VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZHRANÍ PRO HRY S DATAPROJEKTOREM A LEAP MOTION

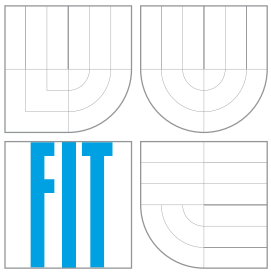
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ADAM TOMEČEK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZHRANÍ PRO HRY S DATAPROJEKTOREM A LEAP MOTION

GAME INTERFACE WITH DATA PROJECTOR AND LEAP MOTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ADAM TOMEČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ ZAHRÁDKA
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce se zabývá tvorbou herního rozhraní pro dataprojektor a snímač pohybu dlaní Leap Motion. Popisuje princip mapování obrazu na objekty a různé druhy mapování a jejichy využití. Praktická část se zabývá návrhem herního rozhraní pro Leap Motion a dataprojektor a návrhem demonstračních her. Rovněž je zde popsán postup testování implementovaného rozhraní a her s uživateli a výsledky testování.

Abstract

This thesis deals with creating of game interface for projector and palm motion sensor Leap Motion. It describes principles of projection mapping on objects and different types of projection mapping and its use. Practical part deals with designing game interface for Leap Motion and projector and designing of demonstration games. It also describes the procedure of testing created game interface and games programmatically and with users.

Klíčová slova

Leap Motion, dataprojektor, mapování obrazu, počítačová hra, herní rozhraní, Unity

Keywords

Leap Motion, projector, projection mapping, game interface, video game, Unity

Citace

Adam Tomeček: Rozhraní pro hry s dataprojektorem
a Leap Motion, bakalářská práce, Brno, FIT VUT v Brně, 2015

Rozhraní pro hry s dataprojektorem a Leap Motion

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Zahrádky

.....
Adam Tomeček
19. května 2015

Poděkování

Děkuji vedoucímu práce panu Ing. Jiřímu Zahrádkovi za odbornou pomoc a vstřicnost. Děkuji také svým blízkým, kteří se účastnili uživatelského testování a dávali mi cennou zpětnou vazbu.

Obsah

Kapitola 1

Úvod

Už několik desítek let se snaží výrobci hardware a počítačových her přicházet s revolučními ovladači, které nevyužívají klasický koncept ovládání her pomocí klávesnice a myši nebo gamepadu, ale zaměřují se na zpracování pohybu hráče tak, aby hra, ovládání, i pocity zanechané v hráči působily co nejrealističtěji. Tyto pokusy v minulosti často skončily neúspěchem kvůli nekvalitě zařízení a/nebo jejich ceně. V posledním desetiletí se však vyrojilo několik velmi úspěšných pohybových ovladačů, mezi které patří například Microsoft Kinect prodávaný jako příslušenství ke konzoli Xbox. Snímač Leap Motion je svým řešením nejblíže právě zmíněnému Kinectu s tím rozdílem, že místo snímání pohybů celého těla snímá pouze a dlaně.

Cílem této práce je navrhnout univerzální rozhraní pro ovládání počítačových her využívající Leap Motion s mapováním pohybu do obrazu z dataprojektoru a otestovat jej na k tomu navržených a vytvořených počítačových hrách. V tuto chvíli existují jen experimenty s mapováním Leap Motion a obrazu z dataprojektoru, ale žádná počítačová hra, která by kombinaci těcho zařízení využívala.

Práce je rozdělena následujícím způsobem. Ve druhé kapitole je popsána funkcionalita Leap Motion a popsány nástroje, které byly k tvorbě rozhraní a počítačových her využity. V kapitole třetí jsou popsány pokusy s mapováním obrazu, které jsem prováděl pro nalezení ideálního způsobu instalace Leap Motion a dataprojektoru, omezení použití Leap Motion a popis vytvořeného rozhraní. Kapitola čtvrtá popisuje proces návrhu a tvorby testovacích her. Kapitola pátá popisuje výsledky testování přesnosti Leap Motion, mapování obrazu z dataprojektoru a hratelnosti vytvořených her. Poslední kapitola, závěr, shrnuje zjištěné skutečnosti a poznatky při vývoji rozhraní, testovacích her a práci s Leap Motion.

Kapitola 2

Mapování obrazu, Leap Motion a použité nástroje

2.1 Mapování obrazu

Mapování obrazu je proces, kdy se promítaný obraz přesně promítne na vybraný objekt. Často se jako objekt používají budovy ale také složité 3D objekty. Na namapovaný objekt se pak promítá obraz, který může daný objekt naprosto změnit, rozpohybova či jinak opticky upravit. Mapovací software tak musí znát strukturu a polohu objektů, na které se bude obraz mapovat. Toto mapování bývá zpravidla statické bez interakce uživatele.

Obrázek 2.1: promítání obrazu na známou operu v Sydney (zdroj [18])

2.1.1 Interaktivní mapování

Interaktivní mapování je mapování obrazu na pohybující se objekt. Pro toto mapování je třeba znát tvar i polohu objektu v prostoru, do kterého se obraz promítá. Tvar objektu se dá opět mapovacímu sofware nastavit dopředu a poté stačí jen správně zjišťovat polohu objektu pomocí různých metod. Pokud ale chceme mapovat obraz na různé objekty, jejichž tvar předem neznáme, musíme použít nějaký druh snímače.

Protože se tato práce zabývá využitím pohybového snímače Leap Motion a promítaného obrazu v počítačových hrách, zaměřil jsem se na podobné práce využívající nejznámější pohybový senzor Kinect firmy Microsoft, pro který existuje spousta prací využívajících tento snímač pro mapování obrazu. Technickým řešením se navíc velmi podobá Leap Motion.

2.1.2 Kinect

Kinect je pohybový snímač primárně sloužící pro snímání pohybů těla k ovládání her na konzoli Xbox.

Ke snímání prostoru využívá Kinect infračervené světlo. Zařízení má v sobě infračervený zářič, dva infračervené snímače hloubky a barevnou kameru snímající rychlostí 30 snímků za sekundu. Díky snímačům hloubky je Kinect shopný přesně zjistit, jak daleko od něj předměty jsou.

Díky volnému API se však Kinect nevyužívá jen pro ovládání videoher, ale často také pro skenování objektů nebo právě mapování obrazu. K tomu se dá využít například nástroj `ofxKinectProjectorToolkit` Gene Kogana. Kalibrační postup je zdlouhavý a komplikovaný, ale výsledná přesnost mapování je podle dostupných videoukázek[20] dostatečná.

Zajímavě vypadá například využití v testovací hře RoomAlive, která využívá několik projektorů a snímačů Kinect k pokrytí celé místnosti, ze které se tak stane virtuální herní hřiště.[19]

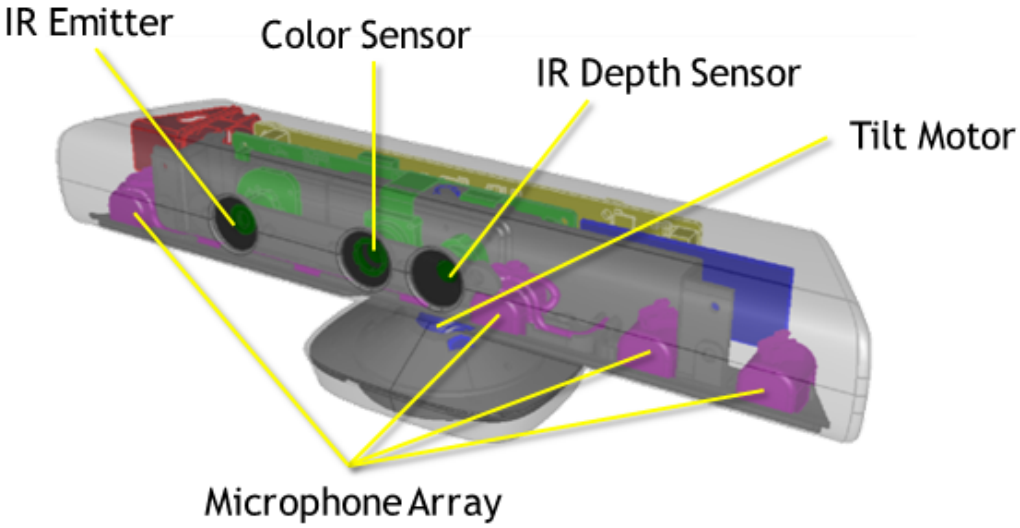
V testovacím prototypu RoomAlive použili autoři 6 jednotek, z nichž každá obsahuje snímač Kinect a dataprojektor, pro kompletní pokrytí pokoje včetně nábytku. Automatizovaný kalibrační proces kalibruje postupně každou z těchto jednotek. Pro zlepšení 3D efektu mapovaného obrazu navíc sofware sleduje polohu hráčovy hlavy a projekci na základě její pozice upravuje.

