ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Katedra biomedicínské informatiky

Software pro podporu rehabilitace ruky

Software for hand rehabilitation support

Bakalářská práce

Studijní program:	Biomedicínská a klinická technika
Studijní obor:	Biomedicínská informatika
Autor bakalářské práce:	Ondřej Antoš
Vedoucí bakalářské práce:	Ing. Jan Mužík, Ph.D
Konzultant práce:	Mgr. Jakub Pětioký, Rehabilitační ústav Kladruby

Kladno 2019

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Antoš**	Jméno:	**Ondřej**	Osobní číslo:	**465562**
Fakulta:	**Fakulta biomedicínského inženýrství**				
Garantující katedra:	**Katedra biomedicínské informatiky**				
Studijní program:	**Biomedicínská a klinická technika**				
Studijní obor:	**Biomedicínská informatika**				

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Software pro podporu rehabilitace ruky

Název bakalářské práce anglicky:

Software for hand rehabilitation

Pokyny pro vypracování:

Pro operační systém Windows 10 vytvořte aplikaci pro podporu rehabilitace ruky u pacientů s hemiparézou horní končetiny. Aplikace bude využívat senzoru LeapMotion k přesnému zachycení pohybů zdravé ruky a pomocí realistického zobrazení zrcadlově převráceného 3D modelu bude na monitoru simulovat pohyb paretické ruky. Aplikace umožní výběr různých realistických pozadí scény a výběr různých modelů ruky. V aplikaci budou pro další vyhodnocení zaznamenávaný časy a délky jednotlivých cvičení včetně trajektorií všech významných bodů ruky. Aplikace bude umožňovat vyšetření funkčního rozsahu pohybů prstů a zápěstí, dle předem strukturované definice pohybu a záznam hodnot vyšetření v časové ose. Aplikace bude obsahovat seznam pacientů s možností editace, vytvoření klinického profilu pacienta a seznam strukturovaných cvičení. Před každým cvičením bude možno cvičícího pacienta vybrat a přiřadit mu cvičení ze seznamu cvičení, aplikace bude umožňovat záznam terapeutického cvičení pacienta ve virtuálním prostoru.

Seznam doporučené literatury:

[1] Eller, F., C# - začínáme programovat, ed. 1, Grada, Praha, 2002, ISBN 80-247-0324-6
[2] Petzold, Ch., Programování Microsoft Windows v jazyce C#, ed. 1, Softpress, Praha, 2003, ISBN 80-86497-54-2
[3] František Vaverka, Základy biomechaniky pohybového systému člověka, ed. 2, Vydavatelství Univerzity Palackého, 1997, 40 s., ISBN 8070677279

Jméno a příjmení vedoucí(ho) bakalářské práce:

Ing. Jan Mužík, Ph.D.

Jméno a příjmení konzultanta(ky) bakalářské práce:

Mgr. Jakub Pětioký, Rehabilitační ústav Kladruby

Datum zadání bakalářské práce: **19.02.2019**
Platnost zadání bakalářské práce: **20.09.2020**

doc. Ing. Zoltán Szabó Ph.D.
podpis vedoucí(ho) katedry

prof. MUDr. Ivan Dylevský, DrSc.
podpis děkana(ky)

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem „*Software pro podporu rehabilitace ruky*“ vypracoval samostatně a použil k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k bakalářské práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně dne ....

Ondřej Antoš

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce za odborné vedení a cenné rady v průběhu vývoje. Mgr. Jakubovi Pětiokému a MUDr. Markétě Janatové děkuji za konstruktivní zpětnou vazbu a vhled do problematiky. Velké díky za psychickou a materiální podporu patří mé rodině.

ABSTRAKT

Software pro podporu rehabilitace ruky

Cílem práce bylo navrhnout a vytvořit programové vybavení určené k rehabilitaci horní končetiny s využitím pohybového senzoru od společnosti Leap Motion. V úvodu je popsána rehabilitační metoda zvaná zrcadlová terapie a její dosavadní aplikace a následně je srovnána s možným rozšířením aplikace této metody s pomocí počítače a prvků virtuální reality. Ze srovnání vyplývají možnosti využití počítačové aplikace zrcadlové terapie, které byly využity k tvorbě návrhu programového vybavení. Následující část práce podrobně popisuje proces implementace jednotlivých součástí návrhu. Dále je popsáno provedené testování výsledného technického řešení a v závěru jsou zhodnoceny výsledky práce.

Klíčová slova

Zrcadlová terapie, vizuální zpětná vazba, rehabilitační software, Leap Motion, Unity

ABSTRACT

Software for hand rehabilitation support

The aim of this thesis was to design and create an application for hand rehabilitation using motion sensor by Leap Motion, Inc. Thesis begins by introducing mirror therapy, its current application in rehabilitation. Following that is comparison of conventional mirror therapy with its novel application using computers and aspects of virtual reality. This comparison serves as a basis for designing an application that takes advantage of the arising possibilities. Next part describes in detail the process of implementing those designed features in the finished application. Finally, the testing phase is described, results are discussed and evaluated in conclusion.

Keywords

Mirror therapy, mirror visual feedback, rehabilitation software, Leap Motion, Unity

Obsah

Seznam zkratek **9**

1 Úvod **10**

2 Přehled současného stavu **11**

2.1 Zrcadlová terapie 11

2.1.1 Průběh rehabilitačního sezení 12

2.2 Způsoby aplikace zrcadlové terapie 12

2.3 Pohybový senzor Leap Motion 14

2.3.1 Porovnání s klasickou aplikací zrcadlové terapie 15

2.3.2 Využití senzoru v programovém vybavení 17

3 Cíle práce **18**

4 Návrh aplikace **19**

4.1 Analýza požadavků 19

4.2 Funkční specifikace 19

4.2.1 Datový model 19

4.2.2 Popis funkcionality 20

4.2.3 Grafické uživatelské rozhraní 21

4.3 Technické specifikace 24

4.3.1 Diagram aplikačních komponent 24

4.3.2 Diagram případu užití 24

4.3.3 Vývojový diagram hlavní nabídky 25

4.3.4 Databázový model 26

4.3.5 Minimální hardwarové požadavky 26

5 Implementace **27**

5.1 Práce s Unity 29

5.1.1 Hierarchie objektů 29

5.2 Zrcadlení ruky 30

5.3 Změna modelu 31

5.4 Změna pozadí scény 32

5.5 Databáze 33

5.6 Nahrávání a přehrávání 34

5.7 Vyhledávaní 35
5.8 Grafické uživatelské rozhraní 35
5.8.1 Ovládací prvky a ikony 35
5.8.2 Přechod načítání 37
6 Testování 38
7 Diskuse 39
8 Závěr 40
Seznam použité literatury 41
Seznam obrázků 44
Obsah přiloženého CD 45

Seznam zkratek

Seznam zkratek

Zkratka	Význam
API	Application Programming Interface
SDK	Software Development Kit
GUI	Graphical User Interface
CMP	Cévní mozková příhoda
ORM	Objektově relační mapování
OS	Operační systém

1 Úvod

Tato práce se zabývá vývojem počítačového programu, který spolu s pohybovým snímačem tvoří technické řešení pro aplikaci zrcadlové terapie.

Zrcadlová terapie je metoda používaná pro rehabilitaci osob postižených ochrnutím či osob s amputovanými končetinami, která byla objevena americkým neurovědcem V. S. Ramachandranem počátkem devadesátých let.

Zrcadlová terapie spočívá ve stimulaci mozkových center pacienta vytvořením subjektivní iluze, že pohybuje svou ochrnutou končetinou. Výsledkem terapie může být obnovení motorické funkce, zmírnění bolestí a získání samostatnosti při vykonávání každodenních činností. Tato iluze vzniká sledováním pohybů zrcadlového obrazu zdravé, případně schopnější, končetiny, kdy z pohledu pacienta tento obraz nahradí pohled na ruku ochrnutou. Pro vytvoření této iluze je dodnes používána jednoduchá pomůcka tzv. mirror box, který původně zkonstruoval pro tento způsob terapie V. S. Ramachandran.

Mirror box i jeho jednodušší alternativy, také založené na použití zrcadla, však mají určitá omezení, zejména pokud je porovnáme s možnostmi využití virtuální reality a počítačů. Ať už se jedná o způsob sledování a záznamu průběhu cvičení, možnosti předcvičování a předávání instrukcí pacientovi, nebo i o samotnou šíři použití na různá postižení pacientů, kdy jsou například z možnosti využít pouhé zrcadlo vyloučeni oboustranně postižení pacienti.

Přes jednoduchost a cenovou dostupnost pomůcek na bázi běžného zrcadla je motivací k vytvoření počítačové aplikace zrcadlové terapie rozšířit ve spojení s virtuální realitou její potenciál, usnadnit terapeutickou asistenci a umožnit zpětné vyhodnocování cvičení.

Mým cílem bylo vytvořit počítačovou aplikaci, která by tyto rozšířené možnosti poskytovala.

2 Přehled současného stavu

V přehledu současného stavu je detailněji popsána samotná zrcadlová terapie, původní způsob její aplikace a průběh rehabilitačního sezení. Dále jsou popsány způsoby aplikace zrcadlové terapie, porovnání konvenční aplikace s aplikací využívající senzor Leap Motion.

2.1 Zrcadlová terapie

Autorem zrcadlové terapie je neurovědec V. S. Ramachandran, kterou poprvé aplikoval na pacientovi po amputaci horní končetiny v roce 1992. [1]

Při provádění rehabilitace pomocí této metody použil konstrukčně jednoduchou pomůcku, tzv. mirror box. Pomůcka je nenákladné na výrobu, jelikož se skládá pouze ze zrcadla a několika neprůhledných stěn, mezi které pacient vkládá obě své končetiny. Mirror box vytváří při rehabilitaci iluzi pohybu postižené končetiny a tím postupně obnovuje funkční schopnosti skutečné paretické ruky pacienta, zmírňuje bolesti vyskytující se při onemocnění a usnadňuje proces obnovení schopnosti provádět každodenní činnosti. [2] [3]

Obrázek 1 Pacient při rehabilitaci zrcadlovou terapií. Zdroj: [24]

Účinnost zrcadlové terapie spočívá v postupném obnovování funkce končetiny prostřednictvím vizuálně koordinované motorické činnosti postiženého pacienta. Tato činnost spočívá v provádění pohybů silnější končetinou, při které pacient dostává vizuální zpětnou vazbu sledováním odrazu končetiny v zrcadlovém boxu.

Částečné ochrnutí postihující polovinu těla se nazývá hemiparéza. Mezi nejčastější příčiny vzniku hemiparézy končetiny patří cévní mozková příhoda (dále jen CMP), která ročně poprvé postihne přibližně 9 miliónů lidí. [2]

Přibližně 80 % lidí, kteří přežijí CMP trpí zhoršením motorické funkce horní či dolní končetiny. [2]

Při CMP dochází k nedokrevnosti mozku, což má za následek poškození mozku nebo jeho částí, včetně těch, které řídí motorické funkce. Následkem tohoto poškození pak mohou být obtíže provádět každodenní činnosti jako oblékání, nazouvání bot, stravování či používání toalety. [4]

2.1.1 Průběh rehabilitačního sezení

Při rehabilitaci ruky pomocí zrcadlové terapie nejprve pacient položí pomůcku před sebe tak, aby zrcadlo bylo orientované v mediánní rovině těla.

