí odlišných metod rozšířené reality. Druhou hlavní částí je prezentace historických modelů v exteriérové části návrší, tedy ve venkovním prostranství Svatováclavského návrší.

Výsledná mobilní aplikace tak umožní zinteraktivnění exhibice Svatováclavského návrší v Olomouc jak ve venkovní, tak ve vnitřní části exhibice. Využitím mobilních zařízení zvýší interaktivitu exhibice zejména pro mladší část populace.

Na výsledky této práce pak budou moci navázat studenti vysokých škol nebo zaměstnanci z oborů např. kultury, historie, informatiky apod.

2 METODY A POSTUPY ZPRACOVÁNÍ

Použité metody

K vypracování diplomové práce byla z majoritní části využita technologie rozšířené neboli augmentované reality (AR). Toto označení se používá pro obraz reálného světa zachyceného pomocí smartphonu, tabletu nebo chytrých brýlí, který je obohacen virtuálním objektem. Tímto objektem může být jakákoli počítačem vytvořená grafika v podobě textu, videa apod. Každé ze zmíněných zařízení v reálném čase vypočítává a zobrazuje virtuální objekt, který má být zobrazen. Druh zobrazení může záviset buď na uživateli nebo zadání druhu zobrazování AR. Uživatel se tak stává operátorem aplikace, ve které je schována funkcionalita zobrazující AR. V této práci jsou využity následné metody:

- Image Tracking – rozpoznání obrazu a umístění virtuálního objektu na něj
- Plane Tracking – rozpoznání plochy a umístění virtuálního objektu na ni
- GPS Tracking – ukotvení virtuálního objektu na zadané GPS souřadnice

Obrazovým prvkem metody *Image Tracking* je takzvaný marker. Marker je ve své podstatě jednoduchý QR kód nebo libovolný obrázek, do kterého je zakódována informace, kterou má po přečtení mobilním zařízením zobrazit. V tomto případě je to digitální 3D model. *Plane Tracking* funguje na principu, kdy mobilní zařízení detekuje plochu, ukládá jí, přepočítává a posléze na ní lze umístit digitální 3D model. Poslední metoda *GPS Trackingu* se zakládá na umístění digitálního 3D modelu staticky na GPS souřadnice pomocí Unity Assetu **Unity AR+GPS Location**.

Uvedené metody jsou technologicky nenáročné a principiálně jednoduché k pochopení a použití. Neúplná známost odvětví AR v porovnání s virtuální realitou (VR) ovšem vede k omezení funkcionality nástrojů. Mobilní aplikace je funkční pro operační systém Android od verze Android 10. Výsledná publikace aplikace bude k dispozici na platformním obchodě (Google Play).

Použitá data

Pro vypracování diplomové práce byly použity digitální modely Václavského návrší v Olomouci napříč jednotlivými stoletími. Tato data byla poskytnuta Vlastivědným muzeem v Olomouci a vytvořili je studenti katedry geoinformatiky Univerzity Palackého v Olomouci. Zmíněné modely byly ve formátu .stl a vytvořeny v software (SW) SketchUp. Dále bylo pracováno s Extensive MarkUp Language (XML) soubory, které do výsledného řešení zahrnuty nebyly. Jednalo se o soubory s vepsanou GPS lokací, nicméně toto řešení nebylo pro danou část práce vhodné, a proto od něj bylo upuštěno. Veškeré C# skripty byly vytvořené autorem, pokud se nejednalo o skripty nativně uložené v nainstalovaných knihovnách.

Použité programy

Primární úprava modelů byla provedena v SW Blender 2.90.1. Úprava spočívala v texturování modelů a jejich ořezání. Dále zde modely byly otočeny podle osy x z důvodu rozdílnosti souřadnicových systémů SW Blender a Unity. Původní .stl modely byly poté vyexportovány do Unity projektu ve formátu .fbx.

V prostředí Unity 2020.3.15.f2 pak byla vytvářena samotná mobilní aplikace, ale také dodatečná práce s modely v podobě napojení textur a jejich zvětšení, popřípadě

zmenšení. Tyto operace závisely na výskytu modelů v určité části aplikace. Při testování aplikace na zařízení iOS byla nejdřív nutná instalace pomocí programu XCODE 12.5.1. Pro export aplikace pro platformu Android byly v Unity nativně nainstalovány následující knihovny:

- JDK
- Android SDK Tools
- Android NDK
- Gradle 7.0

Pro tvorbu základní funkcionality aplikace byly využity následující balíčky nainstalované v prostředí Unity:

- ARFoundation a AR Subsystems 2.1.16
- Unity MARS 1.3.1
- Unity AR+GPS Location 3.5.5
- ARCore XR Plugin 2.1.16
- ARKit XR Plugin 2.1.16

K tvorbě C# skriptů byl využíván SW Visual Studio 2019. Pro tvorbu textu byl použit produkt ze sady Microsoft Office 365 Word. K tvorbě prezentací pak program ze stejné sady PowerPoint. Webové stránky byly vytvořeny v textovém editoru PSPad a finální poster v SW Vectornator.

Postup zpracování

Na začátku práce bylo důležité upravit stávající modely vytvořené ve SketchUpu. Úpravy se týkaly jejich texturování, ořezání, změny orientace a polohy. Všechny modifikace byly provedeny v SW Blender. Odtud byly upravené modely exportovány do projektu Unity. Klíčové bylo správné nastavení exportu, kde bylo nutné texturovaným modelům přiřadit informace o texturách, zachování nastavených parametrů v podobě orientace, polohy a velikosti. Modely použité pro exteriérovou část aplikace bylo nutné otočit o 180 stupňů, aby korespondovaly s reálným objektem Svatováclavského návrší. Vyexportované modely musely být ve formátu .fbx, jinak nebylo možné s nimi v prostředí Unity operovat.

V SW Unity pak bylo nutné znovu nastavit parametry texturám, aby se co nejvíce podobaly parametrům již nastaveným v Blenderu. Poté byly vytvořeny jednotlivé scény mobilní aplikace, taktéž v SW Unity. Pro zajištění funkcionality bylo zapotřebí stáhnout si jednotlivé balíčky zmíněné výše. Byly vytvořeny C# skripty, které obstaraly funkcionalitu tlačítkům, ať už se jednalo o přepínání scén, vyskakovací okno, umísťování digitálních modelů na detekovanou plochu nebo její vypínání. Aplikace byla rozdělena na dvě stěžejní části – **INTERIÉR** a **EXTERIÉR**, kdy interiérová část byla dále rozdělena na další dvě části – **MARKER** a **PLANE**. Každá z těchto částí má rozdílný účel a funkcionalitu (viz Kapitola 4).

Výsledná aplikace pak byla vyexportována na platformu Android v podobě instalačního souboru .apk. Tento soubor pak byl umístěn na katederní účet Google Play, ze kterého je dostupný pro širokou veřejnost již v podobě aplikace.

3 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Muzea jsou na území České republiky součástí kulturního života již více než 200 let. Za první muzeum bývá považováno Slezské zemské muzeum v Opavě, založené roku 1814. Nicméně některé zdroje jako nejstarší uvádějí Národní galerii v Praze, kde na konci 18. století vznikla soukromá sbírka, která se pro veřejnost otevřela o pár let později. Prvním zlomovým bodem byla druhá polovina 19. století, kdy docházelo k postupnému prohlubování a profesionalizaci muzejní činnosti. S tímto je spojen vznik mnoha vědeckých prací, které ovlivnily budoucí směr rozvoje muzejní vědy. Koncem 19. století vznikají nová specializovaná muzea (např. sklářské muzeum v Novém Boru). Ovšem stejně jako v mnoha jiných odvětvích brzdu dalšímu rozvoji vystavila první a druhá světová válka a s ní spojené majoritní škody na sbírkách. Stagnace rozvoje muzeí byla zopakována v 70. a 80. letech minulého století, kdy zejména kvůli zastaralé technice a prostorám z let minulých dochází k zaostávání oproti západním muzeím a tento trend v některých případech převládá až dodnes. Změnit tento stav je tak jedním z klíčových úkolů českého muzejnictví, které čítá okolo 800 objektů specifikovaných jako muzeum či galerie (dtest.cz, 2020).

K úspěšné modernizaci muzeí tak mají vést tři základní kroky: nejprve je to integrace přírody a kultury, kde příroda je jakýmsi přirozeným pojítkem. Za druhé je důležité ukázat vědecké sbírky veřejnosti, kde důležitou součástí je pečlivě organizovaný archivační systém. Třetím krokem je pak správa, otevírání a oživování sbírek pro vědce a veřejnost, kde právě oživování sbírek je v mnoha ohledech klíčové pro zvýšení atraktivity a potenciálu muzeí (Pittman, 2012). Nicméně opravdovým klíčem k modernizaci je právě využití moderních technologií, jako je například rozšířená realita (AR).

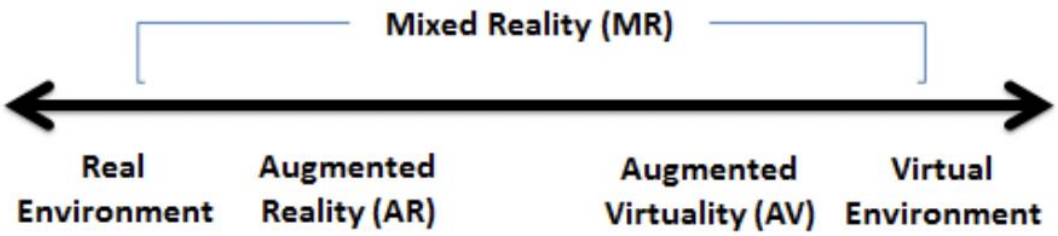
Poprvé se pojem AR objevuje v 50. letech 20. století, kdy kameraman Morton Heilig konstatoval, že kino jako umění by mělo být schopné vtáhnout diváka do dění na plátně. Sám Heilig pak v roce 1955 vytvořil prototyp předchozího nápadu jménem „The Cinema of the Future“ známy jako Sensorama, to vše před vznikem digitální výpočetní techniky (Carmigniani, 2011). Následovalo vytvoření prvního na zařízení připevněného na hlavu Ivanem Sutherlandem, který v roce 1968 vytvořil první fungující prototyp AR (Johnson, 2011). Opravdový přelom znamenalo vytvoření AR laboratoře, která poprvé umožňovala uživatelům jednoduchou interakci s virtuálními elementy. Na začátku 90. let se AR stala vědním oborem a v roce 2000 Bruce Thomas s první mobilní hrou založenou na AR (Furht, 2011). Převratem AR v 21. století byl vznik Google Glass a MicroSoft HoloLens, a následná změna orientace AR na mobilní zařízení. Nejznámějším počinem je tak AR mobilní hra PokémonGO, ovšem nejzásadnějším je vytvoření platforem pro AR ARKit a ARCore v roce 2017, což znamenalo a stále znamená obrovský nárůst mobilních aplikací založených na AR (Nowacki, 2020).

Rozšířená realita je forma objevující se zkušenosti, ve které je reálný svět obohacen počítačově generovaným obsahem vázaným na specifickou lokaci nebo aktivity. AR tím pádem umožňuje digitálnímu obsahu hladké překrytí a smíšení s prvky reálného světa. Kromě 2D a 3D objektů, z nichž mnohé mohou obsahovat digitální aktiva (např. audio, video, textová informace, hmatové informace atd.), která tak obohacují a rozšiřují uživatelovo vnímání reálného světa. Dohromady pak tyto augmentace slouží jako pomůcka a rozšíření vědomostí a porozumění jednotlivců tomu, co se odehrává v jejich okolí. AR se tedy vztahuje k širokému spektru technologií, které

zobrazují počítačově generované materiály jako jsou obrázky, videa nebo třeba texty, ať už pomocí mobilních zařízení nebo chytrých brýlí.

Nicméně takové definice jsou velmi obecné, a proto Azuma a další výzkumníci definovali implementaci AR na základě tří charakteristik. Základem je kombinace reálného světa a virtuálních prvků, u kterých je podmínkou interaktivita v reálném čase, a které jsou registrované ve 3D (zobrazení virtuálních objektů nebo informací je přímo spojené s místem a orientací v reálném světě). Podobně AR systémy definuje Hollerer a Feiner, podle nichž je AR průnik reálné a počítačově generované informace v reálném prostředí, interaktivně a v reálném čase se schopností zarovnat virtuální objekty s reálnými. Nejjednodušší a nejvýstižnější definicí pak může být, že AR jako technologie dovoluje počítačově generované obrazy přesně překrývat fyzické objekty v reálném čase.

Ve výsledku je AR součástí uceleného a fluidního systému zvaného Mixed Reality (MR) (viz Obrázek 1). AR je tedy velmi blízce vázaná jak na reálný svět (reálné objekty), tak na virtuální prostředí (virtuální objekty). Často bývá zaměňována za virtuální realitu (VR). Obě jsou interaktivní, pohlcující a zahrnující informační citlivost. Rozdílem je pojetí uživatelského rámce. Ve VR je uživatelský rámec zcela vázán na virtuální svět, kdežto v AR je uživatelovo vnímání stále vztaženo k reálnému světu a virtuální objekty spolu s reálnými koexistují ve stejném prostoru (Azuma, 1997).



Obrázek 1 Schéma Mixed Reality

Zdroj: https://aquila.usm.edu/cgi/viewcontent.cgi?article=1022&context=jetde

3.1 Druhy AR

Vstupy pro technologii AR mohou být buď vizuální, audio nebo audiovizuální. Na základě těchto vstupů dochází k rozšíření vnímání světa uživatelem. Rozšiřuje se aktuální prostředí za vytváření MR namísto nahrazení reálného světa virtuálním. Dřívější typologie se zaměřovaly spíše na technické zaměření jednotlivých přístupů (Normand, 2012). Aplikace AR jsou tak rozděleny na základě jejich funkčních charakteristik různých aplikací. Podobné klasifikace mohou pomoci výzkumníkům zvážit dané problémové oblasti, které vyhovují jednotlivým typům AR. AR má šest základních typů spadajících do 2 zastřešujících kategorií (Reger, 2016).

3.1.1 Triggered AR

Základem jsou takzvané spouštěče (triggery), což jsou stimuly nebo lépe objekty, které spouštějí augmentaci reálného objektu/prostoru. Do této kategorie patří následující typy AR:

- Markerová AR
    - Podmínkou pro augmentaci je jakýkoliv marker (viz Obrázek 2). Základními typy markerů jsou papírový nebo fyzický objekt, který

existuje v reálném světě. Augmentace vztažené k markerům poté obohacují obraz nebo objekt s pomocí kamery/fotoaparátu s příslušným softwarem (SW), který umožňuje zobrazování AR. Pro marker se nejčastěji používá kombinace černé a bílé barvy, jediným kritériem pro výběr barev markeru je jejich dostatečný kontrast (Katiyar, 2015).

Obrázek 2 Markerová AR

Zdroj: https://www.inaugment.com/what-are-the-types-of-augmented-reality/marker-ar-1024x682/

- Lokačně založená
    - Poloha/lokace je dalším příkladem spouštěče (triggeru). Lokačně založená AR využívá GPS lokaci daného zařízení jako spouštěč pro spárování dynamické lokace s body zájmu za cílem poskytovat relevantní data nebo informace (viz Obrázek 3). Funguje tedy na principu odesílání informací z mobilního zařízení na základě GPS nebo LBS systémů. Tyto systémy pak odešlou zpět informaci do mobilního zařízení a na základě této informace je posléze zobrazena AR (Edwards-Stewart, 2016).

Obrázek 3 Lokačně založená AR

Zdroj: https://medium.com/corebuild-software

- Dynamická augmentace
    - Je responzivní ku zobrazení objektu, který se mění. Pokud má nadefinované sledování pohybu, může docházet k jistému stupňování augmentace, aby virtuální objekt více odpovídal snímanému reálnému objektu.
- Komplexní augmentace
    - Páruje reálný a dynamický pohled na svět pomocí digitální informace, nejčastěji zjištěné pomocí internetu. Jedná se o kombinace markerové AR a dynamické augmentace. Nejtypičtějším příkladem je původní koncept pro Google Glass, kde uživatelé vidí informace o lokálních sítích na základě jejich lokace stanovené GPS. Objekty v zorném poli uživatele jsou tak prezentovány s pomocnými informacemi o jejich okolí ze strany internetu.

3.1.2 View-based

Druhá kategorie zahrnuje buď digitalizované augmentace bez jakékoliv reference na to, co je v náhledu zobrazovacího zařízení (mobil, tablet apod.), nebo jí může být augmentace uloženého statického pohledu. Ve své podstatě se jedná o dynamické povrchy, které detekuje AR aplikace a výslednou informaci pak zobrazuje na displeji zařízení.

- Nepřímá augmentace
    - Augmentace statického náhledu na reálný svět je inteligentní. To znamená, že je zde velmi často zahrnuto augmentování obrazů. Nejtypičtějším příkladem jsou aplikace, které dovolují uživateli například pořídit obrázek místnosti, a poté interaktivně měnit barvu této místnosti (viz Obrázek 4). Další možností je využití senzorově založeného sledování, které značně rozšiřuje možnosti využití tohoto typu AR, ale na úkor horšího sledovacího výkonu (Wither, Tsai, Azuma, 2011).

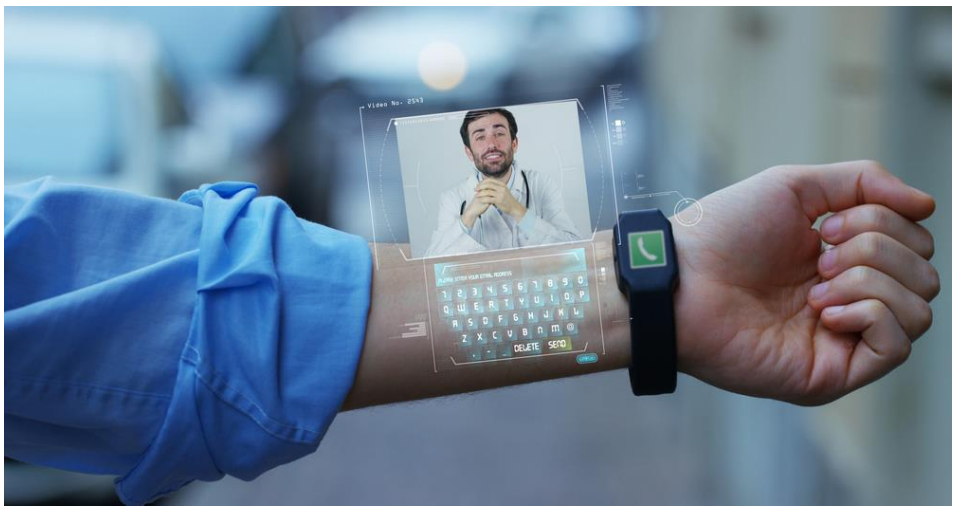
Obrázek 4 Příklad nepřímé augmentace

Zdroj: Azuma, 2011

- Nespecifikovaná digitální augmentace
    - Digitalizuje dynamický pohled na svět bez jakékoliv reference na to, co se zobrazuje. Nejčastěji se používá v mobilních hrách. Uživatel interaguje s augmentací například tak, že prstem poklepává na augmentaci v momentu, kdy se augmentace zobrazí. Neexistuje zde žádná reference v návaznosti na uživatelské prostředí.

3.1.3 Ostatní případy AR

- Projekčně založená AR
    - Nejedná se o technologii, která je ovládána uživatelem, nýbrž o technologii, která projektuje světelné paprsky na fyzický povrch. Využívá zpravidla jeden nebo více projektorů uspořádaných okolo fyzického objektu nebo ve 3D prostoru (Mine, Rose, Yang, 2012).



Obrázek 5 Příklad projekčně založené AR

Zdroj: https://thedailychronicle.in/

- Obrysová AR
    - Identifikuje linie a hranice, které lidské oko není schopno rozeznat. Zakládá se na rozpoznávání objektů pro pochopení uživatelova okolí. Typickým příkladem je zobrazení hrany vozovky za nízké viditelnosti nebo zobrazení struktury budovy z venkovního prostředí (Borkhataryia, 2020).

3.2 Zařízení AR

Rozšířená realita se stává jedním z klíčových hráčů technologické ekonomiky. Podle některých predikcí je hodnota trhu s AR odhadována až na 100 miliard amerických dolarů v roce 2020. Důvodem je příslib přidané virtuální hodnoty napříč všemi vědními spektry. Už pár let AR dokazuje svůj potenciál v řešení různých problémů například ve zdravotnictví, maloobchodu, obchodní logistice, turistice apod. A právě k těmto účelům se využívají následující technologie (Paine, 2018) (viz Obrázek 6).

Technology Generation		Realities	
	Wearable	smart watches, VR-Glasses	AR Smart Glasses
	Mobile	mobile phones, laptop computers, tablets	Mobile AR software applications
	Stationary	desktop computers, gaming consoles	Virtual Mirrors (AR-mirrors)
	Offline	TV, Teletex, newspapers	N/A
		Virtual Reality (VR)	**Augmented Reality (AR)**

Obrázek 6 Základní rozdělení zařízení zobrazujících AR

Zdroj: Paine, 2018

3.2.1 Smartphony a tablety (mobilní)

Systém AR obohacuje nebo augmentuje okolí uživatele za pomocí virtuální informace, která je registrovaná v 3D prostoru a vypadá jako by koexistovala s reálným světem (Azuma, 2001). Na rozdíl od tradičnějších zařízení (headsety) smartphony a tablety dokonale kombinují všechny technologie pro augmentaci v malém kompaktním zařízení, kterým zejména smartphony jsou. A právě například v turistice byla mobilní zařízení prvním svého druhu, která představila svět AR (Seo, 2011). Typickým příkladem je využití při venkovní turistice, kdy uživatel jednoduše namíří smartphone s aktivní GPS směrem k libovolnému objektu ve svém okolí. Potom je zařízení schopné zobrazit další informace, které překrývají obraz reálného světa. Největším benefitem tak je bezprostřední zobrazení dodatečných informací o nějakém objektu. Problémem je nedostání se širšího povědomí o AR směrem k veřejnosti, jelikož právě veřejnost se spoléhá na mnohem tradičnější zdroje informací, kterými jsou například informační centra nebo brožury (Olsson, 2011).

Druhým přístupem pro tuto skupinu je identifikace objektu na základě skenování markeru. Nejčastější je využití čtvercového markeru. Výhodou je vysoká jednoduchost naskenování a na základě designu i rozpoznání pro vyhledání potenciálních markerů. Pro výběr toho správného se využívá metoda rozhodovacího stromu nebo morfologická analýza na základě analýzy naskenovaného obrazu. Každý potenciální marker je pak transformován do záběru kamery a porovnán s předdefinovanými šablonami (Kato, 1999). Další možností pro rozeznání markeru může být i hybridní přístup, ve kterém se kombinuje identifikace pozice a orientace mobilního zařízení používající GPS a kompas za použití zjednodušené analýzy obrazu (Seo, 2006).

Pro zobrazení augmentací mohou tyto zařízení využívat jak připojení k internetu, odkud následně vykreslují digitální informace, tak lokální úložiště. Naprostým základem je ale kamera nebo pole senzorů, které zachycují data, která mají být augmentována. Data jsou následně analyzována na základě hlediska, jestli se jedná o obrazovou analýzu markeru nebo identifikaci GPS polohy, které bude výsledný augmentovaný objekt připsán. Po vyhodnocení jsou stažena data z lokálního nebo cloudového úložiště, kterým je přidán virtuální obsah. Vyrenderovaná data pak vytváří augmentovaný obraz, který se zobrazí na uživatelově zařízení.