2.1.3 Snímání kamerou

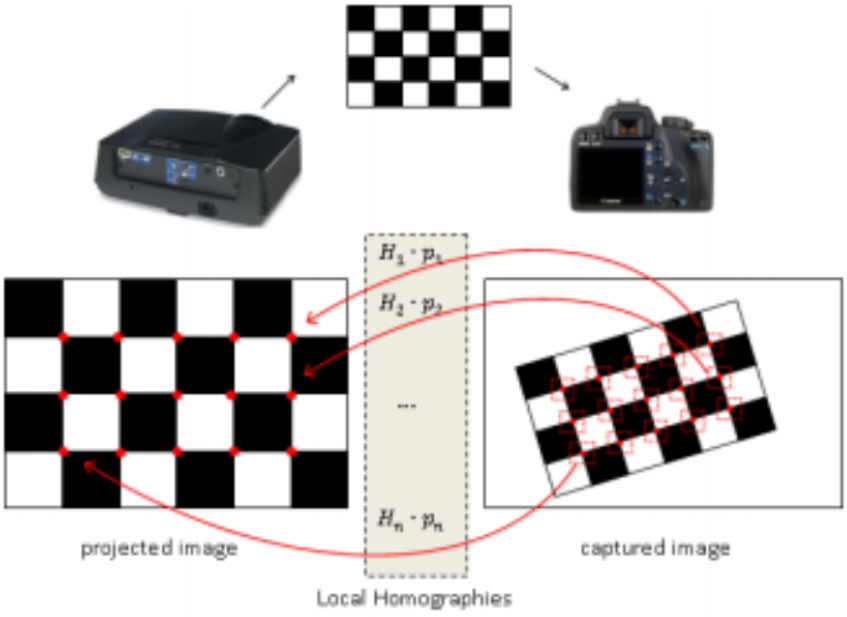
Pohybové snímače se při mapování dají nahradit kamerou správně zkalibrovanou s projektorem, která může být doplněna o další zařízení pro zvýšení přesnosti, například hloubkový snímač jako v Kinectu. V tomto případě tak pohybový snímač nahrazuje obraz natáčený kamerou zpracovaný metodou pro rozpoznávání obrazu. Rozpoznávání obrazu se dá navíc snadno rozšířit o rozpoznávání dalších objektů na rozdíl od proprietálních pohybových snímačů, jako jsou právě zmíněný Kinect nebo Leap Motion, s uzavřeným API, které většinou zpracovává informace o pohybu lidského těla nebo jeho části.

Obecné metody kalibrace kamery a dataprojektoru používají pro kalibraci šachovnicovou podložku známé velikosti, podle které se kalibrují interní parametry kamery (zkreslení způsobené optikou kamery) a promítání stejného šachovnicového vzoru, podle kterého se dá snadno rozpoznat rotace a naklonění obrazu vůči snímači, tedy pozice dataprojektoru vůči kameře (obr. 2.3).

Pro rozpoznávání objektů ve snímaném obraze je nutné předem vědět, jak daný objekt vypadá. To je snadné u objektů, které mají daný tvar. Obtížnější je to u snímání



Obrázek 2.2: schéma snímače Kinect (zdroj [9])



Obrázek 2.3: ukázka kalibrační podložky a princip použití, zdroj [22]

pohybujícího se lidského těla.

Například práce John MacCormicka a Michaela Isarda využívá pro zpracování dlaně v obraze v reálném čase metodu postupného vzorkování. Vzorky se porovnávají s databází předem vytvořených a upravených tvarů částí dlaně a postupně se tak rozpozná poloha, natočení a gesto jednotlivých částí, které se poskládají do finálního tvaru.[21]

Obrázek 2.4: ukázka sledování dlaně v obraze snímaném kamerou, zdroj [21]

2.1.4 Mapování pomocí Leap Motion

V případě této práce slouží jako objekt k mapování dlaň resp. dlaně uživatele ovládajícího hru pomocí k tomu vytvořeného herního rozhraní. Mapováním obrazu v práci tedy chápu jako přepočet metrických souřadnic pozice z reálného světa vrácených Leap Motion API do pixelových souřadnic obrazu tak, aby pozice částí dlaně přesně odpovídala pozici v promítaném obraze. Pokud tedy na pozici libovolné části dlaně namapuji objekt, je promítán přímo na vybranou část dlaně, která v tu chvíli překrývá část promítaného obrazu.

Pro správné mapování obrazu je napřed potřeba nasnímat body v ploše, které budou sloužit pro vzájemnou kalibraci zařízení. Počet potřebných kalibračních bodů se odvíjí od použité metody, počtu rozměrů a složitosti objektu, na který se má obraz mapovat.

V herním rozhraní jsem použil mapování do 2D obrazu promítaného dataprojektorem na rovnou plochu, pro které stačí nasnímat tři kalibrační body z dané plochy.

Obrázek 2.5: snímač Leap Motion (zdroj [17])

2.2 Leap Motion

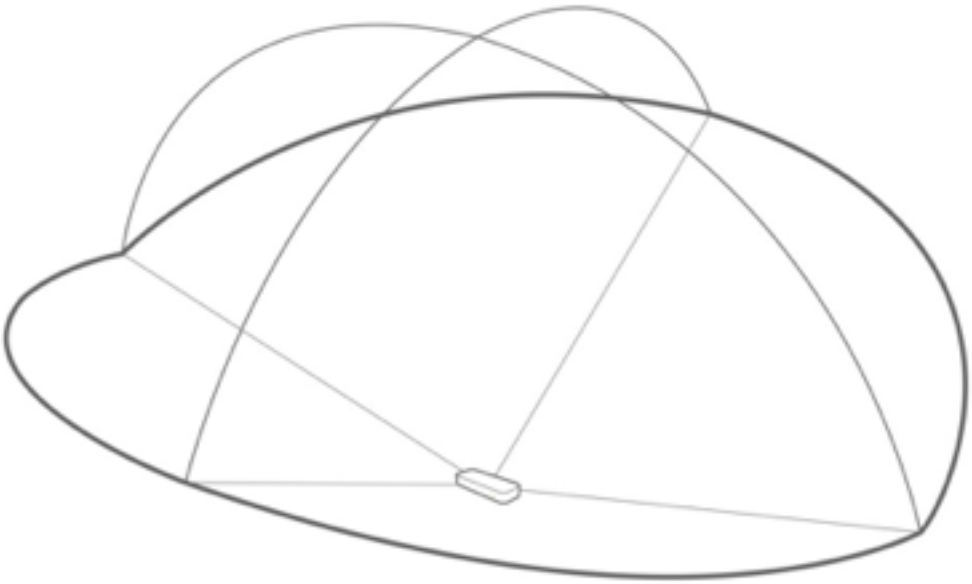
Leap Motion je senzor pohybu firmy Leap Motion, Inc zaměřený na snímání pohybu rukou, dlaní a prstů.

2.2.1 Hardware

Samotné zařízení má velikost přibližně 7.5 cm. Uvnitř se nachází tři infračervené LED a dvě infrakamery schopné snímat rychlostí až 150 snímků za sekundu.[17]

Díky širokoúhlým čočkám jsou infrakamery schopny snímat obraz v úhlu až 150° šířky a 120° hloubky do výšky přibližně 60 cm nad zařízením. To dává dohromady plochu až o velikosti 2.5 m^2.

Nasnímaná data ukládá Leap Motion do vnitřní paměti, provádí potřebné změny rozlišení nasnímaných obrázků a data poté posílá do počítače přes USB sběrnici.



Obrázek 2.6: ukázka rozsahu snímání Leap Motion (zdroj [17])

2.2.2 Software

Leap Motion umí fungovat ve více módech snímání, které se odvíjí od světelných podmínek a rychlosti počítače. Základní, tzv. balanced mód nabízí kompromis mezi rychlostí snímání a přesností. Přes USB 3.0 posílá data rychlostí 150 snímků za sekundu. Další mód, tzv. low-resouce je určen především pro hledání problémů nebo pro pomalé počítače. O 60% snižuje spotřebu energie, rychlost zasílání dat a vytížení počítače. Robustní mód pak dokáže zpracovat obraz za cenu menší přesnosti i při špatných světelných podmínkách s rychlostí až 140 snímků za sekundu.[5]

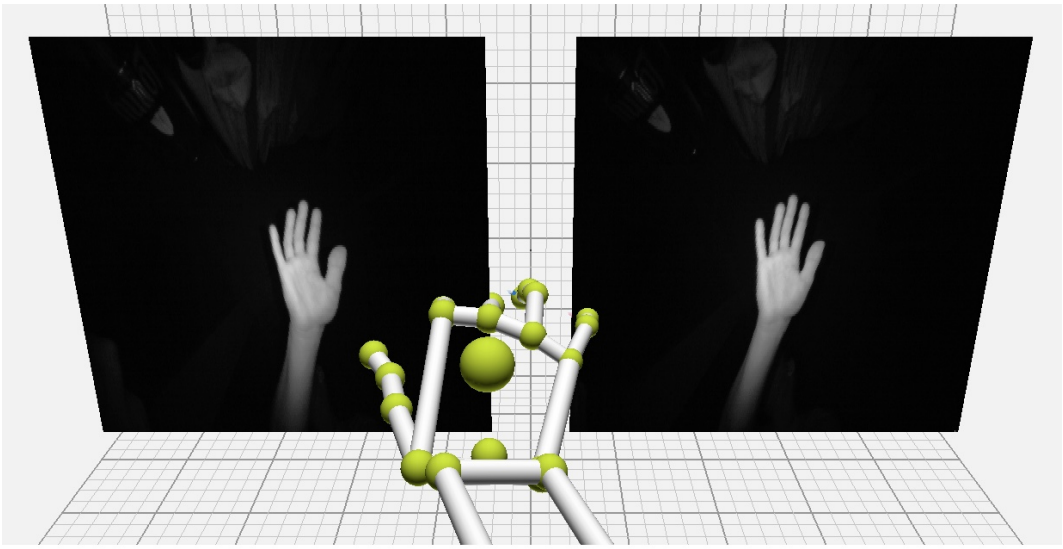
Po nastreamování dat do počítače začne software zpracovávat nasnímané obrázky pomocí algoritmu, který není veřejný. Na rozdíl od Kinectu však Leap Motion nevyužívá snímač hloubky. Rekonstruuje obraz na základě obrazu ze dvou vzájemně posunutých kamer.

Po odstranění objektů v pozadí snímané oblasti je obraz analyzován a vytvořena jeho 3D rekonstrukce. Z této rekonstrukce pak software extrahuje informace o objektech jako jsou prsty nebo nástroje držené v ruce.

Výsledná data jsou pak posílána pomocí transportního protokolu. Ten funguje na bázi služby využívající lokální TCP socket. Protokol pak využívá objektové Leap Motion API.

2.2.3 Vizualizér

Pro rychlou kontrolu snímání nabízí Leap Motion aplikaci Vizualizér. Aplikace zobrazuje model nasnímané dlaně se všemi význačnými body tak, jak se dají získat přes API. Navíc si uživatel může zobrazit aktuálně snímaný obraz z obou kamer pro detekci potenciálně špatně nasvětlených míst a odlesků, které můžou mít vliv na přesnost snímání. Právě snímaná data se také dají pozastavit pro lepší detekci a zjišťování anomálií a nepřesností.