Pak položí své ruce tak, aby v zrcadle viděl obraz své zdravé ruky. Poté už pacient může začít se samotnými pohyby svou nepostiženou rukou a při tom se dívá do zrcadla pomůcky a sleduje zrcadlené pohyby své nepostižené ruky. Konkrétní cviky, které pacient provádí, se liší podle stupně jeho ochrnutí. Cviky můžou být opakované monotónní pohyby ruky běžně vykonávané při každodenních činnostech, např. sevření v pěst, otevření a zavření dlaně, kroužení zápěstím, roztáhnutí prstů a natáhnutí ruky, nebo cviky s využitím vhodných předmětů, např. psaní tužkou, mačkání míčku, čištění zubů kartáčkem.

Doba trvání zrcadlové terapie se liší podle konkrétního nasazení terapeutem. Jeden z pokusů o vytvoření doporučeného postupu pro terapii uvádí, že pro zhodnocení účinků terapie je nutné cvičení provádět minimálně 10 minut jednou denně alespoň po dobu 5 až 6 týdnů. [4]

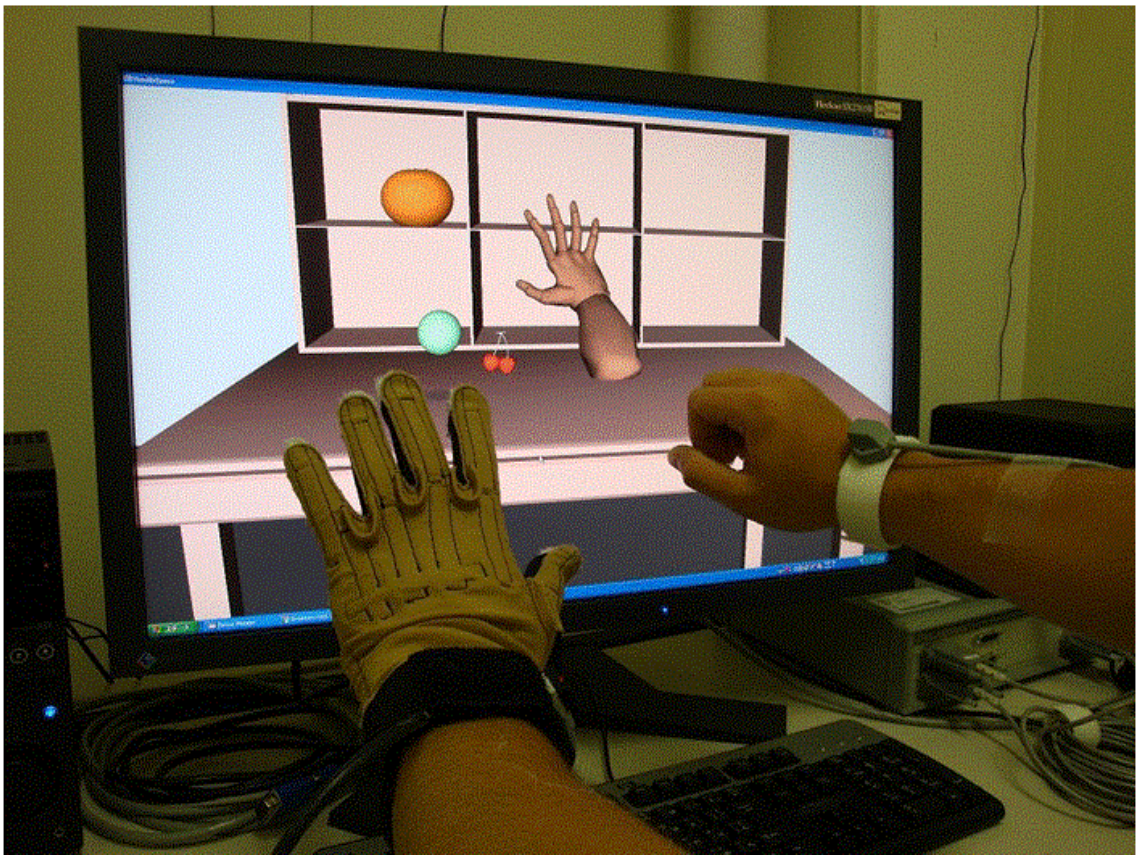
2.2 Způsoby aplikace zrcadlové terapie

Zrcadlová terapie se dodnes provádí s pomocí fyzického mirror boxu v rehabilitačním zařízení, kde je možnost asistence fyzioterapeuta či lékaře. Po seznámení pacienta s instrukcemi pro cvičení je možné rehabilitaci provádět přímo u pacienta v domácím prostředí, kde již není potřeba pro asistenci odborníka.

Zrcadlová terapie se dá aplikovat i s využitím pokročilých technických prostředků na bázi počítačových technologií a virtuální reality. [5] Společným rysem těchto řešení je vytvoření virtuálního prostředí a použití snímače pohybu,

hardwarového nebo softwarového s využitím kamer a zpracování obrazu v reálném čase. Pro navození obrazové iluze se používají zobrazovací zařízení pro virtuální realitu, jako jsou VR brýle nebo běžné počítačové monitory a projektory. Jako vstupní zařízení pro snímání pohybu pacientovy končetiny se používají různé typy snímačů. Mezi ně patří bezdotykové snímače, např. Kinect [6], Leap Motion [6] nebo snímače na mechanickém základě, jako jsou různé modely snímacích rukavic, primárně určených jako příslušenství pro VR brýle.

Snímací rukavice využil např. tým z Univerzity v Okajamě. Ke snímání použil rukavici od společnosti CyberGlove. [5]



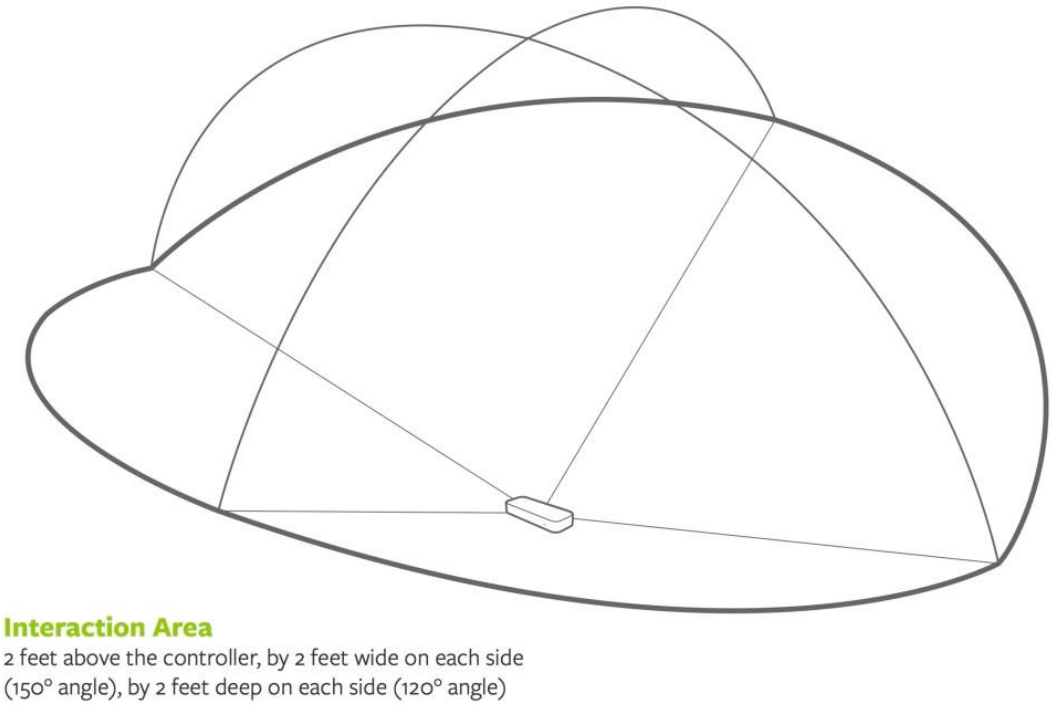
Obrázek 2 Použití snímací rukavice. Zdroj: [5]

2.3 Pohybový senzor Leap Motion

Díky rozvoji technologií virtuální reality existuje řada nových snímačů pro interakci s virtuálním prostředím, kterých lze využít i bez virtuálních brýlí, s použitím běžných zobrazovacích zařízení. Mezi nimi je výjimečný pohybový senzor od firmy Leap Motion, protože dokáže přesně snímat ruku od lokte až po jednotlivé prsty.

Obrázek 3 Pohybový senzor Leap Motion. Zdroj: [27]

Snímací prostor má přibližně tvar kužele s výškou 80 centimetrů a úhlem záběru přibližně 120 až 150 stupňů. Výška snímaného kuželu je omezena ztrátou intenzity vyzařovaného infračerveného záření s rostoucí vzdáleností od senzoru.



Obrázek 4 Prostor snímání senzoru Leap Motion. Zdroj: [26]

Senzor se skládá z dvou širokoúhlých kamer, tří infračervených LED a hliníkového šasi.

Snímání funguje pomocí zachycení odrazu infračerveného světla od ruky dvěma kamerami. Senzor posílá datový tok ve formě stereo obrazu přes USB rozhraní do připojeného počítače.

Následně probíhá zpracování těchto obrazových dat v dodávaném proprietárním softwaru. Kvůli tomu, že zpracování dat probíhá z větší části v softwaru mimo samotný senzor, obnovovací frekvence datového toku je závislá na výkonu počítače, ke kterému je snímač připojen.

Senzor dokáže snímat s udávanou přesností na méně než jeden milimetr. Přesnost se liší podle okolních podmínek místa, ve kterém se senzor využívá a je snižován při výskytu jiných zdrojů infračerveného záření, např. slunce, wolframové i halogenové žárovky.

Nejlepší kvality snímání lze dosáhnout při absenci přímých či odražených zdrojů světla v poli snímání, vyhrnutí rukávů, sundání šperků a jiných doplňků. [7]

2.3.1 Porovnání s klasickou aplikací zrcadlové terapie

Díky rozvoji zařízení pro virtuální realitu a jejich klesající pořizovací ceně a rostoucí dostupnosti se otevírá možnost nasazení těchto technologií i v oblasti rehabilitace. Výjimkou není ani zrcadlová terapie.

V rehabilitačním zařízení může odborník dohlížet na průběh rehabilitačního sezení, poskytovat podporu a vést pacienta při cvičení. Tato osobní péče terapeuta je časově náročná, což dává prostor novým metodám, které nevyžadují tolik času s fyzioterapeuty a daly by se použít jako doplňující rehabilitační prostředek ke konvenční fyzioterapii. [2]

- Uchovávání záznamů o průběhu cvičení
    - Při konvenční metodě je obvykle zaznamenávána četnost a doba provádění jednotlivých cviků. V nepřítomnosti terapeuta nelze dohlížet na jejich správnost a dostatečnost.
    - V počítačové aplikaci lze shromažďovat trajektorie všech významných bodů ruky a je možné dodatečně vyhodnotit správnost a dostatečnost prováděných cviků.
- Využití předmětů
    - Při konvenční metodě není problém se zrcadlením manipulovaného předmětu, pokud jeho rozměry umožňují snadnou manipulaci v prostoru v blízkosti zrcadla, ale především pokud postižení pacienta nebrání snadné manipulaci s předměty.
    - Ve virtuálním prostředí lze oproti tomu pacientovi umožnit manipulaci s libovolně objemným tělesem, a to i v případě závažnějšího postižení, kdy lze manipulaci s předměty usnadnit programově, ale pacient je ochuzen o haptické vjemy a vnímání

hmotnosti předmětu. Tyto vjemy však nepociťuje ani u postižené skryté končetiny během konvenční aplikace, a proto je sporné a otázkou dalšího zkoumání, která z těchto rozdílných vjemových situací má efektivnější dopad při terapii.

