Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Zpracování obrazu z 360 stupňové kamery

Bakalářská práce

Autor: Jan Janás
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Bruno Ježek, Ph.D.

Hradec Králové duben 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne 26.4.2021 Jan Janás

Poděkování:
Děkuji vedoucímu bakalářské práce Ing. Bruno Ježkovi, Ph.D. za metodické vedení, přátelské konzultace a jeho povzbudivý přístup. Také chci poděkovat svým přátelům a spolužákům za jejich zpětnou vazbu k aplikaci, která vznikla v rámci této práce.

Anotace

Bakalářská práce se zabývá tématem zpracování obrazu z 360 stupňové kamery. V úvodní části práce je popsána historie panorama. Uvedeny jsou také různé způsoby jeho využití. Poté jsou představeny principy a metody záznamu panorama, stejně tak jako metody jeho zobrazení (2D reprezentace). V této části jsou také představeny technologie, které jsou využity v praktické části této práce. Praktická část se nejdříve zabývá zobrazením panorama v jednoduché scéně s použitím grafické knihovny OpenGL, kde jsou představeny dva způsoby mapování panorama na trojrozměrné těleso. Poté řešením rozsáhlejší prezentace reálného objektu formou webové aplikace využívající grafickou knihovnu WebGL. V závěru práce je zhodnocena tato implementace a je porovnána s dalšími alternativami.

Klíčová slova: Panorama, 360, vizualizace, 360° kamera, virtuální prohlídka

Annotation

Title: Panorama image processing

The bachelor thesis is aimed at processing of the image from the 360-degree camera. First part of this work consists of the History of the Panorama and panorama use cases. Followed by principles and methods of capturing panorama as well as methods for its 2D representation. This part also describes the technologies used in the practical part of this work. The practical part of this work solves two scenarios. First the display of the panorama in a simple scene using the OpenGL graphical library. Which presents two methods of panorama mapping onto a three-dimensional geometry. And second. Creating a web application for the representation of a real-world environment using the WebGL library. Resulting work is compared with other solutions.

Keywords: Panorama, 360, visualization, 360° camera, virtual tour

Obsah

1 Úvod 1

2 Motivace, cíl práce a metodika zpracování 2

2.1 Cíl práce 2

2.2 Metodika zpracování 2

3 Vývoj a použití panorama 3

3.1 Historie 3

3.2 Využití 5

3.2.1 Google Maps Street View 5

3.2.2 Virtuální prohlídky 5

3.2.3 Virtuální realita 6

3.2.4 Image based lighting 7

3.2.5 Produkční technologie LED Stage 8

4 Metody snímání, reprezentace a zobrazení panorama 10

4.1 Záznam panorama 10

4.1.1 Technika segmentů 10

4.1.2 Technika vypuklého zrcadla 11

4.1.3 Technika Scanner camera 12

4.1.4 Dual-fisheye lens 360° camera 13

4.1.5 Další metody záznamu 14

4.2 2D reprezentace panorama 17

4.2.1 Equirectangular 17

4.2.2 Cubemap 18

4.3 Technologie pro zpracování panoramatických snímků 19

4.3.1 Rozhraní pro programování aplikací (API) a grafické knihovny 20

4.3.2 Software pro tvorbu virtuálních prohlídek 23

5 Praktická část 25
5.1 Implementace zobrazení panorama v jednoduché scéně 25
5.2 Implementace zobrazení panorama v rozsáhlejší scéně 34
5.2.1 Pořízení snímků 35
5.2.2 Webová aplikace 36
5.2.3 Adresářová struktura 38
5.2.4 Obsluha panorama pomocí Three.js 39
6 Shrnutí výsledků 44
7 Závěry a doporučení 47
8 Seznam použité literatury 48
9 Přílohy 52

Seznam obrázků

Obr. 1 Panorama Londýna – Robert Barker Převzato z: [1] 3
Obr. 2 Ukázka realizace patentu Panorama Převzato z: [3] 4
Obr. 3 Google Maps Street View Převzato z: [6] 5
Obr. 4 Ukázka virtuální prohlídky Sixtinské kaple Převzato z: [8] 6
Obr. 5 Ukázka VR headsetu – Oculus Quest Převzato z: [9] 6
Obr. 6 Ukázka techniky IBL Převzato z: [12] 7
Obr. 7 LED Stage z produkce seriálu The Mandalorian Převzato z: [13] 8
Obr. 8 Ukázka techniky segmentů Převzato z: [14],[15] 11
Obr. 9 Ukázka techniky vypuklého zrcadla Převzato z: [17],[18] 12
Obr. 10 Ukázka techniky Scanner camera Převzato z: [19],[20] 13
Obr. 11 Ukázka techniky Dual Fishey lens 360° kamery Převzato z: [21] 14
Obr. 12 Ukázka techniky Chrome ball Převzato z: [23] 15
Obr. 13 Mobilní aplikace PhotoSphere Převzato z: [24] 15
Obr. 14 Ukázka HDRI Převzato z: [25] 16
Obr. 15 Sférické souřadnice Převzato z: [28] 17
Obr. 16 Equirectangular projekce Převzato z: [30] 18
Obr. 17 Cubemap projekce Převzato z: [33] 19
Obr. 18 Three.js – examples Převzato z: [40] 22
Obr. 19 Pano2VR prohlídka s navigací Převzato z: [41] 24
Obr. 20 SkyBox ve hře The Elder Scrolls III Převzato z: [43] 26
Obr. 21 Cubemap směrový vektor Převzato z: [43] 27
Obr. 22 OGLBuffer geometrie a topologie krychle Zdroj: Autor 28
Obr. 23 Načtení textur kubické mapy Zdroj: Autor 28
Obr. 24 Transformační matice Zdroj: Autor 29
Obr. 25 Vertex shader Zdroj: Autor 29
Obr. 26 Fragment shader Zdroj: Autor 30
Obr. 27 Ukázka implementace kubické mapy Zdroj: Autor 30
Obr. 28 Ukázka panorama ve formátu equirectangular Zdroj: Autor 31
Obr. 29 Grid buffer – geometrie gridu Zdroj: Autor 32
Obr. 30 IndexBuffer – topologie gridu Zdroj: Autor 33

Obr. 31 Vertex shader Zdroj: Autor 33
Obr. 32 Ukázka transformace plochy Zdroj: Autor 34
Obr. 33 Ukázka finální aplikace Zdroj: Autor 35
Obr. 34 Ukázka aplikace PixPro 360 VR Suite Zdroj: Autor 35
Obr. 35 KODAK PIXPR0 ORBIT360 4K VR kamera a její umístění na stojanu Zdroj: Autor | Převzato z: [46] 36
Obr. 36 Ukázka kódu – CSS Zdroj: Autor 37
Obr. 37 Plány poschodí FIM Převzato z: [49] 37
Obr. 38 Ukázka SVG plánu Zdroj: Autor 38
Obr. 39 Adresářová struktura projektu Zdroj: Autor 39
Obr. 40 Three.js – tvorba kamery Zdroj: Autor 40
Obr. 41 Three.js – vytvoření geometrie a materiálu Zdroj: Autor 40
Obr. 42 LoadingManager Zdroj: Autor 41
Obr. 43 Three.js – animační smyčka Zdroj: Autor 42
Obr. 44 Three.js – změna panorama Zdroj: Autor 43
Obr. 45 Snímek z kamery (vlevo nahoře), ekvirektangulární projekce (vpravo nahoře), kubická projekce (dole) Zdroj: Autor 44
Obr. 46 Ukázka mapování panorama na kouli Zdroj: Autor 45
Obr. 47 Ukázka finální aplikace Převzato z: [50] 46

1 Úvod

Tématem bakalářské práce je zpracování obrazu z 360° kamery. Tímto obrazem je panorama. Panorama je zobrazení, které nám umožňuje obklopit subjekt virtuálním prostředím a pomáhá nám se ponořit do iluze reality. Hlavní výhodou panoramatického zobrazení, oproti klasické fotografii, je možnost rozhlédnout se. Tím pozorovatel dostává větší a pochopitelnější kontext umožňující snazší orientaci v prostoru. Lze si lépe představit, jaké by to bylo ve skutečném světě, pokud bychom stáli v daném prostoru a ten nás obklopoval. Do virtuálního prostoru však nemusíme vsazovat jen sebe, můžeme jím obklopit i jiné subjekty, u nichž nás zajímá, jak by v kontextu takového prostředí vypadaly.

V teoretické části je představena historie panorama, možnosti jeho využití a nasazení. Dále jsou popsány metody pro záznam panorama včetně výhod a omezení jednotlivých metod, způsoby 2D reprezentace panorama a technologie pro zpracování panorama použité v rámci této práce.

V praktické části je popsána implementace zobrazení panorama pomocí volně dostupných grafických knihoven. Nejdříve v jednoduché scéně s použitím shaderů a knihovny OpenGL a poté v rozsáhlejší scéně s použitím webových technologií a knihovny WebGL formou virtuální prohlídky reálného objektu. V rámci implementace jsou zpracovávány vlastní panoramatické snímky pořízené 360° kamerou. Výstupem je webová virtuální prohlídka Fakulty informatiky a managementu UHK. V závěru jsou shrnuty získané poznatky a vlastní implementace je porovnána s komerčním řešením.

2 Motivace, cíl práce a metodika zpracování

Motivací k této práci je autorova předchozí zkušenost s využitím panoramatických snímků v oblasti 3D grafiky. Autor práce chtěl prozkoumat záznam, zpracování a využití panoramatických snímků a zároveň tak uplatnit nové znalosti osvojené při studiu Počítačové grafiky (PGRF), Programování (PRO) a Technologií pro publikování na webu (TNPW).

2.1 Cíl práce

Cílem práce je představit čtenáři 360° panorama, způsoby jeho využití. Popsat metody jeho záznamu a prezentace. Vysvětlit výhody těchto metod. Implementovat zobrazení panorama pomocí volně dostupných grafických knihoven.

Vyzkoušet tvorbu kompletní interaktivní vizualizace vybrané interiérové scény s možností zobrazení více 360° panoramat a přecházení mezi nimi pomocí mapy. Porovnat výsledky s komerčním řešením.

2.2 Metodika zpracování

Postup prací:

1. Prozkoumat principy a metody používané pro snímání panoramatických dat
2. Prozkoumat metody prezentace a využití panoramatických snímků
3. Vytvořit přehled používaných přístupů, nástrojů a dostupných softwarových řešení
4. Navrhnout a implementovat řešení využití panoramatického obrazu v jednoduché prostorové scéně
5. Navrhnout a implementovat řešení pro prezentaci rozsáhlejšího reálného objektu s využitím sad panoramatických snímků
6. Zhodnotit dosažené výsledky

3 Vývoj a použití panorama

První část textu se věnuje principu panoramatického zobrazení. Jsou popsány historické aspekty i současná využití v moderní době.

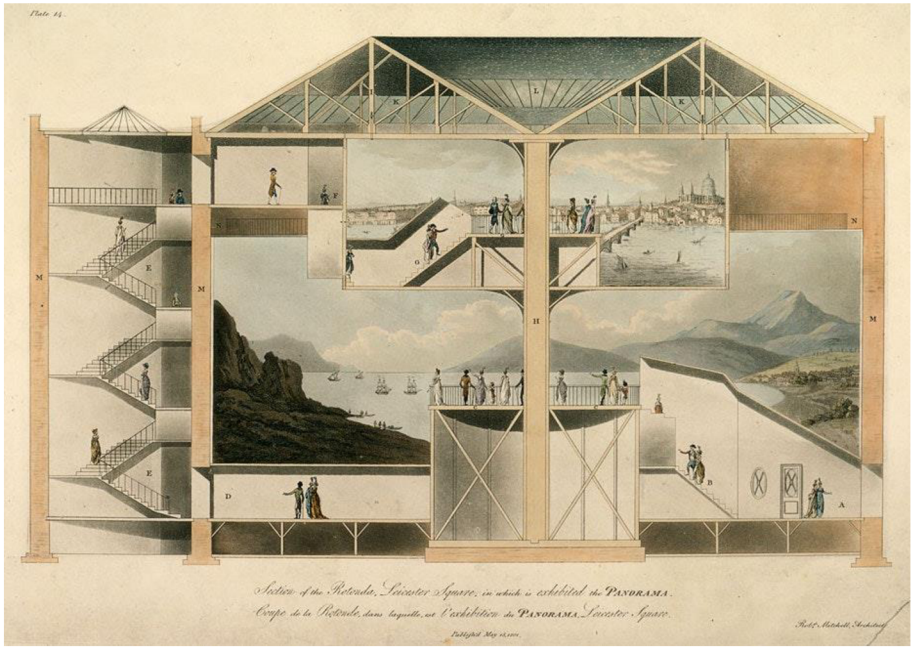
3.1 Historie

Panorama (z řeckého *pan – πᾶν* + *horama – ὅραμα* = view all) je širokoúhlá reprezentace fyzického prostoru. Definice slova vznikla v 18. století, když anglický malíř Robert Barker představil svůj patent.

V malbách se panorama objevovalo již od roku 20 našeho letopočtu, a to například na murálních malbách v Pompeiích. V době osvícenství pak v Evropě vznikaly kartografické experimenty s touto perspektivou. V polovině 19. století se panoramatické malby staly oblíbeným formátem pro zachycení krajiny, topografie a různých historických událostí. Diváci byli uchváceni iluzí obklopení, kterou přinesl patent Roberta Barkera z roku 1787.