Obrázek 2.7: screenshot z aplikace vizualizér

2.2.4 Leap Motion API

Leap Motion nabízí velmi kvalitní a dobře dokumentované API s podporou různých programovacích jazyků a oficiální podprou herních enginů Unity a Unreal.[4]

Mezi podporované jazyky patří:

- JavaScript
- C#
- C++
- Java
- Python
- Objective-C

API Leap Motion je objektově strukturované a umožňuje práci s daty různých částí dlaně. Je tak snadné zjistit směr, pozici a rychlost pohybu každé kosti stejně jako směr a pozici prstů nebo nástrojů držených v ruce. Jako pozice je vrácena relativní vzdálenost od středu Leap Motion v milimetrech ve třech osách.[3]

2.2.5 Změny ve verzi API 2.0

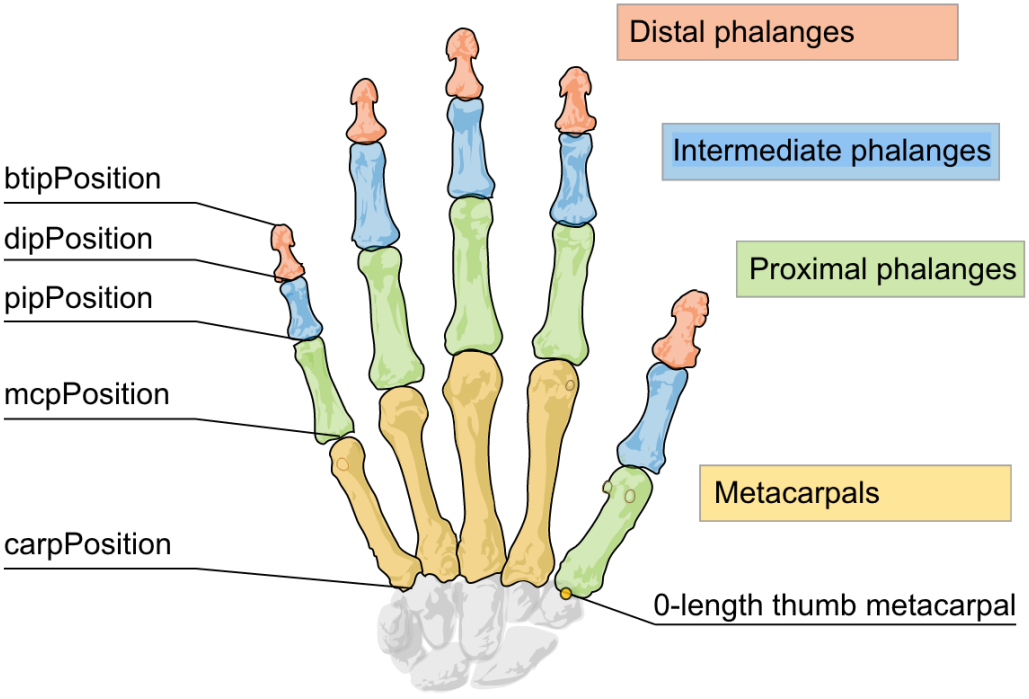
V době zkoumání Leap Motion API vyšla veřejně verze 2.0 resp. 2.1.4, která oproti verzi předchozí nabízí spoustu změn a vylepšení, například: [15]

- robustnější sledování rukou
- skeleton a bone API zpracovávající ruku po jednotlivých kostech
- podpora pro webové prohlížeče
- grafický vizualizér
- rychlejší a méně náročné zpracování obrazu

2.2.6 Skeleton API

Skeleton API je novinka ve verzi Leap Motion API 2.0 resp. 2.1.4. Leap Motion API nově modeluje a simuluje ruku. Díky tomu dosahuje větších přesností a je schopný predikovat pohyb skrytých částí rukou na základě předchozí pozice a předpokládaného chování ruky podle vnitřního modelu.

Sami vývojáři ale upozorňují, že se v aplikacích využívajících Leap Motion má autor vyhnout složitým polohám ruky, především pak polohám, ve kterých nejsou prsty nataženy.[2]



Obrázek 2.8: schéma dlaně v Leap Motion Skeleton API (zdroj [2])

2.3 Herní engine

Herní engine je soubor knihoven či framework, který nabízí funkce pro usnadnění tvorby počítačových her. Základní části enginu jsou zpravidla tato:[13][14]

- modul pro renderování 2D a/nebo 3D grafiky
- modul pro tvorbu 2D a/nebo 3D animací
- modul pro práci se zvukem
- modul pro čtení uživatelova vstupu včetně podpory gamepadů
- fyzikální knihovna a další
- podporu několika skriptovacích jazyků

Herních enginů je velká spousta a jsou mezi nimi velké rozdíly. Běžně se rozdělují do skupin, např. podle typu renderovacího enginu (2D nebo 3D), licence, pod kterou je engine uvolněn, zda je engine multiplatformní, jestli má podporu skriptovacího jazyka pro snadnější tvorbu scén apod.

Při hledání vhodného enginu byl Unity jediný herní engine přímo podporovaný ze strany Leap Motion. Koncem února 2015 byla nově přidána podpora pro Unreal engine.[6]

2.3.1 Unity

Unity patří mezi v současné době nejoblíbenější herní enginy s podílem na trhu větším než 40%. Především kvůli multiplatformnosti, rozsáhlosti, kvalitní dokumentaci a vstřícné cenové politice.[10]

Hru vytvořenou v Unity je možné spustit na více než 20 různých platformách včetně mobilních zařízení nebo webových prohlížečů.[12]

Unity nabízí spoustu modulů maximálně usnadňujících tvorbu počítačových her:

- kvalitní renderovací engine s podporou 2D a 3D grafiky
- animační nástroje
- zvukový engine podporující pokročilou manipulaci se zvuky přímo za běhu hry
- podporu 2D a 3D fyziky za využití Box2D resp. NVIDIA PhysX
- skriptovací jazyky C#, JavaScript a Boo
- robustní editor pro tvorbu scén a správu objektů

Velmi užitečnou a práci usnadňující součástí je tzv. Unity Asset Store. Obchod s různými již hotovými balíčky, které obsahují grafické prvky či 3D modely, zvuky, rozšíření Unity editoru a podobně.[13]

Unity je navíc zcela zdarma v plně funkční verzi pokud výdělky ze hry nedosáhnou limitní částky 100 000 dolarů. Pokročilejší uživatelé si mohou koupit nebo předplatit komerční Pro verzi Unity za 1 500 dolarů ročně resp. 75 dolarů měsíčně. Oproti běžné verzi nabízí Pro verze především lepší technickou podporu, analytické a výkonnostní analyzátory a další.[11]

2.3.2 C#

Jako implementační jazyk jsem použil C#, který je součástí herního enginu Unity a stejně tak je podporován v Leap Motion API. Oproti dalšímu podporovanému jazyku, JavaScriptu, je sice složitější díky striktnímu typování, ale přináší mnohé výhody, jako např. třídně založené objekty nebo syntaktická analýza během kompilace. Tyto výhody jsou pro mě důležitější než méně náročná práce s datovými typy u dynamicky typovaného JavaScriptu.[7][8]

Kapitola 3

Návrh rozhraní a testovacích her

3.1 Omezení Leap Motion

Během zkoumání funkčnosti Leap Motion vyšla nově verze API 2.0[16]. Software měl subjektivně problémy se zpracováním dlaně, která nebyla v základní pozici, kdy Leap Motion snímá plně viditelnou dlaň ze spodní části. Při natočení dlaně nebo snímání dlaně zeshora nebylo dosahováno uspokojivé přesnosti. Tato pozorování velmi ovlivnila návrh rozhraní a testovacích her.

3.1.1 Přesnost snímače

Subjektivně má Leap Motion daleko k dokonalosti. Velmi přesně snímá a zpracováva pohyb plně otevřených a viditelných dlaní směrem ke kamerám při dobrých světelných podmínkách. Pokud ale začnou snímané dlaně provádět složitější pohyby, otáčet se, schovávat prsty a podobně, Leap Motion se velmi často ztratí a není schopný správně rozpoznat, v jakém stavu se dlaně nachází. Na to sami upozorňují autoři Leap Motion (viz kapitola 2.2.6)

Ve verzi softwaru 2.0 sice autoři zapracovali na predikční části zpracování obrazu, kdy i když dokjde k překrytí části dlaně, je software schopný na základě posledních pozic určit, kde se právě neviděné části nacházejí, ale při dlouhotrvajícím překrytí snímaná data o poloze a pohybu ztrácí na přesnosti.

Nová verze software sice vykazuje značně lepší výsledky při snímání ne zcela viditelné nebo natočené dlaně, ale v době prvních pokusů a návrhů prototypů ještě nebyla vydána.

Stejně tak i když specifikace snímače deklaruje rychlost snímání a zpracování až 150 snímků za sekundu, má snímání zpoždění, které se projeví při jakémkoliv pohybu.

