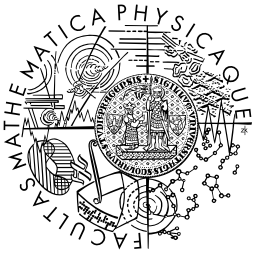


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Adrián Mačák

Point cloud video na HoloLens

Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce:	RNDr. Josef Pelikán
Studijní program:	Informatika
Studijní obor:	Programování a softwarové systémy

Praha 2019

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Chcel by som sa poďakovať predovšetkým môjmu vedúcemu bakalárskej práce RNDr. Josefovi Pelikánovi za odborné rady, obrovskú ochotu a pomoc počas celého obdobia vypracovania. Ďalej by som sa chcel poďakovať Janovi Hovorovi a spoločnosti Pocket Virtuality a.s. za podporu, zapožičanie hardvéru a softvéru. Vďaka patrí aj doc. Ing. Jaroslavovi Křivánkovi, Ph.D. za vedenie ročníkového projektu, na ktorý táto bakalárska práca nadviazala. Tiež by som chcel poďakovať svojej rodine a najbližším za neoceniteľnú motiváciu a podporu.

Název práce: Point cloud video na HoloLens

Autor: Adrián Mačák

Katedra: Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: RNDr. Josef Pelikán, Katedra softwaru a výuky informatiky

Abstrakt: Vývoj headsetov pre virtuálnu a rozšírenú realitu otvára veľa možností použitia týchto technológii v priemysle, zábave a v rôznych iných odvetviach. Cieľom tejto práce je preskúmať možnosti a navrhnúť aplikácie, ktoré zabezpečia, aby pomocou technológii Microsoft Hololens v1 a Microsoft Kinect v2 bolo možné zosnímať, spracovať, preniesť po sieti a vykresliť dynamický point cloud na strane HoloLens. Funkcionalita je rozdelená medzi tri aplikácie. Prvá je desktopová aplikácia, na ktorej beží snímanie, spracovanie a server. Druhá aplikácia, klientská, zabezpečuje rendering a užívateľské rozhranie na strane HoloLens. Tretia aplikácia slúži na testovanie priepustnosti siete, simulujúca prenos point cloud videa. Táto práca opisuje postup od toho ako pripojiť senzor k počítaču, cez pohľad na seba z vtáčej perspektívy, až po nočné videnie, ktoré z toho vzniklo. Výsledkom sú aplikácie, ktoré to dokážu v reálnom čase.

Klíčová slova: Point cloud video rozšířená realita HoloLens Kinect TCP/IP ZeroMQ

Title: Point cloud video for HoloLens

Author: Adrián Mačák

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Josef Pelikán, Department of Software and Computer Science Education

Abstract: The development of virtual and augmented reality headsets opens up many possibilities for using these technologies in industry, entertainment and other sectors. The aim of this work is to explore the possibilities and design the applications, that will ensure that by using Microsoft Hololens v1 and Microsoft Kinect v2 there will be a possibility to capture, process, transfer and render dynamic point cloud on the HoloLens side. The functionality is divided between three applications. The first one is a desktop application on which the capturing, processing, and a server is running. The second application, the client application, provides rendering and the user interface on HoloLens side. The third application is used to test the network throughput, simulating the point cloud video transmission. This work describes the process of how to connect a sensor to the computer, looking at yourself from a bird's eye view, and the night vision that has come out of it. The results are applications that can do these in real time.

Keywords: Point cloud video augmented reality HoloLens Kinect TCP/IP ZeroMQ

Obsah

Úvod

Motivácia

Zdokonalenie headsetov na virtuálnu alebo rozšírenú (augmented) realitu prináša veľa výhod. Použitie v praxi sa stáva čoraz skutočnejším. Skúsme si predstaviť, že by sme taký headset použili na prehrávanie point cloud 1 videa. Zaznamenáme nejaký pohyb, napríklad tanec alebo inštruktážne video a potom sa pomocou toho bude môcť niekto naučiť danú vec sám. Alebo si predstavme, že technik bude vidieť na diaľku okrem obrazu aj 3D štruktúru snímaného objektu.

V prípade tejto práce sa zameriame na zariadenie Microsoft Hololens v1 (2.6). Takú možnosť, zatiaľ štandardný softvér v tomto zariadení neposkytuje. Preto sme sa rozhodli preskúmať možnosti, navrhnúť a implementovať od základu aplikácie, ktoré to umožnia.

Podobné programy

Na webe je možné zatiaľ nájsť rôzne videá s prototypovými aplikáciami. V prípade tejto práce sa bral ohľad na to aby mohla byť neskôr začlenená do už existujúcej grafickej knižnice pre rozšírenú realitu. Taktiež sme sa snažili po sieti prenášať čo najmenej dát, aby bolo point cloud video plynulé a s čo najmenším omeškaním.

Ciele práce

Cieľom tejto práce bolo navrhnúť spôsob ako získať point cloud video, spracovať, preniesť po sieti a následne dáta predať rendereru, ktorý ich vykreslí v headsete pre rozšírenú realitu. Všetko toto by bolo najlepšie stíhať v real-time.

Postup riešenia

Celkovo práca obsahuje 3 aplikácie

- Serverovú aplikáciu, ktorá beží na desktope, zabezpečuje dáta a odpovedá na požiadavky klientov.
- Klientskú aplikáciu, ktorá beží v HoloLens, žiada server o dáta a vykresľuje ich.
- Testovaciu utilitu, ktorú sme pripravili v začiatkoch na otestovanie toho, či bude hardvér prenos dát stíhať. Utilita sa snaží napodobniť reálny prenos point cloud videa generovaním náhodných dát, rovnakej veľkosti. Ďalej umožňuje merať a porovnávať prechodnosť siete na obidvoch TCP stranách.

Obsah práce

V prvej kapitole si definujeme samotný point cloud, jeho získanie a reprezentáciu. V druhej si predstavíme použité technológie, ako hardvér samotný. V tretej kapitole sa pozrieme na použitý softvér, knižnice a frameworky. Ďalšia kapitola

je užívateľská dokumentácia k vytvoreným aplikáciam. V nasledujúcich troch kapitolách si programátorsky detailnejšie rozoberieme 3 aplikácie, ktorých sa táto práca týka. A na záver si všetko zhrnieme a povieme niečo o možnostiach rozšírenia.

1. Point cloud

1.1 Úvod

Pokiaľ by sme sa pokúsili preložiť slovné spojenie „point cloud“ do nášho jazyka, znamenalo by to niečo ako „mrak bodov“. Pre predstavu, mraky na oblohe sú nejaká množina častíc vody v atmosfére. Ak by sme chceli isť veľmi do detailov a mali na to prístroje, vedeli by sme o každej takej častici vody povedať napríklad jej vzdialenosť od povrchu Zeme (výšku).

Poznámka. Existuje všeobecná definícia. Teda „point cloud“ môže byť ľubovoľná množina bodov bez akéhokoľvek napojenia na povrch. V našom prípade budeme používať nasledujúcu obmedzenú definíciu.

Definícia 1 (Point cloud)**.** *Podľa (Lemmens, 2014), point cloud je množina bodov reprezentovaných v určitom súradnicovom systéme. Množina pozostáva zo vzdialenosti diskrétnych bodov zakriveného 2D povrchu v 3D priestore.*

Reprezentácia bodu

Informáciu o danom bode je možné zapísať ako jeho súradnice v 3D priestore, teda XYZ. V prípade fixného zorného uhla, kde súradnice XY sú známe, stačí zapísať len hĺbku - súradnicu Z.

Ďalšia informácia, ktorú je potrebné reprezentovať, je farba daného bodu, napríklad v RGB formáte. Alfa kanál je vhodné nastaviť na nepriehľadný. Na obrázku 1.1 môžeme vidieť príklad point cloud-u.

Obr. 1.1: Príklad point cloud-u.

Získanie point cloud-u

V prípade pár bodov je možné zapísať informáciu ručne. To by prestávalo fungovať, pokiaľ by bodov začalo pribúdať. Pre automatizovanie získavania point cloud dát, je potrebné mať prístroj, ktorý vie zosnímať hĺbku a farbu.

Prístroje snímajúce hĺbku, väčšinou fungujú na báze laseru alebo ultrazvuku, ktoré merajú time-of-flight daného signálu, ktorý vyslali.

Na zosnímanie farby daného bodu nám postačí kamera. Dôležité je potom správne mapovanie hĺbky a farby.

Príklad takého senzoru je Microsoft Kinect v2 (2.1), ktorý vie zosnímať dané informácie. V príslušnom SDK taktiež obsahuje triedu, ktorá sa stará o správne získanie point cloud-u.

1.2 Reprezentácia point cloud-u

Sú rôzne spôsoby reprezentácie point cloud-u, v prípade tejto práce sú dáta serializované v nasledujúcom poradí:

- HeaderLength(INT32) - dĺžka hlavičky.
- TimeStamp(INT64) - časová značka, kedy bol daná frame zaznamenaný.
- Width(UINT32) - šírka frame-u.
- Height(UINT32) - výška frame-u.
- SampleSize(1B) - počet bytov na jeden bod.
- KinectPos - pozícia Kinectu. Použije sa v prípade viacerých Kinectov.
  - X(UINT16)
  - Y(UINT16)
  - Z(UINT16)
- KinectDir - orientácia Kinectu.
  - X(UINT16)
  - Y(UINT16)
  - Z(UINT16)
- HAngle(float) - horizontálny uhol snímania Kinectu.
- VAngle(float) - vertikálny uhol snímania Kinectu.
- Data - poradie bodov je od pravého dolného rohu do lavého horného rohu.
  - point[0,0] - HiColor(INT16) | depth(UINT16)
  - ...
  - point[width-1,height-1] - HiColor(INT16) | depth(UINT16)

Poznámka. Byte order je little-endian. V budúcnosti je možné ošetriť rozpoznanie endianity nejakým príznakom alebo špecifickým reťazcom. Teraz je možné poznať opačnú endianitu tak, že v prípade prečítania prvého čísla HeaderLength(INT32) dostaneme obrovské číslo (Hlavička býva pár desiatok bytov - malé číslo).

Do priečinka sa ukladajú v podobe: názov je unix time-stamp, kedy bol daný frame zaznamenaný a prípona je `.pcl`.

1.3 Point cloud video

Inšpirovali ma princípy z knihy (Pfister a Gross, 2007). Proces získania point cloud dát sme si opísali vyššie. Takto získane dáta, sú zatiaľ len statické pre daný moment. Spojením takých dát do postupnosti nám vytvorí dynamický point cloud.