Obr. 1 Panorama Londýna – Robert Barker
Převzato z: [1]

Barkerovo panorama (patent) byla budova s centrální kruhovou platformou, odkud bylo možné pozorovat 360° malbu na stěně rotundy obklopující pozorovatele. Cílem bylo obklopit diváka věrohodnou reprodukcí lokace, kde se může rozhlížet do všech stran a připadat si, jako by na daném místě skutečně byl. Nešlo tedy o nový druh zobrazení, ale o nový způsob prezentace a interakce diváka s obrazem. Příkladem panorama splňující podmínky Barkerova patentu, které se dodnes dochovalo, je Panorama Mesdag v Hágu v Nizozemí. [2]



Obr. 2 Ukázka realizace patentu Panorama
Převzato z: [3]

Panoramatická fotografie se brzy stala nejpoužívanější metodou k zachycení panorama. Fotografové začali skládat jednotlivé nafocené snímky vedle sebe, čímž vznikal jeden široký snímek. Dnes je již funkce snímání panorama běžná ve všech fotoaparátech i mobilních telefonech.

Stejně jako panoramatická malba je VR panorama 360° zobrazením prostoru či události. Navíc oproti malbě může sloužit jako rozhraní pro interakci s digitálním světem a lze jej kombinovat s dalšími formami médií (hudba, obraz, video). Dalším posunem k důvěryhodnosti a pocitu obklopení byl příchod headsetů pro virtuální realitu. Zařízení jako Oculus Rift a HTC Vive zaměřené na herní průmysl kombinují stereoskopii s 360° prostředím. [4]

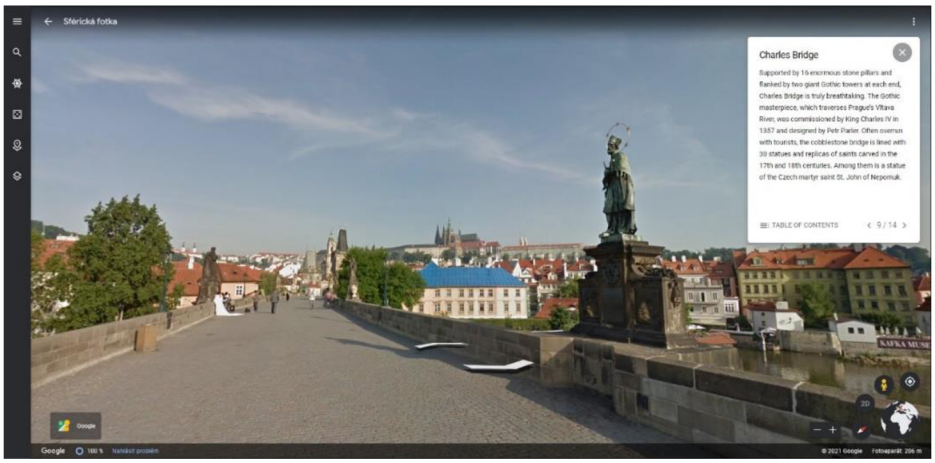
Postupem času se technika Virtual reality (VR) panoramatická fotografie stala oblastí vědy a umění, zabývající se tvorbou interaktivních navigovatelných obrazů. Tyto obrazy obvykle zobrazují nějaký prostor, který obklopuje pozorovatele a umožňuje mu tak ponořit se do virtuálního prostoru. Počátky moderní VR fotografie můžeme přiřadit ke vzniku nástroje Quicktime VR Authoring Tools a softwaru Quicktime Virtual Reality (QTVR) od společnosti Apple Computers představených začátkem roku 1990 pro tvorbu a zobrazení digitálního panorama na displeji. [4]

3.2 Využití

Většina uživatelů se již setkala se sférickým panoramatem na internetu. Virtuální prohlídky dnes nabízejí realitní kanceláře, zoologické zahrady, galerie, muzea, kulturní památky a jiné. Následující kapitoly představují hlavní oblasti nasazení této vizualizační techniky.

3.2.1 Google Maps Street View

Příkladem může být služba Google Maps Street View, která umožňuje uživatelům zvolit si místo na mapě a umístit se do pozice pozorovatele. Takto se může uživatel rozhlížet po ulici, jako by se na ní fyzicky nacházel. Automobily vybavené 360° kamerou jezdí po celém světě a pořizují série panoramatických záběrů. Další sensory zajišťují, aby se panorama správně přiřadila k přesné GPS lokaci. Obsah je tvořen jak společností Google, tak řadou přispěvatelů. [5]

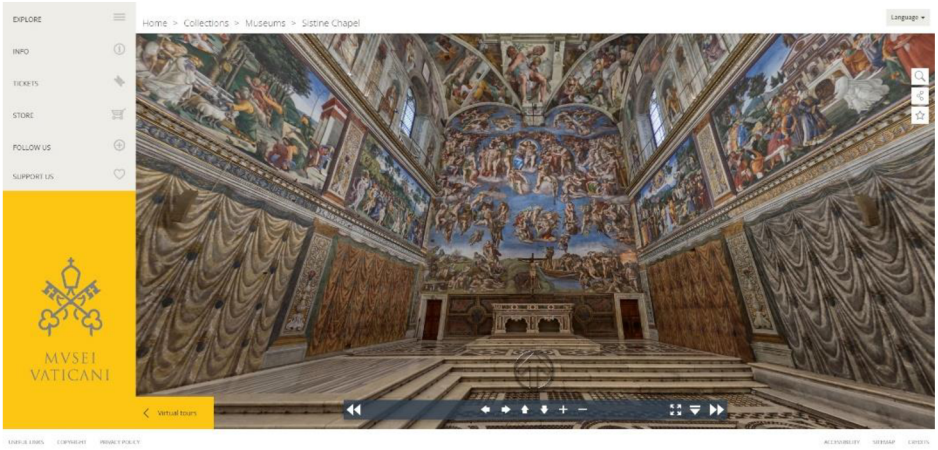


Obr. 3 Google Maps Street View
Převzato z: [6]

3.2.2 Virtuální prohlídky

Díky virtuálním prohlídkám si můžou potencionální zákazníci prohlédnout nemovitosti, byty, restaurace, fitness centra, či jiné nabízené prostory. Můžou tak učinit kdykoliv bez potřeby přesunu na dané místo. Zákazníci získají dostatečnou představu z pohodlí svého domova. Virtuální prohlídky dosáhly většího rozmachu díky aktuální situaci způsobené virem SARS-CoV-2. Mezinárodní karanténa

znemožnila fyzickou přítomnost návštěvníků mnoha institucí a tyto instituce tak využily možnosti virtuálních prohlídek. [7]



Obr. 4 Ukázka virtuální prohlídky Sixtinské kaple
Převzato z: [8]

3.2.3 Virtuální realita

Sférické panorama má své využití také ve světě VR. Ať už jde o zmíněné virtuální prohlídky, kde je středem zájmu pozorovatele, nebo slouží jen jako pozadí v různých hrách a aplikacích, jeho účel je stále stejný. Pomáhá uživateli ponořit se do virtuálního prostředí. Nemusí však být pouze focené, sférické prostředí může být také počítačově generované (CG).

Obr. 5 Ukázka VR headsetu – Oculus Quest
Převzato z: [9]

Cena VR headsetu a dostatečně výkonného počítače je pro většinu populace stále vysoká. Objevují se ale na trhu i nové headsety, které již nevyžadují připojení k počítači, nebo velkou místnost pro umístnění sensorů, které sledují pohyb headsetu a ovladačů. Další bariérou pro tuto technologii je, že některým uživatelům způsobuje používání headsetu nevolnost. To se však zlepšuje díky novým displejům s vyšší obnovovací frekvencí a menší prodlevou. [10]

3.2.4 Image based lighting

Image based lighting (IBL) je technika osvětlení v počítačové 3D grafice, kde se jako zdroj světla používá sférické panorama s vysokým dynamickým rozsahem (HDR). Využití má hlavně ve filmovém průmyslu, kde je potřeba zasadit 3D objekt do reálného prostředí. HDR obraz je namapován na sféru či kupoli, která pak simuluje osvětlení objektu ve scéně. To umožňuje nasvícení scény vysoce detailním osvětlením zachyceným v reálném světě. [11]

Obr. 6 Ukázka techniky IBL
Převzato z: [12]

IBL techniku využívají také herní enginy (Unity, Unreal Engine), aplikace pro tvorbu textur a Look Developement (Mari, Substance Painter/Designer, Marmoset Toolbag, Quixel Suite) a další 3D grafické aplikace.

3.2.5 Produkční technologie LED Stage

Industrial Light & Magic (ILM) a Epic Games (tvůrce softwaru Unreal Engine), společně s partnery produkčních technologií Fuse, Lux Machina, Profile Studios, NVIDIA, a ARRI, odhalili nové filmové paradigma. Ve spolupráci s Golem Creations Jona Favreaua vytvořili seriál The Mandalorian pomocí nové virtuální produkce. Tato technologie umožní filmařům zachytit podstatné množství trikových scén přímo na kameru s využitím technologie real-time game enginu a LED panelů prezentujících dynamické fotorealistické digitální prostředí s kreativní flexibilitou, která byla dříve nepředstavitelná.

Obr. 7 LED Stage z produkce seriálu The Mandalorian
Převzato z: [13]

Více než polovina první série The Mandalorian byla natočena s využitím této technologie. Byla eliminována potřeba shánět lokace pro natáčení, místo toho herci performovali před masivní polokruhovou LED video stěnou a stropem. Praktické kulisy byly zkombinovány s digitálním pozadím. Digitální 3D prostředí, vytvořené firmou ILM se interaktivně přehrávalo na LED stěnách a bylo v reálném čase editováno během natáčení, což umožňovalo na pixel přesný tracking a perspektivně korektní 3D obraz, renderování ve vysokém rozlišení pomocí systému poháněného grafickými kartami (GPUs) od firmy NVIDIA. Prostředí bylo osvětleno a renderováno z perspektivy kamery, což generovalo paralax v reálném čase, jako

kdyby kamera skutečně snímala fyzické prostředí s přesným interaktivním osvětlením herců a kulis. Tato flexibilita umožňovala tvůrcům dělat kreativní rozhodnutí během produkce a vidět změny kompozice přímo v kameře. [13]

4 Metody snímání, reprezentace a zobrazení panorama

V této části textu jsou představeny postupy pro záznam panorama, způsoby 2D reprezentace nasnímaných dat a způsoby jejich zobrazení. Představeny jsou také technologie pro použití panoramatických snímků.

4.1 Záznam panorama

Existuje několik metod, jak zachytit sférické panorama. Tyto metody používají různé typy kamer, objektivů, stativů a postupu focení. Metody se liší kvalitou výstupu, cenou za potřebné vybavení a časem, který zabere snímání.

4.1.1 Technika segmentů

Technika segmentů zaznamenává okolní prostředí po segmentech. Vyžaduje konvenční kameru, širokoúhlý objektiv a stativ s panoramatickou hlavou. Kamera je připevněna na panoramatické hlavě, která umožňuje rotaci ve dvou osách (horizontální a vertikální). Kamera musí být upevněna tak, aby se tyto osy rotace protínaly s uzlovým bodem objektivu (optický střed). Tímto se eliminuje paralax efekt. Velikost segmentu je určena úhlem, o který se kamera otáčí. Ten by měl být zvolený tak, aby se okraje pořízených snímku dostatečně překrývaly. Počet segmentů pak závisí na zvoleném objektivu. Při menší ohniskové vzdálenosti bude pokryta větší plocha prostoru, a tak bude počet segmentů menší než u větší ohniskové vzdálenosti. Menší počet segmentů však zároveň znamená menší rozlišení finálního výstupu.

Obr. 8 Ukázka techniky segmentů
Převzato z: [14],[15]

Technika je časově náročnější, ale výstupem je kvalitní panorama ve vysokém rozlišení. Použití fotoaparátu také umožňuje snímání více expozicí (bracketing) pro výstup s vysokým dynamickým rozsahem (HDR). Ke spojení snímků (stitching) je potřeba speciální software. Využívá se hlavně v oblasti virtuálních prohlídek, kde je vysoké rozlišení klíčové. [16]

4.1.2 Technika vypuklého zrcadla

Technika vypuklého zrcadla vyžaduje konvenční kameru a speciální vypuklé zrcadlo. Kamera je umístěna vertikálně a místo standartního objektivu je osazena speciální optikou s vypuklým zrcadlem. Prostředí se v zrcadle odráží jako na vánoční baňce. Toto zrcadlo může být parabolické, hyperbolické či sférické. Pořízený snímek zrcadla s černou dírou uprostřed a prostředím odraženým kolem středu se poté převede pomocí softwaru do finální podoby. Metoda zaznamenává 360 horizontálních stupňů, je však limitovaná vertikálně a nezabírá tedy kompletní prostor. Jako jediná však nevyžaduje žádnou další úpravu, jako je sešívání snímků nebo odstranění artefaktů a je tedy nejrychlejší metodou. Má však velmi nízké rozlišení a je vhodná pouze pro náhledy prostředí. [16]

Obr. 9 Ukázka techniky vypuklého zrcadla
Převzato z: [17],[18]

4.1.3 Technika Scanner camera

Technika Scanner camera vyžaduje speciální kameru na motorizované vertikální ose. Má sensor s lineárním polem s počtem pixelů 5000-7000 na výšku a jedním pixelem na šířku. Kamera snímá vertikální pixely a postupně se otáčí podél vertikální osy. Snímání probíhá delší dobu a není vhodné pro proměnlivé prostředí. Takto pořízené panorama nevyžaduje další postprodukci. Jde o speciální kameru, jejíž cena se pohybuje kolem 50 000 EUR. [16]

Obr. 10 *Ukázka techniky Scanner camera*
Převzato z: [19],[20]

4.1.4 Dual-fisheye lens 360° camera

Mezi nejdostupnější zařízení pro záznam sférického panorama patří kompaktní zařízení s dvěma čočkami s malým ohniskem zvaným rybí oko (Dual-fisheye lens camera). Klasická metoda pro spojení snímků z více kamer se zakládá na vyhledávání stejných prvků mezi dvěma snímky a jejich následné transformaci do stejné plochy pomocí homografie. Zmíněná metoda však není pro tato zařízení vhodná, jelikož oblast překrytí mezi okraji těchto čoček je velmi malá. Proto většina výrobců těchto kamer nabízí vlastní proprietární software pro konverzi výstupního obrazu, nebo se tato konverze provede přímo v kameře.