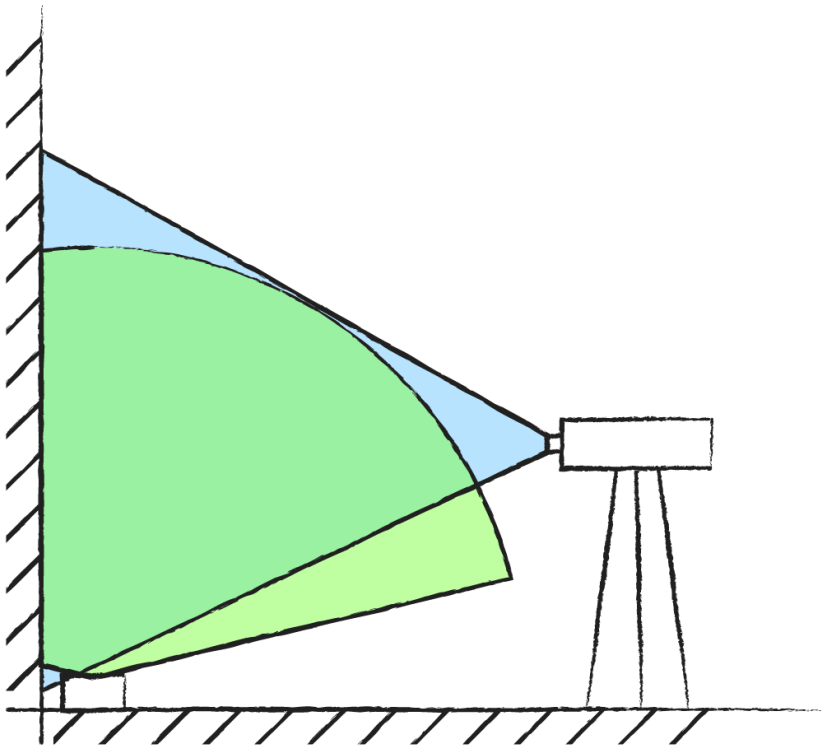
Rychlost a přesnost ovládání jsou však pro počítačovou hru kritické a bylo třeba s nimi při návrhu a vytváření hry počítat.

3.2 Umístění Leap Motion a dataprojektoru

Při experimentování s umístěním Leap Motion a dataprojektoru jsem hledal kompromis s přihlédnutím na co největší přesnost snímání, kvalitu promítaného obrazu a pohodlnosti pro uživatele. Kvůli možnosti mapovat obraz na uživatelovy dlaně jsem musel využívat přední projekci.

3.2.1 Umístění první

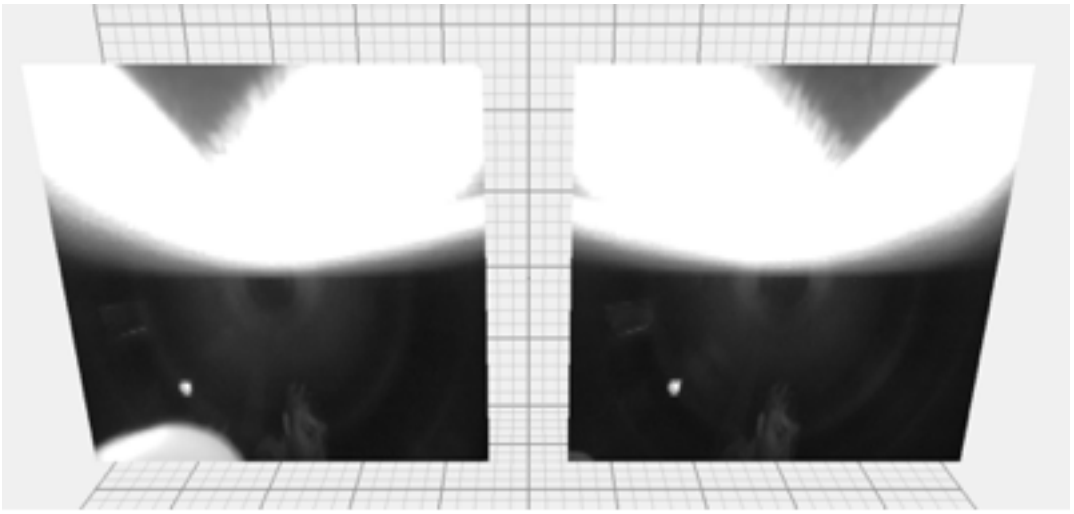
Při prvních pokusech jsem umístil projektor klasickým způsobem, tak by promítal obraz kolmo na zeď. Leap Motion byl umístěn ke zdi těsně pod promítaný obraz (obr. 3.1)



Obrázek 3.1: ilustrační nákres prvního umístění dataprojektoru a Leap Motion

Při tomto rozložení jsem narazil na několik problémů:

- dataprojektor musel být blízko zdi, aby promítaný obraz nebyl příliš velký a Leap Motion tak dokázal pokrýt celou plochu obrazu
- uživatel si příliš stínil a velká část obrazu tak nebyla vidět
- snímání Leap Motion bylo ovlivněno odleskem od zdi a snímání tak nebylo přesné (viz obr. 3.2)
- při snaze si co nejméně stínit obraz se snímaná dlaň dostávala do poloh, které nebyl Leap Motion schopen přesně rozeznat



Obrázek 3.2: ukázka odrazu světla ve snímaném obraze Leap Motion při umístění ke zdi

Jedinou výhodou tohoto umístění je, že se dá dataprojektor umístit prakticky kamkoliv a není třeba využít stativu.

Zjištěné problémy byly velmi omezující kvůli snížené přesnosti snímání, velkému stínění obrazu a nepohodlí uživatele při ovládání. Proto jsem toto umístění zavrhl.

3.2.2 Umístění druhé

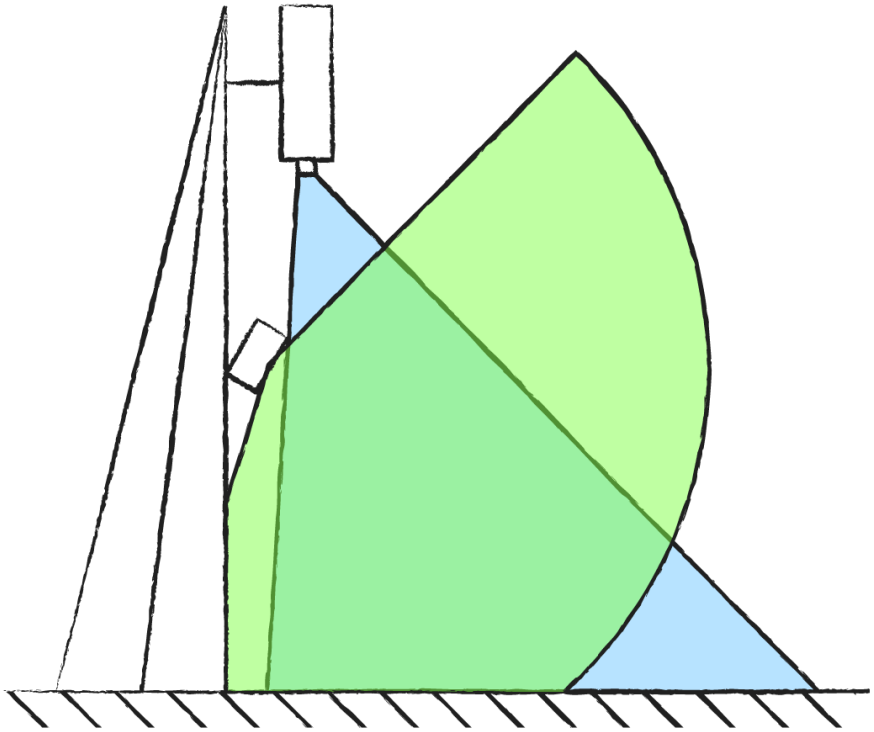
Ve druhém návrhu jsem projektor umístil na stativ tak, aby promítal obraz kolmo dolů na zem. Leap Motion jsem umístil přibližně 50 cm nad zem snímající plochu s promítaným obrazem tak, aby ruce snímal zeshora (obr 3.3)

I při tomto umístění jsem narazil na některé problémy:

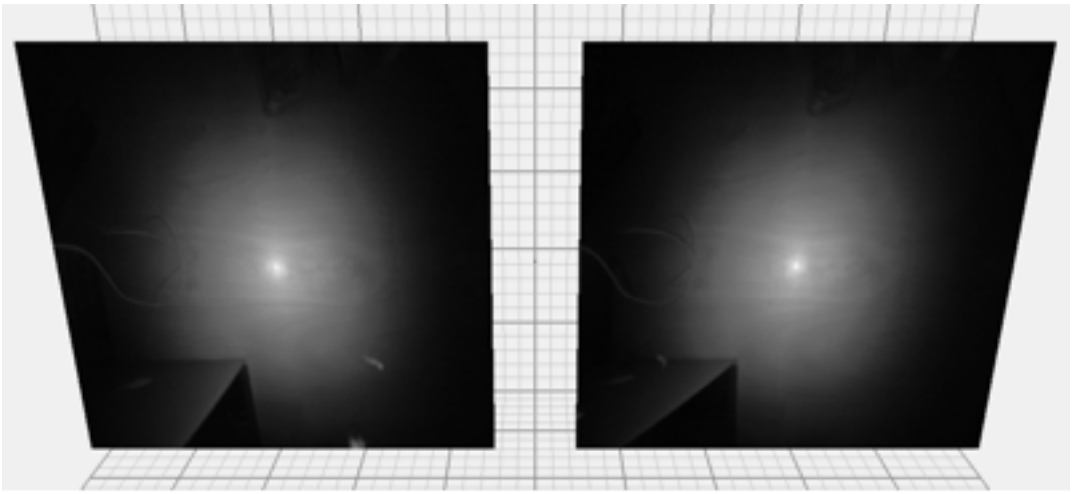
- kvalita snímaného i promítaného obrazu se velmi lišila podle použitého povrchu
- jak jsem popsal výše, Leap Motion měl problém s přesností při snímání dlaně zeshora
- nutnost umístit Leap Motion na konstrukci, která nestíní v obrazu projektoru a neomezuje uživatele v pohybu
- nutnost použít stativ pro upevnění dataprojektoru

Na druhou stranu toto umístění nabízelo i výhody:

- stínění uživatele bylo omezeno na minimum
- pro uživatele přirozenější ovládání