- Dostupnost a náklady
    - Mirror box a jeho alternativy se skládají z cenově dostupných komponent, ale vzhledem k tomu, že pomůcka není vyráběna sériově ve standardizované formě, liší se konkrétní pomůcky svou kvalitou či záleží na šikovnosti zainteresovaných osob, jak dokáží pomůcku sestrojit.
    - Pro počítačovou aplikaci je zapotřebí počítač s odpovídajícím výkonem, senzor Leap Motion a zobrazovací zařízení, např. vestavěná obrazovka počítače či externí monitor. Přidané náklady se mohou lišit podle stávajícího vybavení. Cena samotného senzoru se v roce 2019 pohybuje kolem 2500 Kč.
- Účast terapeuta a asistence
    - I při konvenční zrcadlové terapii lze pro pacienta poskytnout instrukce, např. v podobě videa, tedy bez nutnosti osobní účasti terapeuta. Toto řešení však není centralizované, zpětně kontrolovatelné a např. videozáznam lze těžko upravovat
    - Oproti tomu počítačová aplikace umožňuje nahrávání instrukcí ve formě trajektorie ruky, což je skladnější forma dat a lze s nahrávkami zpětně manipulovat.
- Zábavnost
    - Možnosti mirror boxu udělat ze cvičení zábavnou činnost jsou závislé na individuální kreativitě terapeuta. Škála pohybů a cviků, které dávají samy o sobě smyl je také limitovaná. Rozšíření cvičení o manipulaci s předměty je omezené na prostor v blízkosti zrcadla.
    - Oproti tomu počítačová aplikace zrcadlové terapie má téměř neomezený kreativní potenciál a možnosti měnit chování virtuálního prostředí. Při plnění úkolů ve virtuálním prostředí lze dosáhnout vyšší variability prováděných pohybů a využitím herních prvků upoutat pacientovu pozornost. Další přidanou hodnotou oproti konvenční aplikaci je možnost uchopení jednoho předmětu oběma rukama.

2.3.2 Využití senzoru v programovém vybavení

Pro využití senzoru Leap Motion v softwarové aplikaci výrobce poskytuje SDK pro dva grafické enginy – Unity a Unreal Engine, které zprostředkovávají zobrazování virtuálního prostředí včetně modelů rukou a grafického uživatelského rozhraní. V enginu Unity se vlastní kód aplikace píše v programovacím jazyku C#. V Unreal enginu se píše v jazyce C++. [8]

Při výběru jsem se rozhodl použít Unity, protože jsem během studia s programovacím jazykem C# a vývojovým prostředím Visual Studio už pracoval. Dalším kritériem byla dostupnost výukových materiálů obou enginů. V tomto bodě byly oba enginy téměř vyrovnané, jelikož mají na svých webových portálech rozsáhlou dokumentaci svých API včetně návodů a komunitního fóra, kde lze dotazovat vývojáře i uživatele enginu, ale Unity mělo pro mě srozumitelnější materiály. Oba enginy také obsahují repozitáře s hotovými grafickými a programovacími prvky, které můžou urychlit vývoj aplikace. V repozitářích jsou nabízeny tyto součásti bezplatně či za poplatek, jehož výši určí autor součásti, který ji nabízí. V tomto ohledu měl výhodu engine Unity díky dostupnosti většího množství součástí použitelných pro navrhovanou aplikaci. Enginy mají také rozdílnou formu licencování produktu využívajících jejich technologii. Zatímco Unreal si bere 5% zisku z produktu každý kvartál po překročení zisku 3000 dolarů, Unity poskytuje nekomerční licenci, která lze využít při zisku do výše 100000 dolarů za rok. V tomto ohledu bylo vhodnější využít nekomerční licence Unity. [9] [10]

3 Cíle práce

- Ověřit použitelnost senzoru Leap Motion pro aplikaci v zrcadlové terapii
- Navrhnout a vytvořit rehabilitační aplikaci v souladu s požadavky
    - Navrhnout aplikaci dle provedeného sběru požadavků
    - Vytvořit aplikaci implementující zadanou funkcionalitu

4 Návrh aplikace

Pro vývoj rehabilitační aplikace jsem použil vodopádový model vývoje softwaru, který se skládá z postupně navazujících fází návrhu, implementace, testování a údržby.

4.1 Analýza požadavků

Sběr požadavků probíhal v průběhu práce osobně s vedoucím práce a s externím konzultantem.

Hlavním požadavkem na vytvořenou aplikaci bylo, aby zobrazovala realistický model uživatelovy zdravé ruky a k tomu zrcadlově převrácenou ruku tak, aby vznikla iluze, že uživatel pohybuje oběma svýma rukama, podobně jako při klasické zrcadlové terapii. Dalším požadavkem byla možnost nahrávání a přehrávání záznamů pohybů ruky a ukládání těchto záznamů, včetně údajů uživatele, do lokální databáze.

4.2 Funkční specifikace

4.2.1 Datový model[1]

- Aplikace bude obsahovat databázi pacientů a jejich nahrávek
- Údaje o pacientovi se budou skládat z:
  1. Příjmení a jméno
  2. Léčená ruka
  3. Pohlaví *
  4. Datum narození *
  5. Rodné číslo *
  6. Volný text (např. diagnóza a komentáře) *
- Nahrávky se budou skládat z:
  1. Datum a čas začátku nahrávání
  2. Doba trvání nahrávky v sekundách
  3. Záznam trajektorie ruky
  4. Komentář k nahrávce *
- Každý pacient bude moci mít více nahrávek, ale každá nahrávka bude patřit jen jednomu pacientovi

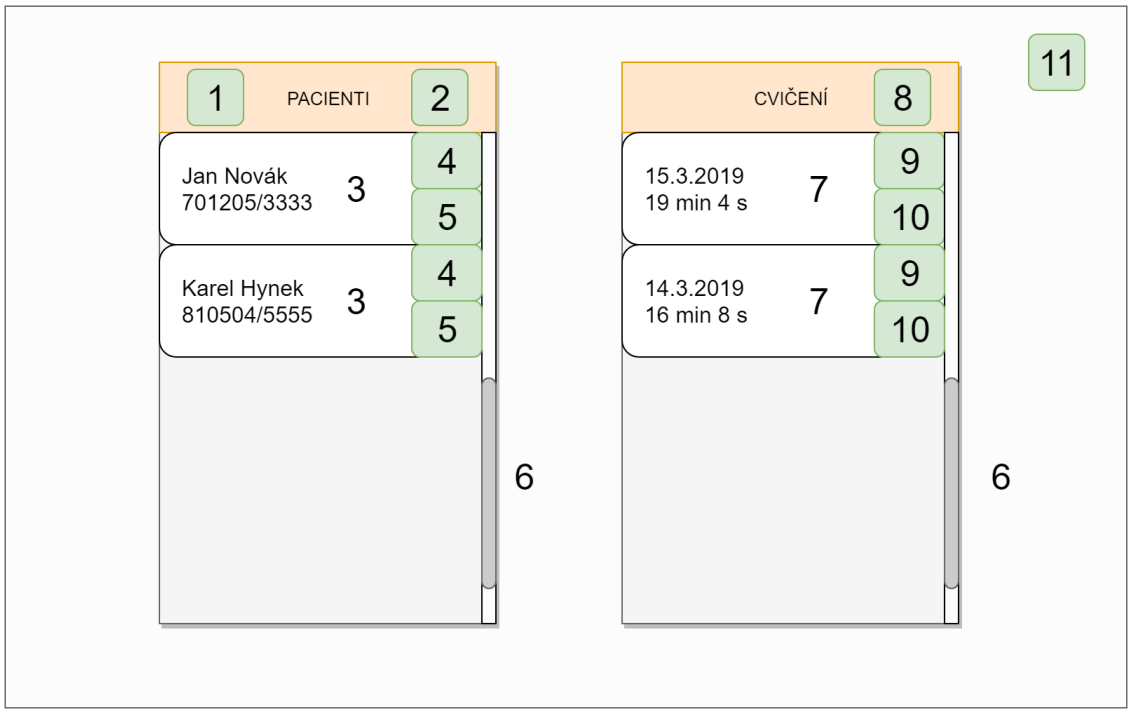
[1] Položky označené hvězdičkou jsou nepovinné

4.2.2 Popis funkcionality

- Pacienty bude možné přidávat, mazat a upravovat jejich údaje
- V seznamu pacientů bude možnost vyhledávat pacienty podle části jména, příjmení či rodného čísla
- Každé nahrávce bude moci uživatel změnit komentář
- U každého pacienta bude možno tvořit tolik nahrávek, kolik dovolí limit databáze či kapacita úložiště stroje, kde aplikace bude v provozu
- Záznam trajektorie ruky nebude možno po dokončení nahrávání dále měnit.
- Pokud se vyskytne chyba při nahrávání nebo dojde k neúmyslnému spuštění, musí uživatel spustit nahrávání nové či zrušit nahrávání
- V aplikaci bude možnost při přehrávání a nahrávání:
    - vybrat si vizuální prostředí scény z několika přednastavených možností
    - vybrat si ze dvou přednastavených variant modelů ruky
    - zapnout a vypnout zobrazení zrcadlené ruky
- V aplikaci bude dále možnost při přehrávání:
    - posouvat se v aktuální nahrávce pomocí posuvníku
- Při ukončení nahrávání k tomuto určeným ovládacím prvkem se nahrávka automaticky uloží do databáze

4.2.3 Grafické uživatelské rozhraní

- Funkční návrh grafického uživatelského rozhraní s popisem ovládacích prvků:
  - Návrh hlavní nabídky:



Obrázek 5 Návrh GUI hlavní nabídky. Zdroj: vlastní

Legenda k obrázku:

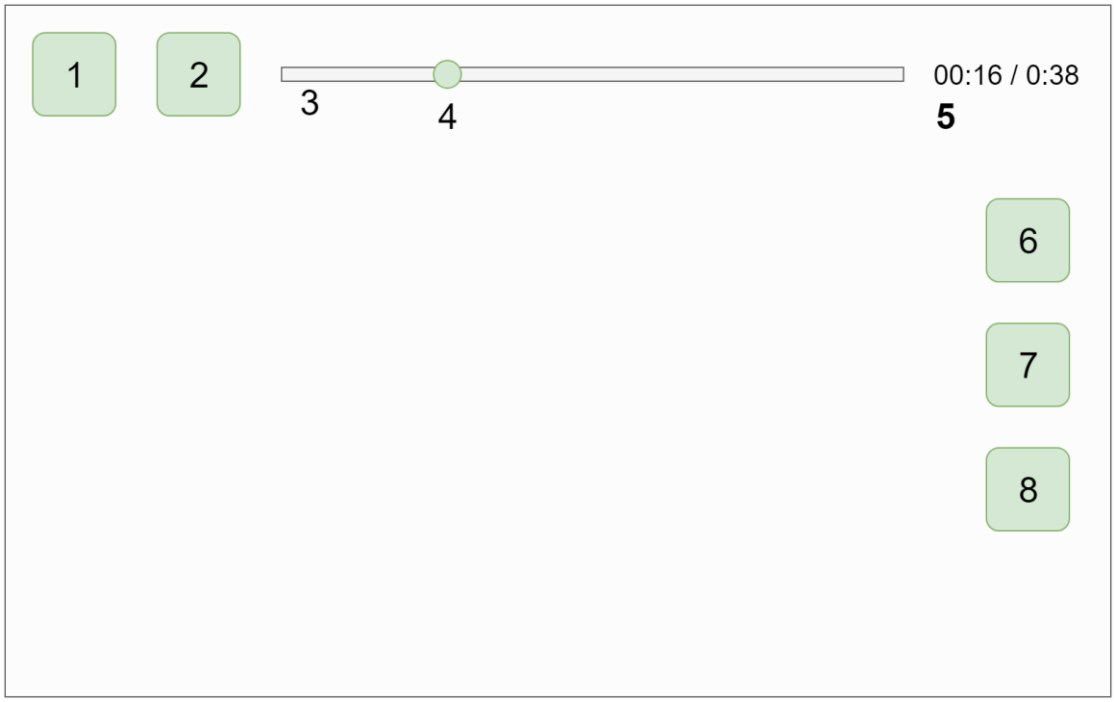
Panel pacientů (vlevo)