Obr. 11 *Ukázka techniky Dual Fishey lens 360° kamery*
Převzato z: [21]

Výstup nemá tak velké rozlišení jako metoda segmentů a v oblast překrytí dvou čoček může mít nechtěné artefakty. Kvalita těchto kamer však neustále stoupá a jejich snadné použití z nich dělá vhodnou alternativu pro nasazení, kdy na metodu segmentů není dostatek času. Například pro snímání osvětlení na lokacích natáčení pro pozdější IBL osvětlení. Tento typ kamery je velmi versatilní a nabízí uživatelsky nejsnadnější cestu k pořízení 360° panorama. Oproti ostatním zmíněným metodám je tato kamera také schopna záznamu 360° videa. [22]

4.1.5 Další metody záznamu

Technika Chrome ball

Technika Chrome ball je velmi dostupná metoda pro vlastníky DSLR fotoaparátu je technika focení lesklé koule. Může se použít profesionální chromovaná koule nebo také čistě obyčejná lesklá vánoční baňka. Zrcadlově lesklý povrch koule je pak potřeba nafotit ze dvou stran tak, aby snímání pokrylo obě polokoule. Je vhodné fotit více expozic pro HDR výstup. Nasnímané polokoule je potřeba rozbalit (unwrap) pomocí speciálního softwaru, jako například HDRShop a spojit jejich přesah. Jde o složitější manuální postup, který vyžaduje znalost softwaru jako Adobe Photoshop. [23]

Obr. 12 Ukázka techniky Chrome ball
Převzato z: [23]

Mobilní aplikace

Fotoaparáty v mobilních telefonech již svojí kvalitou dohánějí klasické fotoaparáty. Panorama je už běžnou funkcí každé fotografické aplikace. Existují ale také aplikace pro snímání sférického panorama. Fungují na principu techniky segmentů a využívají gyroskop telefonu k uspořádání snímků. Výstup má omezený dynamický rozsah a je náchylný na artefakty mezi přesahujícími snímky. Je to však nejdostupnější technika a s dobrým telefonem lze dosáhnout použitelných výsledků.

Obr. 13 Mobilní aplikace PhotoSphere
Převzato z: [24]

High Dynamic Range Imaging (HDR/I)

Další technika, která se využívá při záznamu panorama, se nazývá HDRI. Používá se při zpracování a zobrazení obrazu či videa s širokým rozsahem intenzity mezi nejtmavšími a nejsvětlejšími místy v obraze. Běžné techniky záznamu obrazu ukládají informace o pixelu jedním bajtem na barevný kanál (256 bitů x 3 kanály). V nerovnoměrně osvětlené scéně pak může tato limitace vést k přeexponování příliš světlých míst (slunce, světla) nebo k podexponování v tmavých částech (stíny). Oproti tomu technologie HDR ukládá obraz v plovoucích desetinných jednotkách (floating-point). HDR je nezbytné například při tvorbě panorama pro osvětlovací techniku IBL popsanou výše.

Obr. 14 *Ukázka HDRI*
Převzato z: [25]

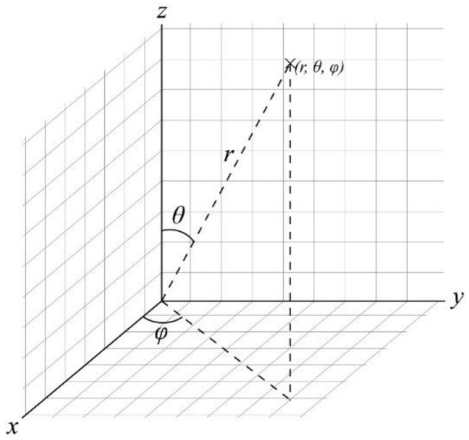
Pro záznam HDR obrazu je třeba snímat scénu s různým nastavením expozice. Kamery dnes již často disponují funkcí sekvenčního snímání více expozic (Auto-exposure bracketing). Výstupem může být 8bit obraz zkombinovaný z pořízených expozic, nebo 16/32bit obraz ve speciálním formátu HDR jako je Radiance (.hdr), OpenEXR (.exr), floating point (.tiff,.psd). [26]

V poslední době se technologie HDR objevuje i v zobrazovacích zařízeních. Technika pro zobrazení obrazu s vysokým dynamickým rozsahem na zařízení s menším rozsahem bez ztráty detailů se nazývá Tonemapping. Využívá adaptivní matematickou funkci, která přepočítá hodnoty všech pixelů tak, aby se vešly

do výstupního LDR barevného prostoru Existuje řada algoritmů pro převod HDR na LDR (linear, logarithmic, reinhard). [27]

4.2 2D reprezentace panorama

K plochému zobrazení 360° panorama se používá sférická projekce. V počítačové grafice lze využít různé plochy pro namapování, jako například krychle (kubická mapa) nebo koule (ekvirektangulární mapa).



Obr. 15 *Sférické souřadnice*
Převzato z: [28]

Pro popsání jakéhokoliv bodu na ploše sféry se používají sférické souřadnice (r;theta;phi). Kde r je vzdálenost bodu od počátku sféry. Úhel θ udává vertikální odklon od osy Z $\langle 0°; 180°\rangle$ úhel φ, který udává horizontální odklon od osy X $\langle 0°; 360°\rangle$. [29]

4.2.1 Equirectangular

Projekce, která mapuje sférickou plochu na plochý obraz, kde horizontální souřadnice jsou zeměpisnou délkou a vertikální souřadnice jsou zeměpisnou šířkou, bez jakýchkoliv dalších transformací. Projekce byla vytvořena kolem roku 100 našeho letopočtu Marinusem z Týru a od jejího vzniku se používala při tvorbě map světa. Vertikální linky zůstávají vertikální a horizont se stane přímkou procházející středem obrazu. Souřadnice obrazu odpovídají lineárně úhlům posunu a náklonu z reálného světa. Póly jsou umístěné v horní a spodní části obrazu a jsou roztažené přes celou šířku obrazu. Oblasti v blízkosti pólů jsou horizontálně roztažené.

Obr. 16 *Equirectangular projekce*
Převzato z: [30]

Ekvirektangulární projekce je výchozím formátem výstupu rotačního panorama. Poměr stran obrazu je 2:1. Projekce se také běžné používá v aplikacích a nástrojích pro zobrazení 360° panorama. [31]

4.2.2 Cubemap

Kubická projekce mapuje část sférické plochy na plochý obraz. Obrazy jsou poskládané jako stěny krychle, každá s rectilineární (přímočarou) projekcí. Krychle je pozorována z jejího středu. Čtyři plochy krychle pokrývají přední, pravou, zadní a levou, jedna pro zenit a jedna pro nadir. Každá obraz pokrývá zorné pole v rozsahu 90x90 úhlových stupňů. Každá plocha zachovává přímé přímky, což je vhodné pro snadnou interpolaci. Kubické mapy se využívají v mnoha aplikacích pro zobrazení 360° panorama a také v herních enginech. [32]

Obr. 17 Cubemap projekce
Převzato z: [33]

V grafických knihovnách, jako je například OpenGL, je kubická mapa přístupná pomocí vektorů vyjádřených jako 3D texture coordinates (S, T, R). Tyto souřadnice jsou automaticky vypočítány pomocí hardwaru. Mezi nejčastější využití kubických map patří mapování prostředí (environment mapping), které se projeví jako odlesky na lesklých objektech. Jejich generování je snadné, a probíhá v reálném čase. Na pozici lesklého objektu, který bude odrážet své okolí, je na základě paprsku od kamery a normály povrchu vypočten odražený paprsek. Podle směru tohoto paprsku je určena barva texelu kubické textury. Výsledkem je dynamicky generovaná mapa prostředí. Dalším využitím je před-výpočet spekulárního či difuzního osvětlení (Illumination map/Diffuse lighting). Zde funguje kubická mapa jako vektorová funkce, která vrací RGBA hodnotu. Kubické mapy většinou není potřeba aktualizovat pro každý realtime renderovaný snímek. [34]

4.3 Technologie pro zpracování panoramatických snímků

Pro zobrazení panorama jsou využívány různé technologie, jako grafické knihovny, programovatelná rozhraní a speciální grafický software.

4.3.1 Rozhraní pro programování aplikací (API) a grafické knihovny

V implementační části této práce jsou použity API OpenGL a WebGL. Pro snadnější práci s těmito API jsou pak použity knihovny LWJGL (pro OpenGL) a Three.js (pro WebGL). V této části jsou tyto API a knihovny popsány.

OpenGL

Grafická knihovna OpenGL je nejrozšířenější adoptovaný grafický standart pro vývoj přenosných, interaktivních 2D a 3D grafických aplikací. Představen byl v roce 1992 společností Silicon Graphics, Inc. a od roku 2006 je spravován společností Khronos Group. Od té doby se stal nejpoužívanějším a nejpodporovanějším API. Vznikly díky němu tisíce aplikací pro různé platformy. OpenGL je inovativní a urychluje vývoj aplikací díky mnoha účinným vizualizačním funkcím pro manipulaci obrazu jako rendering, mapování textur, speciální efekty a další. Vývojáři mohou pracovat na různých platformách jak pro stolní počítače (Windows, Mac OS), tak pro pracovní stanice (Linux). OpenGL jasně specifikuje co má být výsledkem/výstupem jeho funkcí. Jejich implementace se však může u každého vývojáře (NVidia, ARM, Intel, AMD) lišit, ale musí splňovat danou specifikaci.

Aplikace vyžadující maximální grafický výkon, jako 3D animace, simulace, CAD/CAM/CAE, zábavní průmysl, lékařské vizualizace a virtuální realita mohou využít výkon a kvalitu, kterou OpenGL nabízí. Díky své architektuře, specifikaci a široké podpoře v řadě odvětví, je OpenGL skutečně otevřený a nezávislý standart. Jeho aktualizace jsou předem oznámeny a schvalovány. Zpětná kompatibilita zaručuje chod starších aplikací. Aplikace jsou konzistentní mezi kompatibilním hardwarem nezávisle na operačním systému.

Architektura OpenGL umožňuje pomocí API přístup k hardwarovým inovacím, čímž pomáhá prodejcům hardwaru a vývojářům softwaru implementovat nové funkce. Většinou vývojářů jsou výrobci grafických karet. Různé řady karet podporují specifickou verzi OpenGL. Mezi hlavní konkurenty OpenGL patří knihovna Direct3D od firmy Microsoft, který jako grafická platforma vznikl v roce 1995.

API OpenGL má logické a intuitivní příkazy, které usnadňují práci programátorů. Dostupná je i kvalitní dokumentace, spousta publikací a ukázek kódu. OpenGL rutiny zjednodušují vývoj grafického softwaru. Od renderování jednoduchých geometrických primitiv až po komplexní osvětlenou a otexturovanou scénu. OpenGL zpřístupňuje funkce osvětlení, texturování, transformace a další funkce.

OpenGL je stavovým strojem. Kolekce proměnných, které definují, jak má OpenGL pracovat. Jeho stav se označuje jako „OpenGL context“. Při používání OpenGL tento stav měníme nastavením možností, manipulací bufferů a poté renderujeme s použitím aktuálního kontextu. Kdykoliv změníme stav OpenGL změnou kontextu, dalším vykreslením se tato změna aplikuje.