3.2.2 Chytré brýle (nositelné)

Chytré brýle jsou definovány jako nositelné AR zařízení, které se používají jako klasické brýle. Rozdílem je propojení virtuální informace s fyzickou informací v uživatelově zorném poli (Rauschnabel, Brem & Ivens, 2015a, 2015b; Glauser, 2013). V tomto případě se pro zachycení fyzické informace a její augmentaci využívá opět kamera, GPS, ale také například mikrofon. Virtuální informace se získává buď pomocí internetu nebo může být uložena přímo v paměti brýlí. Augmentace pak probíhá primárně na základě polohy, objektu, obličejových a obrazových rozpoznávacích metod. Virtuální informace je poté zobrazena v reálném čase na obrazovce, kterou zde tvoří plastová destička před očima uživatele. Předními zástupci této kategorie jsou Microsoft Hololens a Google Glass (Rauschnabel, 2015).

3.2.3 AR Mirrors (Stacionární)

Podle Rochata a Zahaviho (2011) jsou AR Mirrors (ARM) zvláštní předměty spojené se zvláštními, tajemnými zážitky. Fyzicky jsou zrcadla velmi specifickým povrchem, typicky plochá a lesklá. Na rozdíl od těchto normálních zrcadel dokáží ARM virtuálně změnit odrážený obraz ve zrcadle na základě AR technologie (Portales, 2016). Základem je princip zrcadla, kdy uživatel vidí sám sebe s přidanou virtuální informací. Pro zachycení gest uživatele se používá například Microsoft Kinect, který je připevněn na vrchní straně ARM. Podobná zařízení se nejčastěji používají v marketingu nebo při výuce anatomie (Bork, 2019).

3.3 AR Software Development Kit

Software Development Kit (SDK) neboli framework je soubor nástrojů, které se využívají k vytvoření AR aplikace. SDK poskytují kódovací prostředí, ve kterém je uživatel schopen vytvořit, implementovat veškerou funkcionalitu, která bude kombinovat klíčové základní komponenty, které budou schopny vytvořit AR prostředí. Těmito komponentami jsou myšleny například a spousta dalších:

- Rozpoznávací komponenta – tvoří jádro AR aplikace
- Sledovací komponenta – je „očima“ zkušenosti AR
- Vykreslovací komponenta – zobrazuje virtuální objekty do reálného světa

V další části tak budou představeny konkrétní příklady SDK, které jsou využívány mobilními telefony, tablety nebo i chytrými brýlemi (Amin, Govilkar, 2015).

3.3.1 ARKit

Byl představen spolu s iOS 11 v roce 2017. Jedná se tedy o SDK pro mobilní zařízení od společnosti Apple (iPhone a iPad). Tento framework umožňuje jednoduchou tvorbu jedinečných AR aplikací. Kombinací virtuálních objektů s reálným prostředím bere ARKit aplikace mimo displeje, osvobozuje je od interakce se skutečným světem za použití zcela nových přístupů. ARkit funguje na zařízeních Apple s čipem A9, A10 atd. Využívá technologii Visual Inertial Odometry (VIO) pro přesné sledování okolního světa. Kdy VIO slučuje senzor z fotoaparátu s Core Motion daty. ARKit je poté schopen detekovat horizontální plochy, jako jsou například stoly a podlahy, a dokáže také sledovat a umisťovat objekty na menší charakteristické body (Lotfi, 2019).

Obrázek 7 Logo ARKit

Zdroj: http://zugara.com/arkit-developers-top-arkit-apps

ARKit využívá výhod VIO k efektivnímu prohlédnutí zahrnuté podmínky. VIO propojuje informace zachycenou senzorem fotoaparátu s informací Core Motion. Zkonsolidovaná data dovolují zařízení si přesně uvědomit jeho vývoj bez dalších úprav. Dále dovoluje zařízením zkoumat a pochopit scénu, která byla nasnímána kamerou a rozlišovat tak rovné povrchy. Ploché povrchy například stolů rozpoznané zařízením mohou být využity pro vytvoření zvětšené setkání s realitou, kde lze virtuální objekty umístit na původní povrch. Navíc s pomocí kamerových senzorů měří ARKit osvětlení ve scéně a využívá tato data k optimalizaci osvětlení virtuálního objektu. Efektivita ARKitu může být ještě přiživena využíváním SceneKitu nebo externím programováním v Unity nebo Unreal Engine (Lotfi, 2019).

3.3.2 ARCore

Na rozdíl od ARKitu je ARCore SDK pro všechna zařízení s operačním systémem Android 7.0 (Nougat) a novější (https://developers.google.com/ar/discover). Využívá procedury zvané simultánní odometrie a mapování pro vyhodnocení pozice zařízení vzhledem k jeho okolí. ARCore navenek rozpoznává hlavní místa zachycená na obraze z kamery a využívá těchto ohnisek pro úpravy okolí. Obrazová data jsou spojena s daty inerciálními z inerciální jednotky pro stanovení pozice a orientace kamery vzhledem k okolnímu světu v průběhu času. Vyrovnáváním pozice virtuální kamery, která renderuje 3D obsah z pozice kamery zařízení dané ARCorem umožňuje zejména designérům renderovat virtuální substance z nejlepšího úhlu pohledu. Vyrenderovaný virtuální obraz nebo 3D objekt může být překryt přes obraz pořízený fotoaparátem zařízení. Výsledek pak působí jako kdyby vytvořený virtuální obsah byl součástí reálného světa (Lotfi, 2019).

Obrázek 8 Logo ARCore

Zdroj: https://commons.wikimedia.org/wiki/File:Ar_core.svg

ARCore vždy obohacuje chápání reality pomocí rozpoznávání hlavních bodů a rovin. Vyhledává skupiny charakteristických identifikátorů, které leží na vodorovných nebo svislých površích, podobných deskám. Dále tyto povrchy přetváří a umožňuje jejich použití v aplikacích jako roviny. ARCore může rovněž rozhodnout limit roviny a zpřístupnit data aplikaci. Je mimořádně užitečné umístit virtuální předměty ležící na rovné povrchy. Problémem mohou být povrchy bez textury, například bílé zdi, které nemusí být detekovány kvůli nedostatku hlavních bodů (Lotfi, 2019). Informace o světle prostředí může být efektivně identifikována na základě zadané průměrné intenzity a barevné korekce pro vybraný obraz kamery zařízení.

Jednou z mnoha součástí ARCore jsou takzvané orientační body. Ty umožňují uživateli umisťovat virtuální objekty na zahnuté povrchy. Fungují na principu stanovení hlavního bodu uživatelem po kliknutí na displej. ARCore poté prozkoumá sousední hlavní body a využije je k výpočtu úhlu povrchu v místě uživatelova hlavního bodu. ARCore je schopný vylepšovat chápání prostředí tak dlouho, jak jen se pozice zařízení mění v průběhu času, ve kterém uživatel pracuje s virtuálními objekty ve scéně. Na základě těchto výpočtů ARCore stanoví kotevní bod pro zajištění co nejlepšího sledování pozice objektů (Lotfi. 2019).

3.3.3 Vuforia

Toto SDK patří mezi nejpopulárnější platformy s cílem umožnit uživatelům pracovat na vývoji AR na zařízeních s operačním systémem Windows. Rozpoznává rovinný obraz, stejně tak i ploché, hranaté nebo cylindrické objekty, dále text a okolní prostředí. Konkrétně Vuforia 7 představuje rozpoznávání objektů nové generace a podporu se zařízeními s ARKit a ARCore. Použitím objektového skeneru Vuforia mohou uživatelé skenovat a vytvářet cílové objekty. Proces rozpoznávání může být implementován v databázi. Integrován může být i Unity plugin. Vuforia tak umožňuje uživatelům rozpoznat a sledovat reálný objekt s použitím digitálního 3D modelu téhož objektu. Tento proces je podporován tvarem objektu, který musí být geometricky neměnný a jeho povrch musí být nelesklý (Lotfi, 2019).

Stejně jako hlavní body u augmentovaného obrazu pomocí ARCore SDK má Vuforia svůj vlastní systém zvaný Images Targets. Ten reprezentuje obrazy, které je Vuforia schopná detekovat a sledovat. Na rozdíl od tradičních markerů a QR kódů, tento systém nepotřebuje speciální černobílé kódy k rozpoznání objektu. Rozpoznávání tak funguje podobně jako u ARCore, kdy dochází k detekci hlavních bodů, které jsou následně porovnávány s body uloženými v databázi. Tyto body jsou pak jednoduše exportovatelné v databázi spolu s aplikací. Problémem jsou zde světelné podmínky, kdy zmíněné hlavní body by měly být snímány na rovnoměrně osvětlené ploše.

Obrázek 9 Logo Vuforia

Zdroj: https://innovation-rocks.com/de/services/innovation-rocks-vuforia-logo-2-2/

3.4 Prezentace historických a kulturních památek pomocí AR

Jak z názvu práce vyplývá, tématem je vytvoření interaktivní exhibice, například muzea nebo kulturní památky. Tato podkapitola popisuje dva používané přístupy pro vytvoření mobilní AR aplikace. Prvním typem je bezmarkerový přístup. Ten byl použit například v prezentaci života poeta Ovidia. Aplikace byla vytvořena k výročí 2000 let od smrti Ovidia a obsahuje relevantní historické informace o jeho životě. Druhým typem je markerový přístup, použitý například ve vytvoření průvodce po muzeu v podobě mobilní AR aplikace.

3.4.1 Bezmarkerový přístup

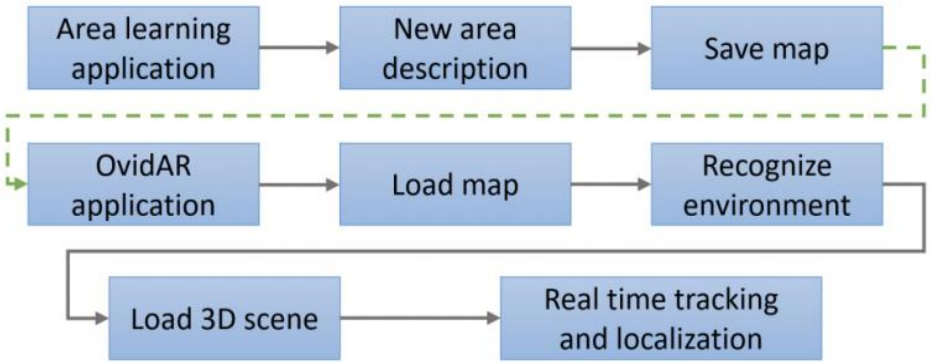
Prvním příkladem prezentace historických a kulturních památek pomocí AR je bezmarkerový přístup. Klíčovými aspekty využití AR pro prezentaci je kombinace reálných a virtuálních objektů v reálném prostředí. Tento přístup je interaktivní, funguje v reálném čase a v neposlední řadě registruje a srovnává reálné a virtuální objekty mezi sebou (Chatzopoulos, 2017). Prvním krokem je zvážení dvou klíčových problémů při vývoji takové aplikace. Prvním je přesnost sledování při použití uživatelem, na který přímo navazuje schopnost registrace 3D modelů s vlastnostmi reálného světa (Bostanci, 2015). Velmi důležitá je stanovení využití mobilní AR aplikace. V tomto případě se jedná o využití bezmarkerového přístupu. Pro zobrazení virtuálního obrazu

využívá GPS v mobilním zařízení nebo systémů pro rozpoznání obrazu pro identifikaci lokace (Mota, 2017).

Využití AR právě v tomto případě požaduje stažení informací z okolního prostředí. Pro získání správných informací a korekci virtuálního objektu s reálným se používá proces zvaný registrace. Existují dvě metody registrace:

- Bez prvotní znalosti – realizována on-line, bez jakékoliv znalosti scény
- S prvotní znalostí – předdefinované vzory jsou vzaty z prostředí s cílem vyhledání vhodného objektu (Wagner, 2009)

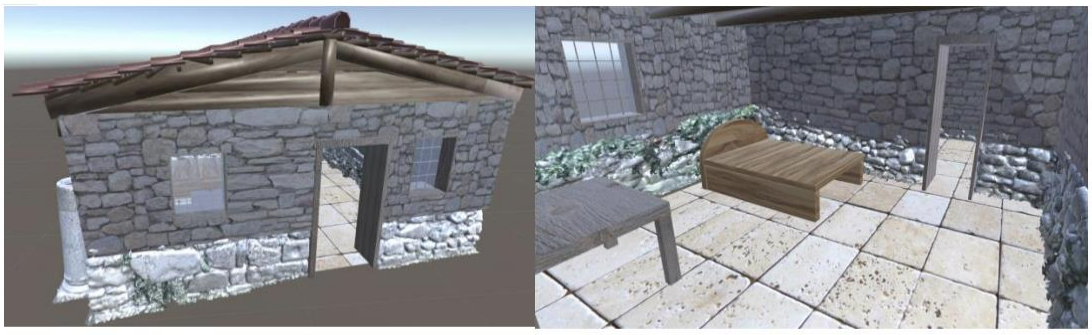
Aplikace vyvinutá pro tento projekt se jmenuje OvidAR využívající sledování pohybu a učení oblasti. Požadavkem pro vytvoření 3D scény byla mapa. Aplikace OvidAR vytvořila 3D scénu za předpokladu, že již dříve došlo k uložení mapy a pokud aplikace detekovala klíčové vizuální vlastnosti, které byly uloženy v souboru popisujícím oblast. Sledování polohy zařízení v reálném čase a lokalizace zajišťuje uživatelovo ponoření do zobrazované scény (viz Obrázek 11). Různorodé animace virtuálního světa skrze OvidAR vytvořily pocit interakce. Další klíčovou částí, která obohacuje uživatelovu zkušenost je zvuk. Speciálně v místech, kde dochází k ozvěně, která rozptyluje zvuk na základě polohy mobilního zařízení.



Obrázek 10 Schéma logického postupu augmentaci v aplikaci Ovid3D

Zdroj: https://www.mdpi.com/2071-1050/11/4/1167

První částí bylo vytvoření 3D objektů, které sloužily jako obsah pro mobilní aplikaci (viz Obrázek 11). Z toho důvodu byly použity historické podklady pro rekonstrukci archeologických artefaktů objevených v oblasti Dobrogea, historického regionu v jihovýchodní části Rumunska. Pomocí fotogrammetrie byly zjištěny základy budovy. 3D model byl vytvořen v SW Autodesk 3D Studio Max.

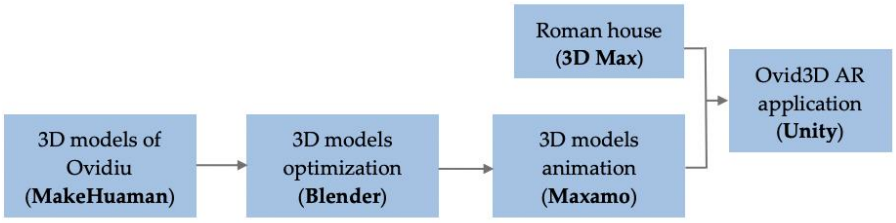


(a) (b)

Obrázek 11 Zobrazení AR v aplikaci Ovid3D

Zdroj: https://www.mdpi.com/2071-1050/11/4/1167

Vytvořen byl i virtuální charakter Ovidia spolu s animacemi a zdrojem audio složka. Pokud tak došlo k setkání uživatele s virtuálním charakterem, zařízení oba zobrazilo na displeji zařízení. Charakter Ovidia byl vytvořen ve 3 fázích jeho života (dítě, dospělý a stařec) pomocí SW MakeHuman a Blenderu. Animace byly vytvořeny v SW Adobe Mixamo a výsledná aplikace vznikla v SW Unity (viz Obrázek 12).



Obrázek 12 schéma postupu práce při tvorbě aplikace Ovid3D
Zdroj: https://www.mdpi.com/2071-1050/11/4/1167

3.4.2 Markerový přístup

Ukázkovou demonstrací tohoto přístupu jsou mobilní průvodci v muzeu, kteří jsou velmi často zahrnuty variabilními funkcemi s cílem umožnit a co nejvíce obohatit využití expozice muzea jak pro návštěvníky, tak pro zaměstnance a kurátory muzea. V případě AR průvodce je jasné, že kromě nových funkcí je potřeba zahrnout i funkce stávající do výsledné aplikace. Z toho důvodu byl vytvořen jakýsi inventář, do kterého byly přidány AR funkce, které byly podrobně diskutovány s profesionály z muzea. Důležité je rozdělení funkcí na viditelné a neviditelné pro návštěvníka muzea spolu s použitím existujících AR funkcí. Nicméně z důvodu neexistující jakékoliv klasifikace AR funkcí došlo k vytvoření klasifikačního procesu:

- Kontextualizace
  - Termín původně využívaný v biblických studiích, ale v 70. letech 20.st trvale přejat kulturními studii a archeologií. Byl vybrán z důvodu vyjádření všech funkcí, které pomáhají návštěvníkovi pochopit objekt v originálním kontextu. Do této kategorie tak spadá vizualizace obrazů, 3D modelů nebo například animace a zvuk.
- Komunikace
  - Komunikační funkce mohou asistovat různé komunikační kanály. Mezi muzeem a návštěvníkem nebo naopak. Mezi více uživateli a ve výsledku tak adresovat části komunikace s výhledem na pozdější konzultace. Komunikační proces tak spojuje návštěvníkovu zkušenost z doby před, při a po návštěvě muzea, a zároveň tak zhutňuje pouto mezi veřejností a muzeem. Dalším příkladem této funkce může být zaslání notifikace o otevíracích hodinách nebo nějaká speciální událost v muzeu.
- Personalizace
  - Personalizace je další výhodou mobilního AR průvodce v muzeu a je možné říct, že je sama svojí funkcí skládající se z dílčích funkcí. Její význam tak lze brát doslova spolu s konfigurací, bez načrtnutí pomyslné hranice mezi kustomizací a adaptabilitou. To vše záleží na individuálním uživateli, který si tak sám může nastavit vlastní preference aplikace (Bowen, 2004).

- Muzejní data management
    - Poslední kategorie je naprosto neviditelná pro návštěvníky, jelikož hraje největší roli v obohacení návštěvníkovy zkušenosti. Pointou souboru těchto funkcí je jejich práce s daty, ať už jsou poskytnuta okolním prostředím nebo prostřednictvím serveru. Principem je logický řetězec uložení, přenosu a zpracování dat.

Na začátku prohlídky návštěvníci dostali mobilního průvodce a spolu se svými „spolunávštěvníky“ se zaregistrují. Tento krok později umožňuje komunikaci mezi jednotlivými účastníky a sdílet tak pozici v prostorách exhibice. Je požadováno, aby se ukládala trasa návštěvníka napříč exhibicí pro vytvoření unikátního a kompletně osobního „suvenýru“. V tento moment dostává návštěvník informace o tom, jak používat mobilní zařízení.

Poté se návštěvník dostává do skutečné prohlídky výstavy, kdy jednoduše namíří mobilní zařízení směrem k markeru obrazu nebo jiného exponátu. Marker je zabrán kamerou zařízení a zpracován příslušným SDK. Po zpracování je virtuální informace (obraz, nabídka, widget, text, video,…) správně přiřazena na zkoumaný objekt (viz Obrázek 13). Uživatel si pak sám vybere, na kterou informaci klikne. Během prohlídky může návštěvník komunikovat s ostatními, nechávat prostorové komentáře, které si ostatní mohou zobrazit. Samo muzeum může během prohlídky návštěvníkům rozesílat notifikace, ať už s cílem zpříjemnění prohlídky, tak za účelem zatraktivnění. Personalizace pomáhá adaptaci obsahu prohlídky exhibice na základě návštěvníkova profilu.

Obrázek 13 Zobrazení kontextových informací v prostředí AR

Zdroj: https://virtualrealitypop.com/ar-in-museums-890b0a48e7a5

Závěrem alespoň v jednom projektu z mobilního průvodce byly poskytnuty nasbíraná data návštěvníkovi jako „suvenýr“ (Sauer et al., 2004). jakmile se návštěvníci registrovali, veškerá data z jejich návštěvy byla sesbírána za účelem předesignování a optimalizace aplikace. Po návratu mobilního zařízení dostal návštěvník pohlednici s vytištěným URL. Poté má návštěvník možnost zobrazit svá data na osobním počítači a vizualizovat jím navštívené objekty spolu s trasou, kterou v muzeu prošel. Tímto způsobem je tak lze expozici přenést za hranice muzea a umožnit tak návštěvníkovi zobrazit a blíže prozkoumat objekty, které navštívil v exhibici (Damala, 2007).

3.5 Software pro tvorbu AR aplikace

3.5.1 Blender

Blender je bezplatná a otevřená sada pro tvorbu 3D. Podporuje celý proces tvorby 3D od modelování, přes animace, simulace, renderování až po motion tracking. Pokročilí uživatelé využívají také Blender Application Programming Interface (API) pro skriptování v Pythonu k přizpůsobení aplikace a psaní specializovaných nástrojů. Jelikož se jedná o otevřenou sadu, tyto nástroje bývají zahrnuty v budoucích verzích Blenderu. Blender je vhodný jak pro jednotlivce, tak pro malá studia, která těží z jeho sjednoceného workflow, široké uživatelské a vývojářské komunity a responzivního vývojového procesu (Blender.org, 2021).

Blender je multiplatformní a běží souběžně na platformách Linux, Windows a Macintosh. Rozhraní využívá OpenGL k zajištění konzistentního zážitku. Jako komunitně řízený projekt v rámcu GNU General Public License (GPL) je veřejnost oprávněna provádět malé i velké změny v kódu, což obecně vede k rozšíření funkcionality, opravám chyb a lepší použitelnosti. Pro AR tvorbu konkrétně lze využít funkcionalitu Blenderu napříč všemi spektry, kdy právě modelování hraje největší roli.

3.5.2 Unreal engine

Tento herní engine obsahuje kompletní sadu vývojových nástrojů pro každého, kdo pracuje s technologií v reálném čase. Od designových vizualizací a filmových zážitků až po tvorbu kvalitních počítačových, konzolových a mobilních her, a v neposlední řadě také tvorba VR a AR Unreal Engine poskytuje vše, co je potřeba pro tvorbu těchto aplikací (Unrealengine.com, 2021).

Pro vytvoření pohlcujících zážitků, které jsou lehce pochopitelné pro lidskou mysl (AR, VR, MR) jsou vyžadovány složité scény. Ty jsou následně renderovány ve výjimečné kvalitě při vysokých snímkovacích frekvencích. Unreal Engine je také navržen pro náročné aplikace, jako je tvorba filmů nebo AAA her a fotorealistické vizualizace.

3.5.3 Unity3D

Jedná se o nejpopulárnější herní engine na světě. Nabízí spoustu funkcí a je dostatečně flexibilní, aby zvládl téměř jakoukoliv hru. S bezkonkurenčními funkcemi pro různé platformy je Unity oblíbený jak u hobby vývojářů, tak i u velkých herních studií. Mezi nejznámější hry vyvinuté pomocí tohoto enginu patří Pokemon Go, Hearthstone nebo třeba Cuphead a spousta dalších. V názvu stojící 3D neznamená orientaci jenom na 3D, Unity také obsahuje nástroje pro vývoj 2D her.

Je oblíbený hlavně mezi programátory kvůli skriptovacímu API C# a integrované sady Visual Studio. Unity také nabízí JavaScript jako skriptovací jazyk a MonoDevelop jako IDE těm, kteří chtějí alternativu k Visual Studio. Oblíbený je i umělci, kterým je zde umožněno vytváření vlastních 2D a 3D scén pomocí výkonných animačních nástrojů.