2. Použité technológie

2.1 Mictosoft Kinect v2

2.1.1 Základné informácie

Názov Kinect vznikol spojením slov „kinetic“ a „connect“. Je technológia vyvíjaná spoločnosťou Microsoft primárne ako doplnok k herným konzolám Xbox na ovládanie pomocou pohybov a zvukov. V tejto práci je použitý Microsoft Kinect v2 (alebo Microsoft for Xbox One), ktorý bol nástupca prvej generácie Microsoft Kinect for Xbox 360. Microsoft Kinect v2 - ďalej už len Kinect. Bol vydaný 22. Novembra 2013.

Nástupcom Kinectu je teraz Kinect for Azure prepojený s na cloud Azure, ktorý bol predstavený 24. februára 2019 a predpokladaná dostupnosť je v priebehu roku 2019.

Tieto údaje boli čerpané z Wikipedia contributors (2019c, Kinect) a Levy (2019).

Obr. 2.1: Microsoft Kinect v2 s osami snímania dát. Zdroj: Microsoft.

Systémove požiadavky

Na používanie Kinectu je potrebné, aby podľa Microsoft (2019) daný počítač splňoval nasledujúce požiadavky:

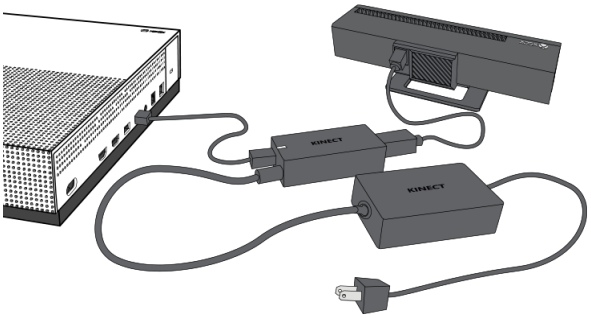
- Kinect for Windows v2 senzor (Obr. 2.1) a Kinect Adapter for Windows (Obr. 2.2)
- OS: min. Windows 8 (64-bit)
- Procesor: Dvojjadrový 3.2 GHz alebo rýchlejší
- RAM: 2GB
- USB: 3.0 - nie všetky sú ale kompatibilné so senzorom. Viac diskusia Goins (2014).
- Software: .NET Framework 4.0, Kinect for Windows SDK 2.0 (Pre obsluhu, nahrávanie, prehrávanie, vývoj, atď.)

Zrušenie podpory pre Kinect.

25. Októbra 2017 bola zastavená výroba a oficiálny predaj Kinectu (Yin-Poole, 2017), čo mi spôsobilo neskôr problémy s kompatibilitou.

2.1.2 Kinect Adapter for Windows

Na pripojenie Kinectu k počítaču (alebo k Xbox One X, Xbox One S) je potrebný adaptér, ktorý dodáva napätie a pridáva rozhranie USB 3.0.



Obr. 2.2: Microsoft Kinect v2 adaptér. Zdroj: Microsoft.

2.1.3 Technické detaily

Údaje v tejto sekcii 2.1.3 sú z dokumentácie Microsoft (2014a) a Microsoft (2014b). Obrázky sú vytvorené pomocou Kinect Studio v2.0 (3.4.1). Kinect dokáže snímať rôzne typy dát z jeho okolia. Pre prehľad uvedieme nasledujúce:

- color
- depth
- body
- body index
- infrared
- long exposure infrared
- audio

Color

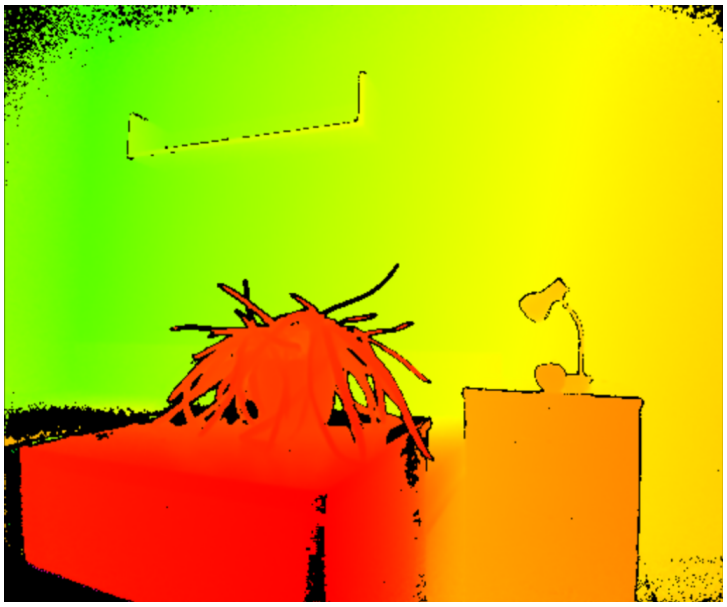
Farbu vie Kinect zosnímať pomocou kamery s rozlíšením Full HD 1920 x 1080 pixelov s farebnou hĺbkou 4 byty. Počiatok x=0, y=0 je v ľavom hornom rohu a koniec v x = 1919, y=1079 pravom dolnom rohu obrázku. Horizontálny uhol snímania je 84.1° a vertikálny je 53.8°.

Obr. 2.3: Farebný stream z Kinectu.

Depth

Hĺbkovú informáciu vie Kinect zachytiť v rozlíšení 512 x 424 bodov typu `UINT16`. Táto hodnota znamená vzdialenosť od pomyselného obdĺžnika rovnobežného s prednou stranou senzoru, kolmého na povrch snímaného priestoru, prekrývajúceho Kinect, nie je to teda vzdialenosť od bodu.

Horizontálny uhol snímania je 70.6° a vertikálny je 60°.



Obr. 2.4: Hĺbkový stream s Color ramp z Kinectu .

Infrared

Infračervený senzor používa rovnaký senzor ako hĺbkový, má teda rozlíšenie 512 x 424 bodov typu `UINT16`. Dáta z infračerveného senzoru je možné zobraziť ako čiernobiely obrázok, zobrazením hodnôt 0..65535 do 0..255 a následným dosadením za R,G,B pre daný bod.

Obr. 2.5: Infračervený stream z Kinectu .

Camera space

Súradnicový systém Kinectu je „right-handed“. IR/Depth senzor sú stredom súradnicovej sústavy, kde x z pohľadu senzoru rastie naľavo (t.j. z pohľadu pozorovateľa doprava), y dohora od senzoru a z v smere od senzoru (viď. Obr. 2.1). FullHD kamera je posunutá od IR/Depth senzoru v smere doprava a má aj iný pomer strán ako je možné vidieť na obrázkoch (2.3, 2.4, 2.5). Korektné namapovanie farby na hĺbku nie je úplne triviálne. Microsoft našťastie poskytuje vo svojom API triedu `CoordinateMapper`, ktorá to dokáže (budú ju zrejme obsahovať aj budúce verzie API).

2.2 Microsoft HoloLens v1

Ďalej už len HoloLens. Sú okuliare na rozšírenú realitu vyvíjané spoločnosťou Microsoft. Výhodou okuliarov je, že sú samostatný, plnohodnotný počítač, ktorý nepotrebuje externé zariadenie na výpočty, ako zatiaľ býva zvykom u okuliarov na virtuálnu realitu. Ako prvá generácia sú dostupné na trhu od 30. marca 2016 v dvoch verziách: Developer a Commercial. Nástupcom sú Microsoft HoloLens 2, ktoré boli predstavené 24. februára 2019 a v čase písania tohoto textu prebiehajú pred-objednávky. Tieto údaje boli čerpané z Wikipedia contributors (2019a, HoloLens). Okuliare je možné vidieť na obrázku 2.6.

Obr. 2.6: Microsoft HoloLens prvej generácie. Zdroj: Microsoft.

2.2.1 Rozšírená realita

Z názvu je zrejmé, že ide o rozšírenie, presnejšie rozšírenie reálneho, fyzického sveta okolo nás, prvkami z virtuálneho, digitálneho sveta. Fyzický svet vieme ešte rozdeliť na dve podkategórie a tou je okolie okolo nás a pohyby, gestá, ktorými je možné ovplyvňovať rozšírenú realitu. Inšpirácia z Bray a kol. (2018).

Vďaka senzorom, ktorými HoloLens disponujú, sú schopné určiť danú polohu k objektom v ich okolí (napr. ich polohu v miestnosti, vzdialenosť od nich). Pri pohľade cez okuliare je možné vidieť statické objekty z digitálneho sveta ukotvené na jednom mieste, bez toho aby sa zmenšovali, pohybovali alebo plávali - čo by bolo neprirodzené až nepríjemné pre sledovateľa. Objekty je možné obchádzať, manipulovať s nimi, pozerať sa na nich z rôznych strán a tak podobne.

2.2.2 Technické detaily

Podľa Pelikán (2018) a Zeller a kol. (2018) o hardvérových detailov, HoloLens ako plnohodnotný počítač, s hmotnosťou 579g má nasledujúcu konfiguráciu:

- CPU: Intel Atom 1GHZ (32-bitový)
- HPU (Holographic Processing Unit) - špeciálny holografický koprocesor s pamäťou 1GB
- RAM: 2GB
- Pamäť: 64GB eMMC
- Pripojiteľnosť: Wi-Fi štandardu IEEE 802.11ac a Bluetooth 4.1 Low Energy.

Vstup

- IMU (inertial measurement unit) - je elektronické zariadenie zložené z gyroskopov, akcelerometrov a magnetometrov merajúce sily v jeho okolí.
- 4 kamery na rozpoznávanie okolia

- 1 hĺbková kamera s uhlom záberu 120°x 120°
- 1 ALS (ambient light sensor) - svetelný senzor
- HD kamera
- 4 audio mikrofóny
- Rozpoznávanie
  - Gestá
  - Prostredie, pohľad
  - Hlas, zvuk

Výstup

HoloLens obsahujú dva HD širokouhlé, priesvitné displeje s rozlíšením celkom 2,3 Mpx. Ďalej obsahujú stereo reproduktory v oblastí uši a 3,5mm jack na pripojenie slúchadiel.

Obmedzenia

Hardvér na HoloLens v1 má ešte rôzne obmedzenia. Ako v závere zhrnul Josef Pelikán (2018): „ Hardware má v současné verzi mnohá omezení (např. malé zorné pole, absence GPS a LTE, poměrně slabý procesor), která se již firma Microsoft snaží odstranit ve verzi 2. HoloLens v2 budou na trhu v roce 2019. “

2.2.3 Software

Microsoft HoloLens používajú operačný systém Windows Mixed Reality - Windows 10. Je možné na nich spustiť takmer každú aplikáciu, ktorá je určená pre UWP (Universal Windows Platform).