Mezi podporované programovací jazyky patří C, C++, Fortran, Ada, a Java. Všechny varianty mají stejnou specifikaci a jsou testovány. Samotné knihovny OpenGL jsou psány v jazyce C. [35, 36]

Lightweight Java Game Library

LWJGL je knihovna v jazyce Java, která umožňuje meziplatformní přístup k populární nativním API, užitečným k vývoji aplikací zaměřených na grafiku (OpenGL, Vulkan), zvuk (OpenAL) a paralelní výpočty (OpenCL). Umožňuje přímý nízko úrovňový přístup, a to přes uživatelsky přívětivé Java rozhraní. Knihovna je open-source s licencí BSD. [37]

Web Graphics Library

WebGL je cross-platformní, volně dostupný webový standard pro low-level 3D grafické API založený na OpenGL Embedded Systems (ES). Využívá ECMAScript k zobrazení v HTML5 canvas elementu a rozhraní DOM. Sémantika je věrná OpenGL ES specifikaci. Větší důraz je kladený na management paměti pro jazyky jako JavaScript. WebGL přináší pro web volně dostupnou 3D grafiku bez nutnosti použití jakýchkoliv pluginů. Implementována přímo v prohlížeči. Hlavní prohlížeče jako Chrome, Edge, Safari a Mozilla jej podporují. WebGL je vyvíjen a spravován organizací Khronos Group. Využívá jej například společnost Google ve své aplikaci Google Maps. [38]

Knihovna Three.js

Three.js je cross-platformní 3D JavaScriptová knihovna a aplikační rozhraní (API), které umožňuje co nejsnadnější implementaci 3D obsahu na webu. K vykreslování 3D grafiky využívá WebGL. Zatímco WebGL je velmi nízko úrovňové, Three.js poskytuje snadné rozhraní. [39]

Obr. 18 Three.js – examples
Převzato z: [40]

Struktura Three.js se skládá z několika objektů. Hlavním objektem je Renderer. Tomu jsou jako argumenty předány Scéna a Kamera. Renderer poté vyrenderuje část 3D scény, která je uvnitř zorného pole (frustum) kamery, jako 2D obraz do canvas elementu. Scéna se skládá z objektů jako Mesh, Light, Group, Object3D a Camera. Objekty mají stromovou hierarchickou strukturu (Rodič/Potomek). Mesh objekt reprezentuje specifickou geometrii se specifickým materiálem. Jak materiál, tak geometrie mohou být použity u více Mesh objektů zároveň. Geometry objekt reprezentuje data o vertexech určité geometrie jako koule, krychle, auto atd. Geometrii lze také načíst ze souboru. Material objekt reprezentuje vlastnosti povrchu, který se vykreslí na geometrii. Jde o vlastnosti jako barva, lesk atd. Materiál může také používat textury. Texture objekt representuje obrazy načtené ze souboru, generované z canvasu nebo renderované ze scény. Light objekt reprezentuje různé druhy světel.

4.3.2 Software pro tvorbu virtuálních prohlídek

Pro tvorbu virtuálních prohlídek existuje řada komerčních i volně dostupných aplikací. Tento software vyžaduje poskytnutí hotových panoramatických snímků a nabízí nástroje na tvorbu interaktivních prohlídek. Mezi tyto nástroje patří prvky uživatelského rozhraní jako jsou body zájmu (hotspots), navigační šipky atd.

Výhodou takového softwaru jsou již předpřipravené ovládací prvky, snadná manipulace, jsou uživatelsky přívětivé a umožňují export prohlídky ve spustitelném formátu, například HTML5. Nevýhodou mohou být finanční náklady a to, že možnosti přizpůsobení jsou většinou omezené na předpřipravené šablony a není snadné dosáhnout vzhledu, který by se lišil od ostatních prohlídek.

Pano2VR

Pano2VR je nástroj k tvorbě virtuálních prohlídek, který konvertuje panorama, 360° fotografie a videa na interaktivní zážitek. Umožňuje sestavit samostatné gigapixelové panorama, nebo virtuální prohlídku s tisícovkou scén. Hotový projekt lze exportovat pro každý moderní prohlížeč, integrovat jej na již existující webovou stránku a zobrazit jej na mobilu, stolním počítači, nebo VR headsetu.

Pano2VR Pro umožňuje propojení panoramatických scén pomocí uzlů a přechodů. Poskytuje rozhraní pro navigaci, díky které se uživatel může pohybovat mezi scénami pomocí kliknutí či dotyku (tap). Nabízí kontext lokace v podobě zabudované mapy prohlídky nebo plánem půdorysu či přímé integrace s Google Maps.

Obr. 19 Pano2VR prohlídka s navigací
Převzato z: [41]

Přímo z aplikace je možné z importovaných snímků extrahovat zvolenou oblast a tu poté bez problémů upravit v libovolné grafickém editoru. Odstranit z nich stativ nebo jiné nechtěné nedokonalosti. Po uložení této retuše jsou pak změny aplikovány na panorama. Aplikace navíc umožňuje přidání interaktivní elementů, jako jsou informační vyskakovací okna (popups), fotografické hotspoty, směrový zvuk a video. K výběru má uživatel množství předpřipravených vzhledů (skins), ale je možné tyto šablony dále upravovat pomocí Skin Editoru. Vytvořit vlastní tlačítka a ovládací prvky pro unikátní design rozhraní, a to vše responzivně. Zvýraznění významných scén pomocí animovaný cest (walk-through), které automaticky navigují uživatele scénami.

Projekt je možné vyexportovat ve formátu HTML5 a nahrát jej přímo na webový server. Nebo pomocí pluginu přímo na WordPress stránky. Program také podporuje WebVR pro tvorbu plně obklopující VR prohlídku pro web. Pano2VR je schopné publikovat prohlídky také přímo do Google Street View.

Pano2VR je k dispozici k vyzkoušení s omezeným počtem scén a vodoznakem. Pro plnou funkcionalitu je nutné zakoupit licenci. Program nabízí dvě verze. Pano2VR 6 light – cena jedné licence je 149.00 euro (bez DPH). Pano2VR 6 pro – cena jedné licence je 349.00 euro (bez DPH). [41]

5 Praktická část

Pro odzkoušení principů zpracování a zobrazení panorama byly navrženy dvě úlohy. První z nich demonstruje zpracování jednoho panoramatického snímku na nízké úrovni s využitím programovatelného řetězce knihovny OpenGL. Jsou testovány metody mapování textury a manipulace s objektem ve scéně. Druhá úloha je zaměřena na komplexnější využití panorama k demonstraci interaktivního zobrazení rozsáhlejší indorové scény.

5.1 Implementace zobrazení panorama v jednoduché scéně

Pro zobrazení panorama v jednoduché scéně je zde použito API programovatelného zobrazovacího řetězce knihovny OpenGL, které je popsáno v předchozí sekci. K implementaci byl použitý programovací jazyk Java a grafická knihovna LWJGL. Panorama se bude zobrazovat namapováním textury na povrch tělesa pomocí programovatelných shaderů. Návrh implementace vychází z ukázek použití shaderových programů využívaných v rámci výuky předmětu PGRF3 v navazujícím magisterském studiu oboru Aplikovaná informatika na FIM UHK.

Programovatelné shadery

Shadery jsou malé programy, které běží na grafické kartě, a to během specifických sekcí grafické pipeline. Zjednodušeně řečeno shader je program, který mění vstupy na výstupy. Zároveň shader představuje část hardwaru grafické karty, na kterém je shaderový program spouštěn. Shadery jsou izolované a nijak mezi sebou nekomunikují, pouze skrze jejich vstupy a výstupy. Shadery v OpenGL jsou psané v jazyce OpenGL Shading Language (GLSL). Je to jazyk podobný jazyku C, který je uzpůsobený pro použití při práci s grafikou a obsahuje užitečné funkce speciálně zaměřené na manipulaci s vektory a maticemi. Shadery začínají deklarací verze, následované řadou vstupních a výstupních proměnných, proměnných typu „uniform“ a hlavní funkce „main“. Tato funkce zpracovává vstupy a jako výsledek vrací výstup. [42]

Existuje více druhů shaderů, kromě vertex a fragment shaderu také tesselation, geometry a compute shader. V rámci navržené implementace budou

použity jen první dva základní. Prvním z nich je vertex shader, který provádí operace nad každým vstupním vrcholem (vertexem). Těmito operacemi jsou transformace souřadnic vrcholů a normál, výpočet souřadnic do textury, osvětlení a barvy. Výstupem jsou souřadnice vrcholu (gl_Position) a další atributy vrcholu. Dalším použitým shaderem je fragment shader. Tento shader provádí operace nad pixely interpolovanými mezi vrcholy. Aplikuje textury a přistupuje k jejich hodnotám. Nastavuje výstupní barvu a průhlednost. Výstupem je barva (gl_FragColor) včetně alpha kanálu a Z hodnota hloubky (gl_FragDepth).

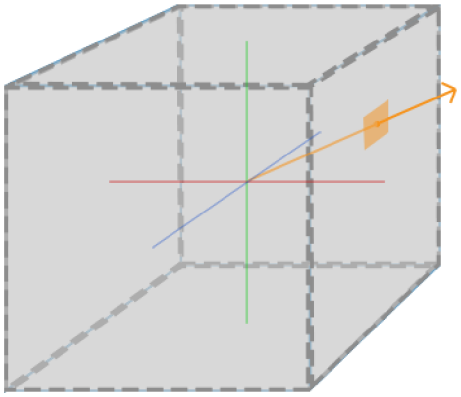
Mapování na krychli (Skybox)

Prvním příkladem mapování na povrch tělesa je mapování na krychli. Své využití má hlavně v herních enginech, kde krychle slouží jako skybox. Skybox je objekt tvaru krychle, který obklopuje celou scénu a na jehož stěny je mapováno šest obrazů okolního prostředí. Při pohledu ze středu krychle dává pozorovateli iluzi, že se nachází uvnitř prostředí, které je daleko rozsáhlejší, než je ve skutečnosti. Příkladem často využívaných skyboxů jsou obrazy hor, oblak a hvězdné oblohy. [43]

Obr. 20 SkyBox ve hře The Elder Scrolls III
Převzato z: [43]

Pro mapování na krychli se používá kubická textura, která obsahuje šest individuálních 2D textur, které tvoří strany krychle. Její užitečnou vlastností je, že ji lze indexovat/vzorkovat pomocí směrového vektoru. Počátek tohoto vektoru leží v centru krychle. Na velikosti vektoru nezáleží, stačí nám směr. OpenGL poté získá

korespondující texely, se kterými se vektor protnul a vrátí vzorkovanou hodnotu textury.



Obr. 21 Cubemap směrový vektor
Převzato z: [43]

Protože jsou pro testování implementace použity snímky z 360° dual fishey lens kamery, které jsou ve formátu equirectangular, je potřeba je převést na kubickou mapu. Pro převod snímků použitých v této implementaci byla použita webová aplikace Panorama to Cubemap [45]. Výstupem je šest snímku představujících strany krychle.

K zobrazení kubické textury pomocí knihovny LWJGL je nejdříve potřeba vytvořit krychli, na kterou je poté textura namapována pomocí fragment shaderu. Krychle je definována pomocí geometrie (vertex buffer) a topologie (index buffer). V případě LWJGL je geometrie zapsána do pole „cube“ podle schéma atributů specifikovaných v atributech „OGLBuffers.Attrib“. Těmito atributy jsou trojrozměrné souřadnice vrcholu (inPosition) a vektor normály (inNormal). Topologie je zapsána do pole „indexBufferData“. Toto pole „cube“ spolu se schématem atributů a polem indexů jsou předány jako argumenty OGLBufferu.

```
void createBuffers() {
    float[] cube = {
            // bottom (z-) face
            1, 0, 0,    0, 0, -1,
            0, 0, 0,    0, 0, -1,
            1, 1, 0,    0, 0, -1,
            0, 1, 0,    0, 0, -1,
            // top (z+) face
            1, 0, 1,    0, 0, 1,
            0, 0, 1,    0, 0, 1,
            1, 1, 1,    0, 0, 1,
            0, 1, 1,    0, 0, 1,
            // x+ face
            1, 1, 0,    1, 0, 0,
            1, 0, 0,    1, 0, 0,
            1, 1, 1,    1, 0, 0,
            1, 0, 1,    1, 0, 0,
            // x- face
            0, 1, 0,    -1, 0, 0,
            0, 0, 0,    -1, 0, 0,
            0, 1, 1,    -1, 0, 0,
            0, 0, 1,    -1, 0, 0,
            // y+ face
            1, 1, 0,    0, 1, 0,
            0, 1, 0,    0, 1, 0,
            1, 1, 1,    0, 1, 0,
            0, 1, 1,    0, 1, 0,
            // y- face
            1, 0, 0,    0, -1, 0,
            0, 0, 0,    0, -1, 0,
            1, 0, 1,    0, -1, 0,
            0, 0, 1,    0, -1, 0
    };

    int[] indexBufferData = new int[36];
    for (int i = 0; i<6; i++){
        indexBufferData[i*6] = i*4;
        indexBufferData[i*6 + 1] = i*4 + 1;
        indexBufferData[i*6 + 2] = i*4 + 2;
        indexBufferData[i*6 + 3] = i*4 + 1;
        indexBufferData[i*6 + 4] = i*4 + 2;
        indexBufferData[i*6 + 5] = i*4 + 3;
    }
    OGLBuffers.Attrib[] attributes = {
            new OGLBuffers.Attrib( name: "inPosition",  dimension: 3),
            new OGLBuffers.Attrib( name: "inNormal",  dimension: 3)
    };

    buffers = new OGLBuffers(cube, attributes, indexBufferData);
    System.out.println(buffers.toString());
}
```

Obr. 22 OGLBuffer geometrie a topologie krychle
Zdroj: Autor

Dále je potřeba načíst kubickou texturu. V tomto případě jde o šest samostatných snímků, každý pro jednu stěnu krychle. Textury jsou předány objektu „OGLTextureCube“ jako jednotlivé „textureID“. O zbytek se postarají shadery.

```
// texture files are in /res/textures/
String[] names = {"textures/px.png",
        "textures/nx.png",
        "textures/py.png",
        "textures/ny.png",
        "textures/pz.png",
        "textures/nz.png"};


textureViewer = new OGLTextureCube.Viewer();
textRenderer = new OGLTextRenderer(width, height);
```