1. Tlačítko pro vyhledávání
2. Přidání nového pacienta
3. Zobrazení nahrávek pacienta
4. Upravení údajů pacienta
5. Smazaní pacienta
6. Posuvník

Panel nahrávek (vpravo)

7. Spuštění přehrávání nahrávky
8. Přidání nové nahrávky
9. Upravení komentáře nahrávky
10. Smazání nahrávky
11. Ukončení aplikace

- Návrh rozhraní v režimu přehrávání:



Obrázek 6 Návrh GUI přehrávání. Zdroj: vlastní

Legenda k obrázku:

1. Návrat do hlavní nabídky
2. Spuštění a pozastavení přehrávání
3. Časová osa nahrávky
4. Posuvník na časové ose
5. Aktuální pozice v čase a celkový čas nahrávky
6. Změna pozadí
7. Zapnutí a vypnutí zobrazení zrcadlené ruky
8. Změna modelu ruky

- Návrh rozhraní v režimu nahrávání:

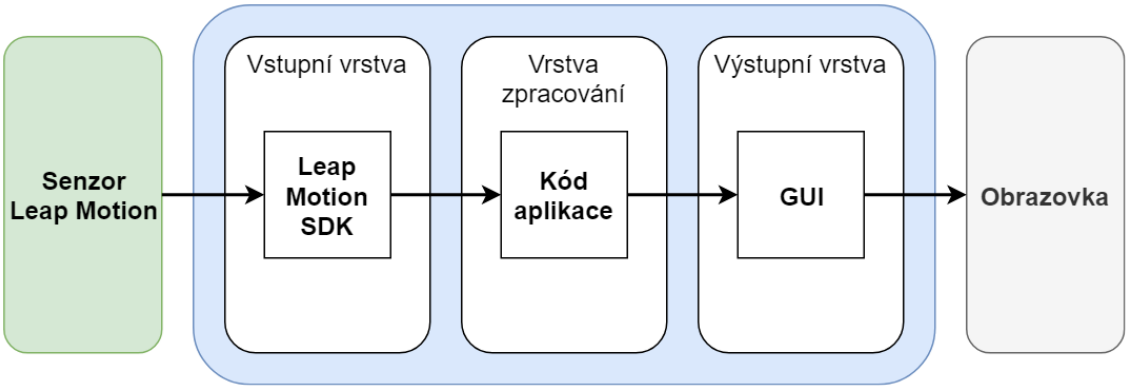
Obrázek 7 Návrh GUI nahrávání. Zdroj: vlastní

Legenda k obrázku:

1. Návrat do hlavní nabídky
2. Spuštění a ukončení nahrávání
3. Aktuální doba nahrávky / maximální doba nahrávky
4. Změna pozadí scény
5. Zapnutí a vypnutí zobrazení zrcadlené ruky
6. Změna modelu ruky

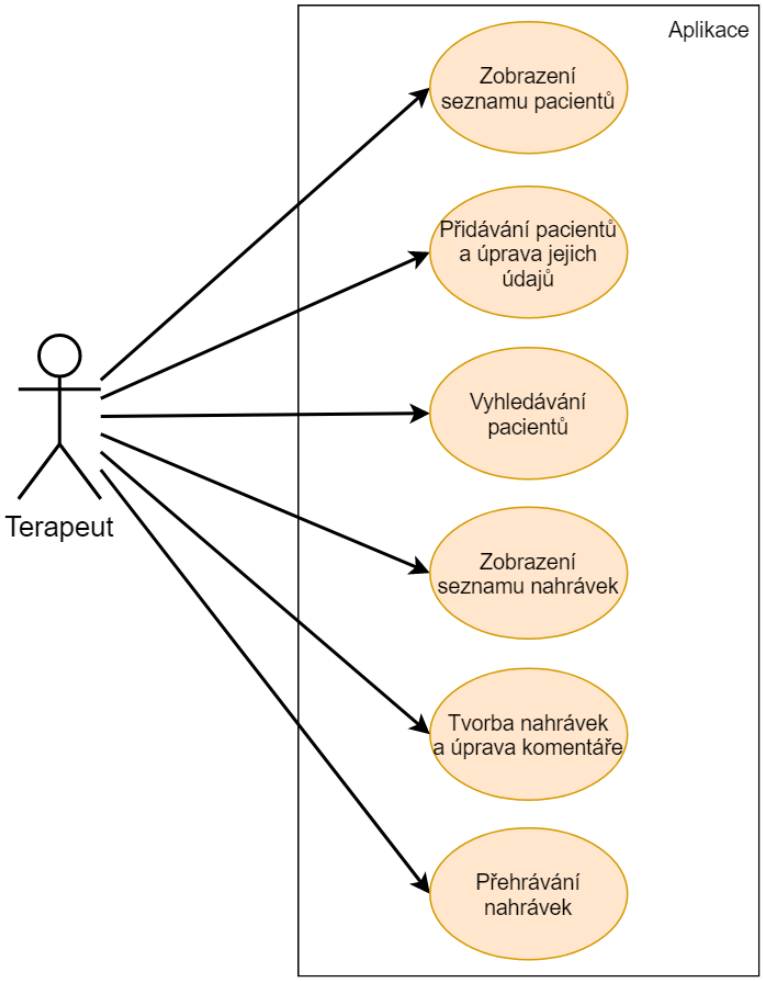
4.3 Technické specifikace

4.3.1 Diagram aplikačních komponent



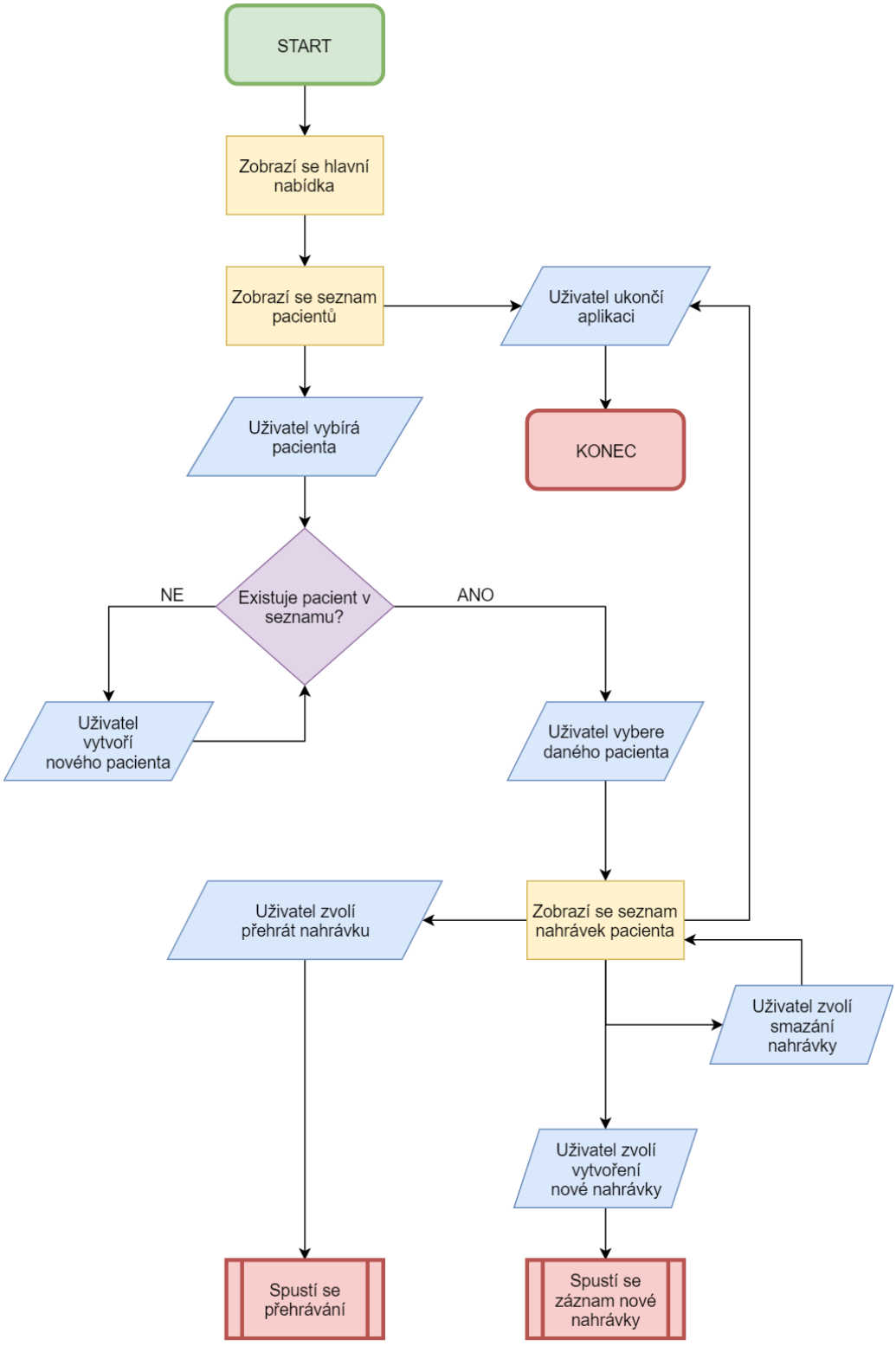
Obrázek 8 Diagram aplikačních komponent. Zdroj: vlastní

4.3.2 Diagram případu užití



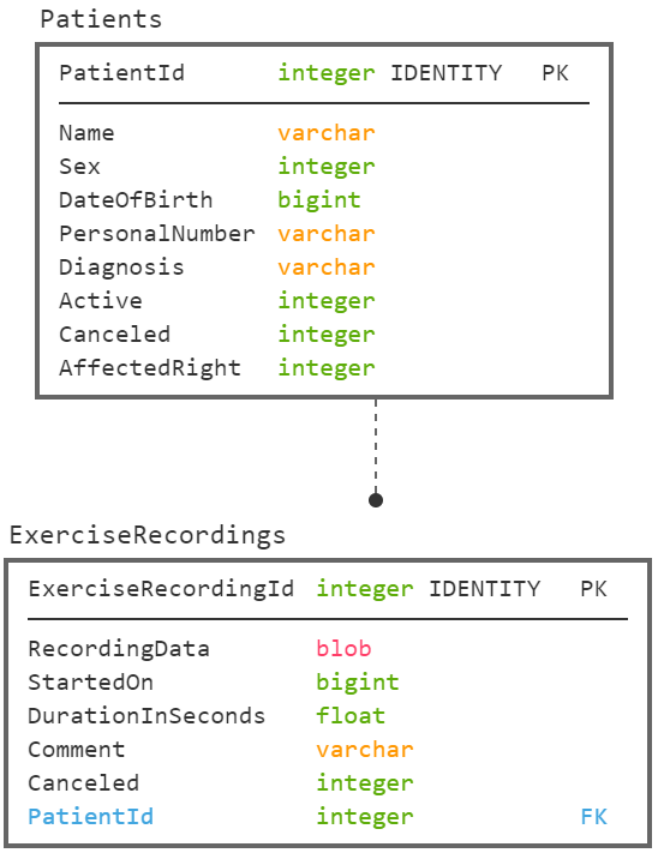
Obrázek 9 Diagram případu užití. Zdroj: vlastní