2.2.4 Ovládanie

HoloLens sú schopné rozpoznávať dve hlavné gestá AirTap a Bloom. Tie je možné neskôr kombinovať na rôzne ďalšie úkony ako je zoom, posúvanie či otáčanie. Nasledujúce informácie boli čerpané z Bray a kol. (2019a) a Bray a kol. (2019b)

AirTap

Je potvrdenie, výber, klik na danú položku. Gesto sa začína ukazovaní ukazovákom dohora s palcom v pravom uhle. Následným spojením ukazováku a palca sa vykoná AirTap.

Bloom

Obdoba Windows button alebo Home button. Otvorenie hlavného/vedľajšieho menu. Gesto začína dlaňou dohora pri spojení všetkých prstov a palca. Následným otvorením dlane sa vykoná Bloom.

Kurzor

Pri nasadení a zapnutí HoloLens je možné vidieť malý krúžok v strede pohľadu. Ten slúži na navigáciu v užívateľskom prostredí, ako v prípade použitia myši na PC.

Hlasové ovládanie

Niektoré gestá je možné nahradiť hlasovým príkazom, ktorý sa zobrazí pri prejdením kurzorom na danú položku. Ďalšou hlasovou službou je Microsoft Cortana, ktorá je takzvaný hlasový asistent vo Windows 10.

Ostatné zariadenia

Ďalšími ovládacími prvkami môžu Motion Controllers alebo pripojená bezdrôtová klávesnica.

2.2.5 Použitie

Využitie Microsoft HoloLens je veľmi široké. Počínajúc priemyslom cez zábavu po komunikáciu a point cloud video - o tom sa dozviete viac v ďalších kapitolách.

3. Použitý software

3.1 C++/WinRT - UWP

UWP (Universal Windows Platform)

Je open source API umožňujúce tvorbu aplikácii pre rôzne platformy s operačným systémom Windows 10. Ide napríklad o PC, Xbox, mobilné zariadenia, HoloLens, VR headsety. UWP je nutnou podmienkou toho, aby aplikácia mohla byť zverejnená v Microsoft Store. Podporuje tvorbu aplikácii v jazykoch C++, C#, VB.NET, F# a Javascript. Tieto údaje boli čerpané z Wikipedia contributors (2019d, UWP).

C++/WinRT

Podľa White a kol. (2019), C++/WinRT je projekcia z Windows Runtime do moderného C++17, ktorá bola uvedená od verzie 10.0.17134.0. Microsoft doporučuje nahradiť predchodcu C++/CX, týmto moderným API. Používa sa pomocou namespace-u `winrt`.

3.2 HoloLens Emulator

Vďaka HoloLens Emulator-u je podľa Microsoft contributors (2019a) možné ladiť a testovať vyvíjanú aplikáciu. Emulátor simuluje aj prostredie v okolí okuliarov HoloLens, takže z pohľadu testovanej aplikácie to vyzerá, že beží na skutočnom zariadení. Užívateľské prostredie (viď Obr. 3.1) sa ovláda pomocou myši, klávesnice alebo iných zariadení.

Emulátor používa technológiu Hyper-V, ktorá podľa Microsoft contributors (2019b) nie je podporovaná na systémoch Windows 10 Home. Je preto potrebné mať verziu Enterprise, Pro alebo Education. Na spustenie emulátoru je potrebné zapnúť súčast systému Windows - Hyper-V.

Obr. 3.1: Hololens Emulátor. Zdroj: Microsoft.

3.3 C#/.NET Framework

.NET Framework je súčasť operačného systému Windows. Konkrétna použitá verzia je .NET Framework 4.6.1. V tejto práci bude použitý programovací jazyk C#/ v jednej z aplikácii.

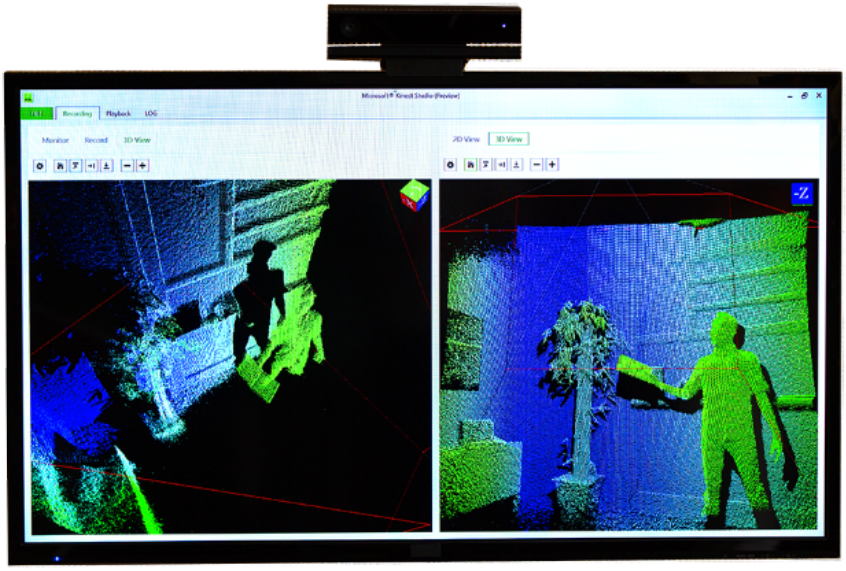
3.4 Kinect for Windows SDK 2.0

Sú nástroje, ktorými je možné ovládať a používať Kinect vo Windowse. Obsahuje SDK Browser v2.0, v ktorom je možné nájsť dokumentáciu, príklady v rôznych programovacích jazykoch a rôzne utility.

Jednou užitočnou z nich je Kinect Configuration Verifier, vďaka ktorej si vie užívateľ overiť, či jeho hardvérová a softvérová konfigurácia je dostačujúca na používanie Kinectu.

3.4.1 Kinect Studio v2.0

Ďalší užitočný program v spomínanom SDK je Kinect Studio v2.0. V ňom je možné prehrávať, nahrávať a editovať nahrávky zo streamov Kinectu (viď Obr. 3.2). Nahrávky generované týmto programom sú vo formáte `*.xef`. Program v prípade prehrávania dokonca emuluje Kinect, čo z pohľadu aplikácie využívajúcej Kinect vyzerá, že je k danému PC pripojený skutočný Kinect.



Obr. 3.2: Kinect studio. Zdroj: Microsoft.

3.5 Knižnica ZeroMQ

Podľa Hintjens (2019), je asynchrónna, cross-platform sieťová knižnica, ktorej jadro je napísané v jazyku C++. Knižnica je message-oriented a poskytuje message frontu. Je to high-level wrapper nad socketmi, ktorý sa stará o fragmentáciu, prenos a následné poskladanie dát na druhej strane. Pokiaľ sa nejaké pakety nedoručia, opakuje sa ich prenos. Ďalej je garantované doručenie všetkých častí danej message, no v prípade, že sa tak nestane, nedoručí sa žiadna. „ZeroMQ guarantees to deliver all the parts (one or more) for a message, or none of them.“

V tejto práci je použitá konkrétne libzmq pre jazyk C++, čo je vlastne core engine ZeroMQ a clrzmq4 čo je .NET wrapper nad natívnou libzmq. Tým je vyriešená komunikácia medzi aplikáciami vytvorených v jazykoch C# a C++.

3.6 Class StreamSockets

Je natívna Windows sieťová trieda, ktorá je vo `Windows.Networking.Sockets` namespace. Podporuje sieťovú komunikáciu cez TCP a Bluetooth sockety.

V tejto práci bola táto trieda použitá z vyššie spomínaného C++/WinRT namespace-u.

3.7 Knižnica FMCore

Knižnica zabezpečujúca uloženie a serializáciu hierarchických 3D dát, ich rendering a užívateľské rozhranie v aplikáciach na HoloLens a iných systémoch.

Je proprietárny softvér vyvíjaný spoločnosťou Pocket Virtuality a.s., o ktorom nie je možné zverejňovať akékoľvek detailnejšie informácie.

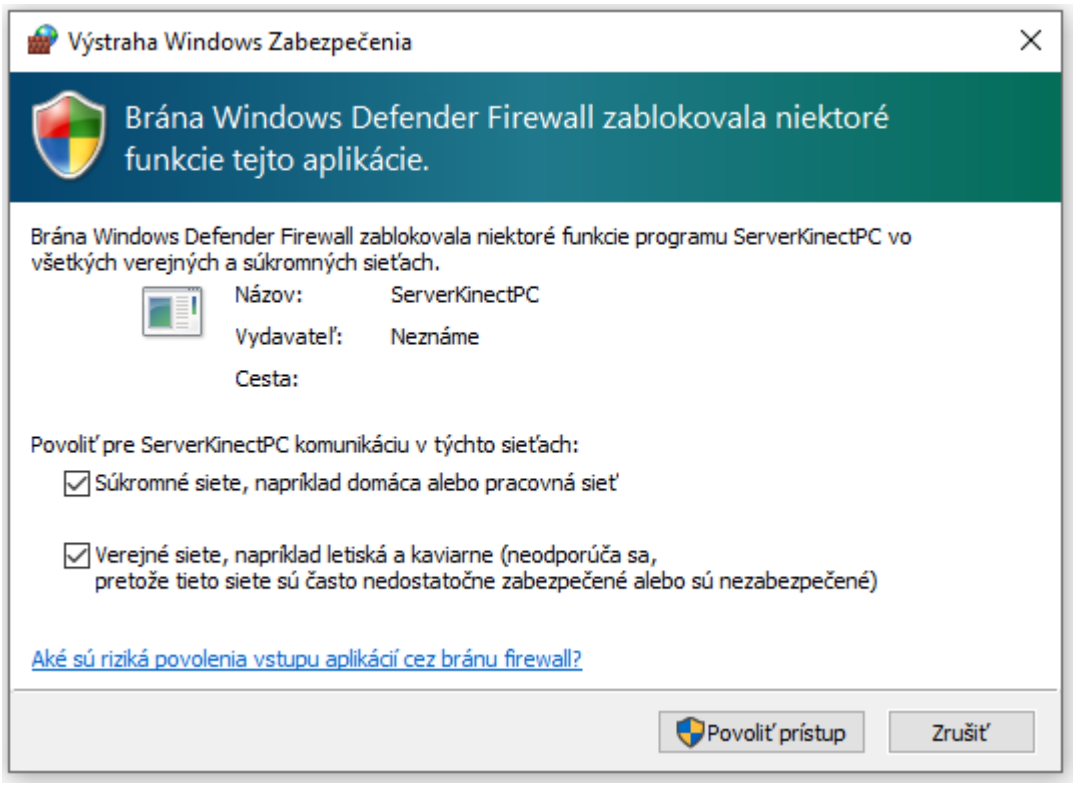
V prípade otázok kontaktujte firmu Pocket Virtuality a.s., napríklad vedúceho práce:
`Josef Pelikán <josef.pelikan@pocketvirtuality.com>`

4. Užívateľská príručka

V tejto kapitole si bližšie povieme, ako sa aplikácie z tejto práce používajú. Pred používaním aplikácii je doporučené túto príručku dôkladne prečítať.