Obr. 23 Načtení textur kubické mapy
Zdroj: Autor

Vertex shaderu jsou jako vstupy předány informace o vrcholech „inPosition“. S vrcholy není třeba manipulovat, proto defaultní výstup vertex shaderu

„gl_Position“ nese nezměněné souřadnice vrcholů vynásobené transformační maticí zahrnující modelovací, pohledovou i projekční transformaci.

```
glUniformMatrix4fv(locMat,  transpose: false,
        ToFloatArray.convert((new Mat4Transl( x: -0.5, y: -0.5, z: -0.5))
                .mul(new Mat4RotX( alpha: Math.PI/2))
                .mul(cam.getViewMatrix()).mul(proj)));
```

Obr. 24 Transformační matice
Zdroj: Autor

Výstupem je pak „vertPosition“, který je ve fragment shaderu použit pro mapování kubické textury. Hodnota vektoru „vertPosition“ se přepočítává z pozice vrcholů krychle, na kterou je následně mapována krychlová textura. Vzhledem k tomu, že pozorovatele umísťujeme do středu krychle je nutné vektory přepočítat tak, aby směřovaly ze středu k jednotlivým vrcholům. Od pozice vrcholu (x, y, z) odečteme vektor pozice středu krychle (0.5,0.5,0.5) a celý výraz vynásobíme dvěma. Tyto výstupy převezme fragment shader jako své vstupy.

```
#version 150
in vec3 inPosition;
out vec3 vertPosition;
uniform mat4 mat;
void main() {
    vertPosition = 2*inPosition.xyz-1.0;
    gl_Position = mat * vec4(inPosition, 1.0);
}
```

Obr. 25 Vertex shader
Zdroj: Autor

Uvnitř fragment shaderu je pomocí kubického sampleru „samplerCube“ a vektoru „vertPosition“ (získaných přepočtem z pozice vrcholů „inPosition“) určen odpovídající texel kubické textury. Fragment shader vrací výstupní barvu „outColor“.

```
#version 150
in vec3 vertPosition;
out vec4 outColor;
uniform samplerCube textureID;
void main() {
    outColor = vec4(texture(textureID, vertPosition).rgb,1.0);
}
```

Obr. 26 Fragment shader
Zdroj: Autor

Tím je zobrazení kubické mapy na krychli kompletní. Stačí umístit kameru do středu krychle a pozorovat tak Skybox zevnitř.

Obr. 27 Ukázka implementace kubické mapy
Zdroj: Autor

Mapování na kouli (SkySphere)

Druhým příkladem mapování panoramatického snímku na geometrii je mapování na kouli. Kromě tvaru tělesa se liší od předchozí metody také typem 2D reprezentace panorama. V tomto případě jde o ekvirektangulární zobrazení.

Obr. 28 Ukázka panorama ve formátu equirectangular
Zdroj: Autor

Ačkoli se metoda jmenuje „Mapování na kouli“, základní geometrií bude v tomto případě plocha. Na tuto plochu bude namapováno ekvirektangulární panorama a plocha s texturou bude pomocí vertex shaderu deformována do tvaru koule. Prvním krokem je tedy vytvoření mřížky (grid) vrcholů, resp. trojúhelníků, které budou tvořit tuto plochu. Aby nedošlo k vizuálním artefaktům, je potřeba vytvořit mřížku s dostatečným počtem trojúhelníkových segmentů. Čím více bude mít plocha segmentů, tím hladší bude povrch koule.

Plocha má rozměry v rozsahu [0-1] v osách X, Y a [0] v ose Z. Parametr „segmentCount“ značí počet segmentů plochy. Tedy počet vrcholů v osách X, Y. Parametr „stepSize“ počítá vzdálenost mezi jednotlivými vrcholy. Pro každý vrchol je potřeba vypočítat souřadnice pozice „inPosition“ a souřadnice do textury „inTextureCoordinates“. Vrcholy jsou vytvořeny po řádcích. Pozice vrcholů nabývají hodnoty (x,y,z) postupně od [0,0,0] do [1,1,0]. Zatímco souřadnice textury (u,v) nabývají hodnoty od [0,1] do [1,0]. Normála je pro všechny stejná [0,0,1].

```
void createBuffers() {
    //quadStrip squareGrid
    int segmentCount = 50;
    float stepSize = 1f/(segmentCount-1);
    float xPos = 0;
    float yPos = 0;
    float uCoord = 0;
    float vCoord = 1;
    int gridAttrib = segmentCount*segmentCount*8;
    float[] grid = new float[gridAttrib];
    for (int i = 0; i < gridAttrib; i +=segmentCount*8) {
        for (int j = 0;j<segmentCount*8;j+=8){
            //per vertex attributes
            grid[i+j] = xPos; //xPos
            grid[i+j+1] = yPos; //yPos
            grid[i+j+2] = 0; //zPOS
            grid[i+j+3] = 0; //xNor
            grid[i+j+4] = 0; //yNor
            grid[i+j+5] = 1; //zNor
            grid[i+j+6] = uCoord; //uTexCoord
            grid[i+j+7] = vCoord; //vTexCoord
            yPos += stepSize;
            uCoord += stepSize;
        }
        yPos = 0;
        uCoord = 0;
        xPos += stepSize;
        vCoord -=stepSize;
    }
```

Obr. 29 Grid buffer – geometrie gridu
Zdroj: Autor

V dalším kroku je potřeba sestavit index buffer. V tomto případě je plocha složena z horizontálních pásů trojúhelníků „GL_TRIANGLE_STRIP“. Indexy pásu tedy vznikají vždy mezi dvěma řádky vrcholů nad sebou zleva doprava. Po dosažení konce řádku je do index bufferu vložen speciální index, který značí, že další index v pořadí je začátkem nového primitiva stejného typu. Další pás tak pokračuje na začátku dalšího řádku. Ještě je potřeba uvnitř fragment shaderu vypočítat výstupní barvu. Jde o podobný postup jako u předchozí metody, liší se pouze typem sampleru. Zde stačí obyčejný „sampler2D“, kterému jsou předány souřadnice do textury „texCoord“. Tím je plocha s texturou hotová.

```
glEnable(GL_PRIMITIVE_RESTART);
glEnable(GL_PRIMITIVE_RESTART_INDEX);
glPrimitiveRestartIndex(65535);
```

```
    OGLBuffers.Attrib[] attributes = {
            new OGLBuffers.Attrib( name: "inPosition",  dimension: 3),
            new OGLBuffers.Attrib( name: "inNormal",  dimension: 3),
            new OGLBuffers.Attrib( name: "inTextureCoordinates",  dimension: 2) };

    int[] indexBufferData = new int[(2*segmentCount*segmentCount)-(segmentCount+1)];
    int index = 0;
    for (int i = 0; i < indexBufferData.length; i+=2) {
        indexBufferData[i] = index;
        indexBufferData[i+1] = index+segmentCount;
        index += 1;
        if(index%segmentCount==0){
            indexBufferData[i+2]=65535;
            i+=1;
        }
    }
    // create geometry with index buffer as the quad strip
    buffers = new OGLBuffers(grid, attributes, indexBufferData);
    System.out.println("buffers \n " + buffers.toString());
}
```

Obr. 30 IndexBuffer – topologie gridu
Zdroj: Autor

Zbývá už jen skrze vertex shader transformovat vrcholy z kartézských souřadnic plochy do sférických (polárních) souřadnic koule. Uvnitř vertex shaderu se nová pozice vrcholu „gl_Position“ vypočítává pomocí následujícího vzorce.

$$x = r * \sin\theta \cos\sigma$$

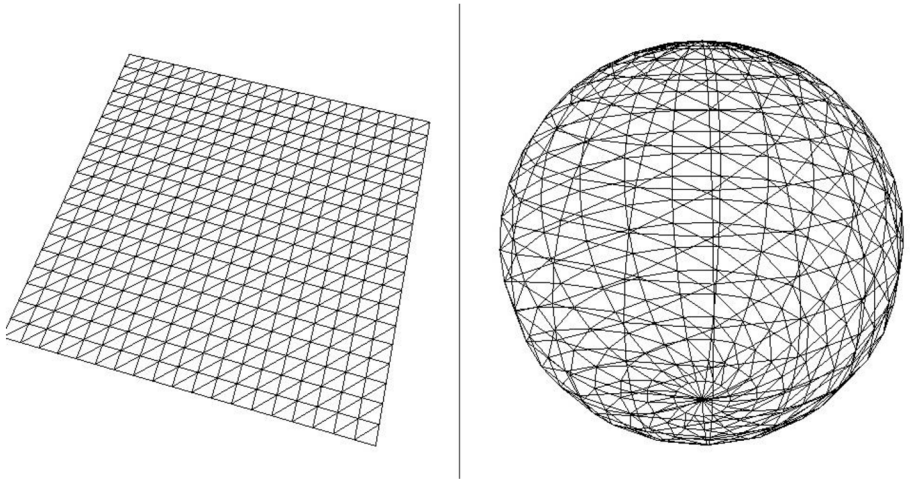
$$y = r * \sin\theta \cos\sigma$$

$$\mathrm{z} = \mathrm{r} * \cos\theta$$

```
#version 330
in vec3 inPosition;
in vec2 inTextureCoordinates;
out vec2 texCoord;
uniform mat4 mat;
const float PI = 3.14159265358979323846264338327950;

void main() {

    float x = -sin(inPosition.x*PI)*cos(inPosition.y*PI*2);
    float y = sin(inPosition.x*PI)*sin(inPosition.y*PI*2);
    float z = -cos(inPosition.x*PI);

    gl_Position = mat * vec4(x,y,z, 1.0);
    texCoord = inTextureCoordinates;
}
```

Obr. 31 Vertex shader
Zdroj: Autor

Kde „r“ je poloměr koule. V tomto případě je roven jedné. Théta (Θ) je úhel zenitu v rozsahu [0-π]. Ve vertex shaderu je zapsán jako π násobek hodnoty „inPosition.x“. Sigma (φ) je azimutní úhel v rozsahu [0-2π]. Ve vertex shaderu je zapsán jako 2π násobek hodnoty „inPosition.y“. Výstupem vertex shaderu je pozice vrcholu s nově vypočítanými sférickými souřadnicemi „gl_Position“ a souřadnice do textury „texCoord“.

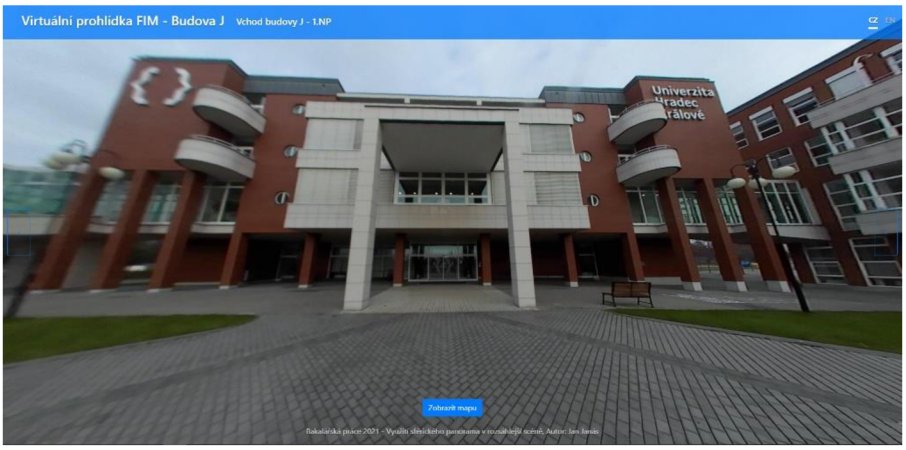


Obr. 32 Ukázka transformace plochy
Zdroj: Autor

Na závěr stačí umístit kameru do středu koule a prohlédnout si panorama namapované na kouli zevnitř.

5.2 Implementace zobrazení panorama v rozsáhlejší scéně

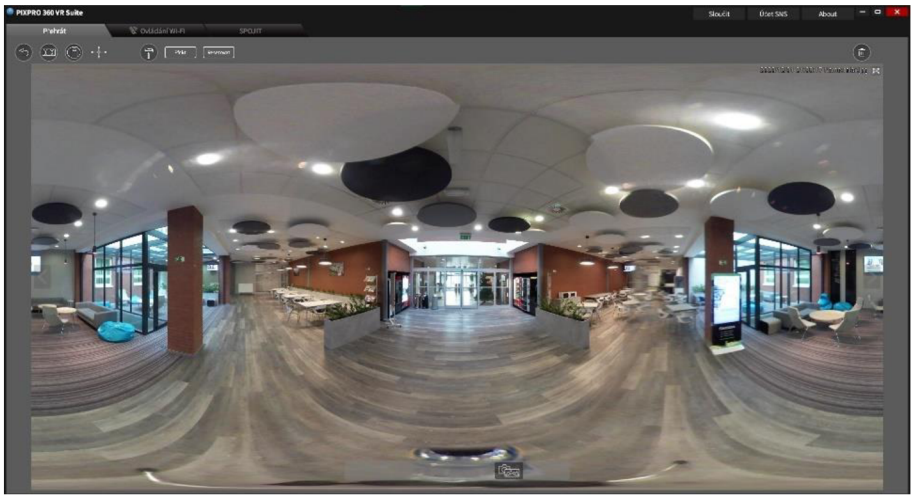
Pro zobrazení rozsáhlejší scény vzniknul v rámci této práce návrh webové aplikace, která by mohla sloužit jako prezentace fakulty Informatiky a Managementu Univerzity Hradec Králové (FIM UHK). Aplikace umožňuje prohlídku místností (učebny, kafeterie, recepce…) na fakultě, a to s využitím 360° fotografií. Aplikace je interaktivní, responzivní a nevyžaduje žádné externí pluginy. Využívá volně dostupnou knihovnu Three.js s licencí MIT, která slouží jako framework pro knihovnu WebGL. Aplikace běží na prohlížeči jak desktopového, tak mobilního zařízení, a může ji tak spustit jakýkoliv běžný uživatel bez nutnosti cokoliv dalšího instalovat. Výhodou je, že knihovna WebGL používá hardwarovou akceleraci zobrazení a je tedy maximálně využit dostupný grafický výkon zařízení.