Vzhledem k tomu, že Unity existuje od roku 2005, došlo k vytvoření obří uživatelské komunity a uživatelských knihoven s různými zdroji. Unity má nejen širokou dokumentaci, ale i obrovské množství videí a tutoriálů online (Unity.com, 2021).

3.5.4 Adobe Creative Suite (CS)

Adobe CS je ukončená softwarová sada aplikací pro grafický design, úpravu videa a vývoj webových aplikací. Každá edice se skládala z několika aplikací Adobe, jako jsou například Photoshop, Acrobat, Premiere Pro, After Effects nebo Illustrator. Tyto aplikace se staly standartními aplikacemi pro mnoho pracovních pozic od grafického designu, přes speciální filmové efekty až po úpravu zvuku (Adobe.com, 2021). Neslouží sice přímo ke tvorbě AR aplikací, nicméně je v nich tvořen grafický design, animace nebo audio právě pro jejich tvorbu.

Obrázek 14 Produkty sady Adobe Creative Suite
Zdroj: https://www.wikiwand.com/en/Adobe_Creative_Suite

3.5.5 C#

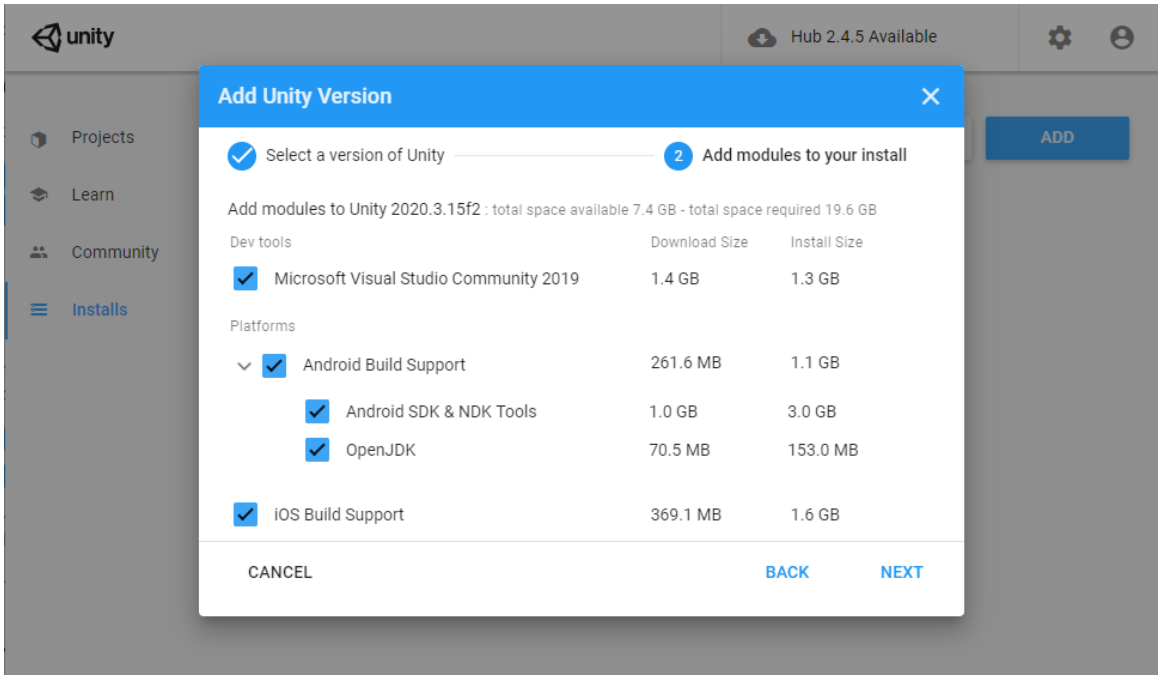
C Sharp je univerzální multi-paradigmatický programovací jazyk zahrnující statické psaní, silné psaní, objektově orientované a komponentně orientované programovací disciplíny. Tento programovací jazyk lze použít k vytváření interaktivních webů, mobilních aplikací, videoher, AR, VR, desktopových aplikací a služeb typu back-end. Například mobilní hra Pokémon Go a web Stack Overflow byly vytvořeny ve frameworcích fungujících na C# (Codeacademy.com, 2021).

4 MOBILNÍ APLIKACE

Stěžejní částí diplomové práce bylo vytvoření mobilní aplikace založené na zobrazení AR. Prvním krokem byla úprava a texturování digitálních modelů Václavského návrší z různých časových období. Modelů bylo konkrétně pět, a to modely zachycující podobu návrší v 12., 13., 17., 19. a 20. století. Modely byly poskytnuty Vlastivědným muzeem v Olomouci, které vytvořili studenti katedry geoinformatiky Univerzity Palackého v Olomouci. Dále následovalo vytvoření projektu v Unity, stažení jednotlivých knihoven potřebných pro vytvoření aplikace a zahrnujících funkce umožňujících zobrazení AR prvků. Tyto knihovny a funkce jsou popsány v následujících podkapitolách. V průběhu práce bylo vyvíjeno User Interface (UI) skládající se z velké části z tlačítek s naprogramovanými funkcemi v jazyku C#. Výsledná aplikace zahrnuje dvě části pro reprezentaci změn Václavského návrší v průběhu zmíněných časových období a byla testována na zařízeních Android a iOS. Výsledná aplikace je pak dostupná na Android zařízení.

4.1 Základní nastavení projektu v Unity

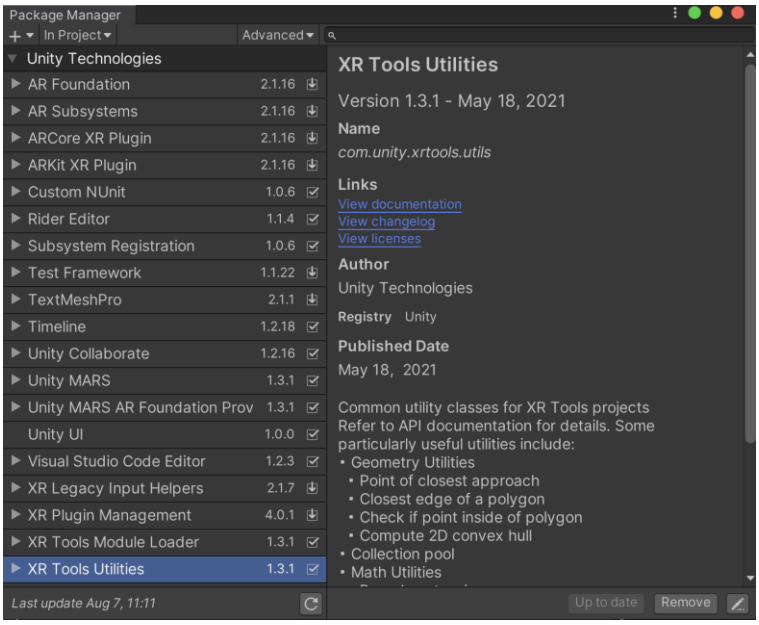
Pro vytvoření mobilní aplikace v prostředí Unity je nutné nainstalovat si, pokud možno nainstalovat nejnovější verzi programu. Dále je důležité do ní nainstalovat balíčky **Microsoft Visual Studio Community**, **Android Build Support** a **iOS Build Support**, bez kterých by nebylo možné aplikaci exportovat do mobilních zařízení (viz Obrázek 15).



Obrázek 15 Základní nastavení unity projektu a staženích potřebných balíčků

Poté už je jen třeba založit Unity projekt, ve kterém se pak víceméně odehrávalo nastavení celé aplikace. Zde bylo kriticky důležité nainstalovat jednotlivé balíčky, které podporují tvorbu AR za pomoci *Package Manager* v Unity. První z nich byl **ARFoundation**, jež umožňuje vytvoření AR a spolupráci s AR subsystémy jak v Unity, tak v mobilní aplikaci. Simultánně s tímto balíčkem byl nainstalován také **AR Subsystems**. Jelikož byla aplikace původně vytvářena pro Android a iOS, bylo nutné nainstalovat **ARCore XR Plugin** (podpora integrace Google ARCore pro využití multiplatformního XR API) a **ARKit XR Plugin** (podpora integrace Apple ARKit pro

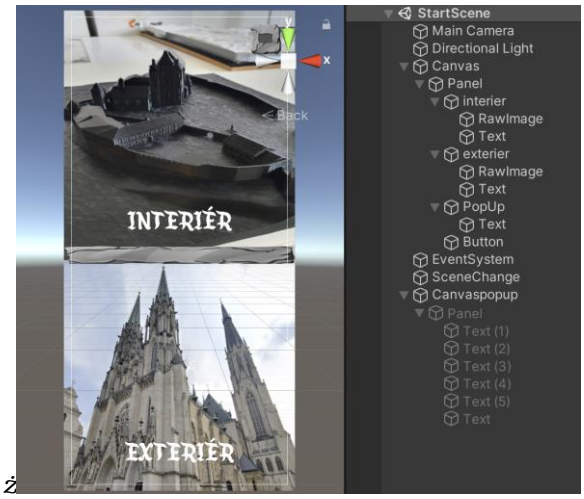
využití multiplatformního XR API). Tím, že tyto pluginy jsou stavěné pro XR, byly simultánně nainstalovány další balíčky, konkrétně **XR Legacy Input Helpers**, **XR Plugin Management**, **XR Tools Module Loader** a **XR Tools Utilities**. V průběhu práce byl přidán ještě balíček **Unity MARS**. Zbytek balíčků byl v projektu nainstalovaný nativně (viz Obrázek 16).



Obrázek 16 Přehled knihoven potřebných pro vypracování diplomové práce

4.2 Úvodní scéna

Jelikož je aplikace rozdělena do dvou částí, byla vytvořena úvodní scéna, která mezi nimi slouží jako rozcestník. Byla rozdělena na dvě poloviny. Objekt *Canvas* byl vytvořen na základě rozměrů zařízení, na kterém byla aplikace testována. Jednalo se tedy o jakousi referenci, podle které byly do scény postupně přidány tlačítka s naprogramovanými funkcemi, obrázky a textová pole. Doprostřed bylo vloženo tlačítko bez funkcionality, které slouží pouze jako grafický rozdělovač mezi dvěma hlavními tlačítky. Objekty *Main Camera* a *Directional Light* jsou do scén vždy přidány defaultně. Konkrétně *Main Camera* je v celé práci umístěna pouze ve scénách, které nevyužívají AR (viz Obrázek 17). Pokud by jakákoliv kamera ve scéně absentovala (*Main Camera* nebo *AR Camera*), nevyskytl by se ve finální aplikaci žádný obraz.



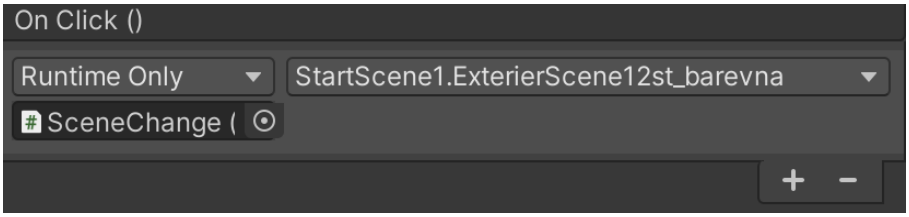
Obrázek 17 Náhled na úvodní scénu v prostředí Unity

První tlačítka s nápisem **INTERIÉR** a **EXTERIÉR** sloužila jako rozcestník mezi zmíněnými částmi aplikace. Tyto nápisy reprezentuje objekt *Text* (font Echanted Land) a na jejich pozadí byly vloženy obrázky, které ilustrují místa, kde a jak je nejlepší aplikaci použít. Sloužila tedy jako přepínač mezi scénami, jehož chod byl zajištěn C# skriptem napsaným v SW Visual Studio (viz Obrázek 18).

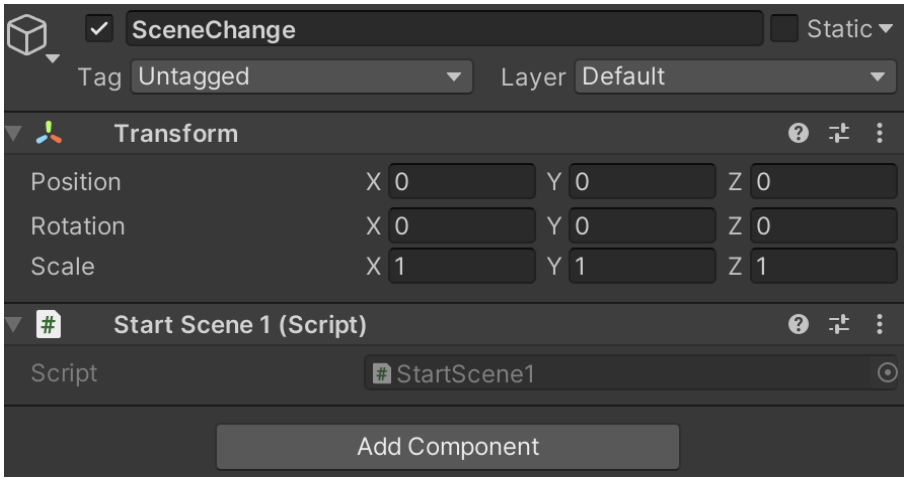
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class StartScene1 : MonoBehaviour
{
    public void InteriorScene()
    {
        SceneManager.LoadScene("InteriorScene");
    }
    public void ExteriorScene12st_barevna()
    {
        SceneManager.LoadScene("ExteriorScene12st_barevna");

    }
```

Obrázek 18 Skript zajišťující funkcionalitu pro přepínání scén

První čtyři řádky skriptu byly defaultně umístěny do skriptu, jelikož byl vytvořen v Unity a jednalo se o načtení základních knihoven zajišťujících funkčnost skriptů vytvářených pro aplikace v Unity. Řádek *public class* definuje název skriptu, základní třída *MonoBehaviour* byla použita, protože se používá jazyk C# a explicitně se z této třídy vychází. Každý další řádek pak udává informaci o názvu scény, na kterou se má odkazovat. Tento skript pak definuje, na jakou scénu se po stisknutí tlačítka má aplikace přepnout (*InteriorScene* a *ExteriorScene12st_barevna*). Tato vlastnost byla každému tlačítku přiřazena v nastavení daného tlačítka (viz Obrázek 19). Byla zvolena možnost *On Click ()*, která udává tlačítku vlastnost, že definovaná operace bude provedena po kliknutí na tlačítko. Nicméně pro úplnou funkčnost tlačítka musel být do scény přidán prázdný *GameObject* pod názvem *SceneChange*, do kterého byl taktéž vložen skript uvedený výše (viz Obrázek 20).



Obrázek 19 Nastavení funkcionality tlačítka pro změnu scény



Obrázek 20 nastavení placeholder objektu SceneChange

Dalším tlačítkem fungujícím na dvou separátních skriptech je tlačítko *PopUp*. Pro jeho běh bylo zapotřebí vytvořit objekt *Canvaspopup* a jemu přidružit objekt Panel, do kterého byly nakonec vloženy texty, které podávají základní informace a instrukce o aplikaci. Oba skripty pak zajistili funkcionalitu korespondující s názvem tlačítka, kdy po jeho stisknutí vyskočí okno se zmíněnými informacemi (viz Obrázek 21).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PanelOpener : MonoBehaviour
{
    public GameObject Panel;

    public void OpenPanel()
    {
     if (Panel != null)
        {
            bool isActive = Panel.activeSelf;

            Panel.SetActive(!isActive);
        }

    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Popup : MonoBehaviour {
    public Canvas canvas;
    public bool a = false;
    public void popup()
{
    if (a == false)
    {
        a = true;
        canvas.enabled = true;
    }else if (a == true)
    {
        a = false;
        canvas.enabled = false;
    }
}
}
```

Obrázek 21 Skripty zajišťující funkcionalitu vyskakovacímu panelu

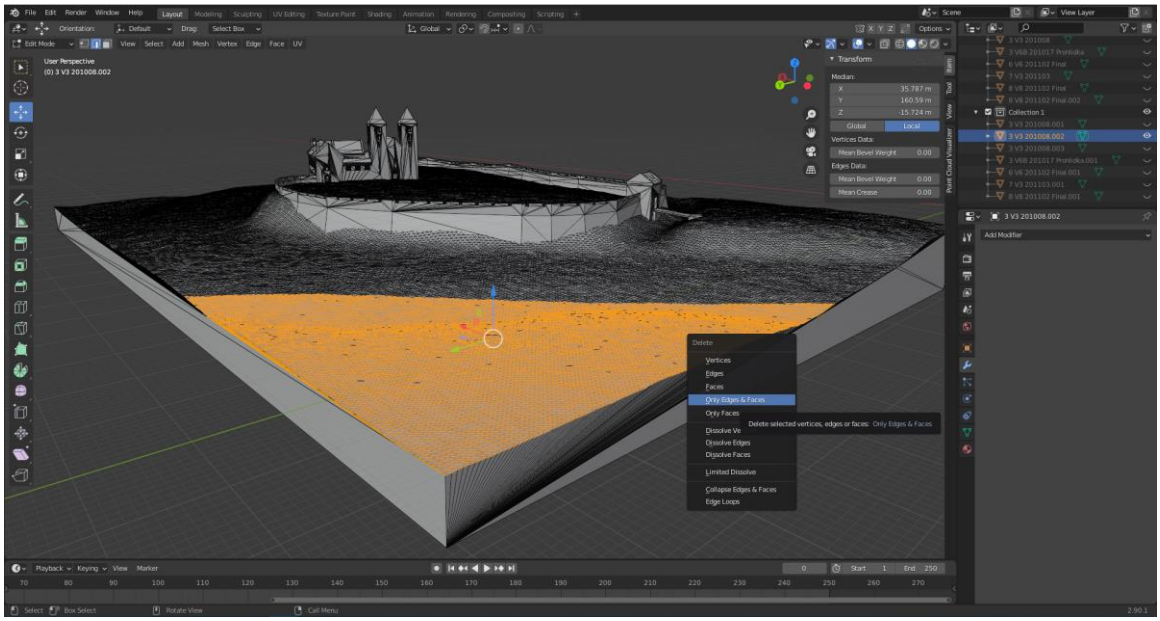
Skript vlevo definuje otevírání objektu *Canvaspopup*, konkrétně jemu přidruženému *Panelu*. Panel jako takový nemůže fungovat sám o sobě, proto se v řešení vždy vyskytuje jako přidružený objekt. Úvod syntaxe je prakticky stejný jako v předchozím skriptu. Řádek *public GameObject Panel* říká, že skript bude operovat právě s daným objektem. Další část pak definuje samotné otevírání *Panelu* po stisku tlačítka. Skript vpravo pak definuje stavy, kterých *Panel* po stisknutí tlačítka nabývá, jinými slovy, jestli je aktivní nebo ne. Toto okno lze jednoduše zavřít jeho opětovným stisknutím (viz Obrázek 22). Design tlačítek byl zajištěn stažením volně dostupných Unity Assetů (Stone UI).

Obrázek 22 Výsledná podoba vyskakovacího panelu

4.3 Práce s modely

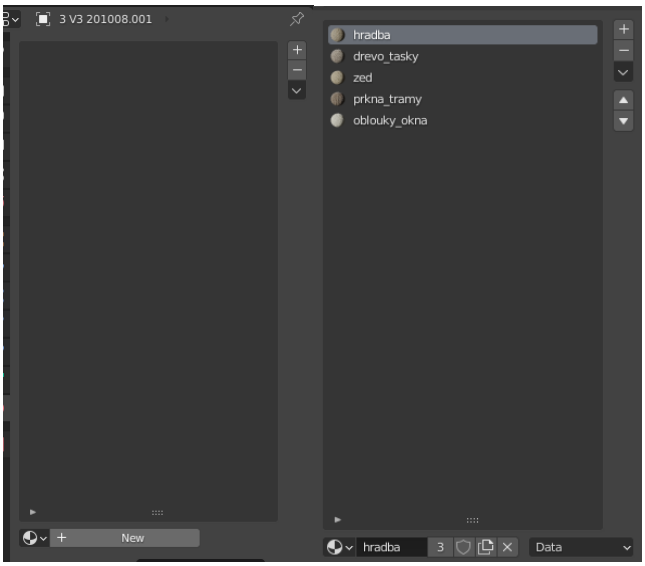
Jak již bylo zmíněno, digitální modely reprezentují Václavské návrší v Olomouci. Vytvořeny byly v SW SketchUp. Pro další práci byly modely ve formátu .stl naimportovány do SW Blender verze 2.90.1. Tento SW byl upřednostněn z důvodu větších možností, co se týká texturování, modelování a následného workflow napojení na Unity.

Nejprve došlo k importu modelů do Blenderu přes nabídku *File -> Import-> Stl*. Tyto modely byly původně určené k 3D tisku, obsahují kromě kostela také terén s podstavou (viz Obrázek 23). Pro interiérové část aplikace se autor po konzultaci s vedoucím práce shodl na prvních úpravách, kdy byla odstraněna podstava a terén ořezán podél obvodu hradeb. U exteriérové části byl terén s podstavou odstraněný kompletně. K tomu došlo jednoduchou operací, kdy byl Blender přepnut do editačního módu v záložce *Layout*. Po vybrání plošek modelu došlo k jejich odstranění vybráním možnosti *Only Edges & Faces* (viz Obrázek 23), která odstraní jak plošky, tak i hrany a vertexy.



Obrázek 23 První úpravy modelu v prostředí Blender – ořezání digitálního modelu

Dalším krokem v úpravě modelů bylo jejich texturování. V Blenderu jej umožňuje záložka *Shading*. Funguje na stejné bázi jako záložka *Layout*, kde je nutné přepnout do editačního módu, kde jsou poté vybraným ploškám přiřazeny vytvořené materiály (textury). Materiály jsou tvořeny v katalogové nabídce kliknutím na tlačítko *New* (viz Obrázek 24). Vytvořený materiál je nejdříve prázdný, proto je potřeba jej nastavit, kde mu budou přiřazeny vlastnosti, co se týče rozlišení, barvy, orientace apod. (viz Obrázek 24).

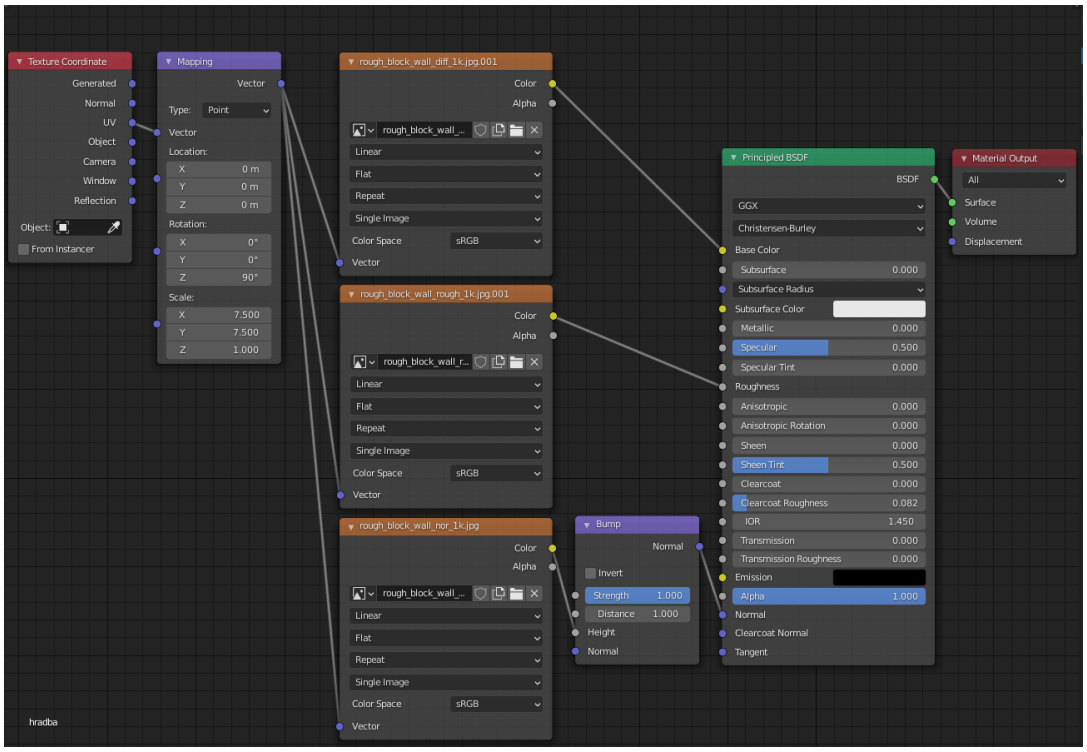


Obrázek 24 Vytvoření materiálů v prostředí Blenderu

Po vytvoření materiálu se otevře nové podokno, ve kterém lze materiál nastavit. Nicméně defaultně se otevřou pouze modifikátory s názvy *Principled BSDF* a *Material Output*, jejichž hodnoty není potřeba měnit. Tyto boxy v defaultním nastavení podobu materiálu vůbec nemění, proto bylo nutné přidat další modifikátory. Prvním modifikátorem byl *Image texture* (oranžová barva), který byl pro nastavení materiálu použitý třikrát. Do tohoto modfikátoru byly posléze nahrány obrázkové soubory formátu .jpg, z nichž každý má jiné vlastnosti (volně dostupné na webové stránce (https://ambientcg.com/) (viz Obrázek 25).