4.1 ServerKinectPC

Pre spustenie aplikácie je potrebné, aby počítač obsahoval systém Windows a .NET Framework 4.6.1, ktorý je už buď súčasťou systému alebo je možné ho získať na stránkach Microsoftu. Ďalej je potrebné aby bol ku počítaču pripojený Kinect a aby splňoval systémové požiadavky, ktoré sú uvedené v kapitole vyššie. Počas behu programu je potrebné povoliť komunikáciu v sietiach, ako môžeme vidieť na obrázku 4.1.

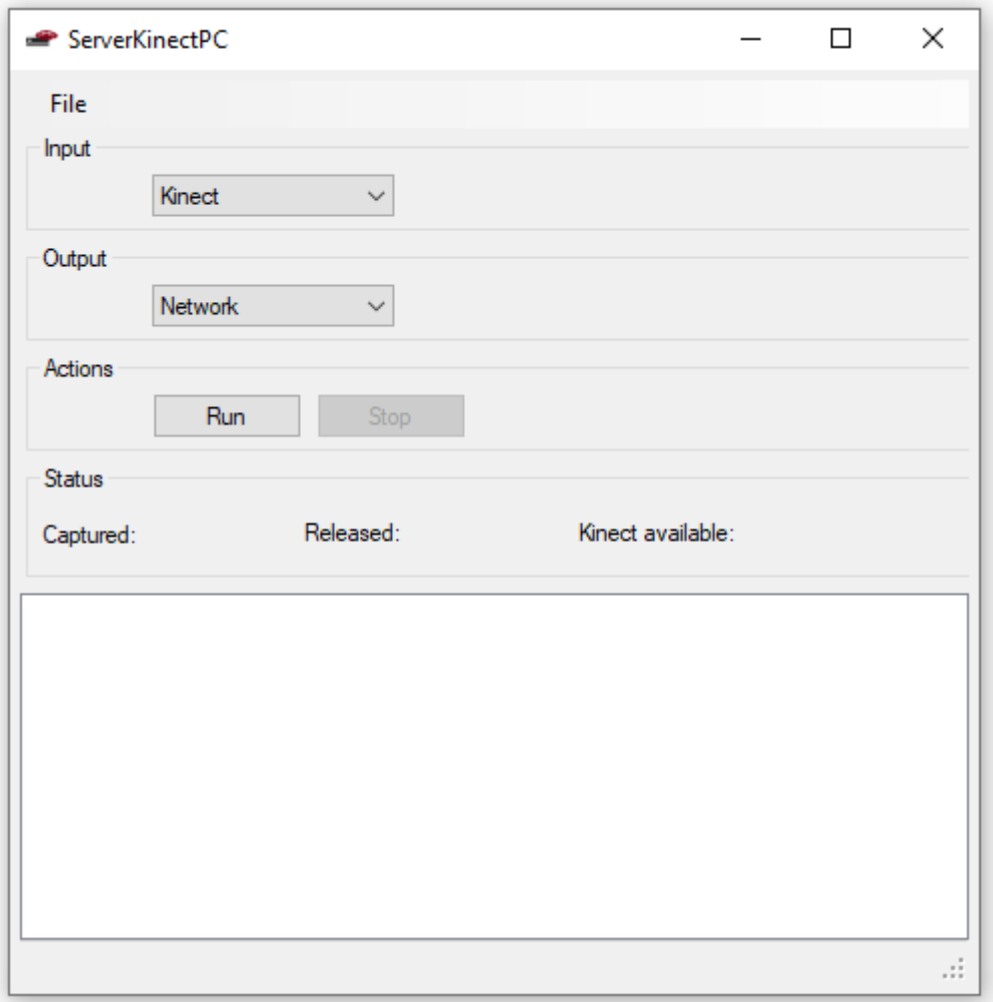


Obr. 4.1: Pridanie výnimky pre sieťovú komunikáciu.

Aplikácia obsahuje 5 hlavných sekcii ako je možné vidieť na obrázku 4.2:

- Input - combo box na výber vstupu (zdroja dát).
- Output - combo box na výber výstupu.
- Actions - tlačidla na vykonanie dostupných akcií.
- Status - zobrazuje základné štatistické údaje o tom ako prebieha daná operácia.

  - Captured - zobrazuje počet frame-ov, ktoré sa podarilo zachytiť zo vstupu.
  - Released - zobrazuje počet frame-ov, ktoré sa podarilo previesť na výstup.
  - Kinect available - zobrazuje či je dostupný Kinect senzor. (Aktívne len v prípade výberu možnosti Input>Kinect.)
- Status Console - zobrazuje log o stave operácii, ktoré prebiehajú.



Obr. 4.2: Hlavné okno aplikácie ServerKinectPC.

Po vybratí vstupu a výstupu, kliknutím na tlačidlo „Run“ sa spustí daná operácia. Počas behu nie je možné meniť formu vstupu/výstupu. Operácia sa ukončí kliknutím na tlačidlo „Stop“.

4.1.1 Funkcie

Aplikácia obsahuje dve metódy vstupu a dve metódy výstupu. Ako bolo spomenuté vyššie, ich výber prebieha cez combo boxy.

Kinect - vstup

Aplikácia bude zaznamenávať dynamický point cloud z Kinectu. Pokiaľ je dostatok svetla, point cloud je farebný. Sleduje sa intenzita okolitého svetla a pokiaľ je svetelnosť nízka, adaptívne sa prepne na čiernobiely (gray-scale) point cloud z infračerveného senzoru a tým pádom je možné zaznamenávať point cloud aj v úplnej tme. V prípade, že svetelnosť stúpne, tak point cloud začne byť znovu farebný.

Na obrázkoch 4.6 a 4.7 môžeme vidieť farebný a čiernobiely point cloud, v závislosti od svetelných podmienok v okolí Kinectu.

Directory - vstup

Po kliknutí na „Run“ sa aplikácia opýta na priečinok, v ktorom sú point cloud dáta. Po výbere sa spustí postupne prehliadanie danej zložky. Pokiaľ sa dôjde na koniec, začne sa prehliadať od začiatku. Nedoporučuje sa akokoľvek meniť štruktúru priečinka s point cloud dátami.

Network - výstup

Zapne sa server, ktorý je pripravený odpovedať na požiadavky klientov. Na server sa môže pripojiť viac klientov, ktorý si budú žiadať frame-y z dynamického point cloud-u. V prípade, že server nemá frame, o ktorý si klient žiada, pošle prázdnu správu. Klient tak vie, že nestratil spojenie, ale musí čakať.

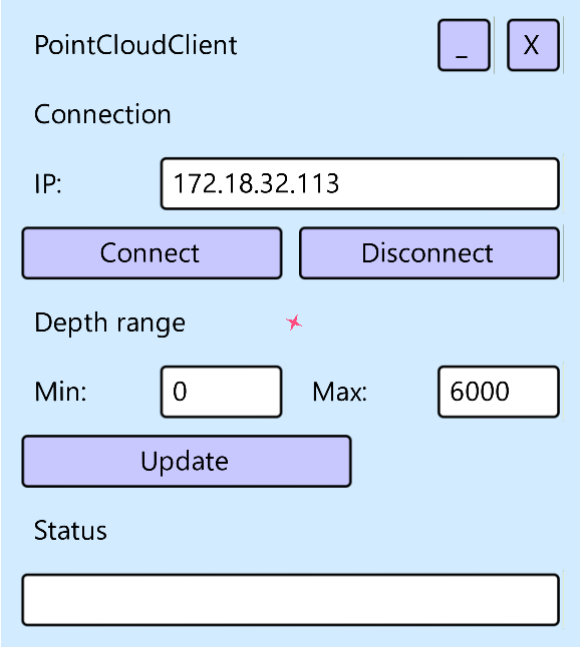
Directory - výstup

Po kliknutí na „Run“ sa aplikácia opýta na priečinok, do ktorého sa majú zapísať point cloud dáta zo zdroja. Dáta sa zapisujú do priečinka postupne, ako ich vracia vstup. Ich štruktúra je každý súbor na frame a názov súboru je časová značka (time-stamp) kedy bol daný frame zaznamenaný, s príponou `.pcl`.

4.2 PointCloudClient

Pre spustenie aplikácie sú potrebné okuliare HoloLens, poprípade HoloLens Emulator. Je dôležité aby obsahovali Windows 10 vo verzii 10.0.17763.0 (min. 10.0.17134.0). Pri spustení alebo počas behu bude potrebné povoliť prístup ku používaniu siete. Aplikáciu nie je možné spustiť na bežnom desktopovom systéme s Windows 10.

Po spustení aplikácie sa zobrazí zelený obdĺžnik, ktorý je vygenerovaný statický point cloud. Menu aplikácie sa zobrazí pomocou dvojnásobného AirTap (dvojkliku) alebo klávesou F5 v prípade, že je pripojená klávesnica. Skryje sa obdobnými spôsobmi alebo stlačením na tlačidlo "_". Aplikácia sa ukončí pomocou tlačidla "X".



Obr. 4.3: Hlavné okno aplikácie PointCloudClient.

Menu obsahuje 3 hlavné sekcie:

- Connection - do textového poľa je možné napísať IP adresu a pomocou nej sa pripojiť alebo odpojiť od daného serveru.
- Depth range - v textových poliach je možné nastaviť počas behu minimálnu a maximálnu hĺbku v milimetroch, ktorá sa bude zobrazovať.
- Status - v textovom poli sa zobrazujú hlášky o statuse serveru.

4.2.1 Funkcie

Funkcie sa obsluhujú pomocou príslušných ovládacích prvkov opísaných v sekcii vyššie.