4.3.3 Vývojový diagram hlavní nabídky



Obrázek 10 Vývojový diagram při spuštění aplikace. Zdroj: vlastní

4.3.4 Databázový model



Obrázek 11 Návrh databázového modelu. Zdroj: vlastní

4.3.5Minimální hardwarové požadavky

- Čtyřjádrový procesor s takt. frekvencí aspoň 1 GHz a podporou instrukční sady SSE2 (např. AMD Phenom™ II nebo Intel® Core™ i3/i5/i7)
- Integrovaná či dedikovaná grafická karta s podporou DirectX 10 (shader model 4.0)
- Operační systém Windows 10
- Aspoň 2 GB operační paměti
- Aspoň 300 MB na úložišti zařízení (velikost aplikace)
- Další volné místo na úložišti pro uložení nahrávek

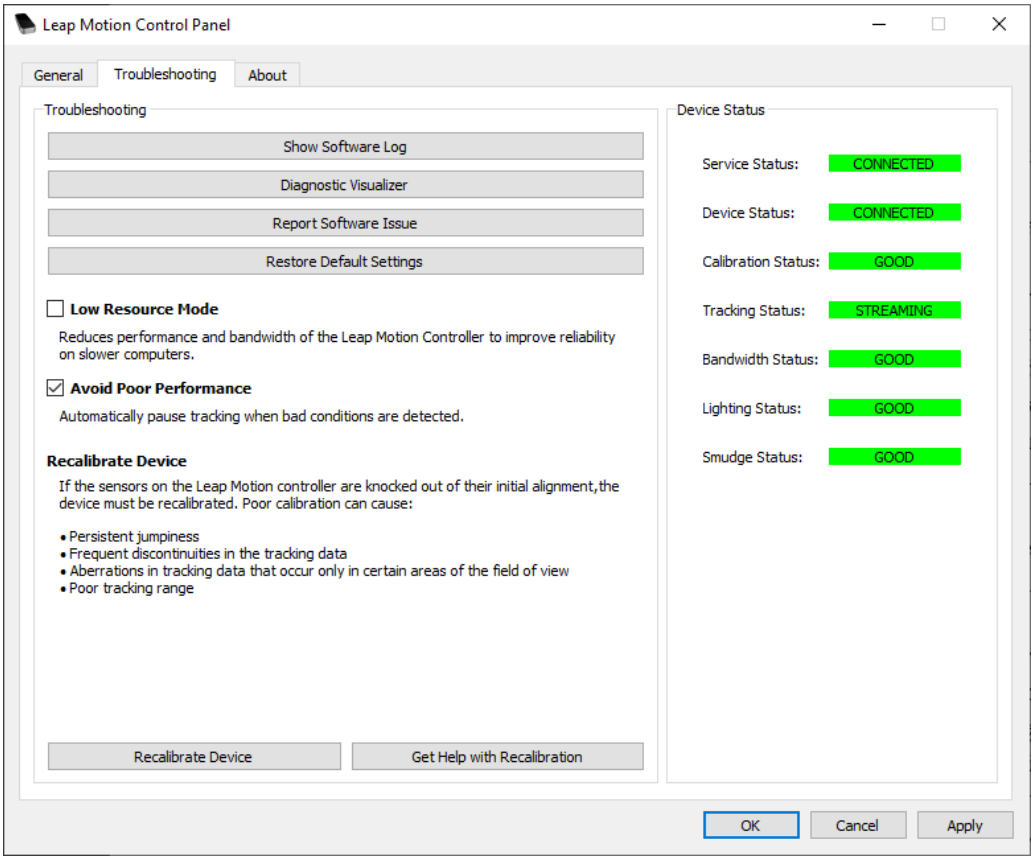
5 Implementace

V této kapitole je popsáno, jak jsem při vývoji aplikace postupoval a jakých prostředků jsem při vývoji využil.

Senzor Leap Motion a PC s OS Windows 10, na kterém jsem mohl vyvíjet aplikaci mi poskytl vedoucí práce.

Při vývoji jsem využil systém pro správu verzí pro sledování změn ve zdrojovém kódu. Zdrojový kód a ostatní soubory projektu jsem synchronizoval s centralizovaným systémem, v mém případě Apache Subversion, který je v provozu na společném pracovišti 1. LF UK a FBMI na Albertově, kde jsem aplikaci vyvíjel.

Výrobce senzoru poskytuje vývojářům aplikací, kteří jej chtějí využívat, ovladač na svém vývojářském portálu. [11] Tento ovladač je ve formátu spustitelného instalačního souboru pro Windows a podporuje verze OS Windows 7 a vyšší. Na vývojovém PC byl nainstalovaný operační systém Windows 10 a ovladač se bez problémů podařilo nainstalovat. Ovladač Leap Motion obsahuje grafický nástroj pro konfiguraci senzoru, panel aktuálního stavu fungování senzoru, kalibračního průvodce a diagnostický vizualizér.

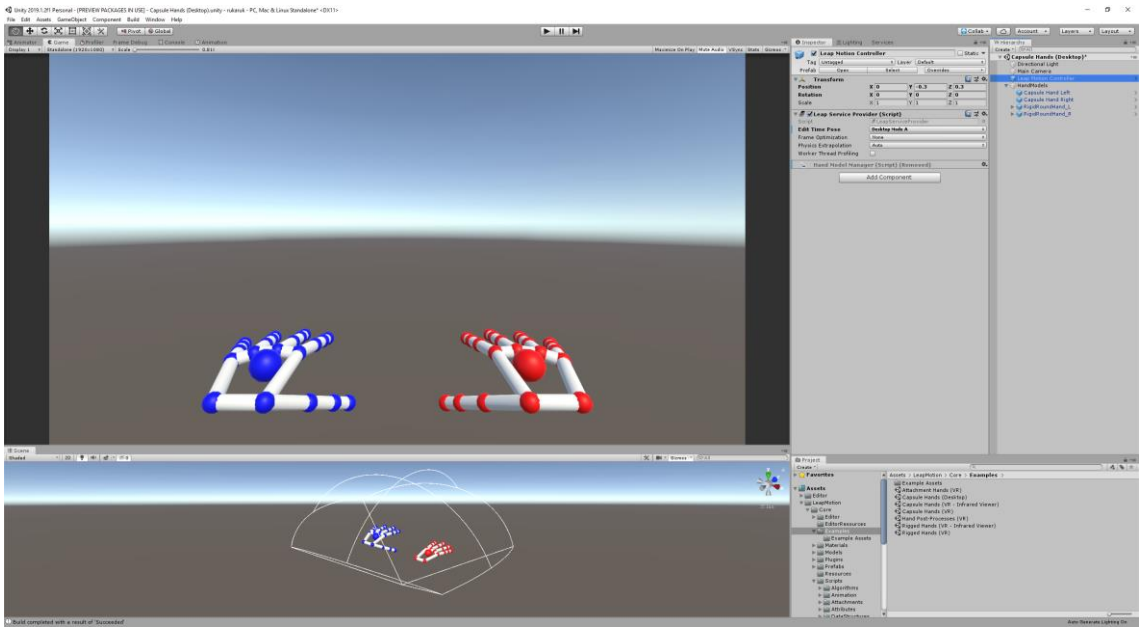


Obrázek 12 Ovládací panel Leap Motion. Zdroj: vlastní

Diagnostický vizualizér slouží k ověření správného fungování senzoru a poskytuje podrobné informace o datovém toku, obnovovací frekvenci, zpoždění zpracování dat a zobrazuje jednoduchou reprezentaci snímaných rukou v reálném čase. Během transportu zařízení ze skladu se může mírně pohnout uspořádání dvou obrazových snímačů uvnitř zařízení, což může vést ke zhoršení přesnosti snímání. Pokud zařízení nefunguje podle specifikací či očekávání, lze v ovladači provést kalibraci senzoru. Kalibrace senzoru se provádí otáčením senzoru, mířícího na lesklou plochu, podle instrukcí v kalibračním průvodci ovladače Leap Motion. Při seznamování se se senzorem jsem spustil diagnostický vizualizér, a přestože nevykazoval známky nesprávného fungování jsem senzor zkalibroval.

Pro vývoj aplikace jsem použil grafický engine Unity, pro který výrobce senzoru poskytuje několik vývojářských balíčků, které mají modulární charakter a jsou ke stažení z vývojářského portálu Leap Motion [12] ve formátu „unitypackage“, což je nativní formát enginu Unity pro import a export.

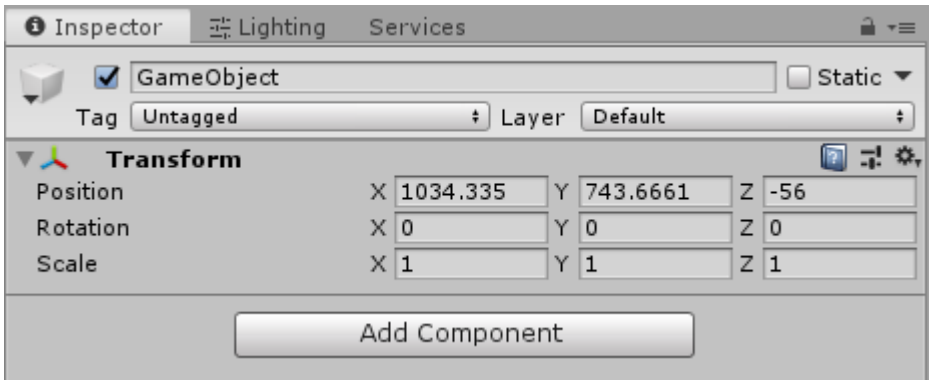
Po nastudování popisu funkcionality a případů použití všech vývojářských balíčků jsem zjistil, že potřebuji jen základní balíček, který je nutný pro všechny aplikace využívající senzor Leap Motion v enginu Unity. Základní balíček obsahuje knihovnu pro Unity v jazyce C#, pomocné skripty pro testování a ukázkové scény pro Unity. V ukázkových scénách jsou připravené a napojené jednotlivé Unity komponenty pro demonstraci fungování senzoru v enginu Unity.