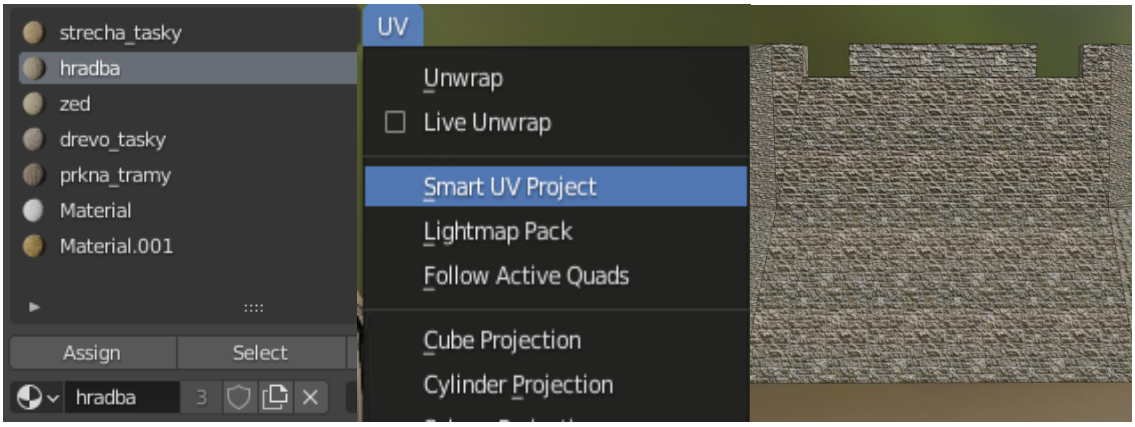
Obrázek 25 Soubory potřebné k vytvoření textury

Do prvního modifikátoru vstupuje obrázek s příponou *_diff (diffuse)* vyjadřující základní hodnotu pixelu textury a byl použit jako základní barva textury. Dalším byl obrázek s příponou *_rough (roughness)*, který ovlivňuje drsnost povrchu textury. Posledním obrázkem vstupujícím do třetího modifikátoru *Image texture* byl obrázek s příponou *_nor (normal)* označující normálu textury. Tento modifikátor byl spojen s dalším přidaným modifikátorem *Bump*, který udává vystouplost dané textury, kterou byla zvolena právě normála. Tímto byly nastavené grafické vlastnosti materiálů. Nicméně pro jejich rozlišení bylo zapotřebí přidat modifikátory *Mapping* a *Texture Coordinate*. *Texture Coordinate* ovlivňuje nastavení souřadnic textury a konkrétně zvolený uzel *UV* umožňuje chytře otexturovat plošky modelu na základě jejich orientace a rozměrů. Modifikátor *Mapping* na druhou stranu určuje stránku rozlišení materiálu. Nastavuje se zde orientace textury a velikost textury (viz Obrázek 26).



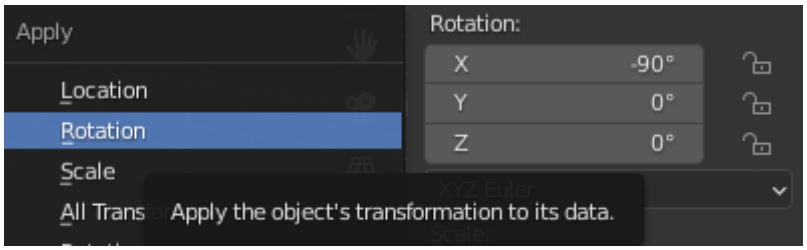
Obrázek 26 Podrobné nastavení parametru textury

Tím bylo nastavení materiálu kompletní. Dalším krokem bylo texturování samotných modelů. Jak již bylo zmíněno, je nutno mít zapnutý editační režim v záložce *Shading*. Následně se vybírají dané plošky, kterým je následně přiřazen materiál pomocí tlačítka *Assign*. Materiál je tedy přiřazen, nicméně je potřeba jít do nabídky *UV -> Smart UV Project*, což zapříčiní zobrazení materiálu na modelu (viz Obrázek 27).



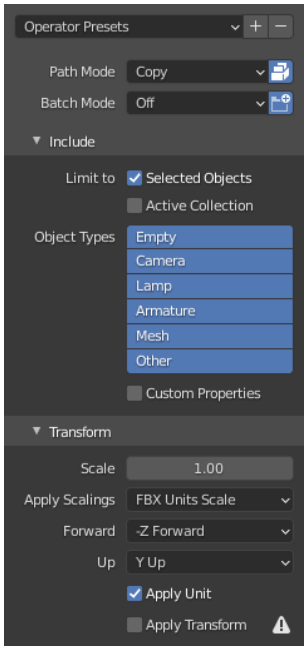
Obrázek 27 Postup práce při texturování modelu

Po texturování (přiřazení materiálů) modelů bylo nutné je vyepoxportovat, jelikož Unity nepodporuje formát .stl. První nařadě byl formát .gltf, který v sobě nativně uchovává informace o přiřazených materiálech. Problémem bylo, že tento formát podporují jen a pouze aplikace vytvářené pro desktop, tím pádem byl pro mobilní aplikaci nepoužitelný. Zvolen tak byl formát .fbx. Před samotným exportem do tohoto formátu je ale nezbytné změnit rotaci modelů, jelikož souřadnicový systém Blenderu a Unity je odlišný. Bylo tedy potřeba otočit každý model o -90° okolo osy x a potvrdit změnu (viz Obrázek 28). Dále pak otočit model o 90° okolo osy x, ale změnu už nepotvrzovat. Poté přišla řada na zmiňovaný export modelu do .fbx. Modely použité v exteriérové části pak musely být otočeny o 180° okolo osy z, aby korespondovaly s umístěním budovy v reálném světě.



Obrázek 28 Potvrzení transformace modelu

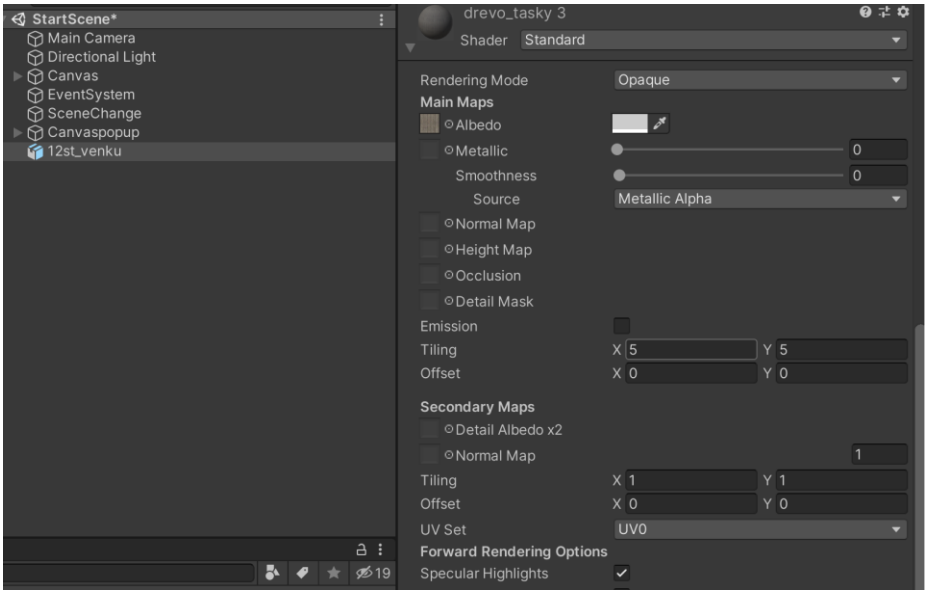
Při nastavení exportu modelů bylo důležité vyvarovat se několika věcem. V kolonce *Path Mode* je důležité vybrato možnost *Copy* a následně kliknout na tlačítko vpravo od této kolonky. Dále je důležité zaškrtnout možnost *Limit to Selected Objects*, pokud by tak nebylo učiněno, vyexportovala by se celá kolekce modelů do jednoho, což bylo pro finální řešení nepřípustné. Poslední dvě věci, na které bylo potřeba dát si pozor je přepnutí *Apply Scalings* na *FBX Units Scale,* pro zachování jednotek z Blenderu a zaškrtnutí možnosti *Apply Unit* (viz Obrázek 29). Výsledné modely byly vyexportovány do složky *Prefabs* v již založeném Unity projektu.



Obrázek 29 Nastavení exportních parametrů pro modely

Tímto krokem skončila práce s modely v prostředí Blenderu a práce v něm celkově. Problémem bylo, že modelům je v Unity potřeba vytvořené materiály (textury) přiřadit znovu. Je patrné, že materiály byly modelu přiřazeny, nicméně ve scéně a výsledné aplikaci se model zobrazuje bez textur a nedá se s nimi nijak operovat. Bylo tedy zapotřebí extrahovat textury do složky *textures*, která je podsložkou složky *Prefabs*. Poté co se textury extrahovaly, bylo nutné extrahovat ještě materiály. Úplně stejným postupem, akorát do složky *Materials*, která byla taktéž podsložkou složky *Prefabs*. Po tomto kroku už měl model přiřazené textury i v Unity a dalo se s nimi dále pracovat.

Velkým nedostatkem ovšem je, že mobilní aplikace vytvářené právě v Unity dokáží pracovat pouze s formátem .fbx. Ten sice přenáší informace o materiálu, ovšem tato informace nebyla kompletní. Přenesla se pouze základní textura, parametry nastavené v Blenderu se nepřenesly (rotace, velikost dlaždice). Proto bylo nutné nastavit zmíněné parametry znovu. Pro jejich nastavení bylo nutné model umístit do scény. Zde pak byl nastavován každý materiál zvlášť. Zde ale utor narazil na další nedostatek, kdy úprava materiálu v Unity je prakticky limitovaná pouze na nastavení velikosti dlaždic jednotlivých textur, nicméně i tak byly použity (viz Obrázek 30).



Obrázek 30 Nastavení materiálu v prostředí Unity

Modely pro interiérovou a exteriérovou část aplikace měly rozdílné vlastnosti, a to zejména ve velikosti zobrazovaného modelu (viz Tabulka 1).

Tabulka 1 Parametry jednotlivých modelů

Název modelu	Část využití v aplikaci	Scale	Formát/Typ modelu
12st_bila	EXTERIÉR	5.5	Prefab
12st_marker	MARKER	0.5	Prefab
12st_textury	PLANE	0.004	.fbx model
12st_venku	EXTERIÉR	5.5	Prefab
13st_bila	EXTERIÉR	5.5	Prefab
13st_marker	MARKER	0.5	Prefab
13st_textury	PLANE	0.004	.fbx model
13st_venku	EXTERIÉR	5.5	Prefab
17st_bila	EXTERIÉR	3.5	Prefab
17st_marker	MARKER	0.5	Prefab
17st_textury	PLANE	0.004	.fbx model
17st_venku	EXTERIÉR	3.5	Prefab
19st_bila	EXTERIÉR	3.5	Prefab
19st_marker	MARKER	0.5	Prefab
19st_textury	PLANE	0.004	.fbx model
19st_venku	EXTERIÉR	3.5	Prefab
20st_bila	EXTERIÉR	3.5	Prefab
20st_marker	MARKER	0.5	Prefab
20st_textury	PLANE	0.004	.fbx model
20st_venku	EXTERIÉR	3.5	prefab

Tu ovlivnilo výsledné umístění modelu, ať už bylo v části **MARKER**, **PLANE** nebo **EXTERIÉR**. Jelikož byly modely z Blenderu vyexportovány ve stejném měřítku, muselo

tak být provedena modifikace jejich velikosti (přescalovaní) a přeuložení, jelikož některé části aplikace nerespektují původní .fbx model a zobrazení modelů stejné velikosti v rozdílných částích aplikace nebylo dostatečně reprezentativní. Zejména v části **EXTERIÉR** byly velikosti modelů rozdílné. Stalo se tak kvůli tomu, že digitální model neodpovídal rozměrům reálného objektu návrší. Z toho důvodu byly modely z 12. a 13. století zvětšeny více než ostatní. K přeuložení pak došlo jednoduchým natažením původního modelu do scény, kde pak byla provedena operace *Unpack Prefab Completely*. Poté bylo ještě nutné výsledný *Prefab* přetáhnout do složky *Prefabs*, jinak by sloužil jako pouhý placeholder a nebyl plně uložený v projektu (viz Obrázek 31).

Obrázek 31 Rozdíl mezi .fbx modelem (nahoře a vlevo) a Prefabem (dole a vpravo)

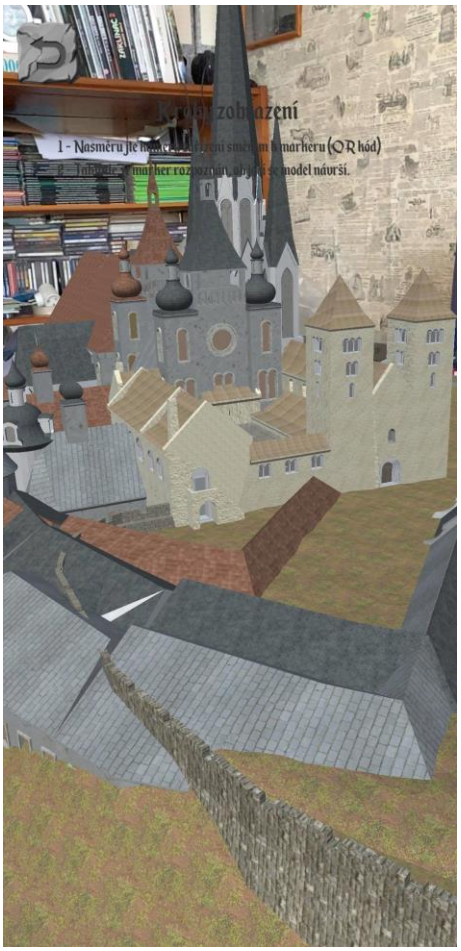
4.4 Interiérová část aplikace

Jelikož i tato část aplikace byla rozdělena na dvě části, nebylo v tomto případě možné se tedy z úvodní scény dostat přímo na vizualizaci modelů pomocí AR. Tato „druhá úvodní scéna“ byla vytvořena pro vstup do části aplikace s názvem **INTERIÉR**. Do této části se lze dostat po kliknutí na tlačítko se stejným názvem v úvodní scéně (viz Obrázek 32). Grafická podoba této scény se prakticky neliší od té úvodní. Došlo pouze k výměně obrázků a textových polí zachycujících funkcionalitu scén, které se objeví po stisknutí těchto tlačítek. Výraznou změnou je absence *Popup* tlačítka, ale přítomnost tlačítka se symbolem zpětné šipky, které slouží pro návrat na úvodní scénu. Veškerá tlačítka náleží objektu *Canvas* a jeho podobjektu *Panel*. Opět nechybělo ani tlačítko bez přidané funkcionality sloužící jako grafický rozdělovač mezi tlačítky *MARKER* a *PLANE* sloužících jako vstup do dalších části aplikace (viz Obrázek 32).

Obrázek 32 Náhled scény pro interiérovou část aplikace v prostředí Unity

Část MARKER

První část aplikace využívající AR nese název **MARKER**. Marker sám o sobě je jakýkoliv obrázek, po jehož naskenování mobilním zařízením se zobrazí digitální model. V tomto případě veškerou funkcionalitu přinesl Asset **Mixed and Augmented Reality Studio (MARS)**. Ten byl v průběhu práce doinstalován na základě nefunkčnosti původního řešení. Původně byla tato část řešena pomocí **ARFoundation**. Bylo založeno základní schéma s pomocí objektů jako *AR Session Origin* a *AR Session* (viz Část PLANE). Od tohoto postupu bylo upuštěno na základě zmíněné nefunkčnosti, kdy marker nebyl detekován mobilním zařízením a docházelo tak k nepříjemným chybám nebo celkovému přetížení systému na platformě iOS (viz Obrázek 33).



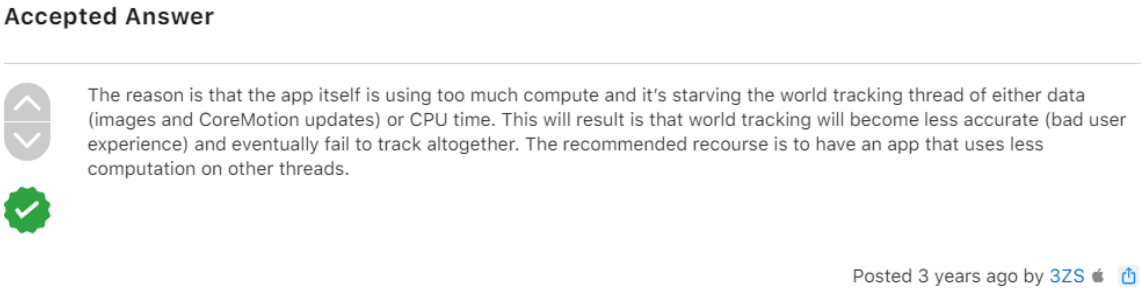
Obrázek 33 Chyba zobrazení modelů v části MARKER na zařízení iOS

Chyba spočívala buď v načtení všech modelů, ač nebyli ve scéně, nebo v přetížení systému s následujícími chybovými hláškami:

[Technique] World tracking performance is being affected by resource constraints [0]

[Technique] World tracking performance is being affected by resource constraints [1]

Dělo se tak po instalaci aplikace na iOS zařízení pomocí programu XCODE. Jednalo se o problém, který způsobovala výpočetní náročnost některých operací, které byly pro aplikaci stěžejní, konkrétně právě detekce markerů nebo přetížení takzvaných „CoreMotion updates“. Těmi jsou myšleny například akcelerometr a gyroskop zabudovaný do mobilního zařízení, které využívají data produkovaná HW zařízením (viz Obrázek 34). Na základě těchto dat se pak přetěžuje CPU, což v případě této práce vyústilo k výslednému nedetekování markerů nebo k již zmíněnému bugu s modely.

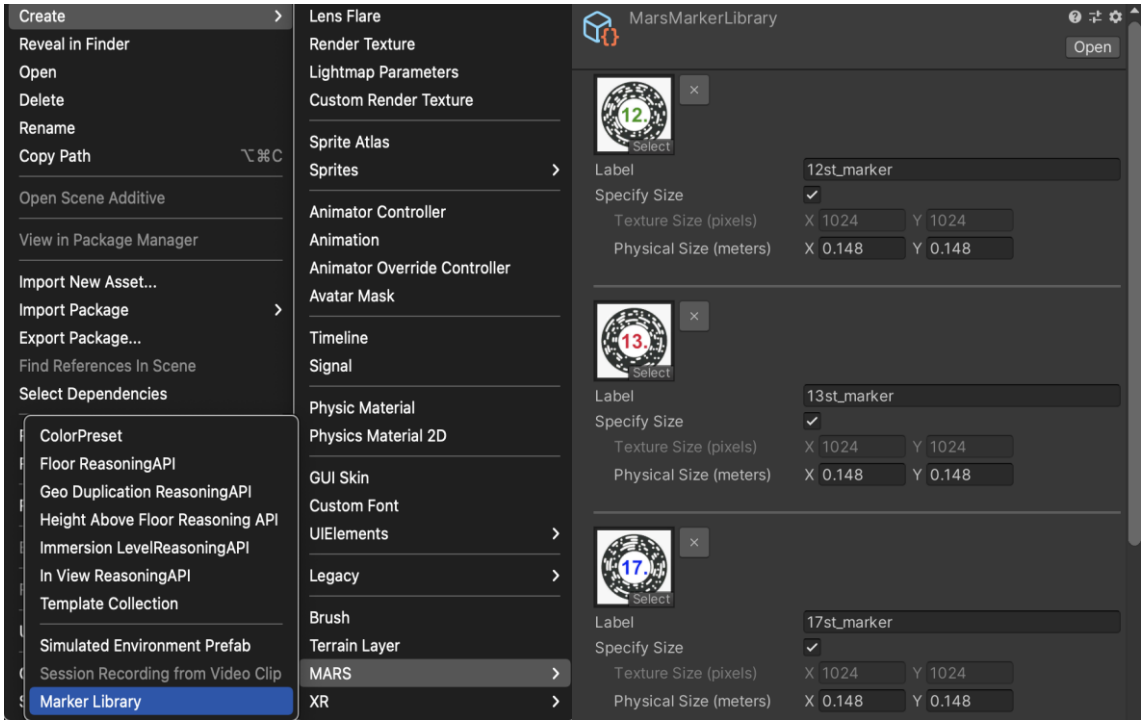


Obrázek 34 Vysvětlení objevující se chyby

Zdroj: https://stackoverflow.com/questions/45511838/technique-world-tracking-performance-is-being-affected-by-resource-constraints

Z toho důvodu autor v průběhu práce přešel právě na Asset **MARS**. Jedná se o obdobu **ARFoundation**, která poskytuje uživatelsky přívětivé rozhraní a veškerá funkcionalita pro tuto část je zde naprogramována. Prvním krokem bylo její stáhnutí, jedná se verzi s plnou funkcionalitou na 45 dní. Pro tvorbu diplomové práce dostačující časový úsek, kdy i po vypršení této verze veškerá funkcionalita zůstává vepsaná v aplikaci. Pokud by byly nutné další úpravy v Unity projektu, bylo by potřeba zakoupit plnou verzi Assetu.

Základ scény tak tvoří objekt *MARS Session*, kde je automaticky vytvořena *Main Camera*. Tento název mohl být zavádějící, jelikož jak autor uvedl, *Main Camera* je objekt defaultně vytvořený v Unity. V tomto případě se ovšem jednalo o zmíněnou knihovnu **MARS**, kde objekt má sice stejný název, nicméně obsahuje funkcionalitu umožňující zobrazit digitální model v AR prostředí. Nejdůležitější část ovšem tvoří objekty *Image Marker*. Do nich jsou zakódovány markery v podobě QR kódu. Za tímto účelem byla vytvořena *MarsMarkerLibrary*, což je ve své podstatě knihovna schraňující všechny markery v podobě obrázků (viz Obrázek 35).