Connection

Do textového pola je potrebné zadať IP adresu servera. Pomocou tlačidla „Connect“ sa zaháji to, že aplikácia žiada server o frame-y a prijaté potom vykreslí. Pokiaľ server nemá daný frame, tak odpovie prázdnou správou, klient na to reaguje tak, že nie je potrebný re-connect, len musí chvíľu počkať. Tým sa zachová živé spojenie. Kliknutím na tlačidlo „Disconnect“ sa zastaví prenos a ostane vykreslený posledný prijatý frame.

V prípade straty spojenia klient počká 2,5 sekundy a automaticky sa pokúsi pripojiť znovu. To sa zopakuje 3 krát a pokiaľ sa spojenie nenadviaže tak sa zastaví. Potom je nutný re-connect od užívateľa. To isté nastáva aj pokiaľ je zadaná zlá IP adresa alebo daný server neexistuje/neodpovedá.

Depth range

Po úspešnom pripojení je možné počas behu dynamického point cloud-u meniť jeho minimálnu a maximálnu hĺbku, ktorá ma byť viditeľná. Do príslušných textových polí sa zadá hodnota a zmena sa prejaví po kliknutí na tlačidlo „Update“.

Hodnoty sú v milimetroch, je to vzdialenosť od Kinect senzoru. Zadávajte hodnoty od 0 do 65535.

Pre vysvetlenie, ak chceme napríklad zobraziť osobu, ktorá sa nachádza 1,5 m od senzoru a chceme, aby pozadie za ňou nebolo vidno, tak možná konfigurácia je Min: 0 a Max: 3000.

Na nasledujúcich obrázkoch 4.4 a 4.5 môžeme vidieť rôzne nastavenia zobrazovanej hĺbky. Na obrázkoch 4.6 a 4.7 môžeme vidieť farebný a čiernobiely point cloud, v závislosti od svetelných podmienok v okolí Kinectu.

Obr. 4.4: Point cloud s hĺbkou 6m.

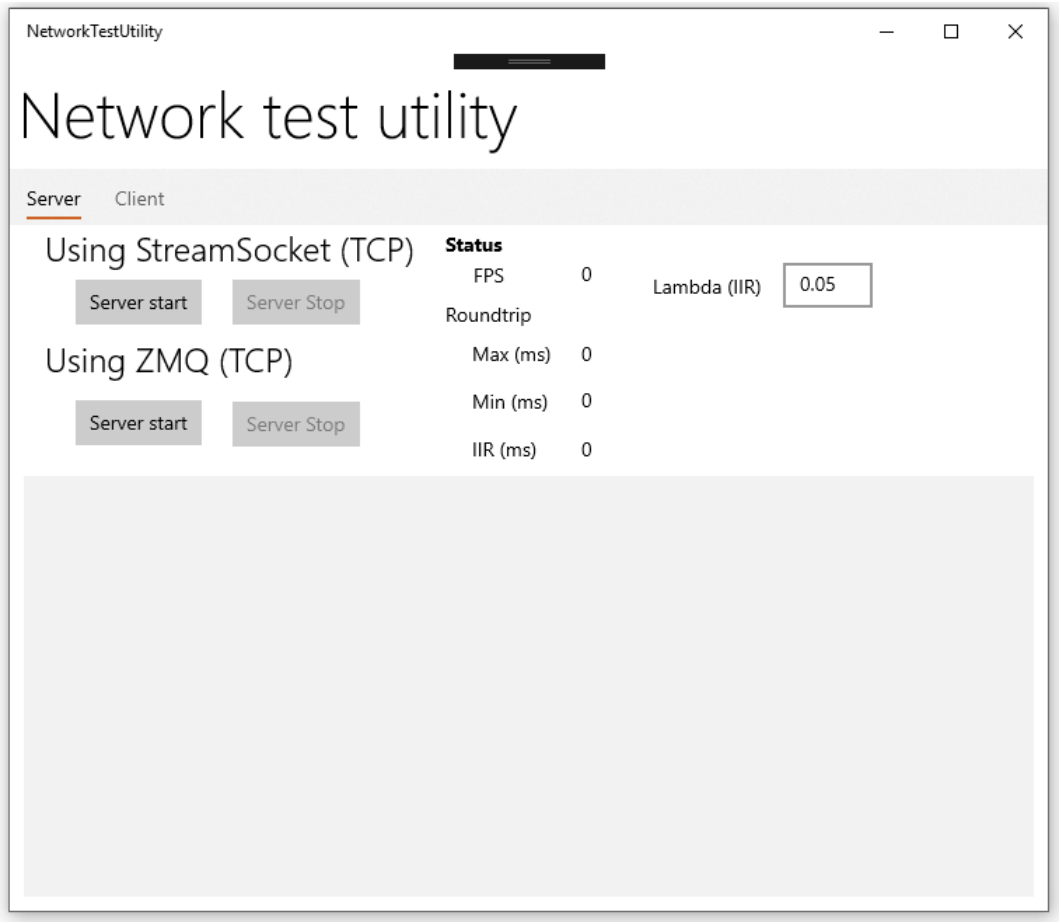
Obr. 4.5: Point cloud s hĺbkou 3m.

Obr. 4.6: Point cloud farebný.

Obr. 4.7: Point cloud čierno-biely - „nočné videnie“.

4.3 NetworkTestUtility

Aplikácia je typu Universal Windows Platform. Je možné ju teda spustiť na desktope, ako aj na okuliaroch HoloLens, poprípade Hololens Emulator. Je dôležité aby obsahovali Windows 10 vo verzii 10.0.17763.0 (min. 10.0.17134.0). Pri spustení alebo počas behu bude potrebné povoliť prístup ku používaniu siete.



Obr. 4.8: Hlavné okno aplikácie NetworkTestUtility.

Aplikácia obsahuje 4 hlavné sekcie:

- Dve hlavné karty:
  - Server - obsahuje tlačidlá na obsluhu servera pomocou danej sieťovej knižnice.
  - Client - obsahuje tlačidlá na obsluhu klienta pomocou danej sieťovej knižnice.
- Status - zobrazuje základné štatistické údaje o tom, ako prebieha daná operácia.
- Konzola - v list boxe sa zobrazujú informácie o stave serveru/klienta.

4.3.1 Funkcie

Funkcie sa obsluhujú pomocou príslušných ovládacích prvkov opísaných v sekcii vyššie.

Server

Obsahuje tlačidlá na obsluhu serveru pomocou príslušnej knižnice: ZeroMQ alebo StreamSockets. Nedoporučuje sa, aby na jednom stroji bežalo viac serverov súčasne.

Client

Do „IP“ text boxu je potrebné napísať IP adresu serveru, na ktorý sa má klient pripojiť. V „Message size“ text boxe je možné meniť veľkosť dát, o ktoré si bude klient žiadať na serveri. Ďalej obsahuje tlačidlá na obsluhu klienta pomocou príslušnej knižnice: ZeroMQ alebo StreamSockets. Nedoporučuje sa, aby na jednom stroji bežalo viac klientov súčasne.

Status

V sekcii status je možné vidieť štatistiky týkajúce sa serveru/klienta. Všetky sú v milisekundách. V prípade serveru hodnoty v sekcii „Roundtrip“ popisujú ako dlho je server nečinný. V prípade klienta tieto hodnoty popisujú aký čas prejde od toho keď si klient požiada nejaký frame, do vtedy kým ho dostane. Minimum a maximum sa po nejakom počte operácii resetuje, aby sa upravili veľké výkyvy. Hodnota IIR je filter s nekonečnou odozvou popísaný v definícii 2.

5. Server

Touto kapitolou začína sekvencia interných programátorských dokumentácii k softvérovým dielam, ktoré boli vytvorené v rámci bakalárskej práce.

5.1 Základné informácie

Na stranu získavania, spracovania dát a odpovedania na požiadavky klientov sme navrhli serverovú aplikáciu s užívateľským rozhraním, ktorá používa .NET Framework 4.6.1 s programovacím jazykom C#. Aplikáciu je možné spustiť len na platforme x64, ktorá obsahuje spomínaný framework (viac v sekcii 2.1).

Okrem štandardných knižníc .NET Frameworku, sú použité ešte dve ďalšie knižnice:

- Microsoft.Kinect - pre obsluhu Kinectu (viac sekcii v 3.4).
- clrzmq4 (ZeroMQ) - pre sieťovú komunikáciu (viac sekcii v 3.5).

Po štarte aplikácie je potrebné získať spôsoby vstupu a výstupu dát. Obsahuje 4 funkcionality:

- získavanie dát z Kinectu
- získavanie dát z priečinka
- distribúciu získaných dát do siete
- zapisovanie získaných dát na disk do priečinka

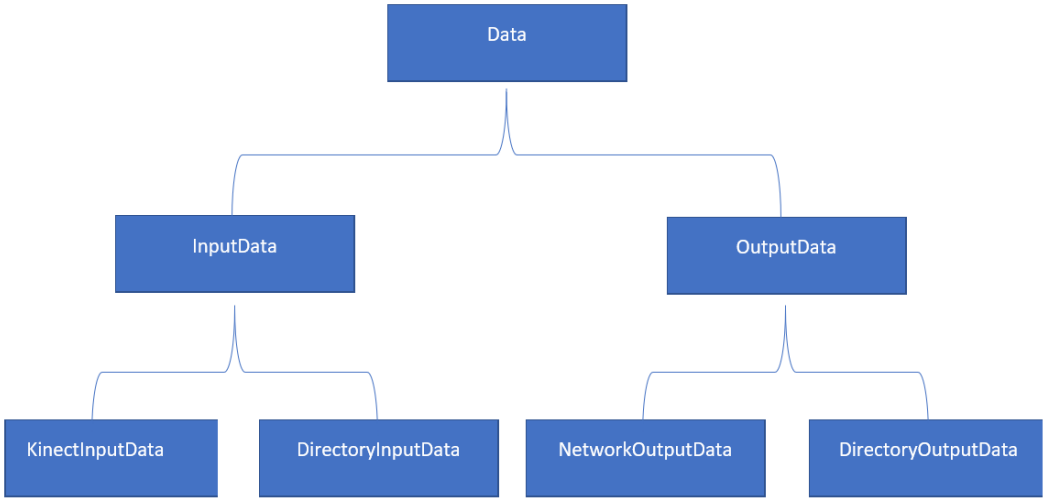
5.2 Prehľad tried v aplikácii.

Pripomíname, že naše dáta sú point cloud video, čo je vlastne postupnosť point cloud snímkov nasnímaných v rôznych časových okamihoch. Každý snímok je teda nejaké pole `byte`, ktoré vieme získať spracovaním surových dát z Kinectu alebo prečítať zo súboru, kde sú uložené v podobe jeden frame/súbor. Táto aplikácia má na starosti správu takýchto dátových operácii.