Obrázek 13 Ukázková Unity scéna Leap Motion. Zdroj: vlastní

5.1 Práce s Unity

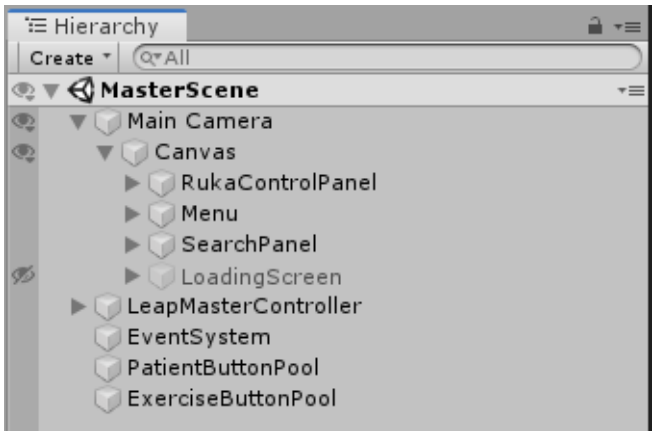
Každý projekt v enginu Unity se skládá z jedné či více scén. Každá scéna je kontejnerem pro jednotlivé prvky zvané entity, které lze uspořádat do hierarchických stromových struktur. Každá entita je modulární prvek složený z komponent, které řídí její vlastnosti a chování. Jedinou nutnou komponentou každé entity je komponenta „Transform“, která udává pozici, rotaci a relativní velikost každé entity ve scéně. Entita sama o sobě ale nemá grafickou podobu nebo tvar, takže není ve scéně vidět, dokud entita neobsahuje komponentu, která jí tuto podobu určí.



Obrázek 14 Ukázka entity v Unity. Zdroj: vlastní

5.1.1 Hierarchie objektů

Hierarchie scény slouží především k formální organizaci objektů scény a není pro funkci vyžadována, přesto je však vhodné objekty uspořádat podle jejich funkční souvislosti, protože i jednoduchá scéna v praxi obsahuje mnoho entit. Například moje výsledná aplikace je tvořena jednou scénou, která obsahuje přes sto entit. Při vývoji jsem ze začátku měl potíže s organizací objektů v hierarchii, což jsem postupně vylepšoval v průběhu práce a ve výsledné aplikaci jsem dokázal zredukovat počet objektů v hierarchii.



Obrázek 15 Hierarchie projektu v Unity. Zdroj: vlastní

Navigaci v hierarchii s velkým počtem prvků usnadňuje klávesová zkratka „Alt + Levé tlačítko myši“, která rozbalí, čí zabalí všechny vnořené entity v hierarchii. [13]

5.2 Zrcadlení ruky

Při implementaci funkce zrcadlení ruky jsem nejprve vyzkoušel nejjednodušší řešení, které mě po seznámení se s Unity a API Leap Motion napadlo, což bylo vytvořit ve scéně kopii původní ruky a vynásobit v „Transform“ komponentě kopie velikost v ose x konstantou -1. Při tom jsem narazil na problém. Toto řešení nebylo funkční, jelikož byla zrcadlena jen pozice ruky, nikoliv její rotace. Pokusil jsem se tedy najít řešení někoho se stejným problémem na komunitních vývojářských fórech Leap Motion a Unity [14] [15], ale nikdo podobný problém neřešil. Našel jsem ale na fóru příspěvek vývojáře Leap Motion, který odpovídal uživateli fóra na jiný problém s tím, že má zkusit využít abstraktní

```
public class MirrorProvider : PostProcessProvider
{
// offset rukou na ose x (frontální rovina) [metr]
[Range(-0.2f, 0.2f)]
public float offset = 0f;
  // Funkce pro vytvoření zrcadlové kopie ruky
  private Hand FlipHand(Hand hand)
  {
  // znaménko translačního vektoru podle chirality
  int sign = hand.IsLeft ? 1 : -1;
  // translační vektor (pouze osa x)
  Vector translation = new Vector(sign * offset, 0f, 0f);
  // vytvoření zrcadlové transformace v ose x
  // (pomocná funkce LeapMotion)
  LeapTransform mirrorLT =
    new LeapTransform(translation, LeapQuaternion.Identity);
  mirrorLT.MirrorX();
  // Vytvoření zrcadlové kopie vstupní ruky
  Hand mirrorHand =
    TransformExtensions.TransformedCopy(hand, mirrorLT);
  // Obrácená translace původní ruky (zachování symetrie)
  hand.Transform(new LeapTransform(
    -translation, LeapQuaternion.Identity));
  // Obrácení parametru chirality výstupní ruky
  mirrorHand.IsLeft ^= true;
  // změna Id (předejití duplicity)
  mirrorHand.Id = hand.Id / 2 + 1;
  return mirrorHand;
  }
}
```

Obrázek 16 Ukázka kódu třídy MirrorProvider. Zdroj: vlastní

třídu PostProcessProvider z knihovny Leap Motion, která umožňuje transformovat ruce před jejich grafickým zobrazením. Po nastudování této třídy se mi podařilo využít ji k vytvoření vlastní komponenty, která správně zrcadlí obě ruce, čímž jsem problém zrcadlení ruky vyřešil. Tato komponenta funguje jako poskytovatel transformované ruky, který lze zapínat a vypínat při běhu aplikace.

5.3 Změna modelu

Do projektu Unity je možné importovat vlastní 3D modely, včetně modelů ruky s definovanou kostrou a vzájemnými geometrickými a dynamickými vazbami. Tento proces tvorby vlastního realistického modelu ruky v externí aplikaci by ale byl časově náročný, takže jsem hledal alternativní metodu, jak získat hotové realistické modely ruky do mé implementace. Řešení jsem nalezl v obchodě s komponenty, který je součástí platformy enginu Unity. Při vývoji jsem použil modely ruky z tohoto obchodu, díky čemuž jsem se mohl zaměřit na programování aplikační logiky a uživatelského rozhraní.

Balíček modelů ruky obsahuje celkem dvě varianty, kterými jsou levá a pravá ruka s dvěma texturami pro světlou a tmavou pleť. Balíček je přímo určen pro vývoj aplikací v Unity, takže bez problémů fungoval ihned po importu projektu.

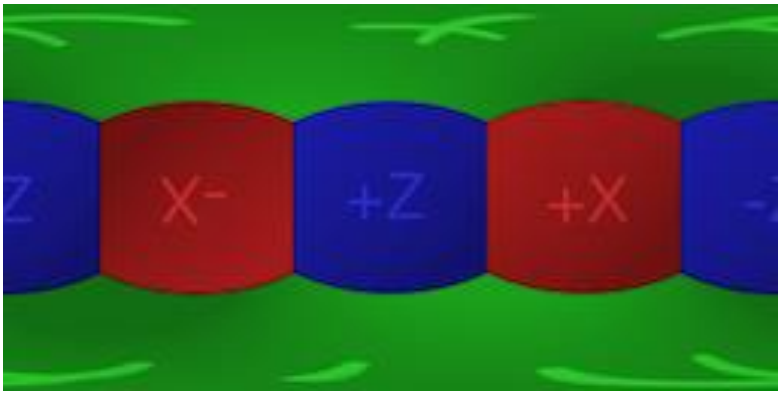
Při implementaci funkce změny modelu rukou jsem neměl tak velký problém jako při implementaci zrcadlení. Vytvořil jsem pro tuto funkcionalitu vlastní jednoduchou funkci, která za chodu dokáže cyklicky měnit modely z mnou předdefinovaného pole modelů.

```
public class CycleHandModels : MonoBehaviour
{
  public string[] GroupNames;
  public int CurrentGroup
  {
    ...
  }
  // Funkce, která cyklicky mění modely ruky
  public void CycleHands()
  {
    CurrentGroup = (CurrentGroup + 1) % GroupNames.Length;
  }
}
```

Obrázek 17 Ukázka funkce pro změnu modelu ruky. Zdroj: vlastní

5.4 Změna pozadí scény

Po nastudování podporovaných formátů textur pozadí v Unity jsem pro implementaci realistického pozadí využil službu poskytující textury v tzv. HDR formátu. [16] Engine Unity umí automaticky importovat mnoho obrazových formátů, včetně několika typů HDR obrazů, u kterých rozpozná jejich specifický formát a rozložení a lze je použít jako pozadí celé 3D scény.



Obrázek 18 HDR obraz v cylindrické projekci. Zdroj: [25]

Tvorba takových textur je časově i finančně náročná, protože se textury tvoří pomocí fotografování zrcadlové koule z několika úhlů přímo na místě, které pak slouží jako pozadí scény. Moje řešení mi usnadnilo práci použitím už hotových textur ze služby „HDRi haven“, která poskytuje realistické HDR obrazy pozadí v cylindrické projekci a bez nutnosti licencování či poplatků.

Obrázek 19 Ukázka cylindrického HDR obrazu. Zdroj: [16]

5.5 Databáze

Každá aplikace, která uchovává data, což platí i o mojí aplikaci, využívá některou z nespočet databázových technologií. Po konzultaci s vedoucím jsem se rozhodl použít tzv. „embedded“ databázi SQLite, která nevyžaduje databázový server, ale ukládá se lokálně do souboru, který se nachází na úložišti zařízení, na kterém je spuštěna samotná aplikace. K usnadnění propojení databáze s kódem aplikace se v softwarovém inženýrství využívá technika „Objektově relační mapování“, která slouží jako spojka mezi konkrétní databázovou technologií a objektově-orientovaným jazykem. Tato technika urychluje práci s databází tím, že mapuje databázové třídy na jejich dané tabulky v konkrétní databázi.

Pro moji aplikaci jsem po konzultaci s vedoucím vybral ORM Entity Framework 6, který byl vhodný pro použití v mojí aplikaci, protože je určen pro jazyk C#, ve kterém jsem psal i vlastní kód aplikace.

Engine Unity používá programovací jazyk C# a vývojové prostředí Visual Studio, v němž se při vývoji dá použít tzv. NuGet balíčků k importování knihoven do projektu, ale Unity tyto balíčky nepodporuje. Jelikož doporučený postup pro implementování databáze SQLite do projektu Visual Studia je použít právě tyto NuGet balíčky, narazil jsem při vývoji své aplikace na problém. Hledal jsem řešení v dokumentaci SQLite [17] a na komunitních fórech [15] [14], kde jsem nalezl mnoho návodů, jak zprovoznit databázi SQLite v projektu Unity. Vyzkoušel jsem všechny navrhované postupy od uživatelů fóra, ale žádný z nich nefungoval. Některé postupy byly zastaralé a některé byly sice aktuální, ale žádný z nich nezmiňoval propojení všech tří součástí, tedy Unity, SQLite a Entity Framework 6, společně. Řešení jsem nalezl s pomocí knihovny „SQLite-net“ [18], která slouží jako ORM a C# wrapper nad knihovnou SQLite a funkčně nabízela pro moji implementaci všechno, co bylo potřeba. Knihovna umožňuje komunikovat s databází v jazyce C# a obsahuje standardní funkce pro CRUD operace a dotazování, což byly mé požadavky.

Po úspěšném importování knihovny SQLite-net do enginu Unity jsem při modelování databázových tříd narazil ještě na jeden problém. Knihovna SQLite-net neposkytovala atributy pro modelování vztahů „1 ku n“ a „n ku 1“, které bylo potřeba využít v modelování databázových tříd. Tento problém jsem vyřešil pomocí připojení rozšiřující knihovny „SQLite-net Extensions“ [19], která tyto rozšířené atributy poskytuje. Problém s rozšiřujícími funkcemi mělo mnoho dalších uživatelů komunitního fóra, na kterém jsem toto řešení našel.

5.6 Nahrávání a přehrávání

Data o trajektorii ruky poskytuje senzor Leap Motion až stokrát za sekundu a v balíčku pro Unity jsou tyto data poskytovány jako .NET objekty třídy „Frame“. Implementace nahrávání tedy spočívala v uchování těchto dat v databázi SQLite. Bylo ale nutné převést data z formy .NET objektu v paměti do formy, která se dá uložit jako jeden z datových typů databáze.