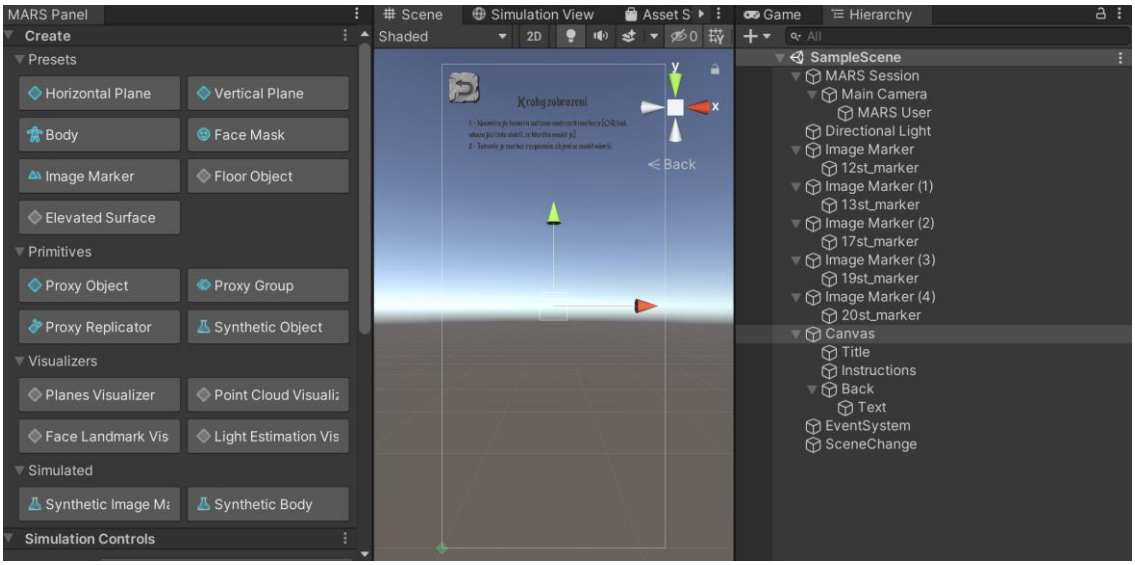


Obrázek 35 Tvorba a výsledná podoba MarsMarkerLibrary

U těchto obrázku záleželo na zvolení správného formátu, kterým byl .jpeg. Kromě .jpg se u ostatních formátů vyskytovaly problémy v podobě špatného snapování

virtuálního modelu přímo na marker (formáty .png a .tiff). Těmto obrázkům poté byl nastaven parametr v podobě jejich fyzické velikosti. Ta byla pouze orientační, jelikož ve výsledné aplikaci tyto rozměry nejsou podstatné. Zařízení po naskenování QR kódu dokázalo rozpoznat jakýkoliv smysluplný rozměr fyzického markeru. Testován byl i marker v podobě 3D modelu, nicméně v tomto případě šlo o nefunkční marker, jelikož kamera zařízení měla problém se zachycením textury zmíněného 3D modelu. Z tohoto důvodu autor zůstal u využití 2D markerů popsaných výše.

Dalším krokem bylo vytvoření pěti objektů *Image Marker*, což je objekt nativně umístěný v balíčku **MARS**. Ten byl vytvořen za pomoci *MARS Panelu* v levé části Unity (viz Obrázek 36).



Obrázek 36 Náhled scény MARKER, její hierarchie a panelu MARS v prostředí Unity

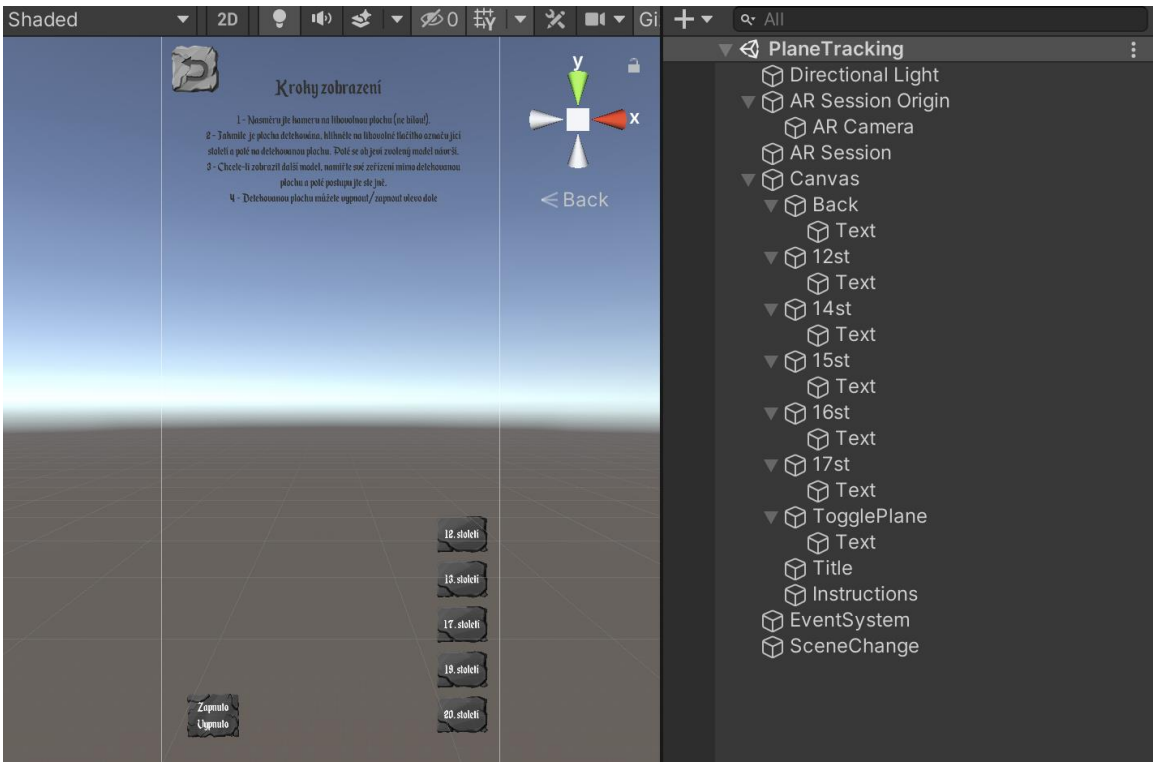
Pět jich bylo vytvořeno z důvodu prezentace modelů Svatováclavského návrší v pěti vybraných století. Poté bylo nutné přiřadit *Image Markerům* správný obrázek, který reprezentuje model z určitého století, který se má zobrazit. Vhodné reprezentace bylo dosaženo pomocí umístění barevných číslovek doprostřed QR kódu. Po přiřazení markeru pak bylo nutné vytvořit *Prefab* jednotlivých modelů z původních..fbx modelů (viz Kapitola 4.3). *Prefaby* pak byly umístěny do scény jako přidružený *Image Markerům*. Ikdyž jsou *Prefaby* na první pohled staticky umístěny do scény, právě balíček **MARS** umožňoval jejich zobrazení až po naskenování markeru mobilním zařízením. Pokud by nebylo **MARS**, vyskytovaly by se modely stejným způsobem, jako tomu bylo při výskytu bugu v aplikaci na platformě iOS (viz Obrázek 34). Právě **MARS** umožňuje celou situaci simulovat v prostředí Unity přes záložku *Simulation View*. Zde bylo vyzkoušeno zobrazení modelu, který se objevil po namíření virtuální kamery na zmiňovaný marker v prostředí Unity.

Další komponenty scény tvoří jednoduchý text s instrukcemi, jak s danou částí aplikace pracovat (viz Obrázek 45). Druhou komponentou je opět tlačítko se symbolem šipky zpět v levém horním rohu, které uživatele vrací na scénu, která je rozcestníkem pro interiérovou část aplikace. Pro funkčnost tlačítka byl opět připojen skript zajišťující změnu scén po kliknutí na zmíněné tlačítko. Jeho plnou funkčnost zajišťuje placeholder objekt v podobě *SceneChange*, který je opět prázdný pouze s připojeným skriptem na přepínání scén.

Obrázek 37 Instrukce pro uživatele, jak pracovat s částí aplikace MARKER

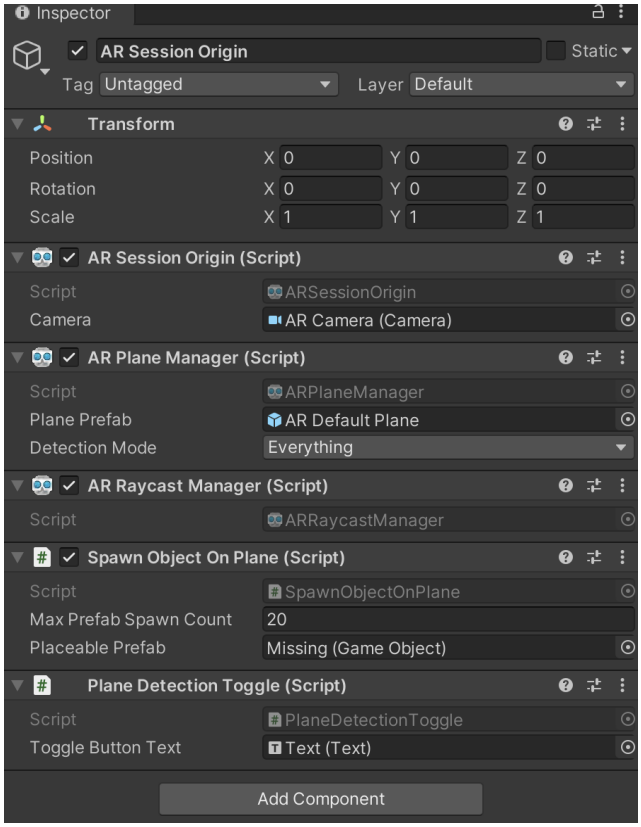
Část PLANE

Druhá část interiérové části aplikace využívající AR nese název **PLANE**, z důvodu detekce plochy, na kterou pak bylo možné libovolně umístit digitální modely návrší. Bylo tedy nutné vytvořit aplikační funkcionalitu, na základě, které mobilní řízení dokázalo detekovat plochy fyzických objektů. K tomu byla využita, na rozdíl od **MARS** balíčku, knihovna **ARFoundation**. Z toho důvodu byly do scény poprvé přidány objekty *AR Session Origin* a *AR Session*. Dále byla přidána *AR Camera*, která musela být označena jako hlavní kamera (viz Obrázek 38).



Obrázek 38 Náhled scény části aplikace PLANE v prostředí Unity

Důvod byl prostý, kamera mobilního zařízení není primárně nastavená na zobrazování AR, přidáním tohoto prvku tak byla primární funkce v aplikaci přenastavena právě na schopnost zobrazovat AR. V řešení části **MARKER** celou tuto funkcionalitu zajistil objekt *MARS Session*. Jelikož **ARFoundation** neobsahoval celou požadovanou funkcionalitu, bylo zapotřebí nastavit objekt *AR Session Origin* (viz Obrázek 39).



Obrázek 39 Nastavení objektu AR Session Origin

Prvním krokem bylo nastavení části *AR Plane Manager*, do které byl přidán *Prefab* z knihovny **ARFoundation**, konkrétně *AR Default Plane* a v kolonce *Detection Mode* byla vybrána možnost *Everything*. Tím bylo zajištěno snímání veškerých ploch a umožnění mobilnímu zařízení tyto plochy skenovat. Části *AR Session Origin* a *AR Raycast Manager* byly nastaveny defaultně. Stěžejním úkolem bylo vytvoření skriptu *Spawn Object On Plane*. Tento skript zajistil, že po zvolení modelu v sloupci tlačítek kliknutím a následným kliknutím na detekovanou plochu se zobrazí model návrší. Číslovka 20 v kolonce *Max Prefab Spawn Count* udává počet modelů, které je možné umístit na detekovanou plochu. Do další kolonky pod názvem *Placeable Prefab* nebyl umístěn žádný model ani *Prefab*, a to z toho důvodu, že pro tuto část nebylo vhodné, aby byl staticky bez jakéhokoliv přičinění uživatele zobrazovaný libovolný model. *Plane Detection Toggle* skript pak sloužil k vypnutí/zapnutí naskenované plochy.

Úvod byl prakticky stejný jako v minulých ukázkách skriptů. První tři řádky byly vytvořeny defaultně, jelikož skript byl vytvořen v Unity. Čtvrtý a pátý řádek byl přidán na základě využívání knihoven **ARFoundation** a **ARSubsystems** (viz Obrázek 40).

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

Obrázek 40 Úvod do skriptu zajišťující funkcionalitu části aplikace PLANE

Další část se zaměřuje na zadání kritérií nastavitelných v Unity. Důležité bylo zavolání objektu z knihovny **ARFoundation**, kterým byl *ARRaycastManager*. Ten zapříčiní spuštění sekvence skenů na mobilním zařízení, které skenují okolní fyzické prostředí. Dalším důležitým parametrem byl *GameObject*, který u umísťovaných modelů definoval, co se s nimi mělo stát, konkrétně jejich vygenerování

se na detekované ploše. K tomuto parametru byla přidána třída *List*, která modely umístila do seznamu, ze kterého se pak mohl vybírat libovolný model k umístění. Příkazy umístěné do třídy *[SerializeField]* pak vytvořili parametry zmíněné výše: *Max Prefab Spawn Count* a *Placeable Prefab*. Datový typ *int* pak určil, že do se do kolonky mohlo zadat pouze celé číslo. Poslední řádek potom definoval, co se má stát po kliknutí na detekovanou plochu, tedy že se má vygenerovat zvolený model (viz Obrázek 41).

```
[RequireComponent(typeof(ARRaycastManager))]
public class SpawnObjectOnPlane : MonoBehaviour
{
    private ARRaycastManager raycastManager;
    private GameObject spawnedObject;
    private List<GameObject> placedPrefabList = new List<GameObject>();

    [SerializeField]
    private int maxPrefabSpawnCount = 0;
    private int placedPrefabCount;

    [SerializeField]
    private GameObject placeablePrefab;

    static List<ARRaycastHit> s_Hits = new List<ARRaycastHit>();
```

Obrázek 41 Část skriptu definující zadavatelné parametry v prostředí Unity

Třetí část skriptu na začátku nejdříve zavolala *ARRaycastManager*, kvůli funkcím zmíněným výše. Dále definovala, co se stane, když uživatel klikne na detekovanou plochu nebo mimo ni. Tedy pokud klikne mimo ni, neobjeví se žádný model a požadavek se opakuje, dokud se neklikne na plochu (viz Obrázek 42).

```
private void Awake()
{
    raycastManager = GetComponent<ARRaycastManager>();
}
bool TryGetTouchPosition(out Vector2 touchPosition)
{
    if(Input.GetTouch(0).phase == TouchPhase.Began)
    {
        touchPosition = Input.GetTouch(0).position;
        return true;
    }

    touchPosition = default;
    return false;
}
```

Obrázek 42 Část skriptu definující stav kliknutí na detekovanou plochu

Předposlední část skriptu navazuje přímo na předchozí. Pokud bylo kliknuto na detekovanou plochu, čtení skriptu se přesunulo do skupiny *private void Update()*. Zde bylo řízeno generování modelu na detekovanou plochu. Tedy pokud se na ní kliklo, vygeneruje se model. Počet modelů byl podmíněn proměnnou *maxPrefabSpawnCount*. Ta udávala maximální počet modelů k umístění, jak autor již dříve zmínil (viz Obrázek 43).

```
private void Update()
{
    if(!TryGetTouchPosition(out Vector2 touchPosition))
    {
        return;
    }

    if(raycastManager.Raycast(touchPosition, s_Hits, TrackableType.PlaneWithinPolygon))
    {
        var hitPose = s_Hits[0].pose;
        if(placedPrefabCount < maxPrefabSpawnCount)
        {
            SpawnPrefab(hitPose);
        }
        }
}
```

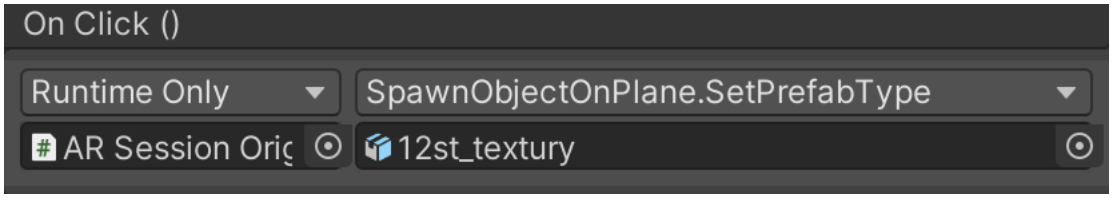
Obrázek 43 Část skriptu vyhodnocující požadavek z předchozí části

Poslední část skriptu už jen zpracovávala výsledky z předchozích částí. Pokud tedy bylo kliknuto na detekovanou plochu, vygeneroval se zvolený model, definováno skupinou *public void SetPrefabType(GameObject prefabType)*. Skupina *private void SpawnPrefab(Pose hitPose)* pak zvolený model umístila na detekovanou plochu v závislosti na rotaci zařízení a pozice kliknutí na ni. Model byl vybrán z definovaného seznamu a za pomoci posledního řádku skriptu pak byl definitivně vygenerován na detekované ploše. Určitým paradoxem bylo, že ve skriptu bylo nutné všude nadefinovat *Prefab*, který ovšem v této části nebyl vyžadován a bylo možné pracovat s .fbx modelem (viz Obrázek 44).

```
public void SetPrefabType(GameObject prefabType)
{
    placeablePrefab = prefabType;
}

private void SpawnPrefab(Pose hitPose)
{
    spawnedObject = Instantiate(placeablePrefab, hitPose.position, hitPose.rotation);
    placedPrefabList.Add(spawnedObject);
    placedPrefabCount++;
```

Obrázek 44 Závěrečná část skriptu zajišťující správné umístění modelu na detekovanou plochu

Nicméně k vytvořenému skriptu bylo třeba vytvořit spouštěč v podobě tlačítka. Jelikož v části **MARKER** bylo prezentováno pět modelů, v této části tomu nebylo jinak. Bylo tedy vytvořeno pět tlačítek, která byla umístěna po pravé straně v dolní části scény, s textem popisujícím název století, ze kterého daný model byl. Každému z těchto tlačítek byl přiřazen předchozí skript *SpawnObjectOnPlane,* který umožňoval jak zvolení modelu před umístěním na detekovanou plochu, tak i jeho umístění. Dále byl přidán model z určitého století korespondujícího s textem tlačítka. Vybrána byla opět funkce *On Click ()*, díky níž došlo k provedení operace po kliknutí, ať už na tlačítko nebo na detekovanou plochu (viz Obrázek 45).



Obrázek 45 Nastavení funkcionality tlačítka v prostředí Unity pro vygenerování modelu na detekované ploše

Tím byla vytvořena základní struktura této části aplikace. Nicméně pro větší pohodlí uživatele došlo k vytvoření tlačítka na vypnutí, potažmo zapnutí detekované plochy. Nejprve bylo zapotřebí vytvořit další C# skript, který tuto funkcionalitu obsahoval. Skript využíval stejné knihovny jako ten předchozí a klíčovou částí tak byla opět třída *[SerializeField]* (viz Obrázek 46). Ta opět definovala volitelný parametr v prostředí Unity stejně jako v minulém skriptu s tím rozdílem, že do něj bylo nutné vložit textový objekt.

```
[SerializeField]
private Text toggleButtonText;
```

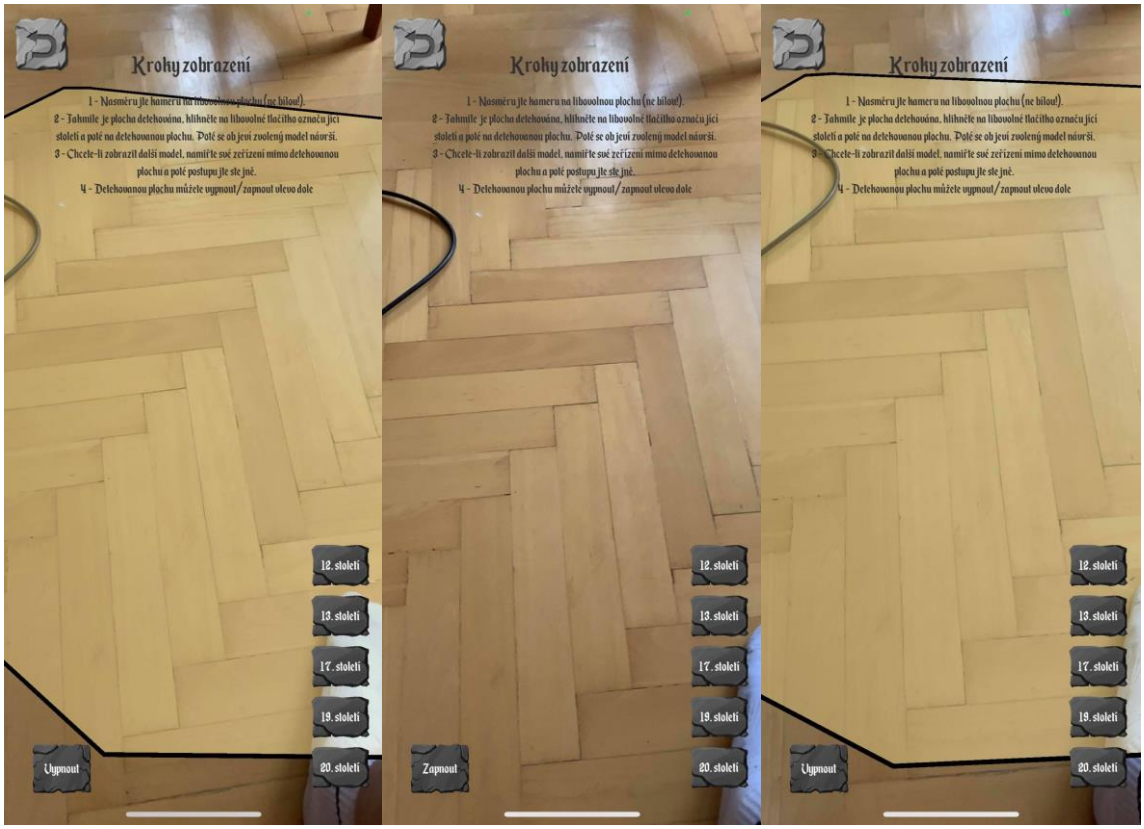
Obrázek 46 Část skriptu definující parametr v prostředí Unity

V tomto textovém objektu se nacházel prostý text Vypnuto/Zapnuto, kde každé slovo bylo nutné umístit na jiný řádek, aby byla zajištěna funkčnost další části skriptu. V té bylo definováno, co se má stát po stisknutí zmíněného tlačítka. Třída *private void Awake()* definuje stav tlačítka při načtení scény, tedy že se detekovaná plocha nejdříve vypíná. Ve třídě *public void TogglePlaneDetection()* je naopak definováno co se má stát při jeho stisknutí, čili pokud byl text v tlačítku „Vypnuto“, měla se po jeho stisknutí detekovaná plocha vypnout a text změnit na „Zapnuto“. Pokud byl text v tlačítku „Zapnuto“, detekovaná plocha se má zapnout a text změnit na „Vypnuto“ (viz Obrázek 47).

```
private void Awake()
{
    planeManager = GetComponent<ARPlaneManager>();
    toggleButtonText.text = "Vypnout";
}

public void TogglePlaneDetection()
{
    planeManager.enabled = !planeManager.enabled;
    string toggleButtonMessage = "";

    if(planeManager.enabled)
    {
        toggleButtonMessage = "Vypnout";
        SetAllPlanesActive(true);
    }
    else
    {
        toggleButtonMessage = "Zapnout";
        SetAllPlanesActive(false);
    }
```

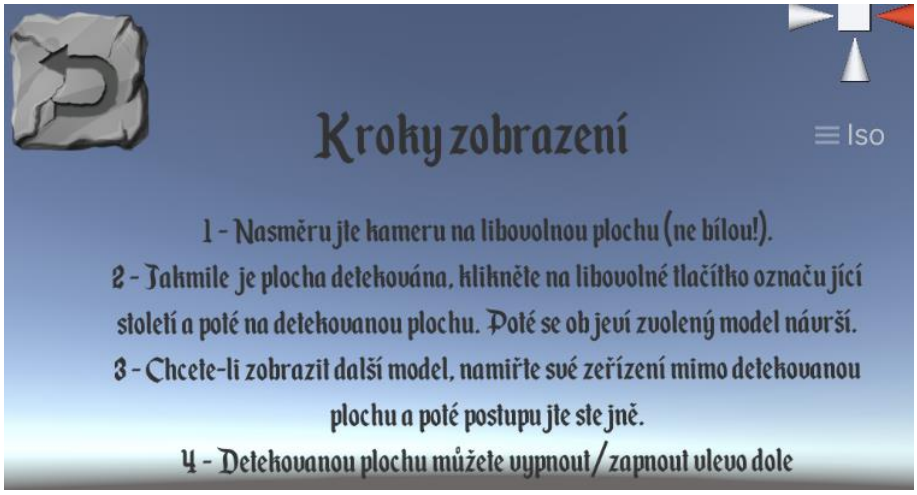
Obrázek 47 Skript zajišťující přepínání názvu tlačítka Vypnuto/Zapnuto

V případě že byla detekována nějaká plocha, je po jejím vypnutí stále uchována v paměti a po opětovném zapnutí se opět objevila v předchozím stavu (viz Obrázek 48).