Pre predstavu si uvedieme prehľad tried, ktoré sa starajú o vstup, výstup a spracovanie dát.

5.2.1 Vstup a výstup dát

Na obrázku 5.1 môžeme vidieť hierarchiu tried. Triedy `Data`, `InputData` a `OutputData` sú abstraktné, zvyšné sú konkrétne, ktoré implementujú danú funkcionalitu.



Obr. 5.1: Hierarchia tried serverovej aplikácie.

Data

Obsahuje položky:

- `public abstract void Stop();` - metóda, ktorá sa volá na zastavenie danej operácie.
- `protected int count;` - udáva počet spracovaných frame-ov.
- `protected MainForm ui;` - referencia na užívateľské rozhranie, berie sa v konštruktore.

InputData

Ako parameter konštruktora berie objekt `MainForm`, pre komunikáciu s užívateľským rozhraním. Obsahuje metódu:

- `public abstract Cache InputNext(long timeStamp);` - získa zo zdroja dát ďalší frame a vráti ho. (Podľa time-stamp-u by mala vrátiť prvý novší, ako je time-stamp.)

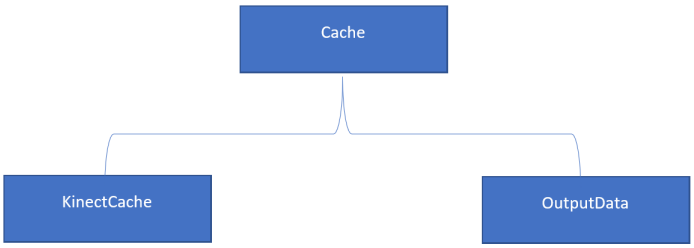
OutputData

Ako parameter konštruktora berie objekt `MainForm`, pre komunikáciu s užívateľským rozhraním. Obsahuje metódu:

- `public abstract void OutputNext(InputData inputData);` - ako parameter berie ľubovoľnú inštanciu triedy `InputData`.

5.2.2 Spracovanie dát

Nasledujúce triedy na obrázku 5.2 slúžia na ukladanie a následne pripravenie dát do výsledného formátu.



Obr. 5.2: Hierarchia tried serverovej aplikácie.

Cache

Je abstraktná trieda, ktorá spracováva a ukladá dáta. Obsahuje položku:

- `public abstract byte[] Release(int count);` - po zavolaní tejto metódy, by mali byť dáta pripravené vo výslednom formáte.

5.3 Rozdelenie aplikácie

Aplikáciu možno rozdeliť na dve hlavné súčasti a tou je užívateľské rozhranie a spracovanie, obsluha dát. Obidve tieto súčasti bežia na samostatných vláknach. Oddelenie má výhodu v tom, že užívateľské rozhranie ostane responzívne a je možné z neho ovládať chod pracovného vlákna.

Užívateľské rozhranie

Vytvorené pomocou Windows Forms, čo je podľa Microsoft (2017) množina knižníc, ktorá sa stará o užívateľské rozhranie vo Windowse s udalosťami riadenou architektúrou. Pri zmenách užívateľského rozhrania sa používa `BeginInvoke()`, aby zmeny prebiehali z vlákna užívateľského rozhrania.

Pracovné vlákno

Ovláda sa pomocou metód `void StartWork()` a `void StopWork()`. Príslušné ovládacie prvky užívateľského rozhrania (napríklad tlačidlá a pod.) tieto metódy volajú. V užívateľskom rozhraní sú ešte dve položky:

- `Thread workerThread;` - pracovné vlákno.
- `volatile bool workerThreadIsRunning;` - premenná udávajúca, či beží daný výpočet.

V metóde `void StartWork()` sa užívateľom vybrané vstupné a výstupné dáta predajú do vlákna (kde už prebieha celá funkcionalita) a vlákno sa spustí.

Vo vlákne sa na základe poskytnutých informácii zostroja už lokálne objekty `InputData` a `OuputData`. Vlákno pracuje s abstraktnými objektami, čiže pri správnej implementácii je možné rozšíriť možnosti vstupu a výstupu dát. (Napríklad v prípade nového Kinectu alebo zápisu dat do iného formátu.)

Následne sa spustí:

```
while (workerThreadIsRunning)
{
        outputData.OutputNext(inputData);
}
```

Výsledné naloženie s dátami už závisí na implementácii konkrétnych tried vstupu a výstupu.

V metóde `void StopWork()` sa položka `workerThreadIsRunning` nastaví na `false` a počká sa na dobehnutie vlákna určitý časový limit. Pokiaľ dovtedy neskončí, tak sa vlákno preruší.

5.4 Konkrétne implementácie

Triedy DirectoryInputData a DirectoryOutputData

Implementácia konkrétnych tried pre prácu s directory je bežné načítanie, zapísanie do súboru. Formát súborov je taký, ako je popísané v sekcii 1.2. V prípade vstupu sa číta od začiatku. V prípade výstupu sa zapisuje pokiaľ súbory už nie sú vytvorené.

Trieda KinectInputData

Získava dáta z Kinectu. Pre každý frame ukladá do objektu `KinectCache` získané surové dáta z Kinectu. Taktiež sa sleduje, koľko svetla je v okolí a podľa toho bude vo výslednom poli buď farebný point cloud alebo čiernobiely point cloud z IR senzoru. Pokiaľ niekto požiada o daný frame, metóda `Release()` spracuje surové dáta a pripraví ich pre renderer, dodá hlavičku a vytvorí z toho jedno veľké pole bytov. Spracovanie prebieha len pri prvom zavolaní `Release()`, pri ďalších volaniach sa už len vrátia spracované dáta.

Obsahuje tiež frontu posledných objektov `KinectCache`, čo je prispôsobená `ConcurrentQueue<T>`, do ktorej sa pridávajú prvky, a pokiaľ sa naplní kapacita, tak sa najstarší z nej zmaže. Metóda `InputNext()`, LINQ-ovým dotazom nájde vo fronte podľa daného time-stamp-u najstarší taký, ktorý ma novší time-stamp, ako je ten poskytnutý. Fronta slúži na to aby server mohol odpovedať viacerým klientom a zároveň keď sa sieťový prenos zbrzdí, je tu ešte možnosť pre klienta spätného „do-načítania“ frame-ov.

Trieda NetworkOutputData

Distribuuje získané `InputData` dáta v sieti. Implementácia používa clrzmq4 (viac v sekcii 3.5). Zavolaním `OutputNext()` prebehne jeden `PollIn()` a pokiaľ obdrží nejakú požiadavku od klienta, tak ju spracuje a pošle odpoveď. Teda pokúsi sa z objektu `InputData` získať požadovaný frame. Pokiaľ ho získa, pošle ho. V prípade že sa nenájde frame, ktorý by vyhovoval, tak sa pošle prázdny. Klient tak potom vie, že nestratil spojenie, len si musí chvíľu počkať.

6. Klient

6.1 Základné informácie

Na klientskú stranu sme navrhli aplikáciu, ktorá je typu Universal Windows Platform, používajúca programovací jazyk C++/WinRT (viac v sekcii 3.1). Aplikáciu je možné spustiť len na HoloLens - architektúra x86.

Hlavnou funkcionalitou je real-time vykresľovanie point cloud videa. Ďalej je možné meniť rozsah zobrazovaného point cloud-u.

Po spustení sa zobrazí zelený obdĺžnik v priestore pred užívateľom. Obsahuje užívateľské rozhranie, pomocou ktorého sa ovláda. Je potrebné sa pripojiť na server, ktorému bude aplikácia posielať požiadavky a následne vykresľovať prijaté dáta.

Okrem štandardných C++ knižníc a namespace-u `winrt` sú použité ešte dve ďalšie knižnice:

- libzmq (ZeroMQ) - zabezpečuje sieťovú komunikáciu (viac v sekcii 3.5).
- FMCore - zabezpečuje uloženie point cloud-u ako 3D dát, ich rendering a užívateľské rozhranie (viac v sekcii 3.7).

6.2 Prehľad tried v aplikácii.

Pre predstavu si uvedieme prehľad tried, ktoré sa starajú o sieťovú komunikáciu a rendering.

Obr. 6.1: Hierarchia tried klientskej aplikácie.

PointCloudScene

Je trieda, ktorá udržuje všetky potrebné údaje o scéne. Parametrami konštruktora sú referencie na objekty pre FMCore (3.7) a `int msg_code`, čo udáva interné číslo pre daný textbox v užívateľskom rozhraní.

Spomenieme tie najdôležitejšie metódy:

- `static FMcore::NodeRef CreatePointCloudScene()` - pomocou tejto metódy sa vytvorí základný zelený point cloud obdĺžnik pri štarte aplikácie.
- `void LogToUI(std::wstring message)` - má za úlohu vypísať daný text do príslušného text boxu v užívateľskom rozhraní.

- `void RenderData(std::shared_ptr<zmq::message_t> reply)` - z príslušnej odpovede `zmq::message_t` zo servera sa pokúsi spracovať a vyrenderovať dáta.

- `bool UpadteDepthRange()` - metóda sa pokúsi naparsovať a zmeniť rozmedzie zobrazovaného point cloud-u. V prípade neplatných hodnôt sa použijú predvolené.

Ďalej ešte táto trieda obsahuje položky a metódy, ktoré sa používajú vo vyššie spomenutých metódach. Taktiež obsahuje shadere, ktoré sa použijú pri renderingu. Položky `minDepth` a `maxDepth` sú typu `std::atomic<float>` aby ich bolo možné používať z viacerých vlákien.

6.2.1 Header

Je trieda, ktorá sa stará o naparsovanie a uloženie informácii o point cloud frame zo správy.

Obsahuje metódu:

- `bool ParseData(std::shared_ptr<zmq::message_t> reply);` - vráti `true` pokiaľ sa jej podarilo naparsovať dáta zo správy `zmq::message_t`.

Pokiaľ `ParseData()` vráti `true` tak položky headru sú v jej dátových položkách a je možné k nim pristupvať. Ďalej ešte obsahuje privátne metódy, ktoré sa starajú o presné naparsovanie daného typu.

6.2.2 ZMQClient

Je trieda, ktorá sa stará o sieťovú komunikáciu. V konštruktore si berie referenciu na inštanciu `PointCloudScene` pre komunikáciu so scénou.

Spomenieme tie najdôležitejšie metódy:

- `void SendRequest(zmq::socket_t & socket)` - pošle požiadavku do daného socketu. Požiadavka je číslo typu `INT64` čo je time-stamp posledného point cloud frame-u, ktorý sme zo serveru dostali.