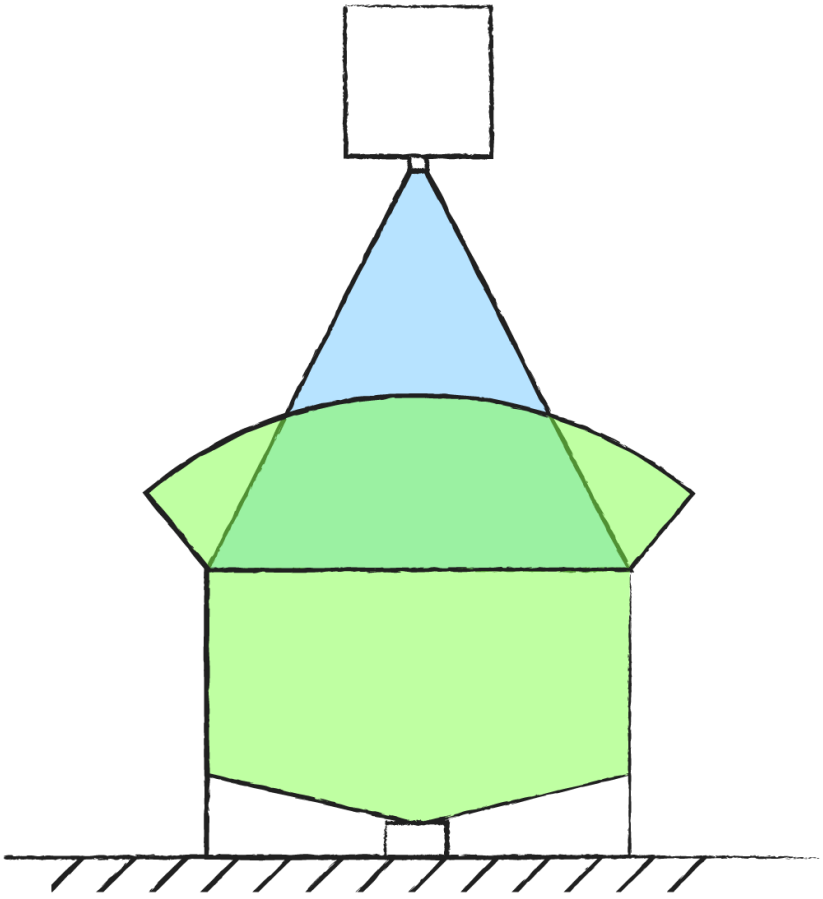


Obrázek 3.3: ilustrační nákres druhého umístění dataprojektoru a Leap Motion



Obrázek 3.4: ukázka odrazu světla ve snímaném obraze Leap Motion při snímání plovoucí podlahy

Díky omezené přesnosti snímání dlaně zeshora jsem po několika experimentech zavrhl i toto umístění.



Obrázek 3.5: ilustrační nákres třetího umístění dataprojektoru a Leap Motion

3.2.3 Umístění třetí

U tohoto návrhu jsem spojil umístění dataprojektoru tak, aby promítal směrem dolů kvůli omezení stínění a zároveň Leap Motion snímající dlaň zespod kvůli vyšší přesnosti. Toho jsem docílil umístěním Leap Motion pod průhlednou promítací plochu (obr 3.5), která zachycuje promítaný obraz a zároveň propouští infračervené světlo vysílané snímačem.

Jako promítací plochu jsem použil fólii sloužící k zakrytí nábytku při malování zakoupenou v místních domácích potřebách. Je průhledná, ale ne zcela čirá. Díky tomu částečně rozptyluje světlo z dataprojektoru a promítaný obraz je dostatečně kontrastní. Naopak dostatečně propouští světlo z Leap Motion umístěného pod ní a při dobrém vypnutí nevytváří výrazný odraz ve snímaném obrazu, který by příliš snižoval přesnost.

Jako kostra pro vypínání fólie je použita lepenková krabice. Díky tomu se dá vše relativně snadno přenášet.

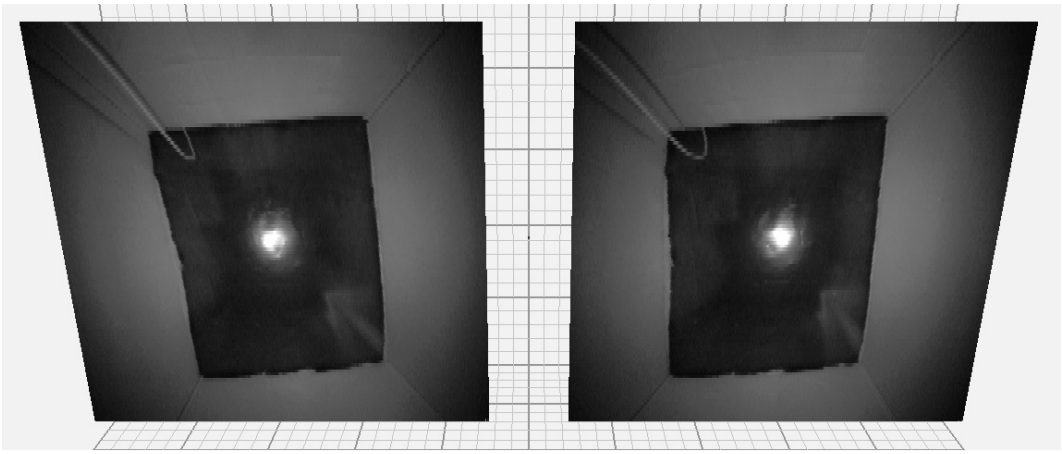
Kvůli omezení odlesku ve fólii jsem zkoušel experimentovat se zaslepením interního osvětlení Leap Motion a použití osvětlení externího, které by dlaň osvětlovalo až za promítací plochou. Ukázalo se, že zaslepit osvětlení Leap Motion není možné bez značného zaslepení širokoúhlých kamer, které příliš omezovalo snímaný obraz.

Experimentoval jsem také s nahrazením fólie tabulkou skla, to ale nedostatečně rozptylovalo světlo promítaného obrazu (obr 3.7). Promítaný obraz nebyl tak kvalitní a manipulace se sklem není přiiš pohodlná.

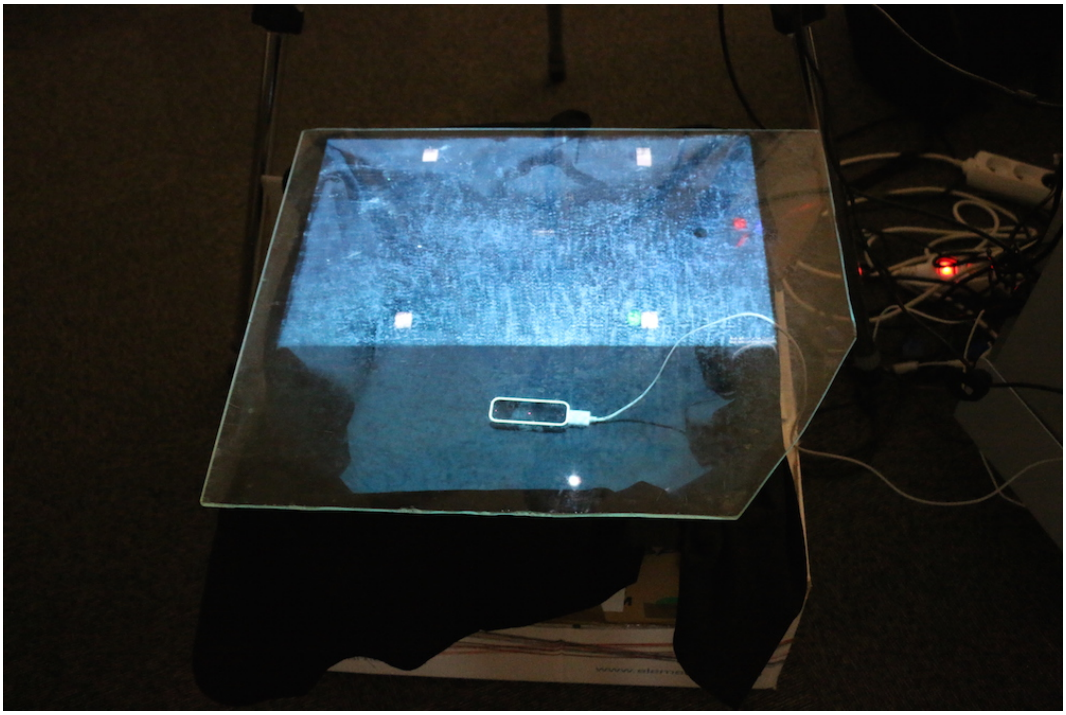
I u finálního rozmístění jsem ale narazil na některé problémy:

- nutnost použít stativ k upevnění dataprojektoru
- obraz promítaný na fólii není tak kvalitní, jako obraz promítaný na plátno či zeď
- snímaný obraz obsahuje slabý odlesk snižující přesnost ve středu obrazu (obr 3.6)
- nutnost použít konstrukci pro vypnutí fólie znamená horší přenositelnost

Větší přesnost snímání díky umístění Leap Motion tak, že snímá dlaň zespod však tyto problémy kompenzuje. U počítačových her je přesnost ovládání to nejdůležitější.