Obr. 33 Ukázka finální aplikace
Zdroj: Autor

5.2.1 Pořízení snímků

Pro pořízení snímků byla použita 360° kamera KODAK PIXPR0 ORBIT360 4K VR. [46] Kamera je osazena dvěma čočkami a jejím výstupem je obraz obsahující dva snímky vedle sebe s perspektivou rybího oka (fisheye). Tento výstup je potřeba převést pomocí proprietárního softwaru k této kameře. Aplikace se jmenuje PIXPRO 360 VR SUITE [47] a umožňuje převedení obrazu z kamery na ekvirektangulární snímek 360x180°.



Obr. 34 Ukázka aplikace PixPro 360 VR Suite
Zdroj: Autor

Kamera byla upevněna na stativu ve výšce očí pozorovatele. Lze ji ovládat dálkově pomocí mobilní aplikace a wifi/Bluetooth připojení. Vzhledem k pandemickému omezením v době pořizování snímků se v budově nenacházeli

žádní studenti. Vstup do některých specializovaný učeben vyžadoval asistenci pověřených pracovníků. Kamera byla umístěna do snímaných prostorů a dálkově byl pořízen snímek. Takto byly nasnímány všechny učebny a některé další prostory budovy FIM.

Obr. 35 KODAK PIXPR0 ORBIT360 4K VR kamera a její umístění na stojanu
Zdroj: Autor | Převzato z: [46]

5.2.2 Webová aplikace

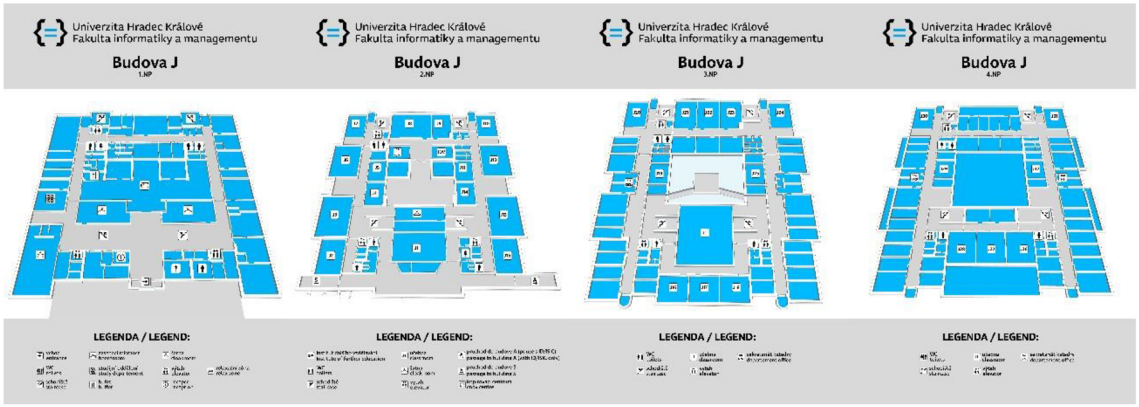
Hlavním prvkem celého webu je panoramatický snímek vybrané místnosti budovy FIM. Je zobrazen na pozadí webu a je interaktivní. Přes něj se vykresluje uživatelské rozhraní. V horní části se nachází pruh s názvem aplikace a informacemi o aktuálním zobrazovaném panoramatickém snímku (název místnosti, poschodí). V této části jsou také tlačítka pro přepínání mezi českou a anglickou verzí webu. Po stranách jsou umístěné šipky pro přecházení na další/předchozí místnost. Ve spodní části se nachází tlačítko pro zobrazení mapy poschodí.

Pro responzivní design aplikace byla použita front-endová sada nástrojů Bootstrap [48]. Za zmínku stojí použitá komponenta Modální okno (Modal), která byla použita pro rozklikávatelnou mapu poschodí. Další přizpůsobení vzhledu bylo implementováno pomocí kaskádových stylů (CSS). K vykreslení panorama na pozadí uživatelského rozhraní bylo potřeba použít některý element jako kontejner pro Three.js. V tomto případě byl zvolen element „body“, kterému byl nastaven styl s názvem „cover-container“. Aby se vykresloval za zbytkem obsahu, byla mu nastavena absolutní pozice.

```css
html,
body {
  height: 100%;
  background-color: #00b1f4;
  margin: 0;
  overflow: hidden;
}

.cover-container {
  position: absolute;
}
```

Obr. 36 Ukázka kódu – CSS
Zdroj: Autor

Uživatelské rozhraní nabízí modální panel, který lze minimalizovat. Panel obsahuje plán zvoleného patra budovy. Při najetí myší na některou z místností je tato místnost zvýrazněna a je možné ji rozkliknout. Po kliknutí na místnost se panel minimalizuje a v pozadí se načte panoramatický snímek zvolené místnosti. Plány poschodí byly převzaty z webu První kroky na FIM. [49]



Obr. 37 Plány poschodí FIM
Převzato z: [49]

Aby plán poschodí mohl být interaktivní, bylo potřeba vytvořit vektorové tvary z jednotlivých místností. Ke zpracování těchto tvarů a jejich export do SVG formátu byl použit software Adobe Photoshop. Ze souboru pak bylo možné získat data a zabalit je do HTML kódu. Jednotlivé SVG tvary pak slouží pro přepínání mezi jednotlivými místnostmi a zvýrazňují, které prostory si lze prohlédnout.

Obr. 38 Ukázka SVG plánu
Zdroj: Autor

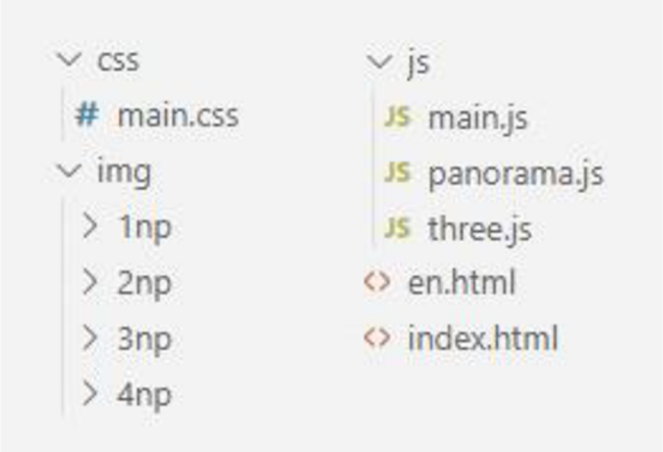
Na desktopové platformě je při interakci s SVG tvary použitý styl zvýraznění tvaru, nad kterým se pohybuje kurzor myši a také se zobrazuje název místnosti jako „tooltip“. Na mobilním telefonu však tuto funkcionalitu nelze uplatnit, a proto jsou tvary, na které lze kliknout barevně zvýrazněné a jejich barva pulzuje pomocí animací uvnitř CSS.

Veškerá interakce s aplikací je obsloužena pomocí Java Scriptu (JS), a to včetně funkcí knihovny Three.js. Při kliknutí na některou z místností je dán pokyn knihovně Three.js k nahrazení textury, která se vykresluje jako panorama. HTML kód je manipulovaný při změně kontextu. Příkladem je změna poschodí v modálním okně s mapou poschodí. Je nutné pomocí JS nahradit SVG tvary stávajícího poschodí, za tvary korespondující s nově zvoleným poschodím.

5.2.3 Adresářová struktura

Kostrou aplikace je soubor „index.html“ a jeho anglická modifikace „en.html“. Tyto soubory obsahují základní HTML strukturu webu. Stylizace webu nad rámec stylů Bootstrap je řešena pomocí kaskádových stylů uvnitř souboru „main.css“ v adresáři „css“. Veškeré panoramatické snímky a plány podlaží jsou uloženy v adresáři „img“ a jsou rozděleny podle jednotlivých nadpodlaží (NP). Pro změnu

stávajícího snímku místnosti za nový, stačí snímek přepsat a dodržet stejný název. Posledním adresářem je „js“, který obsahuje všechny javascriptové soubory. Soubor „three.js“ obsahuje kód knihovny Three.js. Soubor „panorama.js“ pak obsahuje vlastní nastavení Three.js, viz. následující kapitola. A poslední soubor „main.js“ obsluhuje změnu obsahu HTML (viz. změna plánu poschodí, zmiňovaná v předchozí kapitole) a interakci s plánem poschodí.



Obr. 39 Adresářová struktura projektu
Zdroj: Autor

5.2.4 Obsluha panorama pomocí Three.js

Panorama je vykresleno pomocí knihovny Three.js. Jedná se o panorama ve formátu equirectangular. Před použitím této knihovny je potřeba knihovnu stáhnout a poté ji načíst v HTML kódu pomocí script tagu. Three.js je v projektu nastaveno uvnitř java skriptového souboru „panorama.js“.

Aby bylo možné něco zobrazit pomocí Three.js je nejprve potřeba vytvořit scénu, kameru a renderer. Renderer bude vykreslovat scénu z pohledu kamery. Typ kamery je „PerspectiveCamera“ a jejími atributy jsou zorný úhel (FOV), udávaný v úhlových stupních, poměr stran (šířka / výška) a nakonec blízká a vzdálená ořezová plocha (near and far clipping plane). Těmito atributy je dána trojrozměrná oblast, viditelná na obrazovce skrze kameru (frustum). Aplikace používá WebGLRenderer, který vykresluje scénu do HTML canvas elementu. Jeho parametry jsou opět poměr stran a rozměry.

```
function createCamera() {
    // Create a Camera
    width = window.innerWidth;
    height = window.innerHeight;
    aspect = width / height;
    const fov = 90; // Field of View
    const near = 1; // the near clipping plane
    const far = 1100; // the far clipping plane

    camera = new THREE.PerspectiveCamera(fov, aspect, near, far);
    camera.target = new THREE.Vector3(0, 0, 0);
}
```

Obr. 40 Three.js – tvorba kamery
Zdroj: Autor

Pro zobrazení panorama je potřeba vytvořit geometrii, na kterou se panorama promítne a bude sloužit jako „SkySphere“. V tomto případě je vytvořena sférická geometrie s atributy poloměru a počtem segmentů na výšku a na šířku. Geometrii je třeba převrátit, protože se na ni bude kamera dívat zevnitř. Dále je potřeba vytvořit texturu. K tomu slouží „TextureLoader“, kterému je předán ekvirektangulární panoramatický snímek. Textura je poté přiřazena materiálu „MeshBasicMaterial“ jako atribut „map“. Posledním objektem je „Mesh“, kterému je předána geometrie a materiál. Tento „Mesh“ objekt je poté přidán do scény. Světla v tomto případě není potřeba vytvářet.

```
if(geometry==undefined){
    geometry = new THREE.SphereBufferGeometry(500, 60, 40);
    geometry.scale(-1, 1, 1);
}

var texture = new THREE.TextureLoader(loadingManager).load(panoImg);
var material = new THREE.MeshBasicMaterial({
    map: texture
});

if(mesh==undefined){
    mesh = new THREE.Mesh(geometry, material);
    scene.add(mesh);
}else{
    mesh.geometry = geometry;
    mesh.material = material;
}
```

Obr. 41 Three.js – vytvoření geometrie a materiálu
Zdroj: Autor

Při tvorbě objektů „mesh“ a „geometry“ se ověřuje, jestli již byly jednou vytvořeny. Tento kus kódu se totiž znovu využívá při změně panorama a není již potřeba geometrii znovu vytvářet, pouze se změní textura. K načítání textur pomocí „TextureLoaderu“ se používá „loadingManager“. Ten obstarává proces načtení snímku při změně panorama. Funkce „onStart“ zobrazí animaci načítání snímku. Tato animace běží, dokud není snímek připravený. Poté se spustí funkce „onLoad“, která skryje animaci načítání, resetuje nastavení pohledu a nový snímek je tak zobrazen. V případě, že snímek nejde načíst, se spustí funkce „onError“.

```
const loadingManager = new THREE.LoadingManager(() => {});

loadingManager.onStart = function (){
    const loadingScreen = document.getElementById('loading-screen');
    loadingScreen.classList.remove('fade-out');
}

loadingManager.onLoad = function ( ) {

    lon = 0;
    lat = 0;
    isUserInteracting = false;
    camera.fov = 110-camera.aspect*15;
    camera.updateProjectionMatrix();

    const loadingScreen = document.getElementById('loading-screen');
    loadingScreen.classList.add('fade-out');
    // optional: remove loader from DOM via event listener
    loadingScreen.addEventListener('transitionend', onTransitionEnd);

};

loadingManager.onError = function ( url ) {
    console.log( 'There was an error loading ' + url );
    alert( 'There was an error loading ' + url );

};
```