Obrázek 48 Vypínání a zapínání detekované plochy v prostředí aplikace

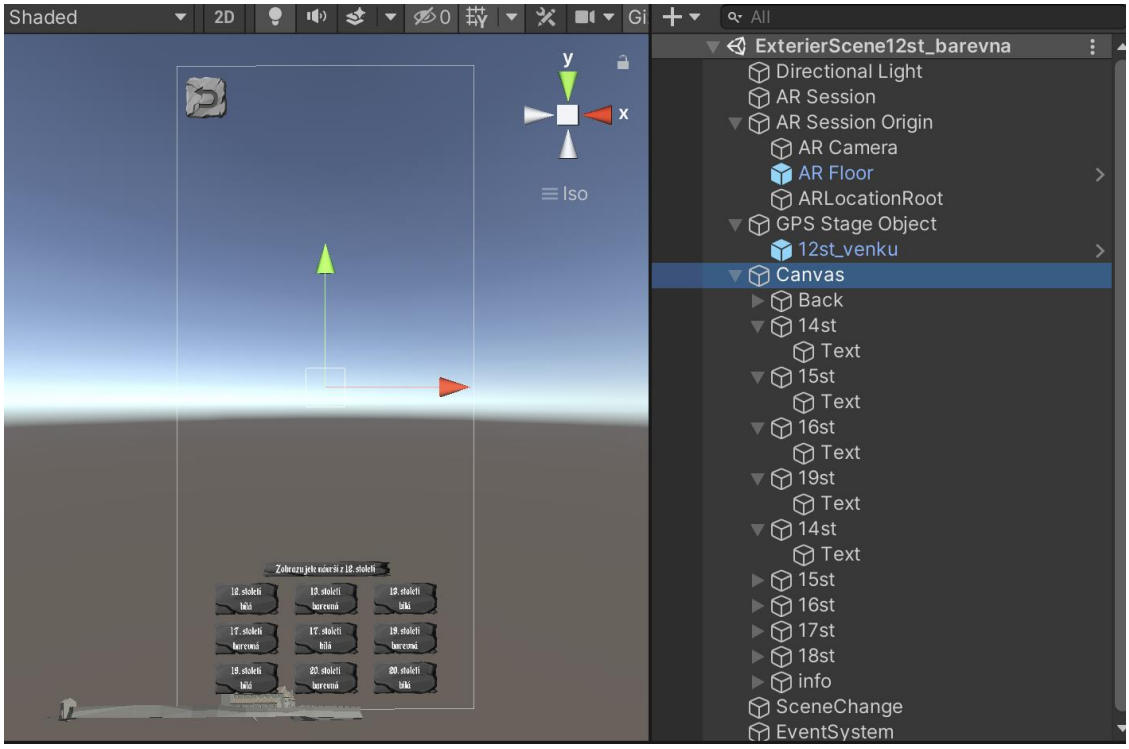
Ve scéně opět nechybělo tlačítko zpětné šipky v levém horním rohu, které po stisknutí vrátilo uživatele zpět na předchozí scénu, kterou byla beze změny předchozí scéna v podobě rozcestníku pro interiérová část aplikace. Funkčnost byla zajištěna opět pomocí placeholder objektu *SceneChange*. Tlačítko mělo připojený skript pro přepínání scén, stejně jako zmíněný placeholder objekt. Nechyběli ani jednoduché instrukce pro uživatele, jak se v této části aplikace pohybovat a jak jí ovládat, v podobě textového objektu (viz Obrázek 49). Do závorky bylo uvedeno, že není doporučeno pokoušet se využívat k detekci plochy místa s bílým povrchem (viz Kapitola 3.3.2).



Obrázek 49 Instrukce pro uživatele, jak ovládat část aplikace PLANE

4.5 Exteriérová část aplikace

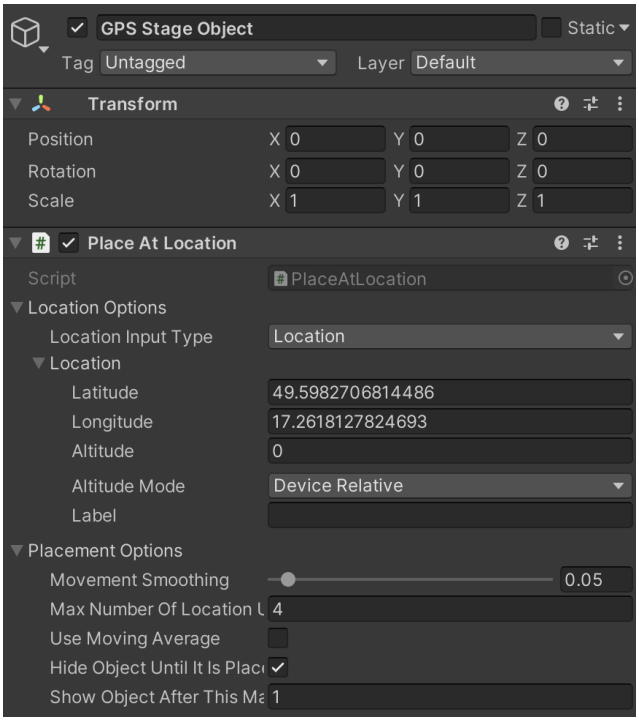
Kromě interiérové části aplikace byla druhou hlavní částí **EXTERIÉR**, na kterou se uživatel dostane po kliknutí na tlačítko **EXTERIÉR** v úvodní scéně. Bylo zde klíčové zobrazení modelů návrší ve venkovní části (viz. Obrázek 50). Základ scény tvořily objekty z knihovny **ARFoundation** v podobě *AR Session* a *AR Session Origin* s přidruženým objektem *AR Camera*. Jelikož se jednalo o venkovní část, bylo zapotřebí získat funkcionalitu umožňující zobrazit digitální model na základě GPS souřadnic. S ohledem na to, že Unity nativně nemá vestavěnou knihovnu nebo jakoukoliv jinou funkcionalitu potřebnou pro tuto část, došlo k zakoupení Unity Assetu **Unity AR+GPS Location**. Ten v sobě obsahoval veškerou funkcionalitu i objekty, které byly zapotřebí k dokončení této části aplikace.



Obrázek 50 Náhled jedné z exteriérových scén v prostředí Unity

Prvním přidaným objektem byl *GPS Stage Object* (viz Obrázek 51). Jelikož obsahoval skript *Place At Location*, který byl obsažen nativně v Assetu **Unity AR+GPS Location**, nebylo nutné programovat další funkcionalitu, jako tomu bylo v části **PLANE**. V něm jsou naprogramovány všechny nutné parametry, které je třeba zadat přímo v Unity. Prvním z nich byl *Location Input Type*, kde byla vybrána možnost *Location*, díky které pak bylo možné zadat další parametry. Prvními z nich byla zeměpisná šířka a délka. Zadané souřadnice byly vybrány tak, aby korespondovaly s umístěním návrší v reálném prostředí. Nadmořské výšce zůstala hodnota 0, jelikož v parametru *Altitude Mode* byla vybrána možnost *Device Relative*. Tento parametr obsahuje dohromady čtyři možnosti výběru:

- Absolute – využívá výšková data zařízení a uvažuje nadmořskou výšku relativní k úrovni hladiny moře
- Device Relative – uvažuje nadmořskou výšku za relativní vůči zařízení
- Ground Relative – uvažuje nadmořskou výšku za relativní k nejbližší detekované ploše
- Ignore – při výpočtech ignoruje nadmořskou výšku



Obrázek 51 Nastavení objektu GPS Stage Object v Unity

Testována byla ještě možnost *Ground Relative*, kde ovšem docházelo ke špatné detekci plochy, na kterou měl být model umístěn (viz Obrázek 52).

Obrázek 52 Vlevo možnost Ground Relative, vpravo Device Relative

Zbytek parametrů byl ponechán defaultně. *Movement Smoothing* byl ponechán na 0.05, kdy se přepočítává poloha modelu na základě dat z GPS. Pokud by hodnota byla 0, přepočítávání polohy by bylo vypnuté. *Max Number Of Location Updates* udávalo, kolikrát se přepočítá poloha zařízení, než se vygeneroval model. *Use Moving Average*

zůstalo nezaškrtnuté, jelikož při testování umísťovalo objekt mimo požadované místo. *Hide Object Until It Is Placed* udává, že se model nevygeneruje, dokud není vypočítána poloha zařízení. Posledním parametrem byl *Show Object After This Many Updates*. Zde bylo zadáno číslo 1. To zadávalo počet aktualizací polohy předtím, než byl vygenerován model, který byl původně skryt.

Dalším krokem bylo přidružení *Prefabu* objektu *GPS Stage Object*. Tím byl *Prefab* z daného století, kterému náležela celá scéna. Pro zobrazení dalšího modelu bylo zapotřebí scénu přepnout pomocí sady tlačítek ve spodní části.

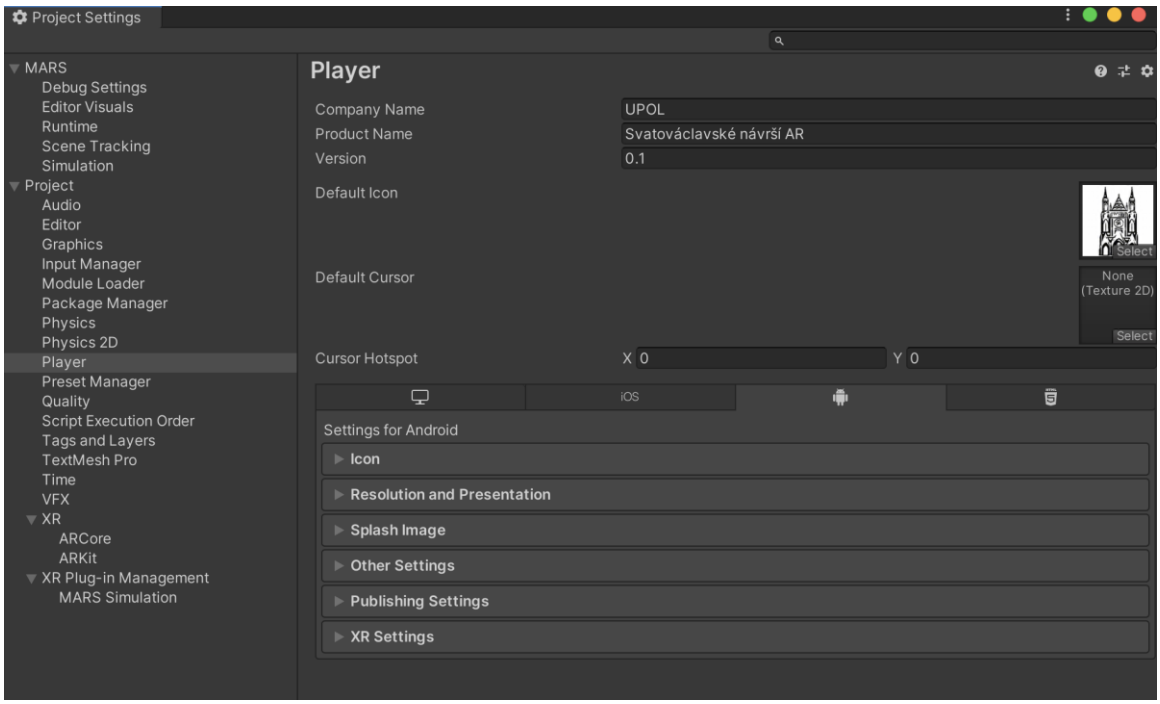
Dokončením výstavby funkcionality pro tuto část aplikace bylo přidružení *ARLocationRoot* a *ARFloor* objektu *AR Session Origin*. První objekt zajišťoval správnou rotaci *Prefabu* na základě GPS souřadnic, na které byl umístěný. *ARFloor* pak byl *Prefab*, který se nativně nacházel v knihovně **Unity AR+GPS Location**. Jeho přidáním se nadefinovala plocha, na kterou se generoval daný model návrší. S kombinací *Altitude Mode*, kde zvolena byla možnost *Device Relative* se jednalo o plochu, která byla v nadmořské výšce relativní vůči mobilnímu zařízení.

Pro přepínání mezi jednotlivými modely byla vytvořena sada tlačítek, která fungovala na bázi změny scény, ve které byl vložen objekt z určitého století. Těchto tlačítek bylo v každé scéně devět, jelikož modelů k zobrazení v této části bylo celkově deset. Bylo to z důvodu, kdy se autor na podněty vedoucího práce rozhodl přidat i netexturované modely. Vytvořeno bylo i stejné množství scén. Během přepínání scén se jednotlivým tlačítkům měnily názvy na základě zobrazovaného modelu, jehož tlačítko vždy ve scéně chybělo. Bylo tak učiněno proto, aby se uživateli nestalo, že se zastaví na jedné scéně a při kliknutí na tlačítko století, které již zobrazuje by se nic nestalo. Nad těmito devíti tlačítky byl umístěn stavový řádek, ve kterém se vždy zobrazil název století, ze kterého byl model zobrazován. Celé přepínání scén funguje na stejném principu jako zpětné tlačítko v předchozích scénách, které nechybí ani v těchto scénách. Bylo toho opět dosaženo pomocí prázdného placeholder objektu *SceneChange*. Tlačítko se symbolem zpětné šipky opět sloužilo pro návrat na úvodní scénu celé aplikace.

4.6 Export aplikace

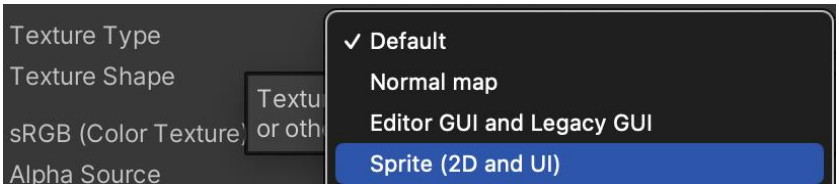
Posledním krokem vlastního řešení bylo vyexportování aplikace. Ta byla testována na zařízeních Android a iOS, nicméně výsledná aplikace byla publikována pouze pro Android z důvodu zmíněné nefunkčnosti části **MARKER** (viz Kapitola 4.4).

Nejprve bylo definováno nastavení pro aplikaci na platformu Android. Před samotným exportem bylo třeba nastavit určité parametry. Prvním krokem byl název společnosti a název produktu: Svatováclavské návrší AR. Poté byla vytvořena ikona aplikace na webu https://imagetosketch.com/, kam byla nahrána fotografie katedrály a následně vytvořena její silueta. Pro nahrání bylo důležité, aby nahraný obrázek byl ve formátu .jpg (viz Obrázek 53).



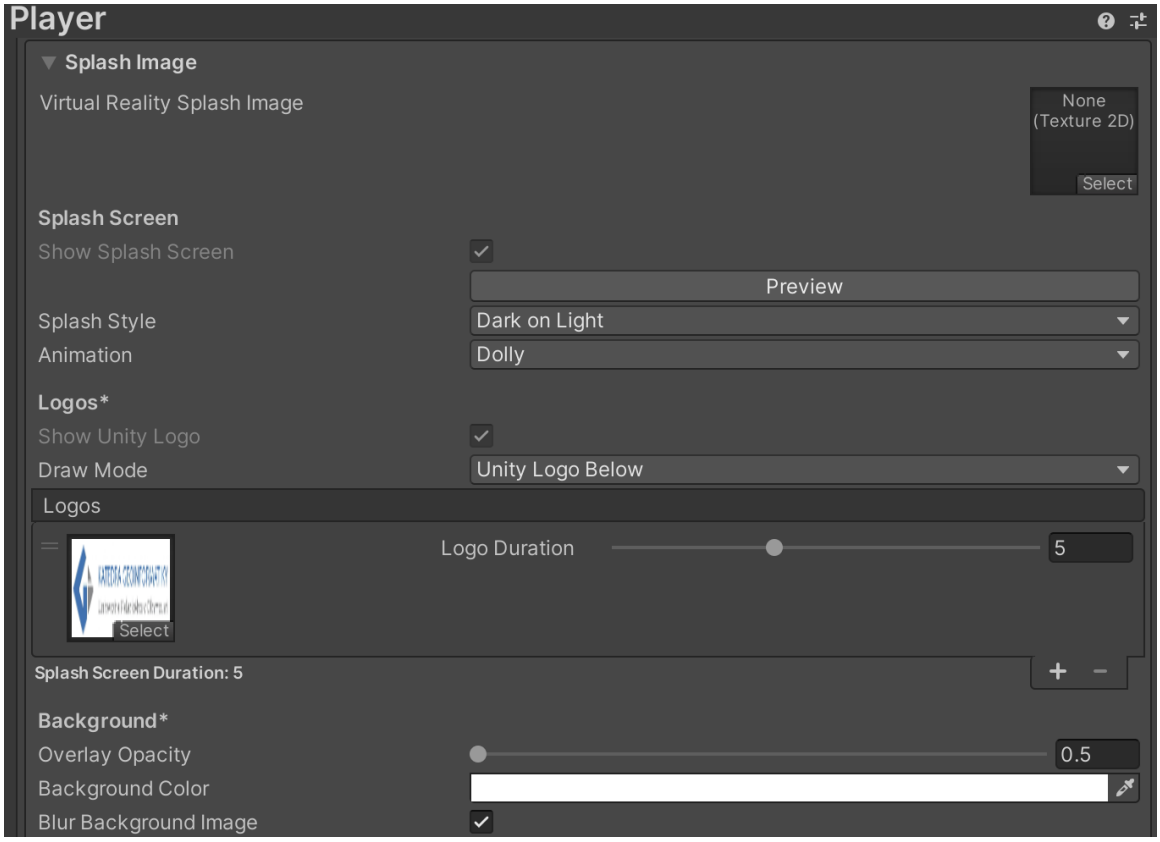
Obrázek 53 Základní nastavení jména a ikony aplikace

Dalším krokem bylo vytvoření takzvaného úvodní obrazovky (dále jako *Splash Screen*), která se načetla při spuštění aplikace. Bylo tak učiněno v záložce *Splash Image*. Do této části bylo nahráno logo katedry, které ovšem nebylo možné nahrát ve formátu .jpg. Bylo tedy nutné logo přeuložit. Podobně jako tomu bylo u modelů, které byly přeuloženy do Unity formátu *Prefab*, bylo nutné logo transformovat do formátu zvaného *Sprite*. Toho se dosáhlo jednoduchou operací, kdy po vybrání obrázku v souborech projektu došlo k přepnutí *Texture Type* na *Sprite (2D and UI)* (viz Obrázek 54).



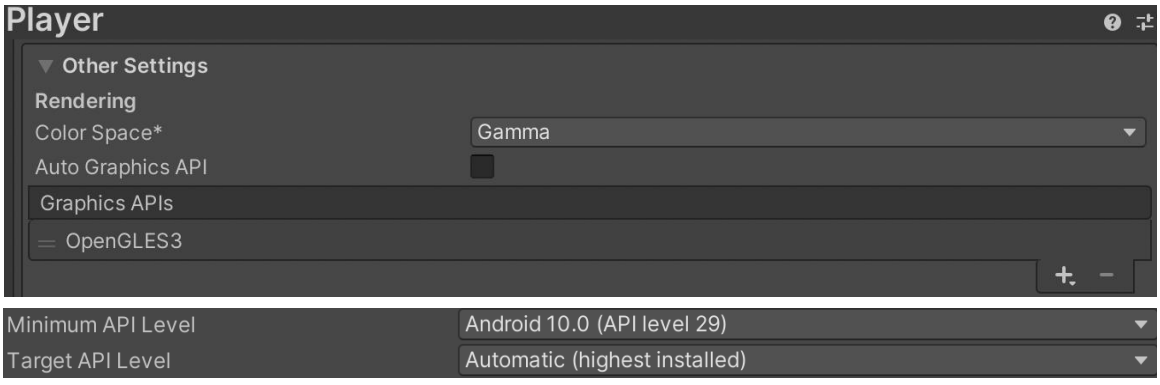
Obrázek 54 Změna formátu obrázku zastupující ikonu aplikace

Byla nastavena doba trvání *Splash Screenu* na 5 sekund, barva pozadí na bílou, zbytek parametrů byl ponechán v defaultním nastavení (viz Obrázek 55).



Obrázek 55 Nastavení Splash Screenu aplikace

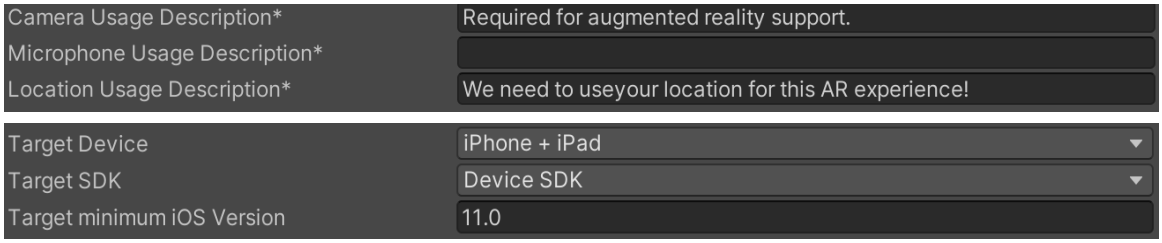
Předposledním krokem bylo nastavení v sekci *Other Settings*. Zde bylo nutné z grafických API odstranit API Vulkan, který výrazně zvyšoval výpočetní náročnost aplikace. Dále pak byl přenastaven *Minimum API Level* na Android 10.0. Byly testovány i dřívější verze, nicméně na nich export aplikace pokaždé hlásil chyby, které vyžadovaly zásah do kódu Unity, právě proto byla vybrána verze 10.0. Jako cílové API bylo vybrané automatické, tedy nejnověji nainstalované, které zajistilo běh i na novějších verzích Androidu (viz Obrázek 56).