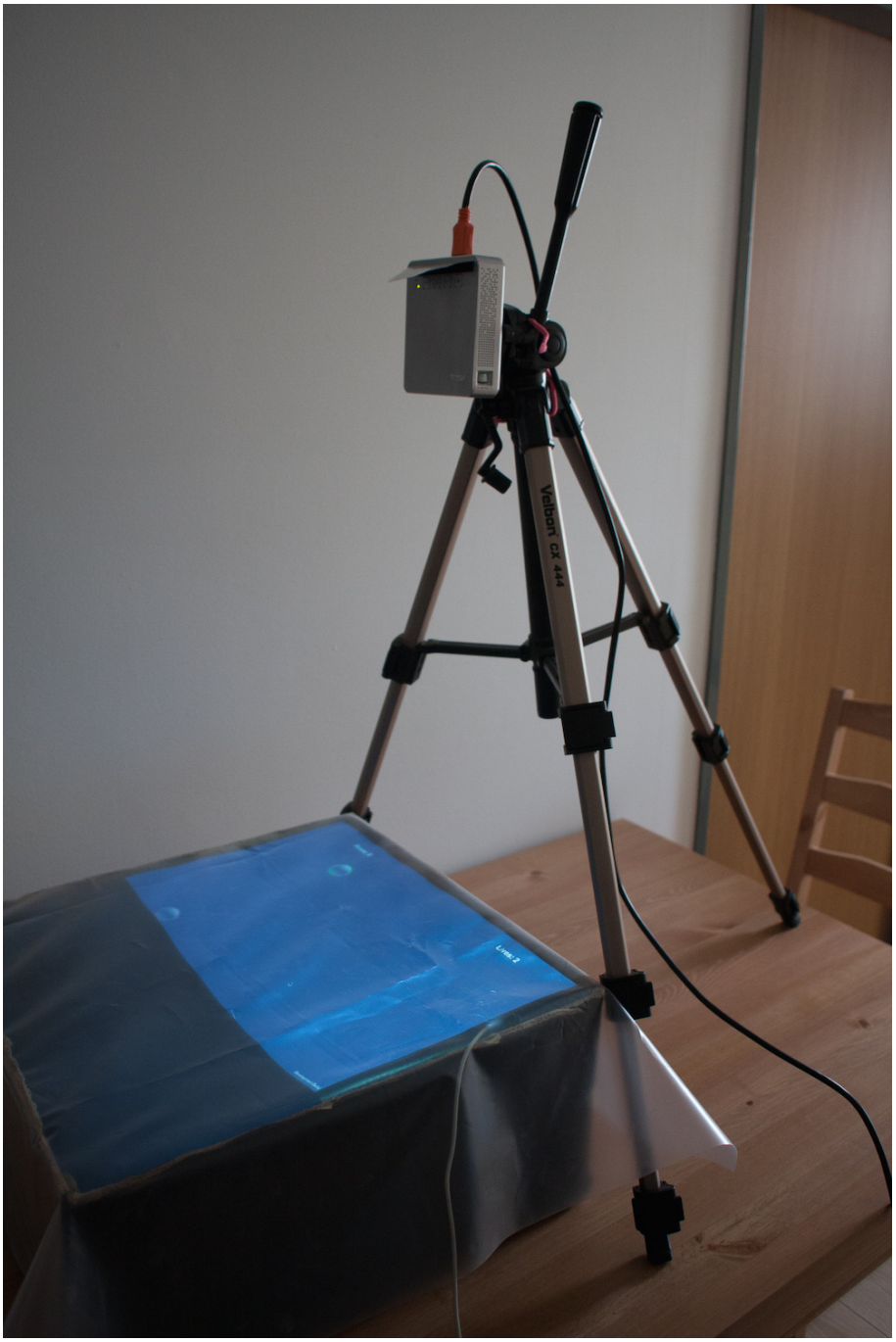


Obrázek 3.6: ukázka odrazu světla ve snímaném obraze Leap Motion umístěného v lepenkové krabici zakryté fólií



Obrázek 3.7: pokus s tabulkou skla místo fólie

Pro zjednodušení kalibrátoru je Leap Motion umístěn přibližně uprostřed promítací plochy a promítací plocha je ve tvaru přesného obdélníku. Dataprojektor a Leap Motion je proto nutno předem správně umístit tak, aby jejich vzájemná pozice neznepřesňovala kalibraci. Správné umístění se dá snadno docílit nastavením stativu s dataprojektorem a uložení Leap Motion v papírové krabici pod promítací plochu, kam nemá uživatel přístup. Tím jsem zjednodušil kalibrační proces a oprostil jej o posun a rotaci snímané plochy vůči ploše promítací a korekci pro promítaný obraz. Zároveň nebude docházet ke zkreslení herního rozhraní kvůli náklonu promítaného obrazu, které by mohlo znamenat nekonzistenci v ovládání her.



Obrázek 3.8: finální podoba instalace dataprojektoru a Leap Motion s promítáním na fólii

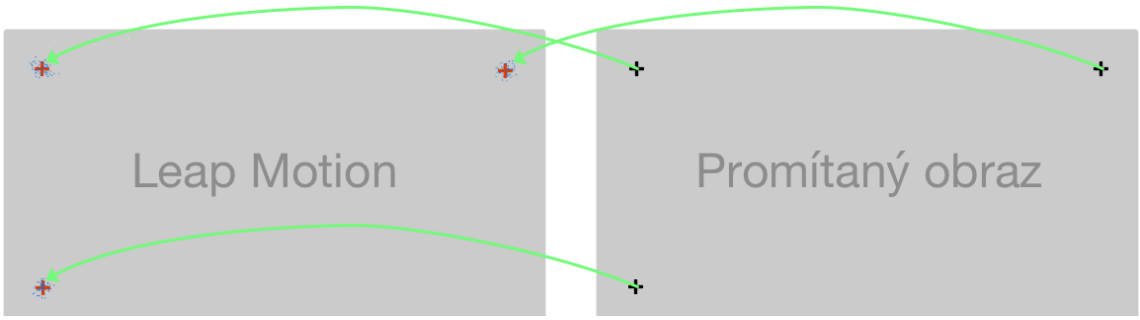
3.3 Návrh rozhraní kalibrátoru

Při návrhu rozhraní kalibrátoru jsem počítal s tím, že všechny testovací hry budou pouze ve 2D. 3D hry vytvořené pro Leap Motion, které jsem hrál, se velmi špatně ovládají. Doplněné o mírně sníženou přesnost snímání díky odrazu ve fólii by pravděpodobně bylo ovládání pro uživatele velmi frustrující.

Kalibrace probíhá jak na hardwarové úrovni správným umístěním dataprojektoru a Leap Motion, tak na softwarové úrovni zjišťováním velikosti obrazu podle souřadnic kalibračních bodů ve snímacím prostoru Leap Motion.

Pro 2D mapování postačuje nasnímat tři kalibrační body v ploše. Proto jsem pro snímání zvolil tři rohové body v promítaném obraze. Kalibrátor tedy postupně snímá polohu špičky prstu uživatele nad jednotlivými, předem vhodně vybranými kalibračními body, jejichž polohu známe. Tyto body slouží pro zjištění velikosti a pozice promítaného obrazu vzhledem k Leap Motion a následnému přepočtu libovolné souřadnice z Leap Motion API na souřadnice obrazu tak, aby pozice částí dlaní v obraze odpovídaly reálným pozicím.

Pro zpřesnění kalibrace a vyloučení lokálních extrémů a nepřesností uživatele se pro každý bod ukládají souřadnice z Leap Motion API snímané po dobu jedné sekundy. Tyto se pak zprůměrují a vytvoří výsledný kalibrační bod.



Obrázek 3.9: Ukázka použití kalibračních bodů. Modrou barvou nasnímané souřadnice z Leap Motion, červenou kalibrační bod vzniklý zprůměrováním nasnímaných souřadnic, černou body v promítaném obraze.

Pokud tedy známe souřadnice snímaných bodů v rozhraní Unity a tedy v promítaném obraze, můžeme snadno přepočítat souřadnice kalibračních bodů nasnímané pomocí Leap Motion na souřadnice Unity resp. souřadnice v promítaném obraze. Jako vstup kalibrace slouží souřadnice tří nasnímaných kalibračních bodů ze snímané plochy Leap Motion jako reálná poloha těchto bodů v promítaném obraze a poloha těchto bodů v Unity. Jak přepočet probíhá za předpokladu, že pozice bodů v obraze jsou na souřadnicích `[-1,1]`, `[-1,-1]` a `[1,1]` ukazují rovnice 3.1.

$$x_{unity} = \frac{x_{leap} - \frac{(x_1+x_0)}{2}}{\frac{x_1-x_0}{2}}$$
$$y_{unity} = \frac{y_{leap} - \frac{(y_0+y_2)}{2}}{\frac{y_0-y_2}{2}} \tag{3.1}$$

Proměnné x_{leap} a y_{leap} jsou souřadnice nasnímané pomocí Leap Motion a x_{unity} a y_{unity} jsou výsledné souřadnice pro Unity.

Tyto rovnice se používají vždy, když je třeba přepočítat souřadnice mezi Leap Motion API a Unity kvůli správnému namapování herních objektů na uživatelovy dlaně.

3.4 Testovací hry

Mým cílem při návrhu testovacích her bylo vytvářet hry takové, které by co nejvíce využívaly pohybové snímání a mapování obrazu promítaného dataprojektorem. Především jsem se chtěl vyvarovat hrám, které by šly snadno přenést a ovládat na dotykových zařízeních.

3.4.1 4kanoid

Původní nápad této hry byl inspirován známou hrou Arkanoid. V této hře uživatel kontroluje pálku, od které se odráží míček a ničí cihličky. Cílem je, aby se míček nedotkl spodní strany obrazovky, což vede ke ztrátě života a zároveň zničil všechny cihličky v levelu.

Při návrhu jsem vycházel z polohy špičky prstu, což je základní funkcionalitou Leap Motion API.

Prostá kopie Arkanoidu, kde hráč ovládá jen jednu pálku mi přišla příliš obyčejná. Do návrhu jsem proto zapracoval čtyři pálky různé barvy. Aby byla hra zajímavější, napadlo mě, že by hráč neodpaloval jeden míček, ale rovnou čtyři - pro každou pálku jeden. To jsem nakonec zavrhl jako příliš obtížné, proto jsem zvolil variantu jednoho míčku nutného odpálit pálkou barvy odpovídající barvě míčku, která se během hry mění.

Důležitou součástí původního Arkanoidu je nereálná fyzika odrážení míčku od pálky.Odrazy se nechovají podle pravidla úhel dopadu = úhlu odrazu. Naopak se míček odráží v opačném směru tak, že čím blíže je míček ke kraji pálky, tím ostřejší je úhel odrazu.