Obr. 42 LoadingManager
Zdroj: Autor

Nakonec je potřeba nastavit animační smyčku (render/animate loop). Smyčka způsobí to, že Renderer vykreslí scénu pokaždé, když se obnoví obraz (obnovovací frekvence). Tedy u běžné obrazovky 60x za vteřinu. Uvnitř smyčky je možné animovat různé parametry. V tomto případě se volá funkce „update“, uvnitř které se při každém vykreslení přičte hodnota k parametru „longitude“, čímž dochází k horizontální rotaci kamery. Tato rotace probíhá, dokud uživatel nezačne sám manipulovat s rotací kamery. Při změně místnosti/panorama se rotace resetuje a kamera se začne znovu otáčet. Pokud proběhne kompletní rotace panorama o 360°, dojde automaticky na přechod do další místnosti.

```
function animate() {
    requestAnimationFrame(animate);
    update()
};

function update() {

    if (isUserInteracting === false) {

        lon += 0.15;
        if(lon>=360.0){
            currentRoom+=1;
            if(currentRoom>58){
                currentRoom = 0;
            }
            changePanoImg(currentRoom);
        }

    }

    lat = Math.max(-85, Math.min(85, lat));
    phi = THREE.MathUtils.degToRad(90 - lat);
    theta = THREE.MathUtils.degToRad(lon);

    camera.target.x = 500 * Math.sin(phi) * Math.cos(theta);
    camera.target.y = 500 * Math.cos(phi);
    camera.target.z = 500 * Math.sin(phi) * Math.sin(theta);

    camera.lookAt(camera.target);

    renderer.render(scene, camera);

}
```

Obr. 43 Three.js – animační smyčka
Zdroj: Autor

Pokud uživatel zvolí na mapě jinou místnost, nebo se na ni překlikne pomocí šipek, panorama této místnosti musí nahradit stávající texturu. Three.js musí dostat pokyn k aktualizaci této textury. To je řešeno voláním funkce „createObjects“. Při změně textury se zobrazí animovaný prvek znázorňující načítání nového panorama.

```
function changePanoImg(room) {
    currentRoom = room;
    spinner.innerHTML = '<section id="loading-screen"><div class="spin
    const loadingScreen = document.getElementById('loading-screen');
    const roomName = document.getElementById('roomName');

    switch (room) {
        case 0:
            src = ('img/1np/vchod.jpg');
            roomName.innerHTML = rooms[0] + ' - 1.NP';
            break;
        if (room <= 15) {
            src = ('img/2np/J' + room + '.jpg');
            roomName.innerHTML = rooms[room] + ' - 2.NP';
        } else if (room > 15 && room <= 25) {
            src = ('img/3np/J' + room + '.jpg');
            roomName.innerHTML = rooms[room] + ' - 3.NP';
        } else {
            src = ('img/4np/J' + room + '.jpg');
            roomName.innerHTML = rooms[room] + ' - 4.NP';
        }
        break;
    }
    createObjects(src);
    $("#exampleModal").modal("hide");
}
```

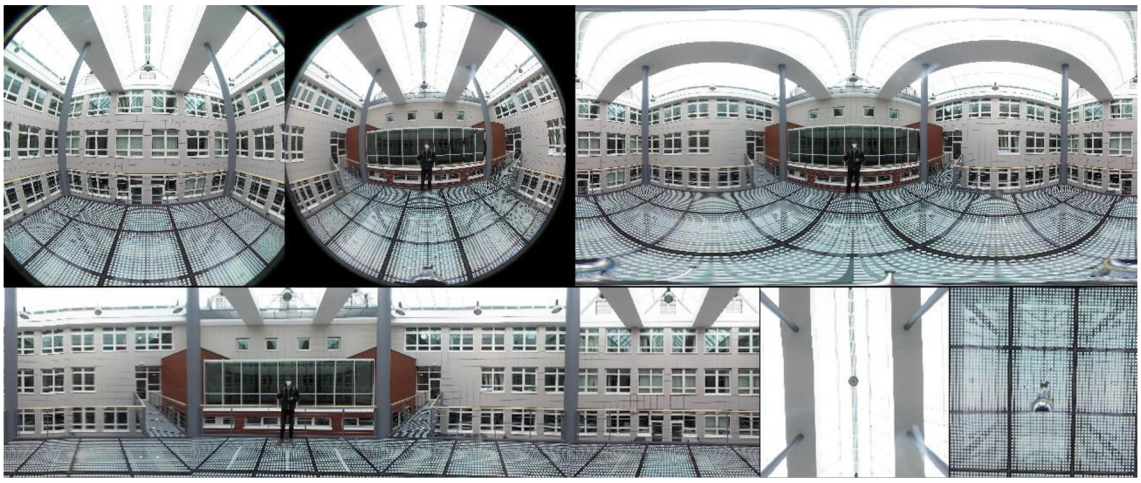
Obr. 44 Three.js – změna panorama
Zdroj: Autor

6 Shrnutí výsledků

Pořízené snímky

V rámci této práce byl zpracováván obraz z 360° kamery KODAK PIXPR0 ORBIT360 4K VR [46], která byla zapůjčená fakultou FIM UHK. Jejím výstupem byly dva snímky ve formátu fishey (rybí oko) s rozlišením 3680x3680 pixelů. Klasická metoda pro spojení snímků z více kamer se zakládá na vyhledávání stejných prvků mezi dvěma snímky a jejich následné transformaci do stejné plochy pomocí homografie. Zmíněná metoda však není pro toto zařízení vhodná, jelikož oblast překrytí mezi okraji těchto čoček je velmi malá. Dalšími problémy můžou být paralax, vinětace a rozdílná světlost dvou snímků. Z těchto důvodů výrobci jednotlivých 360° kamer dodávají své zařízení s proprietárním softwarem pro spojení dvou snímků, nebo jsou tyto snímky spojeny přímo v zařízení.

Pomocí proprietárního softwaru ke kameře PIXPRO 360 VR SUITE [47], byl obraz převeden na ekvirektangulární projekci. Výstupní panorama má poměr stran 2:1 a rozlišení 7360x3680 pixelů. Pomocí webové aplikace Panorama to Cubemap [45] byl ekvirektangulární snímek převeden na kubickou mapu. Výstupem je šest jednotlivých snímků o rozlišení 1840x1840 pixelů. Všechny snímky jsou ve formátu JPEG s bitovou hloubkou 24 bitů. S takto připravenými snímky bylo v rámci práce dále nakládáno.



Obr. 45 Snímek z kamery (vlevo nahoře), ekvirektangulární projekce (vpravo nahoře), kubická projekce (dole)
Zdroj: Autor

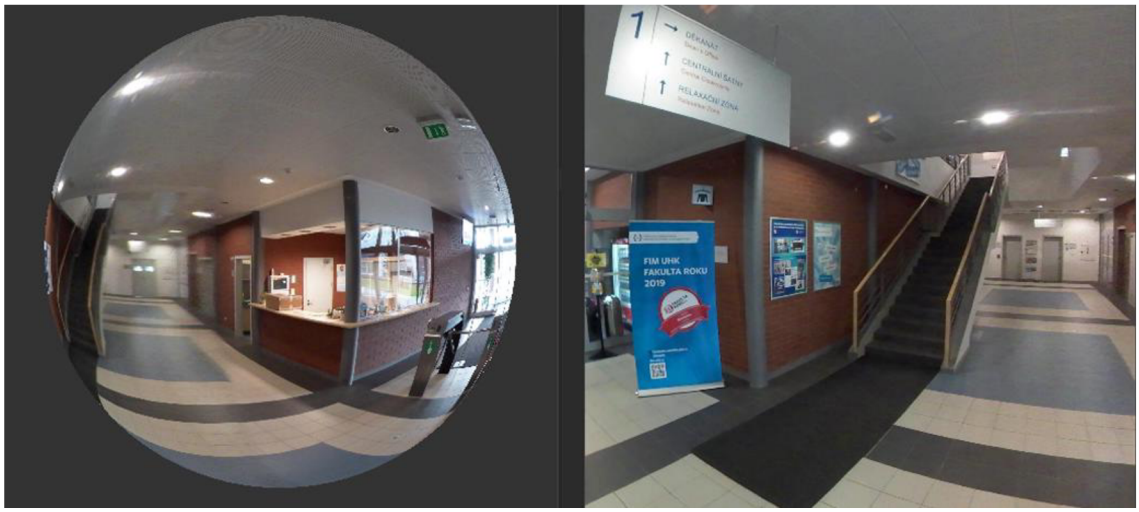
Použitá technika

Snímky z kamery nemají příliš vysoké rozlišení. V oblasti spojů mezi dvěma čočkami je obraz obzvláště rozmazaný. Rozdílné optické vlastnosti dvou čoček jsou také znát v podmínkách s horším osvětlením, kde se snímek z jedné čočky vykazuje větším šumem v obraze jako na snímku z druhé čočky. Kamera také trpí nízkým dynamickým rozsahem, proto jsou na snímcích místností vidět přepaly.

V aplikace lze však stávající snímky snadno nahradit novými. V případě pořízení lepších snímku pomocí kvalitnější kamery je tak možné aplikaci snadno aktualizovat pouhým nahrazením snímků.

Implementace zobrazení panorama v jednoduché scéně

Implementace využívá obě představené 2D reprezentace panorama, equirektangulární a kubickou. Představuje způsob jejich zobrazení ve 3D prostoru s využitím technologií představených v této publikaci (OpenGL a LWJGL). Kubická mapa byla namapována na geometrii krychle pomocí fragment shaderu a kubického sampleru. Ekvirektangulární mapa byla namapována na geometrii plochy, která byla pomocí vertex shaderu transformována na sférickou geometrii. Výsledkem bylo v obou případech 360° panorama pozorovatelné zevnitř geometrie, na kterou bylo namapováno, a to bez viditelných švů.

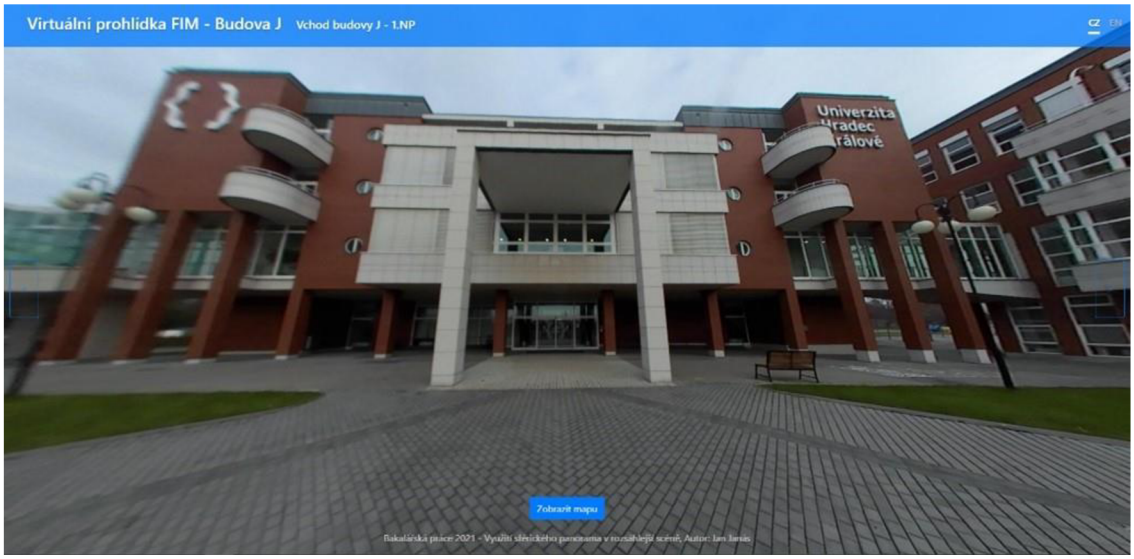


Obr. 46 Ukázka mapování panorama na kouli
Zdroj: Autor

Implementace zobrazení panorama v rozsáhlejší scéně

V rámci této implementace vznikla webová virtuální prohlídka FIM UHK. Bylo pořízeno okolo sedmdesáti panoramatických snímků budovy J s využitím 360° kamery poskytnuté fakultou.

Aplikace byla nasazena a byl jí vyhrazen webový prostor v rámci sítě FIM UHK. Je dostupná pod adresou: https://fim.uhk.cz/virtualne/. Je responzivní, interaktivní a má českou i anglickou jazykovou verzi.



Obr. 47 Ukázka finální aplikace
Převzato z: [50]

Aplikace umožňuje návštěvníkům prozkoumat prostory fakulty nebo najít hledanou učebnu na mapě poschodí. Aplikaci lze také využít pro účely prezentace, jelikož snímky se postupně automaticky otáčejí a střídají. V průběhu tvorby aplikace bylo dbáno na zpětnou vazbu studentů, přátel a vedení fakulty.

7 Závěry a doporučení

V rámci této práce bylo představeno panoramatické zobrazení, jeho historie, využití v praxi, metody jeho snímaní a zpracování. Byly představeny volně dostupné technologie pro práci s obrazem, pomocí kterých bylo implementováno zobrazení panorama v jednoduché scéně a následně k tvorbě rozsáhlejší virtuální prohlídky reálného objektu FIM UHK.