Obrázek 56 nastavení parametrů klíčových pro Android aplikaci

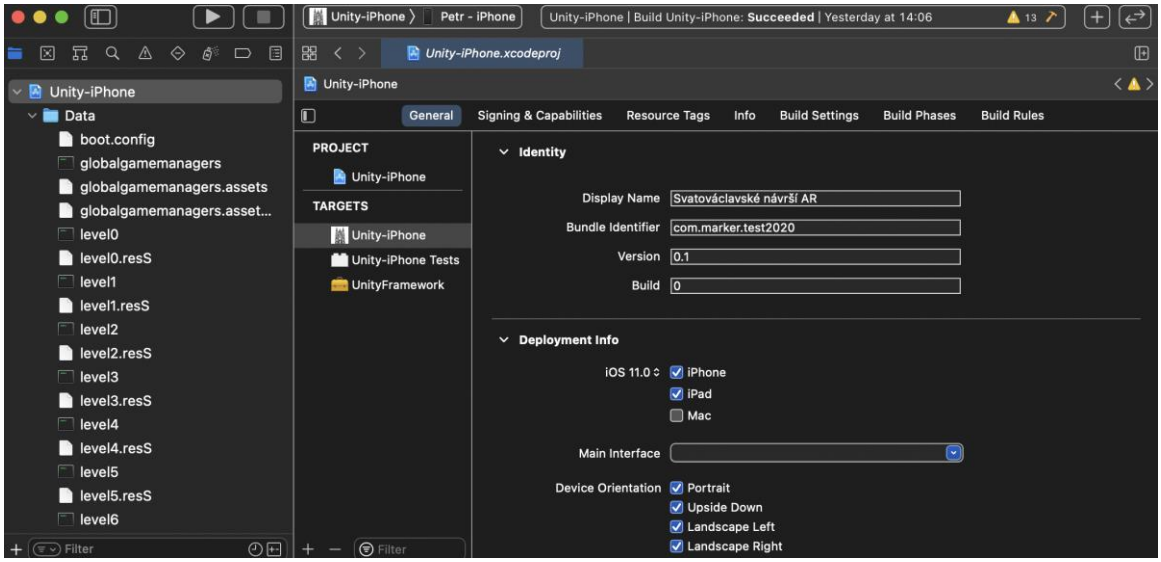
Závěrečným krokem tak už byl samotný export aplikace pro Android. Toho bylo dosaženo v nabídce *File -> Build Settings*. Zde bylo nutné nastavit správnou cílovou platformu, tedy Android a naimportovat scény, které byly nastaveny v předchozích částech práce. Pak už jen stačilo stisknout tlačítko *Build*, díky kterému byl vygenerován instalační soubor .apk.

Pro vytvoření aplikace na platformy iOS byl postup prakticky stejný. Jediným rozdílem bylo nastavení v sekci *Other Settings*. Zde bylo potřeba definovat využití kamery cílového zařízení spolu s využitím polohy. Nastavenými cílovými platformami byly iPhone a iPad s minimální verzí iOS 11.0 (viz Obrázek 57).



Obrázek 57 Nastavení parametrů pro iOS aplikaci

K instalaci aplikace do mobilního zařízení byl zapotřebí další program XCODE. Jednalo se tak o komplikovanější postup než v případě vygenerování instalačního souboru .apk. V prostředí XCODE bylo třeba přihlásit se pomocí AppleID, poté připojit mobilní zařízení a nainstalovat aplikaci pomocí tlačítka *Play* v horní části XCODE (viz Obrázek 58). Po instalaci se aplikace sama spustila.



Obrázek 58 Rozhraní programu XCODE, pomocí kterého byla instalována testovací aplikace na zařízení iOS

Jak již bylo zmíněno v předchozích kapitolách, výsledná aplikace byla publikována pouze pro platformu Android z důvodu nefungující jedné ze stěžejních částí aplikace na platformě iOS. Pro export aplikace na Android bylo zapotřebí mít nainstalované knihovny **Android JDK**, **Android SDK Tools**, **Android NDK** a **Gradle**. Ty ovšem byly nainstalovány defaultně při založení Unity projektu, takže nebylo nutné s nimi nijak operovat.

5 VÝSLEDKY

Teoretická část práce je popsána v současném stavu řešené problematiky (viz Kapitola 3). V ní došlo k popisu historického významu muzeí a v návaznosti na ně jejich modernizaci a implementaci AR. Z pohledu AR byly rozlišeny její jednotlivé druhy a přístupy zobrazování. Definovány byly jednotlivé SDK pro mobilní zařízení a na závěr nejpoužívanější SW pro tvorbu aplikací využívajících AR.

Pilotním výsledkem a výstupem je mobilní aplikace pro platformu Android spolu s detailním popisem její tvorby. Aplikace je instalovatelná na zařízeních s verzí operačního systému Android 10.0. Veškerá její funkcionalita byla vytvořena v prostředí Unity na základě knihoven, kterými byli **ARFoundation**, **MARS** a **Unity AR+GPS Location**. Po spuštění aplikace se zobrazuje vytvořený *Splash Screen* (viz Příloha 1). Dále byla vytvořena úvodní scéna, která rozděluje aplikaci na Interiérovou a Exteriérovou část (viz Příloha 2). V jejím prostředí bylo vytvořeno vyskakovací okno, které obsahuje základní informace o aplikaci a návod, jak ovládat část **EXTERIÉR** (viz Obrázek 22). Další Interiérovou část také obsahuje úvodní scénu, která je rozdělena na část **MARKER** a **PLANE**. V pravém horním rohu je tlačítko, které umožňuje vrátit se zpět na úvodní scénu.

Část **MARKER** pak využívá technologie *Image Tracking*. Tato technologie je založená na skenování fyzického obrazu (markeru). Tyto markery byly vytvořeny v SW Vectornator. Jedná se o QR kód ve tvaru kruhu, kde v jeho středu je číslo označující století, ze kterého je generovaný model. K dalším výsledkům práce tak patří i vytvořená sada markerů dostupných na webových stránkách práce (viz Příloha 16). V Unity byla použita knihovna **MARS**, která obsahuje veškerou funkcionalitu potřebnou pro tuto část aplikace. Scéna v mobilní aplikaci obsahuje pouze jednoduché tlačítko zpět na úvodní scénu Interiérové části aplikace a text s instrukcemi, jak tuto část aplikace ovládat, a jak s ní operovat. Po naskenování markeru je tedy zařízení schopno vygenerovat digitální model, který se objeví přímo na markeru (viz Příloha 4). Naskenované modely se ukládají a zůstávají na displeji zařízení, dokud se uživatel nevrátí zpátky nebo celou aplikaci nevypne.

Druhá část interiérové aplikace nese název **PLANE**. Využívá technologie *Plane Tracking*, která je založená na detekování plochy naskenováním mobilním zařízením. Pro její vytvoření byla v Unity použita knihovna **ARFoundation**. Ta nicméně neobsahuje celou funkcionalitu, tím pádem byly vytvořeny C# skripty, kterými byla tato funkcionalita dovytvořena. Scéna v mobilní aplikaci obsahuje pět tlačítek, pomocí kterých si lze vybrat model z 12., 13., 17., 19. nebo 20. století. Tyto modely pak lze umístit na detekovanou plochu. Pro lepší uživatelský zážitek bylo přidáno tlačítko na vypnutí/zapnutí detekované plochy. Jeho funkcionalita byla též vytvořena pomocí C# skriptu. Ikdyž modelů je pouze pět, limit na počet umístěných modelů byl nastaven na 20. Ve scéně nechybí ani tlačítko pro návrat na úvodní scénu interiérové části aplikace. V horní části je umístěn text, který obsahuje instrukce k tomu, jak operovat s danou částí aplikace (viz Příloha 5).

Poslední částí byla část **EXTERIÉR**, která je dostupná po kliknutí na tlačítko **EXTERIÉR** v úvodní scéně. Po kliknutí na zmíněné tlačítko se objeví texturovaný model z 12. století v lokalitě Svatováclavského návrší v Olomouci. Tato část aplikace využívá Unity Asset zvaný **Unity AR+GPS Location**. Ten přináší funkcionalitu na základě, které je možné vytvořit *Location Based AR*. Ta využívá GPS souřadnic, které byly zadány v prostředí Unity. V dolní části displeje je pak sada tlačítek, která umožňuje měnit

zobrazovaný model, taktéž v lokalitě návrší. Modely jsou jak texturované, tak netexturované. Přepínání zobrazovaných modelů funguje na principu přepínání scén jako v předchozích případech, z toho důvodu bylo vytvořeno celkově 10 scén pro tuto část aplikace. Aby uživatel věděl, jaký model je právě zobrazován, byl nad tlačítka vytvořený stavový řádek podávající informaci o zobrazovaném modelu. V levém horním rohu je opět tlačítko zpět pro návrat na úvodní scénu celé aplikace (viz Příloha 6, 7, 8, 9, 10, 11, 12, 13, 14 a 15).

Výsledná aplikace se celkově skládá ze 14 scén vytvořených v Unity. Úvodní scéna celé aplikace, úvodní scéna pro interiérové část, scéna pro zobrazování markerů, scéna pro detekce plochy a umístění modelů právě na ni, a již zmíněných 10 scén pro exteriérové část aplikace, kde každá scéna obsahuje jeden určitý texturovaný nebo netexturovaný model z daného století. Tato aplikace je ke stažení z platformního obchodu Google Play nebo v podobě souboru .apk z webových stránek diplomové práce. Vytvořeny byly ještě C# skripty, dodávající určitou funkcionalitu (viz Příloha 17, 18, 19 a 20).

Nesmírnou výhodou výsledné aplikace je její interaktivnost a univerzálnost. Konkrétně zobrazení modelů pomocí markerů nebo umístění modelů na detekovanou plochu, kterou je možné použít prakticky všude. Výhodou je exteriérové část aplikace, kde má uživatel možnost prohlédnout si vývoj Svatováclavského návrší v průběhu staletí přímo v Olomouci. Nicméně tato přidaná hodnota může být i nevýhodou, jelikož se uživatel musí dostavit do lokality návrší. Ovšem tyto nevýhody byly prakticky v celém rozsahu odstraněny právě částmi aplikace **MARKER** a **PLANE**.

6 DISKUZE

Práce byla zaměřena především na implementaci geoinformačních technologií do tvorby interaktivní exhibice historických objektů. Lokalitou exhibice bylo Svatováclavské návrší v Olomouci a vybranou technologií byla rozšířená realita.

Tato část byla časově velmi náročná z důvodu importu modelů ze SketchUpu ve formátu .stl. Tento formát je nativním formátem CAD SW, což činilo model rozsekaným, co se plošek týče (např. jedna stěna domku v modelu nebyla jednolitá ploška, ale 50 plošek). Proto bylo nutné označovat jednu po druhé a poté jim přiřadit náležitou texturu. Pro budoucí práce by tak bylo výhodnější využít workflow Blender -> Unity bez předchozího použití SketchUp. U modelů sloužících k prezentaci v exteriérové části exhibice pak došlo úplnému odstranění digitálního modelu reliéfu a posunutí podstav budov. Ty byly původně protažené až na dno podstavy DMR, protože původní modely sloužily k tisku 3D modelů. Další komplikací byl export modelů do prostředí Unity. Z důvodu rozdílných souřadnicových systému, zejména orientaci os, musely být všechny modely otočeny.

Nedostatkem vyexportovaných modelů z Blenderu ve formátu .fbx je, že v prostředí Unity bylo zapotřebí znovu přiřadit a nastavit textury. Jednalo se také o zdlouhavý proces, kdy textury přiřazené k modelu musely být extrahovány v Unity a z nich poté mohly být vytvořeny výsledné materiály reprezentující texturu. V Unity ovšem nejde nastavit otočení dlaždice textury, proto se u některých modelů textura zdi jeví kolmo k zemi, místo aby byla vodorovná se zemí. Tento problém je ovšem záležitostí pouze Unity projektů, které jsou vytvářeny s cílem vygenerování mobilní aplikace. Počítačové aplikace dokáží využívat formát modelů .gltf, který v sobě má veškeré nastavení zakódované, a proto už pak není třeba v Unity cokoliv nastavovat.

Tvorba jednotlivých scén a jejich funkcionality probíhala smysluplně bez větších problémů. Nicméně menší komplikací byl typ modelů, se kterým jednotlivé části aplikace pracují. Bylo tedy zapotřebí otestovat, jestli dané části dokáží pracovat s .fbx modely nebo je třeba je přeuložit jako *Prefab*. Proto musely být přeuloženy modely využívané v částích **MARKER** a **EXTERIÉR**. Vytvoření skriptů uvádějících do chodu jednotlivá tlačítka a téměř celou část aplikace **PLANE** vyžadovalo naučení se základů programovacího jazyka C# i s pomocí fóra Unity anebo StackOverflow, kde je velmi aktivní komunita. V prostředí Visual Studio 2019 pak nebyl problém s hledáním chyb v podobě překlepů nebo absence jakéhokoliv typu závorky. Ovšem jakákoliv chyba v jednom skriptu okamžitě paralyzovala funkčnost ostatních použitých v Unity.

V části **MARKER** byla vytvořena sada 2D markerů v podobě QR kódu. Autor práce tuto formu doporučuje s ohledem na nenáročnost výpočtu na straně mobilního zařízení, tak i s ohledem na problémy identifikace 3D markeru. Nepříjemným překvapením pak bylo zjištění, kdy tato část nefungovala na mobilních zařízeních s operačním systémem iOS. Důvodem je uváděna velká výpočetní náročnost a nadlimitní zatížení CPU, což vyústilo k nerozpoznání markeru v návaznosti na nezobrazení modelu a v některých případech i pád aplikace. To vše je způsobeno závadou ve zdrojovém kódu iOS a SW XCODE. Z tohoto důvodu je aplikace dostupná pouze na zařízení s operačním systémem Android, kde zmíněná část funguje bez sebemenších problémů. Velkým usnadněním pak byl objev a využití knihovny **MARS**, ve které se nacházela veškerá funkcionalita ke splnění kritérií funkčnosti této části aplikace.

Část **PLANE** je založena da detekci ploch, na které je pak možné umístit zvolené modely návrší. Jak bylo uvedeno, funkcionalita této části je založená na C# skriptech a knihovně **ARFoundation**. Zde bylo jediným problémem mírně složitější ovládání ve finální aplikaci. Tlačítka se totiž chovají jako „průhledná“, to znamená že pokud má uživatel vybraný model k umístění na detekovanou plochu a míří na ní svým zařízením, při kliknutí na jakékoliv tlačítko ve scéně se model i tak umístí na detekovanou plochu. Z tohoto důvodu bylo do instrukcí v této části aplikace uvedeno, aby uživatel při změně modelu nebo vypínání/zapínání detekované plochy mířil mimo ni. Umístěné modely totiž není možné smazat a jediným východiskem pro jejich odstranění z detekované plochy je návrat na předchozí scénu nebo vypnutí aplikace. Dále autor práce doporučuje omezit detekování bílých ploch, a to z důvodu, že se jedná o reflektivní povrch, který kamera zařízení není z největší části schopna rozpoznat nebo ho rozpozná špatně.

V závěrečné části aplikace s názvem **EXTERIÉR** byla primárně využívána knihovna **Unity AR+GPS Location** a přidány byly také prvky z **ARFoundation**. Obě obsahují širokou funkcionalitu, nicméně největším problémem první zmíněné je, že se jedná o placenou knihovnu, kterou ovšem pro potřeby práce bylo zapotřebí zakoupit. Důvodem byla absence jakékoliv funkcionality, ať už přímo v Unity nebo pomocí instalovatelných knihoven. Zdlouhavým procesem bylo vybrání přesných GPS souřadnic tak, aby digitální model co nejvíce korespondoval s reálným objektem návrší. Ze stejného důvodu bylo také nutné modely zvětšit. Na to navázal další problém, kdy se zvětšením modelu došlo i k posunutí jeho polohy. Proto opět došlo k přepsání GPS souřadnic. Nedostatkem této části může být kolísavá kvalita GPS Signálu, která způsobuje menší posuny digitálního modelu a s tím související neúplně překrytí reálného objektu. V průběhu práce bylo upuštěno od možného řešení umístění modelu pomocí XML souboru, kdy model ani po zadání stejných souřadnic jako ve stávajícím řešení vůbec nekorespondoval s reálným objektem. Přístup pomocí XML souborů by tak bylo vhodnější použít při zobrazení více menších digitálních modelů, a ne při zobrazení jednoho velkého.

Export samotné aplikace do prostředí Android byl poměrně bezproblémový, což je možné říct i iOS aplikaci, která nefungovala kvůli zmíněnému bugu. Nedostatkem je, že testování samotné aplikace je opět zdlouhavý proces, kdy celou aplikaci je potřeba vyexportovat v případě Androidu do .apk a v případě iOS do SW XCODE a teprve poté je nainstalovat. Tento proces vždy zabral mezi 10-15 minutami času. Bylo by proto dobré, kdyby aplikaci šlo testovat bez exportu přímo v prostředí Unity prostým připojením mobilního zařízení.

I přes velké množství problémů je možné říct, že výsledná aplikace může být přínosem, a to z hlediska zvýšení atraktivity exhibice, tak zvýšením její interaktivity. I když se jedná o aplikaci, která nalezne využití pouze na jednom reálném objektu, stále je možnost využívat interiérové části **MARKER** a **PLANE**, ať je uživatel kdekoliv. K použití první ovšem potřebuje zmíněné markery, nicméně k použití druhé mu stačí prakticky jakákoliv plocha. Přestože je aplikace zaměřena na prezentaci Svatováclavského prostředí v AR, otevírá dveře k využití AR i na jiných historických a kulturních objektech, ale i v jiných vědních oborech s přesahem do soukromé sféry.

7 ZÁVĚR

Diplomová práce byla primárně zaměřena na analýzu potenciálu a aplikování geoinformačních technologií do tvorby interaktivní exhibice historických objektů pomocí technologie rozšířené reality. Hlavním cílem bylo vytvoření mobilní aplikace využívající AR. Dílčím cílem byla optimalizace historických modelů Svatováclavského návrší v Olomouci.

V první fázi došlo k nastudování potřebných materiálů. Následovalo sepsání současného stavu řešené problematiky, ve které byly rozebrány typy AR, historie AR, zařízení AR a SW pro tvorbu AR mobilních aplikací.

Řešení hlavního cíle začalo úpravou historických modelů návrší v prostředí Blenderu. Zde došlo nejprve k jejich úpravě v podobě ořezání částí z digitálního modelu reliéfu (DMR). Následně pak byly jednotlivé modely texturovány, neboť Blender nabízí širokou funkcionalitu pro nastavení jednotlivých textur. Došlo tedy k texturování těchto modelů, úpravě geometrie, změny jejich orientace a přípravě pro export do prostředí Unity. Zde byla stažena funkcionalita pro mobilní aplikaci v podobě knihoven podporujících zobrazení AR. Byly vytvořeny markery ve tvaru kulatého QR kódu s číslicí uprostřed udávající století, ze kterého zobrazovaný model je, a tyto markery pak byly vytištěny. Došlo také k vytvoření části aplikace, kde stažené knihovny a vytvořené C# skripty poskytují funkcionalitu pro detekci plochy a umístění historických digitálních modelů na ni. Závěrečnou částí aplikace je část Exteriér, kde za pomoci funkcionality dostupné ze zakoupené knihovny bylo možné umístit digitální model na GPS souřadnice a prezentovat tak změnu Svatováclavského návrší v průběhu století na reálném objektu návrší.

Výsledná aplikace je dostupná pro všechna mobilní zařízení s operačním systémem Android 10.0 a výše. Ke stažení je na webových stránkách diplomové práce a internetového obchodu pro platformu Android Google Play. Tato aplikace je hlavním výsledkem práce spolu s detailním popisem její tvorby, práce v Unity a optimalizací historických modelů. Aplikace může v budoucnu sloužit k popularizaci kultury a umění, a zároveň také například k výuce nebo jako výchozí bod pro budoucí bakalářské a diplomové práce.

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

Adobe. (2021). Adobe Creative Suite. https://www.adobe.com/.

Amin, D., & Govilkar, S. (2015). Comparative Study of Augmented Reality SDK's. International Journal on Computational Sciences & Applications, 5, 11-26. https://doi.org/10.5121/ijcsa.2015.5102.

AZUMA, R., B. HOFF, H. NEELY a R. SARFATY. A motion-stabilized outdoor augmented reality system. In: Proceedings IEEE Virtual Reality (Cat. No. 99CB36316) [online]. IEEE Comput. Soc, 1999, s. 252-259 [cit. 2021-8-12]. ISBN 0-7695-0093-5. Dostupné z: doi:10.1109/VR.1999.756959.

AZUMA, Ronald T. A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments [online]. 1997, 6(4), 355-385 [cit. 2021-8-12]. ISSN 1054-7460. Dostupné z: doi:10.1162/pres.1997.6.4.355.

Blender Online Community. (2021). Blender—A 3D modelling and rendering package. http://www.blender.org.

BORK, Felix, Leonard STRATMANN, Stefan ENSSLE, Ulrich ECK, Nassir NAVAB, Jens WASCHKE a Daniela KUGELMANN. The Benefits of an Augmented Reality Magic Mirror System for Integrated Radiology Teaching in Gross Anatomy. Anatomical Sciences Education [online]. 2019, 12(6), 585-598 [cit. 2021-8-12]. ISSN 19359772. Dostupné z: doi:10.1002/ase.1864.

BORKHATARYIA, Mansi. 2020 [cit. 2021-02-05]. 6 Types of Augmented Reality: Choose the Best for Your Business. Dostupné z WWW: < https://medium.com/@mansiborkhatariya/6-types-of-augmented-reality-choose-the-best-for-your-business-15798e9c43cd>.

BOSTANCI, Erkan, Nadia KANWAL a Adrian F CLARK. Augmented reality applications for cultural heritage using Kinect. Human-centric Computing and Information Sciences [online]. 2015, 5(1) [cit. 2021-8-12]. ISSN 2192-1962. Dostupné z: doi:10.1186/s13673-015-0040-3.

BYUNG-KUK SEO, JUNGSIK PARK a JONG-IL PARK. 3-D visual tracking for mobile augmented reality applications. In: 2011 IEEE International Conference on Multimedia and Expo [online]. IEEE, 2011, 2011, s. 1-4 [cit. 2021-8-12]. ISBN 978-1-61284-348-3. Dostupné z: doi:10.1109/ICME.2011.6012118.

Carmigniani, J. and Furht, B. (2011) Augmented Reality: An Overview. In: Furht, B., Ed., Handbook of Augmented Reality, Springer, New York, 3-46. https://doi.org/10.1007/978-1-4614-0064-6_1.

CodeAcademy. (2021). Code Academy. https://www.codecademy.com/.

DAMALA, A., HOULIER, P. and MARCHAL, I., 2007. Crafting the Mobile Augmented Reality Museum Guide. In: S. Richir (Editor), VRIC'07, 9th International Conference on Virtual Reality. IEEE, Laval, France, pp. 303-306.

Dtest [online]. 2020 [cit. 2021-02-05]. Dtest. Dostupné z WWW: https://www.dtest.cz/clanek-8011/muzea-a-muzejnictvi.

EDWARDS-STEWART, Amanda. 2016 [cit. 2021-02-05]. Classifying different types of augmented reality technology. Dostupné z WWW: https://www.researchgate.net/publication/315701832_Classifying_different_types_of_augmented_reality_technology.

FURHT, Borko, ed. Handbook of Augmented Reality [online]. New York, NY: Springer New York, 2011 [cit. 2021-8-12]. ISBN 978-1-4614-0063-9. Dostupné z: doi:10.1007/978-1-4614-0064-6.