3.4.2 Cvrnkačka

Během studia Leap Motion API mě napadl návrh na hru, ve které by se cvrnkalo prstem do různých objektů. Leap Motion API by mělo umět zjistit směr a rychlost pohybu prstu.[1]

Inspirace vycházela ze hry, kterou si pamatuju z doby Flashových her - Peanut. V této hře hráč ovládal ruku a cvrnkal s ní do pohybujících se oříšků. Těmi se snažil trefovat veverky a/nebo branku. Pohybové ovládání se pro tuto hru hodí a mělo by přinést větší interaktivitu než jen pouhé ovládání myší.

3.4.3 LaserPopper

Hráč má ve hře za úkol pomocí laseru vycházejícího z jeho prstu praskat balónky pomalu padající z horní části obrazu. Za prasknutý zelený balónek je odměněn zvýšením skóre, za prasknutý červený balónek naopak ztratí život. Stejně tak přijde o život když zelený balónek dosáhne spodního okraje obrazovky.

Návrh této hry vznikl z předpokladu, že Leap Motion API umí detekovat nástroje držené v ruce a získávat informace jako poloha apod. stejně jako u prstu. V původním návrhu tedy měl hráč v ruce držet tužku, která by sloužila jako "laserové ukazovátko"a pomocí laseru pak praskat balónky.

3.4.4 Grafické materiály a zvuky

Protože nejsem zdatný grafik a s tvorbou zvukových efektů nemám žádné zkušenosti, rozhodl jsem se použít do testovacích her volné grafické materiály i zvukové efekty. Hledal jsem dobře strukturované databáze, které mají materiály k dispozici pod licencí, která by mi umožňovala je použít ve své hře bez nutnosti za ně platit.

Jako vhodná databáze pro grafické materiály se jevila webová stránka OpenGameArt.org, kde grafici dávají k dispozici své práce pro použití ve hrách zpravidla pod některou z verzí licence Create Commons. Ta v základu nabízí volnost libovolně používat dílo se zachováním autorství.

Pro zvukové efekty jsem použil databázi FreeSound.org, kde autoři také umisťují své zvuky pod licení Creative Commons.

Obě databázové stránky mají kvalitní vyhledávání s velkou možností filtrování a náhledy, takže vyhledání správné grafiky, resp. zvuku, je otázkou správně zadaného klíčového slova a chvíle vybírání.

Kapitola 4

Implementace

4.1 Kalibrátor a mapování

Součástí kalibrátoru je scéna v herním enginu Unity sloužící pro interakci s uživatelem během procesu kalibrace a třída, která zajišťuje načítání dat a samotný přepočet souřadnic mezi Leap Motion API a Unity tak, aby reálná pozice v obraze odpovídala pozici ve scéně.

Kalibrátor začne odpočítávání v momentě, kdy Leap Motion detekuje hráčovu ruku a poté barevně indikuje, na který bod má uživatel soustředit svou pozornost a které body již byly nasnímány. Mezi snímáním jednotlivých bodů jsou časové rozestupy indikované odpočítáváním uprostřed plochy, aby měl uživatel dostatek času k přesunu prstu na správný bod.

Po dokončení kalibrace se na uživatelovy prsty začnou mapovat barevná kolečka pro vizuální kontrolu přesnosti. Pokud s výslednou přesností není spokojený, může uživatel stisknutím klávesy spustit kalibraci znovu. Pokud vše vyhovuje, stiskem jiné klávesy se přepne zpět do hry.

Rozhraní je vytvořeno tak, aby kalibrace byla co nejsnadnější a trvalá, dokud se s instalací nepohne. Zároveň univerzální pro snadné použití ve více hrách díky umístění do samostatné scény.

4.1.1 Kalibrační scéna

Scéna obsahuje pouze čtyři objekty - v každém rohu krychli znázorňující místa kalibračních bodů, které je třeba načíst.

Pokud Leap Motion detekuje dlaň, začne odpočítávání kalibrace. Spustí se odpočet informující uživatele a krychle znázorňující aktuální bod pro snímání změní barvu. Během načítání pak krychle změní barvu ještě jednou a z Leap Motion API se začnou ukládat body pomocí metody `TipPosition` objektu `Finger`. Načítání probíhá pro nejpřednější prst. Ten se získá z objektu `FingerList` pomocí metody `Frontmost`. Pro větší přesnost se nenačítá pouze jeden bod, ale načítání probíhá sekundu a načtené pozice se poté zprůměrují. To vyloučí extrémy a nepřesnosti, které mohou během snímání nastat. Po načtení kalibračního bodu změní krychle znázorňující daný bod barvu detekující úspěšné načtení bodu a spustí se odpočet pro bod následující.

Po úspěšném načtení tří kalibračních bodů se na uživatelovy prsty začnou mapovat barevné kruhy pro vizuální kontrolu přesnosti kalibrace. Stejně tak se kalibrační body uloží do souboru kvůli trvalosti kalibrace.

Manipulaci objektů ve scéně zajišťuje třída `CalibrationController`. Samotné načítání kalibračních bodů z Leap Motion API a ukládání výsledků má na starosti třída `Calibrator`, se kterou `CalibrationControlller` při kalibraci komunikuje.

Obrázek 4.1: screenshot kalibrační scény po úspěšné kalibraci

4.1.2 Třída Calibrator

Nedůležitější třída herního rozhraní. Pro `CalibrationController` zajišťuje přístup do Leap Motion API a ukládání načtených kalibračních bodů. Pro testovací hry nabízí načítání kalibračních bodů uložených v souboru a přepočet souřadnic mezi Leap Motion API a Unity pro správné mapování obrazu. Přepočet zajišťuje veřejná metoda `GetPosition` výpočtem rovnic 3.1 uvedených v návrhu.

4.2 Testovací hry

Díky použití Unity enginu, se kterým jsem již dříve pracoval na několika hrách, bylo vyvíjení testovacích her velmi rychlé. Během vývoje jsem vytvořil několik základních prototypů, které jsem neustále testoval a vylepšoval tak, aby co nejvíce využily Leap Mootion API zároveň se hry ovládaly co nejpohodlněji a ovládání bylo přesné.

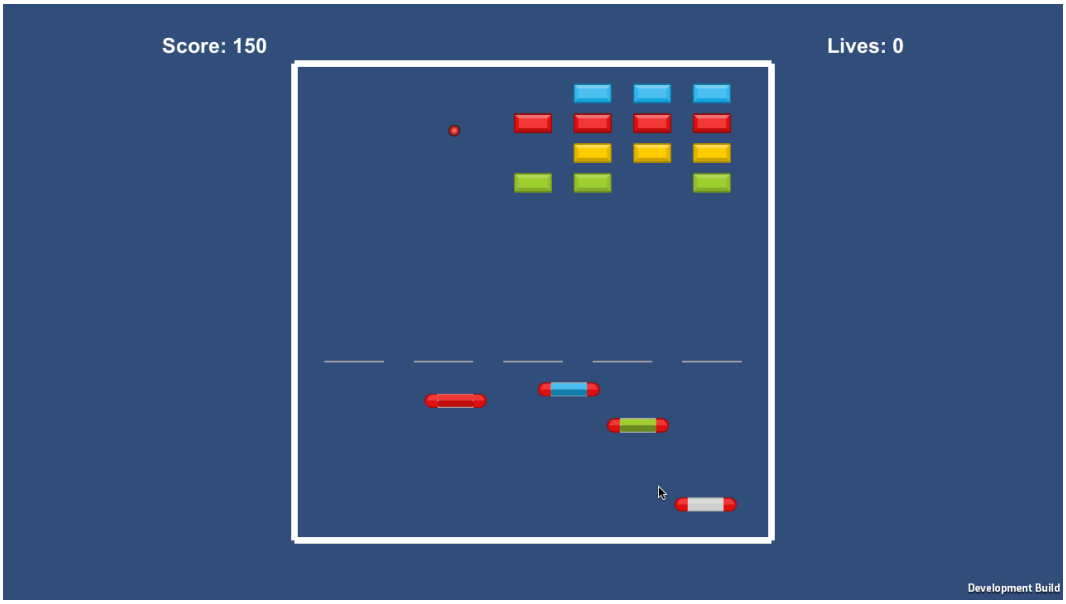
Pro snadnější testování bez neustálé potřeby využívat Leap Motion resp. celé herní rozhraní se dají hry ovládat také klasickým vstupem z klávesnice.

Každá z her využívá objekt `Calibrator` pro přepočet souřadnic načtené pozice z Leap Motion API.

4.2.1 4kanoid

Ve hře 4kanoid hráč ovládá čtyři pálky, které jsou namapovány na polohu prstů. Každá pálka má navíc jinou barvu a koliduje s míčkem jiné barvy, která se náhodně střídá po odpalu. Hráč tak musí sledovat nejen polohu míčku, ale také jeho barvu, aby vybral správnou pálku (prst) k dalšímu odrazu. Pokud se míček nepodaří odrazit a dotkne se spodní části obrazu, ztrácí život. Pokud hráč ztratí všechny životy, hra končí a scéna se restartuje.

Scéna obsahuje několik objektů. Mezi nejdůležitější patří objekt míčku, který má nastavený fyzikální materiál tak, aby se správně odrážel od zdí a cihliček. Jeho ovládací skript zajišťuje správné detekce kolizí a reakce na ně včetně rozdílného chování odrazu při nárazu na pálku. Dalšími důležitými objekty jsou pálky mapované na uživatelovy prsty. Mapování pálek zajišťuje scény. Objekt cihličky a její skript zajišťuje pouze odstranění cihličky po kolizi s míčkem. Samotné mapování pálek na prsty hráče zajišťuje kontrolér scény, který komunikuje s objektem `Calibrator` a nastavuje pálkám pozici tak, aby byla pálka dané barvy vždy namapována na stejný prst.



Obrázek 4.2: screenshot ze hry 4kanoid

4.2.2 Cvrnkačka

Pro správné ovládání hry má hráč cvrnkat prstem do pohybujících se objektů co největší rychlostí. Rychlost a směr pohybu prstu je snadné zjistit přes Leap Motion API.