NULL	Hodnota, reprezentující žádnou hodnotu.
INTEGER	Celočíselná hodnota se znaménkem a dynamickou velikostí 1, 2, 3, 4, 6, až 8 bajtů
REAL	Číselná hodnota s plovoucí řádovou čárkou velikosti 8 bajtů dle specifikace IEEE
TEXT	Textový řetězec s kódováním UTF-8, UTF-16BE nebo UTF-16LE
BLOB	Libovolná hodnota uložená v nezměněné podobě

Tabulka 1 Datové typy databáze SQLite3. Zdroj: [17]

Uchovávání dat z třídy „Frame“ v textové podobě není vhodné, protože na úložišti zařízení zabírá na jednotku dat příliš mnoho místa. Nejvhodnější formát k ukládání nahrávek je binární formát. Využil jsem postupu dle dokumentace frameworku .NET od Microsoftu [20], kterým je použití binárního formátovače „BinaryFormatter“.

Převod objektu v paměti do série bajtů se nazývá serializace. Podle postupu jsem úspěšně implementoval serializaci nahrávek, které obsahují jednotlivé snímky v sérii. Tyto serializované data se mi podařilo uložit do databáze, ale při kontrole uložených dat během testování jsem narazil na další problém. Velikost nahrávky byla příliš vysoká. Začal jsem hledat řešení, jak nahrávky zmenšit kompresí, a na komunitním fóru Unity [15] jsem našel několik uživatelů s podobným dotazem. Jednou z rad bylo využít knihovnu „SharpZipLib“ [21], kterou jsem formou dll knihovny úspěšně importoval do Unity projektu. Použitím této knihovny se mi podařilo zkomprimovat každou nahrávku na polovinu své původní velikosti.

Při testování se ale ukázalo, že serializace pomocí knihovny „SharpZipLib“ je příliš pomalá, což by narušovalo plynulost aplikace. Zaznamenal jsem čas ukládání u testovací minutové nahrávky, který byl 20 sekund. U reálné nahrávky, která by mohla dosahovat kolem třiceti minut by ukládání trvalo příliš dlouho. Řešení tohoto problému jsem nalezl ve výměně serializační knihovny, po vyhledávání na komunitním fóru Unity. Tuto nově nalezenou knihovnu jménem „MessagePackCSharp“ [22] se mi podařilo importovat do projektu Unity a po

úpravě kódu aplikace tak, aby knihovnu využila, se doba ukládání a načítání snížila na několik milisekund.

5.7 Vyhledávaní

Plocha využitelná k zobrazení ovládacích prvků je omezená velikostí obrazovky, takže seznam pacientů se v průběhu používání aplikace a postupným přidáváním nových pacientu zvětšoval. Kvůli tomu bylo stále zdlouhavější najít v seznamu konkrétního pacienta pouze posouváním v seznamu. Tento problém jsem vyřešil implementací vyhledávání pacientů podle části jména nebo rodného čísla. Tuto funkcionalitu jsem přidal pomocí vytvoření funkce, která filtruje zobrazované pacienty s využitím pomocných filtračních funkcí.

Pro vyhledávání jsem vytvořil porovnávací funkci, která je case-insensitive a ignoruje diakritiku. Např. pro vyhledání jména „Havlíček“ postačuje zadat „havlicek“.

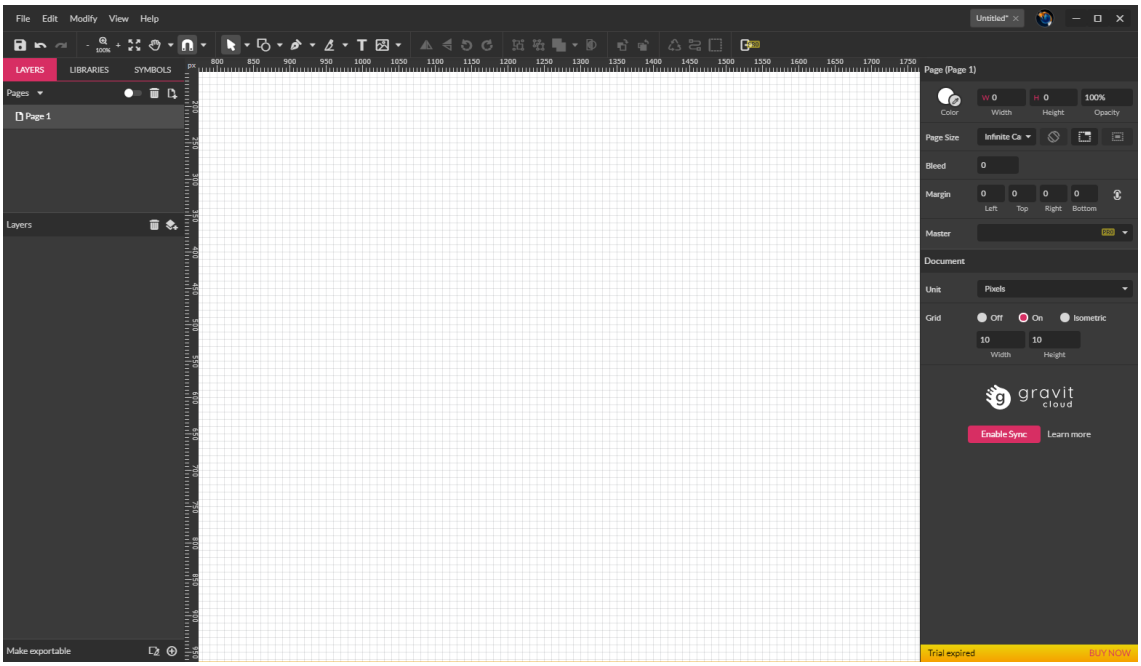
5.8 Grafické uživatelské rozhraní

Unity obsahuje vlastní grafické komponenty, ze kterých jsem sestavil grafické uživatelské rozhraní aplikace. Při jeho implementaci jsem se řídil svým funkčním návrhem, který jsem vytvořil ve fázi návrhu aplikace na základě funkčních požadavků. S ohledem na to, že uživatelé aplikace by mohli mít různé úrovně technických dovedností, jsem při návrhu i implementaci kladl důraz na jednoduchost a funkčnost.

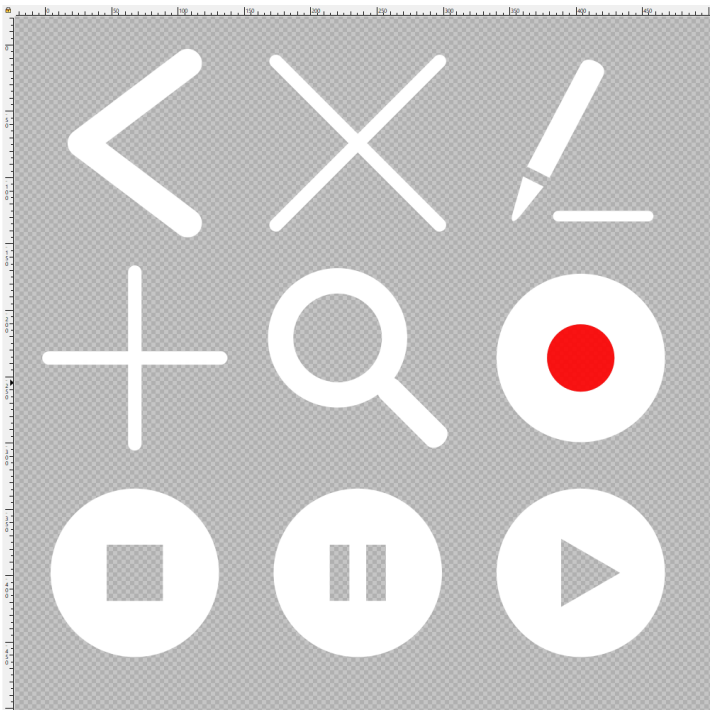
5.8.1 Ovládací prvky a ikony

Pro ovládací prvky typu tlačítka jsem použil jako identifikátor jejich funkce ikony. Nejprve jsem použil ikony dostupné z mediální banky Wikimedia Commons, které byly označeny jako „volné dílo“. Tyto ikony byl funkční, ale nesplňovaly mé kritéria na ergonomii ovládání a nebyly v jednotném grafickém stylu. Rozhodl jsem se proto vytvořit vlastní ikony pomocí vektorového grafického programu Gravit Designer [23] s cílem splnit kritéria ergonomie pomocí sjednocení grafického stylu ikon. S programem jsem měl předchozí zkušenosti, které jsem při tvorbě využil.

Vytvořené ikony jsem uložil ve formátu SVG, aby byla možnost dále je upravovat a také tím ikony neztratily svou flexibilitu při importování, což by se stalo při exportu do rastrovaného obrázku. Importoval jsem vektorové ikony do Unity pomocí vestavěného pluginu „SVG Importer“ a poté je vložil do ovládacích prvků.



Obrázek 20 Program Gravit Designer. Zdroj: vlastní



Obrázek 21 Vlastní vytvořené ikony. Zdroj: vlastní

5.8.2 Přechod načítání

Aby uživateli bylo zřejmé, že aplikace momentálně pracuje na úkonu, při kterém nemůže přijímat uživatelský vstup, jsem vytvořil tzv. „loading screen“ ve formě zastiňující vrstvy s textem „Načítám…“ uprostřed. Upravil jsem kód aplikace tak, aby se tento loading screen zobrazoval během načítání a ukládání nahrávek.

Testováním jsem zjistil, že při načítání a ukládání nahrávek se loading screen vůbec nezobrazuje, ale uživatelské rozhraní zamrzne. Hledal jsem příčinu tohoto problému na komunitním fóru Unity, kde jsem se dozvěděl, že je zamrznutí způsobeno prováděním časově náročných operací, např. ukládání dat, v hlavním vláknu aplikace.

Engine Unity volá funkci Update, která renderuje snímek, několikrát za vteřinu. Všechny funkce Unity sdílí jediné hlavní vlákno a musí čekat na dokončení všech jednotlivých operací než lze vyrenderovat následující snímek. Toto se v jiných aplikacích dá vyřešit využitím více vláken, což Unity implementuje ve formě tzv. C# Job Systému. Alternativním řešením je použití tzv. „Coroutine“, což jsou speciální funkce Unity, které umožňují rozložit dokončení funkce na více renderovaných snímku, čímž se zamrzání GUI odstraní. Upravil jsem svojí implementaci funkcí načítání a ukládání do formy Coroutine, čímž se problém zamrzání odstranil. Tuto variantu jsem zvolil, protože úprava kódu na formu Coroutine byla mnohem rychlejší, jednodušší a vedla ke stejnému výsledku, jako použití C# Job Systému. Díky zvolení varianty Coroutine jsem se také vyhnul možnému riziku zavedení nestability při implementaci vícevláknové varianty.

6 Testování

Při testování jsem si vytvořil několik fiktivních pacientů s běžnými českými jmény pro ověření funkcionality vyhledávání.

Po provedení výpočtu přibližné velikosti nahrávky jsem zjistil, že každá minuta serializované a nezkomprimované nahrávky dosahuje velikosti 20 MB, z čehož vyplývá, že na úložiště o kapacitě 500 GB by se vešlo pouze 400 hodin nahrávek a zápis dat do databáze by trval příliš dlouho k zachování plynulosti chodu aplikace.

	Serializace	Serializace + komprese
.NET BinaryFormatter + SharpZipLib	4 s / 1 min	15 s / 1 min
MessagePackC#	0.05 s / 1 min	0.05 s / 1 min

Tabulka 2 Přibližné rychlosti serializace testovaných metod na 1 minutu nahrávky. Zdroj: vlastní

Moji výslednou aplikaci jsem poskytl k otestování doktorce MUDr. Markétě Janatové, která aplikaci vyzkoušela a odpověděla na otázky z testovacího dotazníku, který jsem pro účely zhodnocení připravil.