HANAFI, Anasse, Lotfi ELAACHAK a Mohammed BOUHORMA. A comparative Study of Augmented Reality SDKs to Develop an Educational Application in Chemical Field. In: Proceedings of the 2nd International Conference on Networking, Information Systems & Security - NISS19 [online]. New York, New York, USA: ACM Press, 2019, 2019, s. 1-8 [cit. 2021-8-12]. ISBN 9781450366458. Dostupné z: doi:10.1145/3320326.3320386.

HEIN, Daniel W. E. a Philipp A. RAUSCHNABEL. Augmented Reality Smart Glasses and Knowledge Management: A Conceptual Framework for Enterprise Social Networks. ROSSMANN, Alexander, Gerald STEI a Markus BESCH, ed. Enterprise Social Networks [online]. Wiesbaden: Springer Fachmedien Wiesbaden, 2016, 2016-06-08, s. 83-109 [cit. 2021-8-12]. ISBN 978-3-658-12651-3. Dostupné z: doi:10.1007/978-3-658-12652-0_5.

CHATZOPOULOS, Dimitris, Carlos BERMEJO, Zhanpeng HUANG a Pan HUI. Mobile Augmented Reality Survey: From Where We Are to Where We Go. IEEE Access [online]. 2017, 5, 6917-6950 [cit. 2021-8-12]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2017.2698164.

KATIYAR, Anuroop. 2015 [cit. 2021-02-05]. Marker Based Augmented Reality. Dostupné z WWW: <https://www.krishisanskriti.org/vol_image/04Jul201511074828%20%20%20%20%20%20%20%20Anuroop%20Katiyar%20%20%20%20%20%20%20%20%20%20%20%20441-445.pdf>.

KATO, H. a M. BILLINGHURST. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99) [online]. IEEE Comput. Soc, 1999, s. 85-94 [cit. 2021-8-12]. ISBN 0-7695-0359-4. Dostupné z: doi:10.1109/IWAR.1999.803809.

LOFTI, R. Bowen a Jim X. CHEN. 2003 IEEE Virtual Reality Conference Guest Editors' Introduction. Presence: Teleoperators and Virtual Environments [online]. 2004, 13(2), iii-iv [cit. 2021-8-12]. ISSN 1054-7460. Dostupné z: doi:10.1162/1054746041382384.

MINE, Mark, ROSE, David, YANG, Bei. 2012 [cit. 2021-02-05]. Projection Based Augmented Reality in Disney Theme Parks. Dostupné z WWW: < https://web.cs.wpi.edu/~gogo/courses/cs525A/papers/Mine_2012_ProjectionAR.pdf>.

MOTA, José Miguel, Iván RUIZ-RUBE, Juan Manuel DODERO a Daniel MOLINA. LEARNING AUGMENTED REALITY IN THE CLASSROOM [online]. In: 2017, s. 8579-8582 [cit. 2021-8-12]. Dostupné z: doi:10.21125/iceri.2017.2332.

NORMAND, Jean-Marie, Myriam SERVIÈRES a Guillaume MOREAU. A new typology of augmented reality applications. In: Proceedings of the 3rd Augmented Human International Conference on - AH '12 [online]. New York, New York, USA: ACM Press, 2012, 2012, s. 1-8 [cit. 2021-8-12]. ISBN 9781450310772. Dostupné z: doi:10.1145/2160125.2160143.

NOWACKI, Paweł a Marek WODA. Capabilities of ARCore and ARKit Platforms for AR/VR Applications. ZAMOJSKI, Wojciech, Jacek MAZURKIEWICZ, Jarosław SUGIER, Tomasz WALKOWIAK a Janusz KACPRZYK, ed. Engineering in Dependability of Computer Systems and Networks [online]. Cham: Springer International Publishing, 2020, 2020-05-12, s. 358-370 [cit. 2021-8-12]. Advances in Intelligent Systems and Computing. ISBN 978-3-030-19500-7. Dostupné z: doi:10.1007/978-3-030-19501-4_36.

OLSSON, Thomas, Else LAGERSTAM, Tuula KÄRKKÄINEN a Kaisa VÄÄNÄNEN-VAINIO-MATTILA. Expected user experience of mobile augmented reality services: a user study in the context of shopping centres. Personal and Ubiquitous Computing [online]. 2013, 17(2), 287-304 [cit. 2021-8-12]. ISSN 1617-4909. Dostupné z: doi:10.1007/s00779-011-0494-x.

PAINE, James. 2018 [cit. 2021-02-05]. 10 Real Use Cases for Augmented Reality. Dostupné z WWW: < https://www.inc.com/james-paine/10-real-use-cases-for-augmented-reality.html>.

PITTMAN, Karin. 2012 [cit. 2021-02-05]. To Modernize a Museum. Dostupné z WWW: https://w2.uib.no/filearchive/to-modernize-a-museum.pdf.

PORTALÉS, Cristina, Jesús GIMENO, Sergio CASAS, Ricardo OLANDA a Francisco GINER MARTÍNEZ. Interacting With Augmented Reality Mirrors. MANAGEMENT ASSOCIATION, Information Resources, ed. Virtual and Augmented Reality [online]. IGI Global, 2018, s. 18-46 [cit. 2021-8-12]. ISBN 9781522554691. Dostupné z: doi:10.4018/978-1-5225-5469-1.ch002.

RAUSCHNABEL, Philipp A., Alexander BREM a Bjoern S. IVENS. Who will buy smart glasses? Empirical results of two pre-market-entry studies on the role of personality in individual awareness and intended adoption of Google Glass wearables. Computers in Human Behavior [online]. 2015, 49, 635-647 [cit. 2021-8-12]. ISSN 07475632. Dostupné z: doi:10.1016/j.chb.2015.03.003.

REGER, Greg. 2016 [cit. 2021-02-05]. Classifying different types of augmented reality technology. Dostupné z WWW: https://www.researchgate.net/publication/315701832_Classifying_different_types_of_augmented_reality_technology.

ROCHAT, Philippe a Dan ZAHAVI. The uncanny mirror: A re-framing of mirror self-experience. Consciousness and Cognition [online]. 2011, 20(2), 204-213 [cit. 2021-8-12]. ISSN 10538100. Dostupné z: doi:10.1016/j.concog.2010.06.007.

SAUER, Frank, Sebastian VOGT a Ali KHAMENE. Augmented Reality. PETERS, Terry a Kevin CLEARY, ed. Image-Guided Interventions [online]. Boston, MA: Springer US, 2008, 2008, s. 81-119 [cit. 2021-8-12]. ISBN 978-0-387-73856-7. Dostupné z: doi:10.1007/978-0-387-73858-1_4.

SEO, Jinseok, Namgyu KIM a Gerard J. KIM. Designing Interactions for Augmented Reality Based Educational Contents. PAN, Zhigeng, Ruth AYLETT, Holger DIENER, Xiaogang JIN, Stefan GÖBEL a Li LI, ed. Technologies for E-Learning and Digital Entertainment [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 2006, s. 1188-1197 [cit. 2021-8-12]. Lecture Notes in Computer Science. ISBN 978-3-540-33423-1. Dostupné z: doi:10.1007/11736639_149.

Unity3D. (2021). The leading platform for creating interactive, real-time content. https://unity.com/.

Unreal Engine Online. (2021). The world's most open and advanced real-time 3D creation tool. https://www.unrealengine.com/en-US/.

WITHER, Jason, Rebecca ALLEN, Vids SAMANTA, et al. The Westwood Experience: Connecting story to locations via Mixed Reality. In: 2010 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities [online]. IEEE, 2010, 2010, s. 39-46 [cit. 2021-8-12]. ISBN 978-1-4244-9339-5. Dostupné z: doi:10.1109/ISMAR-AMH.2010.5643295.

WAGNER, Daniel a Dieter SCHMALSTIEG. Making Augmented Reality Practical on Mobile Phones, Part 2. IEEE Computer Graphics and Applications [online]. 2009, 29(4), 6-9 [cit. 2021-8-12]. ISSN 0272-1716. Dostupné z: doi:10.1109/MCG.2009.67.

YUEN, Steve Chi-Yin, Gallayanee YAOYUNEYONG a Erik JOHNSON. Augmented Reality: An Overview and Five Directions for AR in Education. Journal of Educational Technology Development and Exchange [online]. 2011, 4(1) [cit. 2021-8-12]. ISSN 1941-8035. Dostupné z: doi:10.18785/jetde.0401.10.

PŘÍLOHY

SEZNAM PŘÍLOH

Vázané přílohy:

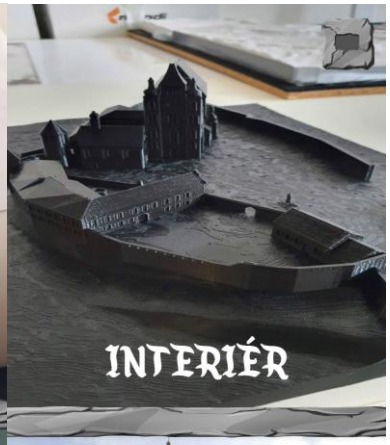
Volné přílohy

Elektronické přílohy

KATEDRA GEOINFORMATIKY
Univerzita Palackého v Olomouci

Made with unity

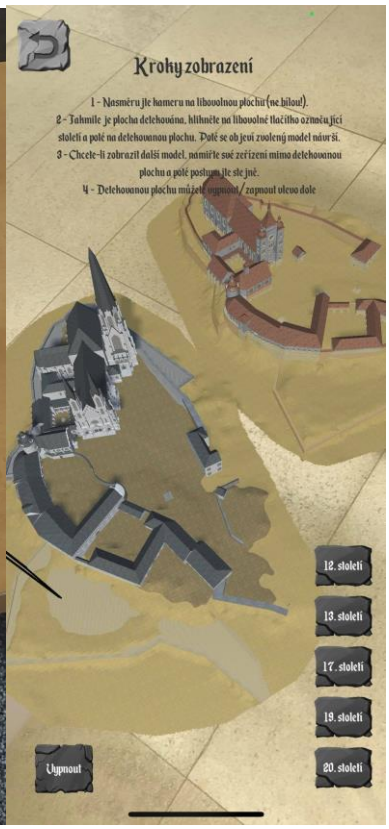
Příloha 1



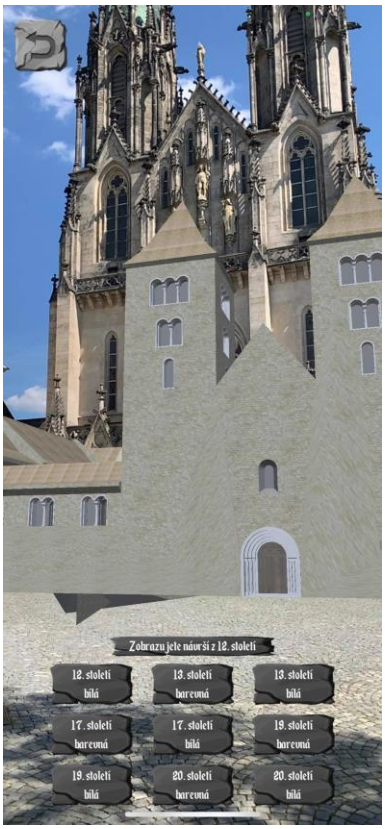
Příloha 2

Příloha 3

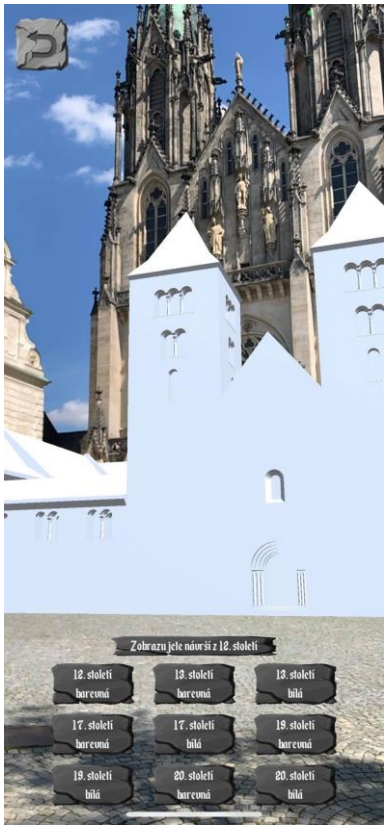
Příloha 4



Příloha 5

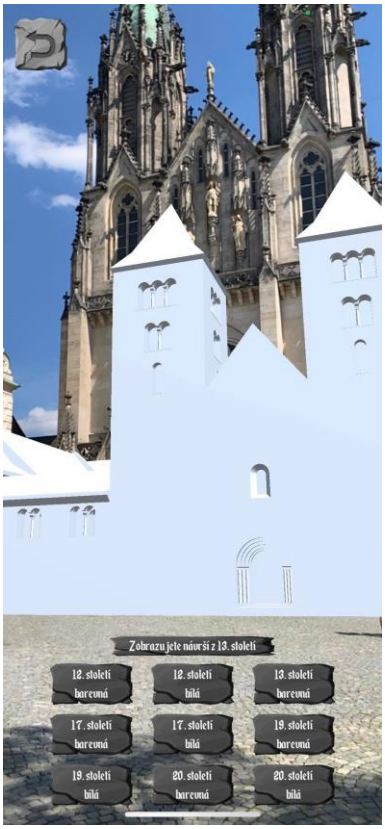


Příloha 6

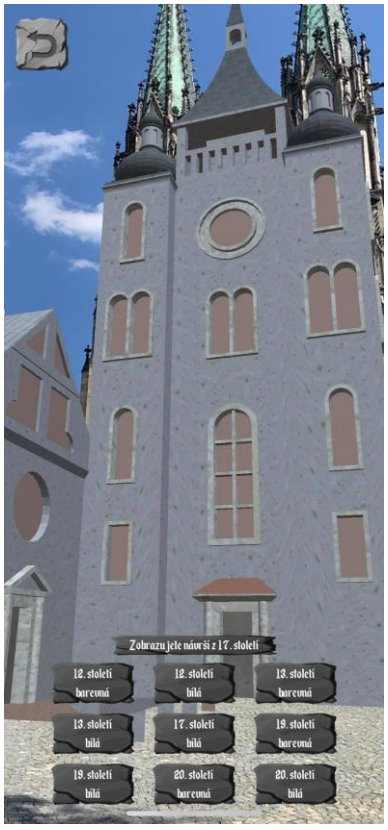


Příloha 7

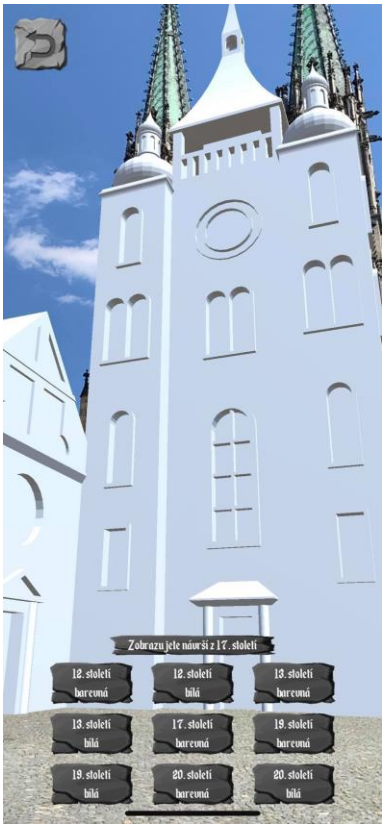
Příloha 8



Příloha 9



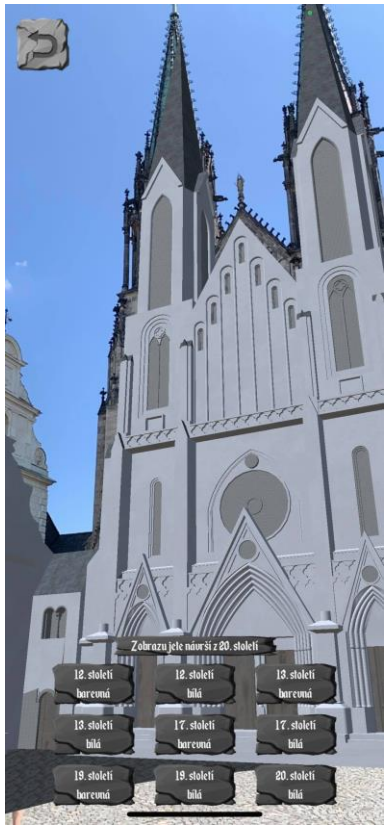
Příloha 10



Příloha 11

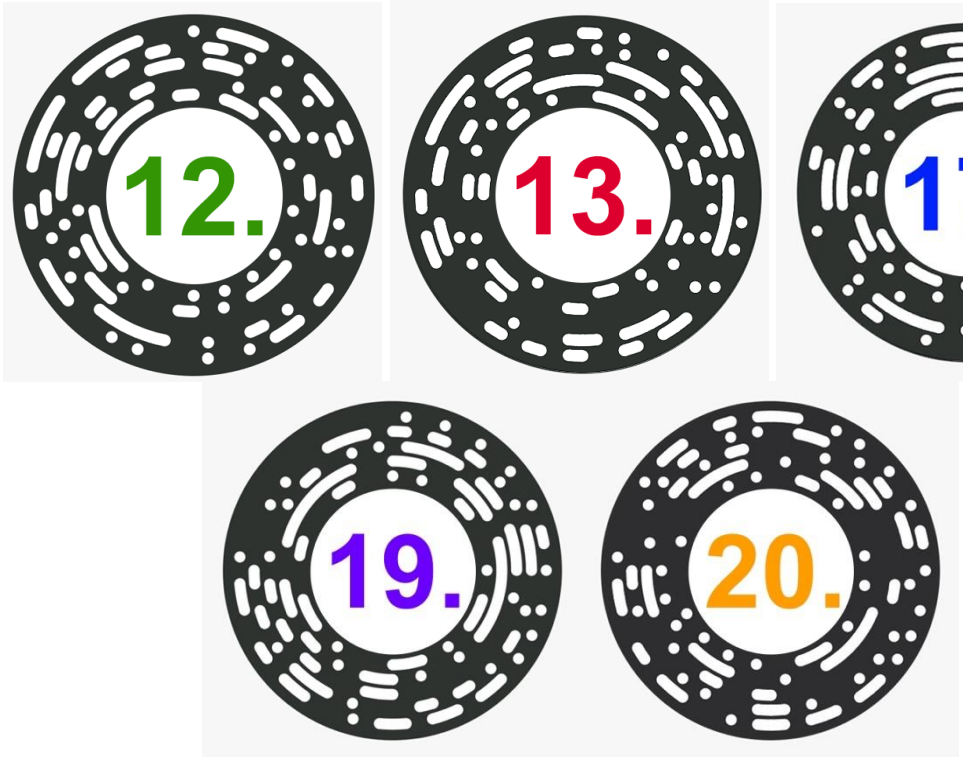
Příloha 12

Příloha 13



Příloha 14

Příloha 15



Příloha 16

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class StartScene1 : MonoBehaviour
{
    public void InterierScene()
    {
        SceneManager.LoadScene("InterierScene");
    }
    public void SampleScene()
    {
        SceneManager.LoadScene("SampleScene");
    }
    public void PlaneTracking()
    {
        SceneManager.LoadScene("PlaneTracking");
    }
    public void StartScene()
    {
        SceneManager.LoadScene("StartScene");
    }
    public void ExterierScene12st_barevna()
    {
        SceneManager.LoadScene("ExterierScene12st_barevna");
    }
    public void ExterierScene12st_bila()
    {
        SceneManager.LoadScene("ExterierScene12st_bila");
    }
    public void ExterierScene13st_barevna()
    {
        SceneManager.LoadScene("ExterierScene13st_barevna");
    }
    public void ExterierScene13st_bila()
    {
        SceneManager.LoadScene("ExterierScene13st_bila");
    }
    public void ExterierScene17st_barevna()
    {
        SceneManager.LoadScene("ExterierScene17st_barevna");
    }
    public void ExterierScene17st_bila()
    {
        SceneManager.LoadScene("ExterierScene17st_bila");
    }
    public void ExterierScene19st_barevna()
    {
        SceneManager.LoadScene("ExterierScene19st_barevna");
    }
    public void ExterierScene19st_bila()
    {
        SceneManager.LoadScene("ExterierScene19st_bila");
    }
    public void ExterierScene20st_barevna()
    {
        SceneManager.LoadScene("ExterierScene20st_barevna");
    }
    public void ExterierScene20st_bila()
    {
        SceneManager.LoadScene("ExterierScene20st_bila");
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PanelOpener : MonoBehaviour
{
    public GameObject Panel;

    public void OpenPanel()
    {
     if (Panel != null)
        {
            bool isActive = Panel.activeSelf;

            Panel.SetActive(!isActive);
        }

    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Popup : MonoBehaviour {
    public Canvas canvas;
    public bool a = false;
    public void popup()
{
    if (a == false)
    {
        a = true;
        canvas.enabled = true;
    }else if (a == true)
    {
        a = false;
        canvas.enabled = false;
    }
}
}
```

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

[RequireComponent(typeof(ARRaycastManager))]
public class SpawnObjectOnPlane : MonoBehaviour
{
    private ARRaycastManager raycastManager;
    private GameObject spawnedObject;
    private List<GameObject> placedPrefabList = new List<GameObject>();

    [SerializeField]
    private int maxPrefabSpawnCount = 0;
    private int placedPrefabCount;

    [SerializeField]
    private GameObject placeablePrefab;

    static List<ARRaycastHit> s_Hits = new List<ARRaycastHit>();

    private void Awake()
    {
        raycastManager = GetComponent<ARRaycastManager>();
    }
    bool TryGetTouchPosition(out Vector2 touchPosition)
    {
        if(Input.GetTouch(0).phase == TouchPhase.Began)
        {
            touchPosition = Input.GetTouch(0).position;
            return true;
        }
        touchPosition = default;
        return false;
    }
    private void Update()
    {
        if(!TryGetTouchPosition(out Vector2 touchPosition))
        {
            return;
        }
        if(raycastManager.Raycast(touchPosition, s_Hits, TrackableType.PlaneWithinPolygon))
        {
            var hitPose = s_Hits[0].pose;
            if(placedPrefabCount < maxPrefabSpawnCount)
            {
                SpawnPrefab(hitPose);
            }
            }
    }

    public void SetPrefabType(GameObject prefabType)
    {
        placeablePrefab = prefabType;
    }

    private void SpawnPrefab(Pose hitPose)
    {
        spawnedObject = Instantiate(placeablePrefab, hitPose.position, hitPose.rotation);
        placedPrefabList.Add(spawnedObject);
        placedPrefabCount++;
}
}
```

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

[RequireComponent(typeof(ARPlaneManager))]
public class PlaneDetectionToggle : MonoBehaviour
{
    private ARPlaneManager planeManager;
    [SerializeField]
    private Text toggleButtonText;

    private void Awake()
    {
        planeManager = GetComponent<ARPlaneManager>();
        toggleButtonText.text = "Vypnout";
    }

    public void TogglePlaneDetection()
    {
        planeManager.enabled = !planeManager.enabled;
        string toggleButtonMessage = "";

        if(planeManager.enabled)
        {
            toggleButtonMessage = "Vypnout";
            SetAllPlanesActive(true);
        }
        else
        {
            toggleButtonMessage = "Zapnout";
            SetAllPlanesActive(false);
        }

        toggleButtonText.text = toggleButtonMessage;

    }
    private void SetAllPlanesActive(bool value)
    {
        foreach(var plane in planeManager.trackables)
        {
            plane.gameObject.SetActive(value);
        }
    }
}
```