- `void ReceiveData(std::shared_ptr<zmq::message_t> reply)` - v tejto metóde sa volá `RenderData()` zo scény. Teda prijaté dáta sa pokúsia vyrenderovať.

- `void Start(const std::wstring& ipAddress)` - táto metóda spustí samostatný task, v ktorom si klient opakovane vo `while` cykle, žiada server o dáta, ktoré potom spracováva a renderuje. Po poslaní požiadavku potom prebieha čakanie pomocou `zmq::poll()` v prípade, že nič neprišlo, tak sa klient automaticky pokúsi pripojiť znova a požiadavku opakovať. To všetko sa opakuje trikrát. Ak sa nepodarí pripojiť aj tak, preruší sa.

- `void Stop()` - v metóde sa pomocou príznaku typu `std::atomic<bool>` zastaví task a počká sa na jeho dokončenie.

6.2.3 PointCloudClientListener a PointCloudClientMain

Sú triedy potrebné na spoluprácu našej aplikácie s FMCore (3.7).

6.3 Užívateľské rozhranie

Je vytvorené pomocou FMCore (viď na obrázku 4.3). Z obsluhy užívateľského rozhrania vedie `std::shared_ptr<ZMQNetwork::ZMQClient>`, ktorý v prípade kliknutia „Connect“ nastaví a spustí klienta. V prípade „Disconnect“ sa do danej referencie priradí `nullptr` a následne sa ukončia a vyčistia zdroje. Pomocou ovládacích prvkov „min“ a „max“ nastavíme zobrazovanú vzdialenosť point cloud-u. Stlačením tlačidla „Upadte“ sa prejaví daná zmena.

7. Sieťová utilita

7.1 Zakladné informácie

Na testovanie sieťového prenosu sme ešte na začiatku experimentálne navrhli aplikáciu, ktorá je typu Universal Windows Platform, používajúca programovací jazyk C++/WinRT (viac v 3.1). Aplikáciu je možné spustiť teda na desktope, ako aj na HoloLens. Aplikácia generuje náhodne dáta špecifikovanej veľkosti a distribuuje ich v sieti, čím sa simuluje prenos napríklad frame-ov point cloud videa.

Má dve hlavné funkcionality, dokáže byť server alebo klient. Ďalej nám poskytuje štatistiku daného prenosu. V prípade klientov meră roundtrip od poslania requestu, do vtedy kým nepríjme odpoveď. V prípade serveru sa merá doba nečinnosti serveru.

Používa dve sieťové knižnice:

- libzmq (ZeroMQ) - zabezpečuje sieťovú komunikáciu. Linkuje sa ako dynamická knižnica (viac v 3.5).
- StreamSockets - zabezpečuje sieťovú komunikáciu. Je sieťová trieda v `namespace winrt::Windows::Networking::Sockets` (viac v 3.6).

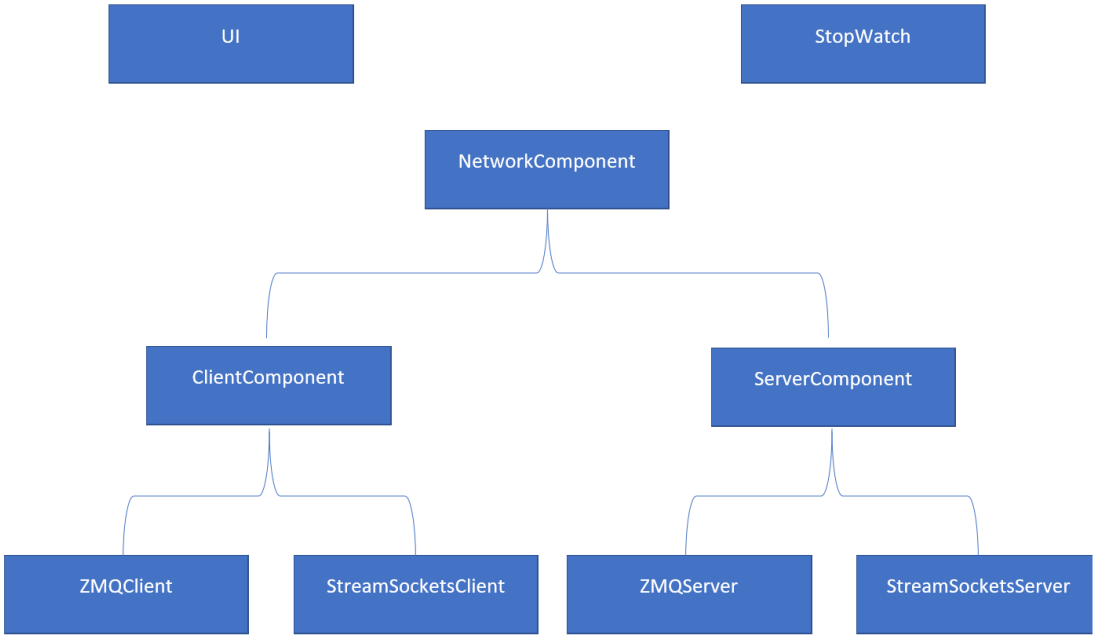
Lambda

Lambda znamená konštanta pre IIR filter alebo tiež filter s nekonečnou odozvou. Vzorec sa prepočítava podľa nasledujúcej definície.

Definícia 2 (Filter s nekonečnou odozvou)**.** *Alebo po anglicky IIR (infinite impulse response). Nech a je aktuálna hodnota a b je nová hodnota.*
Potom: $a = a \ * (1 - \lambda) + b * \lambda, \lambda \in \mathbb{R}$
(Inšpirácia z Wikipedia contributors (2019b, IIR).)

7.2 Prehľad tried v aplikácii.

Sieťové prvky majú medzi sebou veľa spoločných metód a vlastností. `NetworkComponent`, `ClientComponent` a `ServerComponent` sú abstraktné triedy, ostatné triedy už implementujú dané metódy pre konkrétne použitie. Hierarchia tried vyzerá následovne:



Obr. 7.1: Hierarchia tried sieťovej utility.

UI

Je trieda, ktorá obsahuje referencie a metódy na obsluhu užívateľského rozhrania.

- `void LogToListBox(const hstring message)` - postará sa o to aby daná správa typu `winrt::hstring` bola pridaná zo list boxu - „serverovej konzoli“.
- `void ChangeRoundtrips()` - postará sa o to, aby dané roundtripy boli zapísané do užívateľského rozhrania.

Zmena v prvkoch užívateľského rozhrania prebieha za pomoci volania `.Dispatcher().RunAsync()`, aby k tomu dochádzalo z vlákna daného formulára.

StopWatch

Je trieda, ktorá nám vytvára API pre uľahčenie používania triedy `std::chrono::time_point<std::chrono::steady_clock>`.
Obsahuje metódy:

- `void Start()` - zapne stopky.
- `double Stop()` - zastaví stopky a vráti uplynutý čas typu `double`.

NetworkComponent

Obsahuje všeobecné metódy a položky pre prácu so sieťou. Spomenieme tie najdôležitejšie metódy a položky:

- `std::shared_ptr<UI> output;` - referencia na inštanciu triedy `UI`, pre komunikáciu s užívateľským rozhraním.
- `std::shared_ptr<StopWatch> stopwatch;` - pre prístup k stopkám z viacerých metód.
- `void OnTick();` - volá sa pri „tiku“ timer-u každú sekundu, čím sa aktualizujú štatistické hodnoty.
- `void UpdateRoundtrip(const double currentRoundTrip)` - aktualizuje hodnoty roundtripu.
- `virtual void Stop() = 0;` - metóda pre zastavenie, potomkovia budú musieť implementovať vlastnou implementáciou.

ClientComponent

Rozširuje `NetworkComponent` metódami a položkami pre obsluhu sieťového klienta.

Spomenieme dôležité metódy:

- `virtual void Start() = 0;` - implementácia inicializácie a začatia posielania požiadavkov na server s danými parametrami sa necháva na konkrétnu implementáciu.

ServerComponent

Rozširuje `NetworkComponent` metódami a položkami pre obsluhu sieťového serveru.

Spomenieme dôležité metódy:

- `static void fill_with_random_data();` - do príslušného vektoru `data` vygeneruje náhodne dáta danej veľkosti.
- `virtual void Start() = 0;` - implementácia inicializácie a štartu serveru s danými parametrami sa necháva na konkrétnu implementáciu.

7.3 Rozdelenie aplikácie

Aplikáciu možno rozdeliť na dve hlavné súčasti a tou je užívateľské rozhranie a obsluha siete. Obidve tieto súčasti bežia na samostatných vláknach. Oddelenie má výhodu v tom, že užívateľské rozhranie ostane responzívne a je možné z neho ovládať chod sieťového vlákna.

Užívateľské rozhranie

Je vytvorené pomocou XAML. Z `MainPage` vedú `std::unique_ptr` na dané inštancie tried. V prípade kliknutia na „start“ špecifickej komponenty sa vytvorí nová inštancia `std::unique_ptr` na danú komponentu. Pri „stop“ sa naopak tento pointer zahodí priradením `nullptr`. V užívateľskom rozhraní je možné meniť IP adresu serveru, veľkosť daného frame-u a konštantu lambda. Tieto hodnoty

sa predávajú metóde `Start()`. Spomínané hodnoty nie je možné meniť počas behu.

Obsluha siete

V daných komponentách bežia samostatné tasky, v ktorých bežia while cykly. Zastaviť ich je možné pomocou `concurrency::cancellation_token`.

7.4 Meranie siete

Nasledujúce meranie prebiehalo pri konfigurácii:
`x86 | Release | Start without debugging`
medzi desktopom a HoloLens.

Veľkosť frame-u (Bytes)	**StreamSockets**	**ZeroMQ**
1024 (1kB)	18	300
2048 (2kB)	16	290
4096 (4kB)	17	280
8192 (8kB)	15	220
16384 (16kB)	15	210
32768 (32kB)	26	180
65536 (64kB)	18	130
131072 (128kB)	17	88
262144 (256kB)	15	54
524288 (512kB)	5	30
1048576 (1MB)	4,75	16
2097152 (2MB)	3,6	8,5
4194304 (4MB)	2	3,2
8388608 (4MB)	1,2	1,6
16777216 (16MB)	0,75	0,9

Pozn: Udáva FPS(frames per second) pri danej veľkosti. $FPS = \frac{1}{IIR}$. *Viac v definícii 2.*

Tabuľka 7.1: Tabuľka merania siete medzi desktopom a HoloLens. Pozor! Utilita bola vytvorená na experimentálne účely a implementácia nemusí byť optimálna pre StreamSockets.