Z teoretické části vyplývá, že 360° panorama je od svého vzniku velmi atraktivním formátem obrazu, o jehož vývoj a využití se dnes zajímají různá technologická odvětví. Zábavní průmysl (film, VR), služby (Google Maps), vizualizace (architektura, virtuální prohlídky), 3D grafika (IBL, LED Stage) a další.

Praktická část práce uplatňuje některé z popsaných metod snímání a zpracování panorama a zároveň je ukázkou reálného nasazení těchto technologií. Zobrazení panorama v jednoduché scéně je ukázkou základních principů pro zobrazení panorama v 3D prostoru a webová virtuální prohlídka je ukázkou jednoho z možných využití tohoto zobrazení. Komerční řešení pro tvorbu virtuálních prohlídek nabízí spoustu předpřipravených šablon, které se dají přizpůsobit podle libosti. Z mého pohledu je však tato publikace důkazem toho, že i pomocí volně dostupných technologií lze dosáhnout podobného výstupu, jehož vizuální stránka se výrazně liší od většiny komerčních řešení.

Pokud by se aplikace měla v budoucnu dále rozšiřovat, bylo by vhodné začít u kvality samotných panoramatických snímků. Zde se nabízí buďto technika segmentů s použití DSLR a panoramatické hlavy, nebo investice do novější 360° kamery. Výkon aplikace by se mohl navýšit přechodem z API WebGL na ohlášené API WebGPU, které by mělo lépe využít potenciál grafického hardwaru.

Tvorbou virtuální prohlídky FIM UHK vznikla aplikace, která má dle mého názoru praktické využití a je pro mě něčím, co můžu využít. Slovy mých spolužáků, kteří aplikaci vyzkoušeli: „Konečně můžu rodičům ukázat, jak to vlastně vypadá na mojí fakultě.“ Stejně tak věřím, že aplikace poslouží mojí fakultě k reprezentativním účelům a bude se časem dále rozvíjet.

8 Seznam použité literatury

[1] *Soubor:Panorama of London Barker.jpg* [online]. nedatováno [vid. 2021-01-11]. Dostupné z: https://cs.wikipedia.org/wiki/Soubor:Panorama_of_London_Barker.jpg

[2] "Unlimiting the Bounds": the Panorama and the Balloon View. *The Public Domain Review* [online]. [vid. 2021-01-18]. Dostupné z: https://publicdomainreview.org/essay/unlimiting-the-bounds-the-panorama-and-the-balloon-view/

[3] MITCHELL, robert. *English: Cross-section of the Rotunda in Leicester Square in which the panorama of London was exhibited (1801).* [online]. 1801date QS:P571,+ - -00T00:00:00Z/9 1801 [vid. 2021-01-18]. Dostupné z: https://commons.wikimedia.org/wiki/File:Cross-section-of-the-rotund_0.jpg

[4] THOMPSON, Seth. VR Panoramic Photography and Hypermedia: Drawing from the Panorama's Past. In: . 2015.

[5] Objevte krásy Street View a přispějte do Map Google svými vlastními snímky. *Google Maps Street View* [online]. [vid. 2020-08-17]. Dostupné z: https://www.google.com/streetview/

[6] *Google Earth* [online]. [vid. 2021-01-11]. Dostupné z: https://earth.google.com/web/@50.08663702,14.41032496,206.24537659a,0d,60y,305.95761611h,98.57599337t,0r/data=CjQSMhIgNDJlNjFkZjg3OGFkMTFlOTg4YTU0ZDFmNmYyMjAxNmEiDmNoYXJsZXMtYnJpZGdlIhoKFkRqcnI3bVRRajB6SzhlTndwRFlMVkEQAg

[7] GUTOWSKI, Piotr a Zuzanna KŁOS-ADAMKIEWICZ. Development of e-service virtual museum tours in Poland during the SARS-CoV-2 pandemic. *Procedia Computer Science* [online]. 2020, **176**, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020, 2375–2383. ISSN 1877-0509. Dostupné z: doi:10.1016/j.procs.2020.09.303

[8] *Virtual tour „Sistine Chapel"* [online]. [vid. 2021-02-08]. Dostupné z: http://www.museivaticani.va/content/museivaticani/en/collezioni/musei/cappella-sistina/tour-virtuale.html

[9] *Oculus Quest 2: náš nejpokrokovější autonomní headset | Oculus* [online]. [vid. 2021-01-11]. Dostupné z: https://www.oculus.com/quest-2/

[10] *Motion Sickness in VR: Why it happens and how to minimise it* [online]. [vid. 2021-02-08]. Dostupné z: https://virtualspeech.com/blog/motion-sickness-vr

[11] *Image-Based Lighting.pdf* [online]. [vid. 2020-08-17]. Dostupné z: https://ict.usc.edu/pubs/Image-Based%20Lighting.pdf

[12] *HDRIs: All | HDRI Haven* [online]. [vid. 2020-11-28]. Dostupné z: https://hdrihaven.com/hdris/

[13] Groundbreaking LED Stage Production Technology Created for Hit Lucasfilm Series „The Mandalorian". *Industrial Light & Magic* [online]. [vid. 2021-01-11]. Dostupné z: https://www.ilm.com/hatsrabbits/virtual-production-on-the-mandalorian/

[14] *How to stitch a panorama ?* [online]. [vid. 2020-09-06]. Dostupné z: https://www.panoramic-photo-guide.com/contents-stitch-a-panorama.html

[15] SOOD, Naveksha. Image Stitching to create a Panorama. *Medium* [online]. 31. červenec 2019 [vid. 2021-01-12]. Dostupné z: https://medium.com/@navekshasood/image-stitching-to-create-a-panorama-5e030ecc8f7

[16] SCHMIDT, Lutz. What's the meaning of VR photography / VR panoramas? nedatováno, 5.

[17] 0-360.COM. 0-360.com. 360 degree photography and video. *0-360.com* [online]. [vid. 2020-09-06]. Dostupné z: http://www.0-360.com/

[18] PANORAMACAMERAS. Types of Panoramic Photography. *Panorama Cameras* [online]. 27. listopad 2012 [vid. 2021-01-12]. Dostupné z: https://panoramacameras.wordpress.com/2012/11/27/types-of-panoramic-photography/

[19] *SpheroCam HDR - Panoramania, DIARY* [online]. [vid. 2020-09-06]. Dostupné z: http://panoramania.co.jp/diary/archives/2006/09/spherocam-hdr.html

[20] *Panoscan VRs – Bohonus VR Photography* [online]. [vid. 2021-01-12]. Dostupné z: https://www.bohonus.com/vr-galleries/panoscan-vrs/

[21] *Insta360 ONE X2* [online]. [vid. 2020-11-28]. Dostupné z: https://www.insta360.com/product/insta360-onex2

[22] HO, Tuan. Dual-fisheye Lens Stitching for 360-degree Imaging & Video. nedatováno, 60.

[23] MENDOZA, Antonio. The Trickster's Project: HDRI Process. *The Trickster's Project* [online]. [vid. 2021-01-21]. Dostupné z: http://thetricksterproject.blogspot.com/2013/07/hdri-process.html

[24] DESIGN, Marziah Karch Writer Marziah Karch is a former writer for Lifewire who also excels at Serious Game, Develops Online Help SYSTEMS, MANUALS a

interactive training modules our editorial process Marziah KARCH. Android Photo Sphere: What It Is & How to Use It. *Lifewire* [online]. [vid. 2021-01-21]. Dostupné z: https://www.lifewire.com/what-is-android-photo-sphere-1616136

[25] What Is Bracketing and How to Use It in HDR Photography. *ExpertPhotography* [online]. [vid. 2021-02-08]. Dostupné z: https://expertphotography.com/bracketing-photography/

[26] *High Dynamic Range Imaging - 2nd Edition* [online]. [vid. 2020-08-05]. Dostupné z: https://www.elsevier.com/books/high-dynamic-range-imaging/reinhard/978-0-12-374914-7

[27] *Tonemapping* [online]. [vid. 2021-03-26]. Dostupné z: http://dativ.at/logmap/#overview

[28] *Sférická soustava souřadnic* [online]. 2019 [vid. 2020-11-23]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Sf%C3%A9rick%C3%A1_soustava_sou%C5%99adnic&oldid=17279740

[29] WEISSTEIN, Eric W. *Polar Coordinates* [online]. [vid. 2021-02-08]. Dostupné z: https://mathworld.wolfram.com/PolarCoordinates.html

[30] *Big_ben_equirectangular.jpg (600×300)* [online]. [vid. 2020-11-02]. Dostupné z: https://wiki.panotools.org/images/0/01/Big_ben_equirectangular.jpg

[31] *Equirectangular Projection - PanoTools.org Wiki* [online]. [vid. 2020-11-02]. Dostupné z: https://wiki.panotools.org/Equirectangular_Projection

[32] *Cubic Projection - PanoTools.org Wiki* [online]. [vid. 2020-11-02]. Dostupné z: https://wiki.panotools.org/Cubic_Projection

[33] *Big_ben_cubic.jpg (600×450)* [online]. [vid. 2020-11-02]. Dostupné z: https://wiki.panotools.org/images/9/91/Big_ben_cubic.jpg

[34] NVIDIA Developer. *NVIDIA Developer* [online]. [vid. 2021-01-26]. Dostupné z: https://developer.nvidia.com/

[35] *OpenGL Overview* [online]. [vid. 2020-11-23]. Dostupné z: https://www.opengl.org/about/#1

[36] *LearnOpenGL - OpenGL* [online]. [vid. 2020-11-23]. Dostupné z: https://learnopengl.com/Getting-started/OpenGL

[37] *LWJGL - Lightweight Java Game Library* [online]. [vid. 2021-01-27]. Dostupné z: https://www.lwjgl.org/#learn-more

[38] WebGL - OpenGL ES for the Web. *The Khronos Group* [online]. 19. červenec 2011 [vid. 2020-11-25]. Dostupné z: https://www.khronos.org/webgl/

[39] *three.js – JavaScript 3D library* [online]. [vid. 2020-11-25]. Dostupné z: https://threejs.org/

[40] *three.js examples* [online]. [vid. 2021-01-19]. Dostupné z: https://threejs.org/examples/#webgl_animation_keyframes

[41] Pano2VR - Virtual Tour Software. *Garden Gnome* [online]. [vid. 2021-01-26]. Dostupné z: https://ggnome.com/pano2vr/

[42] *LearnOpenGL - Shaders* [online]. [vid. 2021-03-03]. Dostupné z: https://learnopengl.com/Getting-started/Shaders

[43] *LearnOpenGL - Cubemaps* [online]. [vid. 2020-11-25]. Dostupné z: https://learnopengl.com/Advanced-OpenGL/Cubemaps

[44] javascript - Convert 2:1 equirectangular panorama to cube map. *Stack Overflow* [online]. [vid. 2021-02-01]. Dostupné z: https://stackoverflow.com/questions/29678510/convert-21-equirectangular-panorama-to-cube-map

[45] *Panorama to Cubemap* [online]. [vid. 2021-02-01]. Dostupné z: https://jaxry.github.io/panorama-to-cubemap/

[46] *ORBIT360 4K 360 VR Camera | KODAK PIXPRO Digital Cameras* [online]. [vid. 2021-01-28]. Dostupné z: https://kodakpixpro.com/cameras/360-vr/orbit360-4k

[47] *KODAK PIXPRO Cameras | Support - Software and Manuals* [online]. [vid. 2021-01-28]. Dostupné z: https://kodakpixpro.com/support/downloads/

[48] CONTRIBUTORS, Mark Otto, Jacob Thornton, and Bootstrap. *Bootstrap* [online]. [vid. 2021-02-01]. Dostupné z: https://getbootstrap.com/

[49] *První kroky na FIM - První týden na FIM* [online]. [vid. 2020-11-28]. Dostupné z: https://fim2.uhk.cz/kroky/cz/groupm/start-na-univerzite/prvni-tyden-na-fim

[50] *Virtuální prohlídka FIM* [online]. [vid. 2021-03-18]. Dostupné z: https://fim.uhk.cz/virtualne/

9 Přílohy

Zdrojové kódy webové aplikace s adresářovou strukturou (virtualTour.zip)

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Jan Janás**
Studium:	I1700090
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Zpracování obrazu z 360 stupňové kamery**
Název bakalářské práce AJ:	Panorama image processing

Cíl, metody, literatura, předpoklady:

Cíl práce:

Prozkoumat přístupy a techniky pro zpracování obrazu z 360 stupňové kamery a metody vizuální prezentace.

Postup prací:

1. Prozkoumat principy a metody používané pro snímání panoramatických dat
2. Prozkoumat metody prezentace a využití panoramatických snímků
3. Vytvořit přehled používaných přístupů, nástrojů a dostupných softwarových řešení
4. Navrhnout implementovat řešení využitím panoramatického obrazu v jednoduché prostorové scéně
5. Navrhnout a implementovat řešení pro prezentaci rozsáhlejšího reálného objektu s využitím sad panoramatických snímků
6. Zhodnotit dosažené výsledky

SZELISKI, Richard, 2010. *Computer Vision: Algorithms and Applications*. B.m.: Springer Science {\& Business Media. ISBN 978-1-84882-935-0.

HO, Tuan a Madhukar BUDAGAVI, 2017. Dual-fisheye lens stitching for 360-degree imaging. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP): 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* [online]. s. 2172–2176. ISSN 2379-190X. Dostupné z: doi:10.1109/ICASSP.2017.7952541

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Bruno Ježek, Ph.D.
Datum zadání závěrečné práce:	4.5.2020