První dojem hodnotila jako velmi dobrý a iluzi pohybu obou rukou jako velmi věrnou. Ergonomii uživatelského rozhraní shledala velmi dobrou. Na otázku porovnání klasické zrcadlové terapie a mojí výsledné aplikace odpověděla, že klasický mirror box působí realističtěji, ale že aplikace má výhodu v poutavém grafickém zpracování, které pacienta více motivuje. Dále ocenila možnost nahrání pohybů ruky terapeutem pro účel samostatné domácí rehabilitace.

V dotazníku byl prostor pro vlastní komentáře a připomínky k aplikaci, ve kterém zmínila, že by byla přínosná možnost nahrávat i paretickou ruku pro diagnostiku účinků terapie. Dotazník je k dispozici v elektronické příloze práce.

7 Diskuse

Kdybych práci začínal s nynějšími zkušenostmi, nejdříve bych vytvořil tzv. Proof of concept, kterým bych ověřil, že lze splnit zadání a pokračoval bych tvorbou malých funkčních prototypů. Dále bych dříve začal testovat propojení Unity s databází, což byla nejsložitější část implementace, kterou jsem ale nakonec překonal a získal cenné zkušenosti. Také bych více dbal na dodržení oddělení zodpovědností některých částí aplikace, což je jedna ze zásad objektově-orientovaného programování. Kdybych podrobně prostudoval všechny části dokumentace Unity a Leap Motion, vývoj by pravděpodobně probíhal plynuleji a chyby v návrhu či implementaci bych odhalil snadněji.

Pro analýzu pacientova pokroku při rehabilitaci či vyšetření funkčního rozsahu je ve výsledné aplikaci nutné vizuální hodnocení terapeutem. Aplikace by se dala rozšířit o funkcionalitu automatické diagnostiky, která by využívala dat trajektorie ruky k porovnání relativního zlepšování či zhoršování pacientova stavu v průběhu rehabilitace.

Dále by se aplikace dala rozšířit implementováním funkce tvorby předdefinovaných cvičení. Fyzioterapeut by pomocí této funkce mohl s předstihem sestavit cvičení na míru pro svého pacienta. Sestavování takových modulárních cvičení by mohlo mít podobu textových instrukcí, či vzorových nahrávek, které by fyzioterapeut sestavil z pohybů své ruky.

Výsledná rehabilitační aplikace bude poskytnuta Rehabilitačnímu ústavu Kladruby, se kterým bylo dohodnuto nasazení do klinické praxe. Na tuto práci by se dalo vhodně navázat provedením randomizované kontrolované studie o využití výsledné rehabilitační aplikace pro pacienty s fantomovými bolestmi, regionálním bolestivým syndromem či centrální parézou HK v důsledku poškození mozku.

8 Závěr

Cílem této práce bylo vytvořit funkční technické řešení aplikace zrcadlové terapie s pomocí pohybového senzoru Leap Motion a ověřit použitelnost senzoru v programovém vybavení.

Výsledné programové vybavení splňuje funkcionalitu ze zadání práce a senzor nijak neomezil implementaci všech částí zadané funkcionality.

Rehabilitační aplikace bude distribuována formou spustitelného souboru. Vytvořil jsem k aplikaci návod k použití s popisem ovládacích prvků, který je součástí přiloženého CD.

Vlastní zdrojový kód aplikace i spustitelný soubor je poskytnut s licencí MIT License, jejíž plné znění se nachází na přiloženém CD.

Seznam použité literatury

[1] RAMACHANDRAN, V., D ROGERS-RAMACHANDRAN a M STEWART. Perceptual correlates of massive cortical reorganization. *Science* [online]. 1992, **258**(5085), 1159-1160 [cit. 2019-05-05]. DOI: 10.1126/science.1439826. ISSN 0036-8075. Dostupné z: http://www.sciencemag.org/cgi/doi/10.1126/science.1439826

[2] THIEME, Holm, Jan MEHRHOLZ, Marcus POHL, Johann BEHRENS a Christian DOHLE. Mirror therapy for improving motor function after stroke. *Cochrane Database of Systematic Reviews* [online]. 2012, (3) [cit. 2019-05-05]. DOI: 10.1002/14651858.CD008449.pub2. ISSN 14651858. Dostupné z: http://doi.wiley.com/10.1002/14651858.CD008449.pub2

[3] EZENDAM, Daniëlle, Raoul BONGERS a Michiel JANNINK. Systematic review of the effectiveness of mirror therapy in upper extremity function. *Disability and Rehabilitation* [online]. 2009, **31**(26), 2135-2149 [cit. 2019-05-05]. DOI: 10.3109/09638280902887768. ISSN 0963-8288. Dostupné z: http://www.tandfonline.com/doi/full/10.3109/09638280902887768

[4] ROTHGANGEL, Andreas a Susy BRAUN. *Mirror Therapy: Practical Protocol for Stroke Rehabilitation* [online]. b.r. [cit. 2019-05-06]. Dostupné z: DOI: 10.12855/ar.sb.mirrortherapy.e2013

[5] SATO, Kenji, Satoshi FUKUMORI, Kantaro MIYAKE, Daniel OBATA, Akio GOFUKU a Kiyoshi MORIT. A Novel Application of Virtual Reality for Pain Control: Virtual Reality-Mirror Visual Feedback Therapy. GHOSH, Subhamay, ed., Subhamay GHOSH. *Pain in Perspective* [online]. InTech, 2012, s. 238-245 [cit. 2019-05-04]. DOI: 10.5772/51139. ISBN 978-953-51-0807-8. Dostupné z: http://www.intechopen.com/books/pain-in-perspective/a-novel-application-of-virtual-reality-for-pain-control-virtual-reality-mirror-visual-feedback-thera

[6] SWEE, Sim, Lim YOU, Benny HANG a Desmond KIANG. Development of rehabilitation system using virtual reality. In: *2017 International Conference on Robotics, Automation and Sciences (ICORAS)* [online]. IEEE, 2017, s. 1-6 [cit. 2019-05-05]. DOI: 10.1109/ICORAS.2017.8308045. ISBN 978-1-5386-1908-7. Dostupné z: http://ieeexplore.ieee.org/document/8308045/

[7] Operating Environment. *Leap Motion* [online]. b.r. [cit. 2019-05-09]. Dostupné z: https://support.leapmotion.com/hc/en-us/articles/223783708-Operating-Environment

[8] Unity Modules Documentation. *Github* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://leapmotion.github.io/UnityModules/

[9] Personal License. *Unity* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://unity3d.com/unity/activation/personal

[10] Frequently Asked Questions. *Unreal Engine* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://www.unrealengine.com/en-US/faq

[11] Leap Motion SDK. *Leap Motion* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://warehouse.leapmotion.com/apps/4621/download

[12] Leap Motion Unity Assets. *Leap Motion* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://developer.leapmotion.com/unity

[13] Hierarchy Window. *Unity Documentation* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://docs.unity3d.com/Manual/Hierarchy.html

[14] Leap Motion Forums. *Leap Motion* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://forums.leapmotion.com/

[15] Unity Forum. *Unity* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://forum.unity.com/

[16] HDRis. *HDRi Haven* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://hdrihaven.com/hdris/

[17] SQLite Documentation. *SQLite* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://www.sqlite.org/docs.html

[18] SQLite-net. *Github* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://github.com/praeclarum/sqlite-net

[19] SQLite-net Extensions. *Bitbucket* [online]. b.r. [cit. 2019-05-12]. Dostupné z: https://bitbucket.org/twincoders/sqlite-net-extensions

[20] .NET guide: Serialization guidelines. *Microsoft* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/serialization/serialization-guidelines

[21] SharpZipLib. *Github* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://github.com/icsharpcode/SharpZipLib

[22] MessagePack-CSharp. *Github* [online]. b.r. [cit. 2019-05-12]. Dostupné z: https://github.com/neuecc/MessagePack-CSharp

[23] Gravit Designer. *Gravit* [online]. b.r. [cit. 2019-05-06]. Dostupné z: https://www.designer.io/

[24] RAMACHANDRAN, V. a E. ALTSCHULER. The use of visual feedback, in particular mirror visual feedback, in restoring brain function. *Brain* [online]. 2009, **132**(7), 1693-1710 [cit. 2019-05-05]. DOI: 10.1093/brain/awp135. ISSN 0006-8950. Dostupné z: https://academic.oup.com/brain/article-lookup/doi/10.1093/brain/awp135

[25] Cubemap. *Unity Documentation* [online]. b.r. [cit. 2019-05-13]. Dostupné z: https://docs.unity3d.com/Manual/class-Cubemap.html

[26] How Does the Leap Motion Controller Work?. *Leap Motion* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

[27] Designing the Leap Motion Controller. *Leap Motion* [online]. b.r. [cit. 2019-05-06]. Dostupné z: http://blog.leapmotion.com/designing-leap-motion-controller/

Seznam obrázků

Obrázek 1 Pacient při rehabilitaci zrcadlovou terapií. Zdroj: [24] 11
Obrázek 2 Použití snímací rukavice. Zdroj: [5] 13
Obrázek 3 Pohybový senzor Leap Motion. Zdroj: [27] 14
Obrázek 4 Prostor snímání senzoru Leap Motion. Zdroj: [26] 14
Obrázek 5 Návrh GUI hlavní nabídky. Zdroj: vlastní 21
Obrázek 6 Návrh GUI přehrávání. Zdroj: vlastní 22
Obrázek 7 Návrh GUI nahrávání. Zdroj: vlastní 23
Obrázek 8 Diagram aplikačních komponent. Zdroj: vlastní 24
Obrázek 9 Diagram případu užití. Zdroj: vlastní 24
Obrázek 10 Vývojový diagram při spuštění aplikace. Zdroj: vlastní 25
Obrázek 11 Návrh databázového modelu. Zdroj: vlastní 26
Obrázek 12 Ovládací panel Leap Motion. Zdroj: vlastní 27
Obrázek 13 Ukázková Unity scéna Leap Motion. Zdroj: vlastní 28
Obrázek 14 Ukázka entity v Unity. Zdroj: vlastní 29
Obrázek 15 Hierarchie projektu v Unity. Zdroj: vlastní 29
Obrázek 16 Ukázka kódu třídy MirrorProvider. Zdroj: vlastní 30
Obrázek 17 Ukázka funkce pro změnu modelu ruky. Zdroj: vlastní 31
Obrázek 18 HDR obraz v cylindrické projekci. Zdroj: [25] 32
Obrázek 19 Ukázka cylindrického HDR obrazu. Zdroj: [16] 32
Obrázek 20 Program Gravit Designer. Zdroj: vlastní 36
Obrázek 21 Vlastní vytvořené ikony. Zdroj: vlastní 36

Obsah přiloženého CD

Keywords.pdf ..... Klíčová slova
Abstrakt_CZ.pdf ..... Abstrakt v českém jazyce
Abstrakt_EN.pdf ..... Abstrakt v anglickém jazyce
Zadani_BP.pdf ..... Zadání bakalářské práce
Fulltext.pdf ..... Kompletní bakalářská práce
Uzivatelsky_navod.pdf ..... Uživatelský návod
Source.zip ..... Vlastní část zdrojového kódu
Program.zip ..... Spustitelná aplikace
License.txt ..... Plné znění MIT License
Dotaznik.pdf ..... Dotazník k testování aplikace