Záver

7.5 Zhrnutie

V tejto práci sme použili zariadenia Microsoft Hololens v1 a Microsoft Kinect v2. Zaoberali sme sa spôsobmi od toho, ako dáta získať z Kinectu, až po vykreslenie point cloud videa v HoloLens.

Cieľom tejto práce bolo navrhnúť spôsob ako v reálnom čase získať point cloud video, spracovať, preniesť po sieti a následne dáta predať rendereru, ktorý ich vykreslí v headsete pre rozšírenú realitu - HoloLens.

Naše bádanie začalo už pri ročníkovom projekte, v ktorom sme sa pokúsili najprv len zosnímať dáta a fyzicky ich skopírovať do HoloLens. Navrhli sme preto aplikáciu pomocou Unity3D, ktorá prehrávala lokálne súbory. Čo sa týka siete, tak pomocou low-level API UNET sa nám nepodarilo dosiahnúť veľkej rýchlosti, ale dáta bolo možné preniesť.

Preto sme začali skúmať, testovať priepustnosť siete a hľadať možné riešenia, s ktorými by sa nám to podarilo. Vznikla tak vyššie spomenutá aplikácia NetworkTestUtility, ktorá slúžila hlavne ako experimentálna súčasť projektu a jej implementácia možno preto nie je úplne ideálna. Zapáčila sa nám knižnica ZeroMQ, ktorá je cross-platform a ponúkla nám pomerne jednoduché API, množstvo príkladov, podporu v oboch jazykoch C# a C++.

Potom sme sa už začali naplno venovať prenosu point cloud videa. Okúsil som na vlastnej koži, že pridanie do projektu a spojazdnenie knižníc, čo by človek rád použil nemusí byť úplne triviálne.

Keď sa nám to podarilo už celé spojazdniť, tak rýchlosť ešte nebola úplne uspokojúca. Potom bolo ešte potrebné dôkladne všetko prejsť a rozmyslieť si čo by mohlo prenos brzdiť.

Popridávali sme a zlepšili užívateľské rozhrania daných aplikácii, aby bolo možné aplikáciu naplno využívať. Pridali sme možnosť nastavenia rozmedzia vzdialenosť zobrazovaného point cloud-u, lebo niekedy je vhodné niečo nezobrazovať.

Ku koncu sme sa rozhodli využiť ešte jednu vlastnosť, ktorú má Kinect a tou je infračervený stream. Adaptívne prepínanie medzi farebným streamom z kamery alebo čierno-bielym streamom z IR senzoru môže mať zaujímavé využitie pri snímaní point cloud videa v nie úplne ideálnych svetelných podmienkach (napr. vonku, v pivnici, v zlom počasí, a tak podobne).

Vznikli teda aplikácie a spôsob, ktorý sa približuje tomu, o čom sme na začiatku len rozprávali: serverová - ServerKinectPC, klientská - PointCloudClient a testovacia - NetworkTestUtility.

7.6 Možnosti budúceho rozšírenia

Aplikáciu ServerKinectPC vie pracovať s abstraktnými triedami. Je možné ju teda rozširovať implementovaním `InputData` alebo `OutputData` a začlenením novej triedy medzi terajšie. Počítali sme s tým, že v budúcnosti príde nový Kinect alebo nejaký iný senzor na získanie point cloud-u. Rovnako aj v prípade, že by sme sa rozhodli používať iný formát súborov nemusíme ten doterajší prepisovať,

ale nový naimplementujeme osobitne a zachováme tak aj spätnú kompatibilitu. Ďalej by sa mohol z Kinectu snímať aj zvuk a prenášať spolu s obrazom.

Rozšírenie PointCloudClient by mohlo prebiehať z rôznych strán. Skúmali sme aj použitie senzoru „Kinect“, ktorý je na HoloLens. Prístup k jeho dátam je zatiaľ možný len cez Research mode, ktorý je určený na výskumné účely. Nie je však garantované podľa Gedye a kol. (2018), že v budúcich verziách API alebo zariadenia ako samotného bude táto možnosť podporovaná.

Zoznam použitej literatúry

Bray, B., McCulloch, J., Schonning, N. a Zeller, M. (2018). What is mixed reality? [Online]. URL https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality. [cit. 11.05.2019].

Bray, B., McCulloch, J., Zeller, M., rwinj, r. a grbury, g. (2019a). Gestures. [Online]. URL https://docs.microsoft.com/en-us/windows/mixed-reality/gestures. [cit. 11.05.2019].

Bray, B., McCulloch, J., Zeller, M., rwinj, r. a Turner, A. (2019b). Gaze. [Online]. URL https://docs.microsoft.com/en-us/windows/mixed-reality/gaze. [cit. 11.05.2019].

Gedye, D., Schonning, N. a Zeller, M. (2018). Hololens research mode. [Online]. URL https://docs.microsoft.com/en-us/windows/mixed-reality/research-mode. [cit. 14.05.2019].

Goins, D. (2014). Confirmed list of usb 3.0 pci-e cards/laptops/configurations which work for kinect v2 (during preview). [Online]. URL https://social.msdn.microsoft.com/Forums/en-US/bb379e8b-4258-40d6-92e4-56dd95d7b0bb/confirmed-list-of-usb-30-pcie-cardslaptopsconfigurations-which-work-for-kinect-v2-during?forum=kinectv2sdk. [cit. 09.05.2019].

Hintjens, P. (2019). Ømq - the guide. [Online]. URL http://zguide.zeromq.org/page:all. [cit. 11.05.2019].

Lemmens, M. J. P. M. (2014). Features of point clouds and functionalities of processing software. In *14th International Scientific and Technical Conference "From Imagery to Maps: Ditigal Photogrammetric TechnologiesÖctober 20-23, 2014, Hainan, China.* URL https://repository.tudelft.nl/islandora/object/uuid:2c76978c-51ac-49eb-8452-b8997bc0cf38/datastream/OBJ/download. [Online] [cit. 09.05.2019].

Levy, N. (2019). Microsoft unveils next-generation hololens headset and $399 'azure kinect' camera for developers. [Online]. URL https://www.geekwire.com/2019/microsoft-unveils-next-generation-hololens-headset-399-azure-kinect-camera-developers. [cit. 09.05.2019].

Microsoft (2014a). Kinect api overview. [Online]. URL https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn782033%28v%3dieb.10%29. [cit. 09.05.2019].

Microsoft (2014b). Coordinate mapping. [Online]. URL https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn785530(v%3dieb.10). [cit. 09.05.2019].

Microsoft (2017). Windows forms overview. [Online]. URL https://docs.microsoft.com/en-us/dotnet/framework/winforms/windows-forms-overview. [cit. 09.05.2019].

Microsoft (2019). Set up kinect for windows v2 or an xbox kinect sensor with kinect adapter for windows. [Online]. URL https://support.xbox.com/en-US/xbox-on-windows/accessories/kinect-for-windows-v2-setup. [cit. 09.05.2019].

Microsoft contributors (2019a). Using the hololens emulator. [Online]. URL https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-hololens-emulator. [cit. 11.05.2019].

Microsoft contributors (2019b). Install hyper-v on windows 10. [Online]. URL https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v. [cit. 11.05.2019].

Pelikán, J. (2018). Rozšířená realita s brýlemi hololens. In *Czech-Slovak conference on geometry and graphics 2018*. ISBN 978-80-214-5652-5. URL https://2018.csgg.cz/files/csgg_2018.pdf. [Online].

Pfister, H. a Gross, M. (2007). *Point-Based Graphics*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann. ISBN 9780123706041.

White, S., plieblang, p. a Satran, M. (2019). Introduction to c++/winrt. [Online]. URL https://docs.microsoft.com/en-us/windows/uwp/cpp-and-winrt-apis/intro-to-using-cpp-with-winrt. [cit. 11.05.2019].

Wikipedia contributors (2019a). Microsoft hololens — Wikipedia, the free encyclopedia. [Online]. URL https://en.wikipedia.org/wiki/Microsoft_HoloLens. [cit. 11.05.2019].

Wikipedia contributors (2019b). Infinite impulse response — Wikipedia, the free encyclopedia. [Online]. URL https://en.wikipedia.org/w/index.php?title=Infinite_impulse_response&oldid=878925108. [cit. 12.05.2019].

Wikipedia contributors (2019c). Kinect — Wikipedia, the free encyclopedia. [Online]. URL https://en.wikipedia.org/w/index.php?title=Kinect&oldid=895874764. [cit. 09.05.2019].

Wikipedia contributors (2019d). Universal windows platform — Wikipedia, the free encyclopedia. [Online]. URL https://en.wikipedia.org/w/index.php?title=Universal_Windows_Platform&oldid=896520499. [cit. 11.05.2019].

Yin-Poole, W. (2017). Kinect officially dead. [Online]. URL https://www.eurogamer.net/articles/2017-10-25-kinect-officially-dead. [cit. 09.05.2019].

Zeller, M., Turner, A., Bray, B. a grbury, g. (2018). Hololens (1st gen) hardware details. [Online]. URL https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details. [cit. 11.05.2019].

Zoznam obrázkov

A. Prílohy

A.1 Rozdelenie solutionu, projektov a zdrojových súborov.

Priečinok FMPointcloud obsahuje 3 priečinky, v ktorých sú jednotlivé projekty (aplikácie). Solution FMPointcloud.sln je závislý ešte od dvoch projektov FMCore a libzmq (vyznačené hierarchicky v prílohe). Vďaka tomu je možné knižnice zlinkovať a deploy-núť do zariadenia HoloLens.

V prílohe bakalárskej práce nie sú obsiahnuté:

- libzmq - je open-source knižnica, voľne dostupná webe (viac 3.5).
- FMCore - ako bolo spomenuté v 3.7.

Projekt bez týchto súčastí nie je možné skompilovať a deploy-núť. Tvorilo sa to na dátach a súčastiach programu, ktoré nie je možne zverejniť. V prípade otázok kontaktujte firmu Pocket Virtuality a.s., napríklad vedúceho práce:
`Josef Pelikán <josef.pelikan@pocketvirtuality.com>`

Poznámka pre oponenta: Na kompletné predvedenie a spustenie aplikácii sú potrebné zariadenia HoloLens a Kinect. Pre predvedenie aplikácii prosím kontaktujte autora alebo vedúceho práce:
`Adrián Mačák <adrian.ado.macak@gmail.com>`

A.2 Ukážkové video

Príloha ďalej obsahuje video, v ktorom je možné vidieť prenos point cloud videa v reálnom čase a funkcie aplikácie na HoloLens.