Na základě tohoto předpokladu jsem vytvořil jednoduchý prototyp, kterým jsem chtěl ověřit správnou funkčnost této části API a především její přenost. Bohužel se ukázalo, že teorie popsaná v API v praxi příliš nefunguje. Pokud se zrovna Leap Motion neztratil při sledování kvůli prudkým pohybům a natočení dlaně, vracelo API naprosto nepřesné hodnoty.

Z tohoto důvodu jsem v dalším vývoji této hry nepokračoval, protože i přes několik experimentů a úprav v prototypu nebyla hra pomocí Leap Motion ovladatelná.

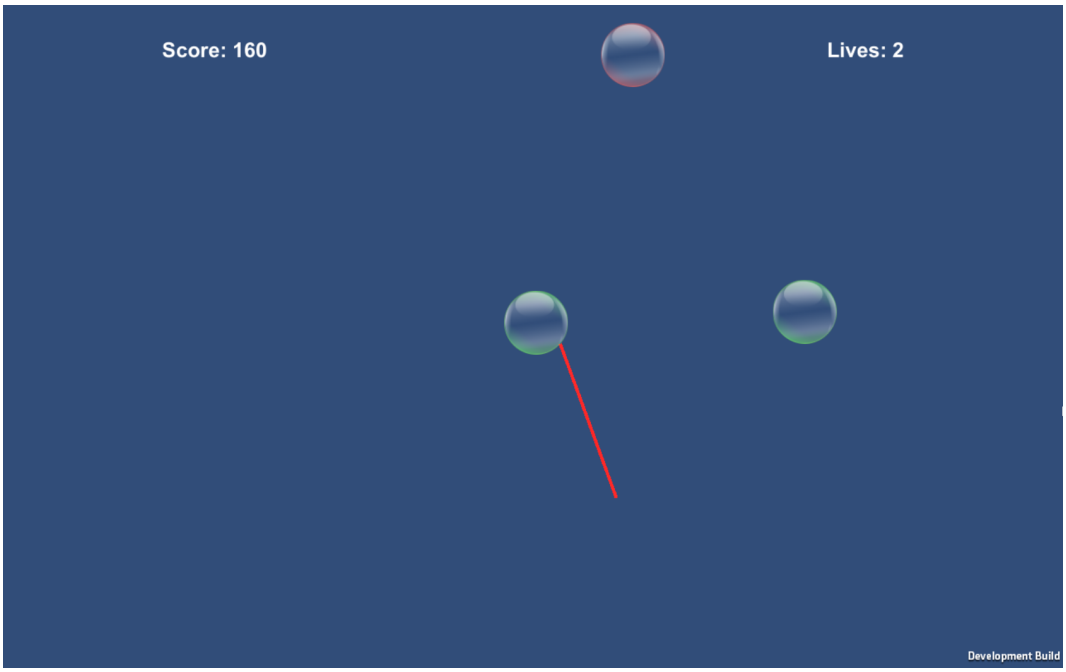
4.2.3 LaserPopper

Původní návrh hry vycházel z toho, že Leap Motion umí přesně detekovat předměty držené v ruce. Uživatel měl v ruce držet tužku či něco podbného a z té se měl promítat laserový paprsek. Po pár experimentech se bohužel ukázalo, že Leap Motion nástroje prakticky nedetekuje a zbývající informace o dlani se držením nástroje znepřesnily.

Podobný efekt se ale dá dosáhnout pomocí části API, která vrací informace o jednotlivých kostech v dlani. Nakonec jsem se tedy rozhodl laser kreslit na základě pozice kostí v ukazováčku a to od hlavice záprstní kosti (latinsky se skupina záprstních kostí nazývá Metacarpus, anglicky Metacarpals viz obrázek 2.8 z Leap Motion dokumentace) po špičku ukazováčku.

V horní části obrazu se náhodně objevují balónky, které díky nastavenému fyzikálnímu materiálu automaticky padají dolů. Nastavení materiálu se pro každý balónek náhodně generuje. Pomocí raycastingu se pak kreslená čára zastavuje přesně na hraanici balónků tak, aby nebyla přerušena kolize čáry a balónku. Po sekundě kolidování balónek praská a hráči je přičteno skóre resp. odečten život, pokud praskl balónek špatné barvy. Genrování balónků z předem vytvořených objektů zajišťuje kontrollér scény. Kreslení a mapování laseru má na starosti `LaserController`, který komunikuje s objektem `Calibrator`. Ovládací skript balónků pak zajišťuje jejich odstraňovaání ze scény v případě, že doba nepřerušené kolize s laserovým paprskem přesáhla jednu sekundu.

LaserPopper je oproti 4kanoidu teoreticky nekonečná hra. Neustále se zrychluje a zmenšují se prodlevy mezi vytvářením nových balónků.



Obrázek 4.3: screenshot ze hry LaserPopper

Kapitola 5

Testování

Základní testování funkčnosti a přenosti ovládání jsem prováděl neustále během návrhu rozhraní i implementace kalibrátoru a her. Většinu testování, kde jsem ověřoval přesnost Leap Motion v různých situcích jsem prováděl v Leap Motion vizualizéru. Především jsem pak testoval kontinuitu snímání dlaně při různých pohybech a natočení dlaní.

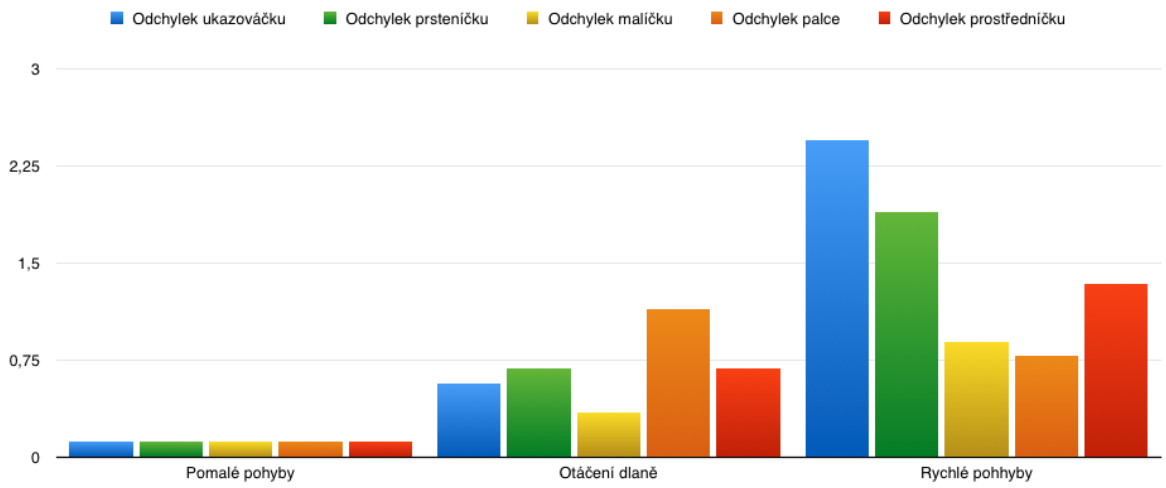
Obrázek 5.1: praktická ukázka - hráč hrající hru 4kanoid

5.1 Přesnost snímání kalibrace

V samotných prototypech jsem sledoval především přesnost snímání, zda se objekty opravdu mapují tam, kam mají a jak často a z jakého důvodu dochází ke krátkodobým chybám a nepřesnostem při snímání.

Pokud Leap Motion snímal dlaň správně, fungovalo mapování s přesností s rozptylem několika milimetrů. Prováděl jsem také programové testování hledající velké odchylky vzhledem k posledním nasnímaným souřadnicím a sledující, jak často Leap Motion přestane sledovat prst resp. prsty.

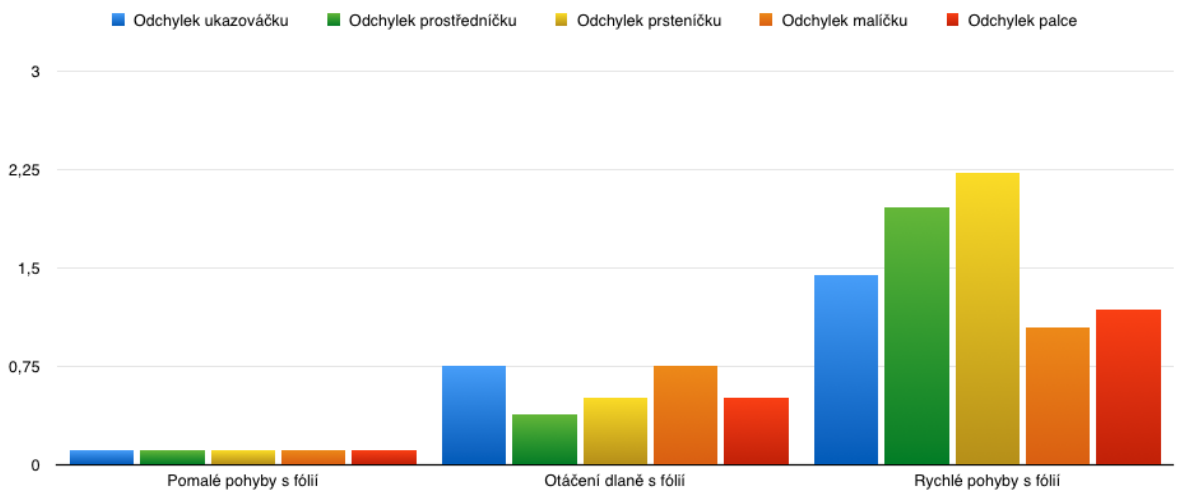
Program snímal polohy všech prstů po dobu 15 sekund. Mezi nasnímanými souřadnicemi hledá výrazné odchylky oproti poslední naměřené hodnotě a sleduje, jestli Leap Motion API vrací informace o všech prstech. Měření jsem prováděl nad snímačem Leap Motioin umístěném na stole bez žádné překážky, umístěném v krabici s fólií a snímajícím dlaň shora. Testoval jsem na jednoduchých pomalých pohybech až po rychlé náhodné pohyby dlaní. Výsledky testování jsou v grafech (5.2, 5.3 a 5.4), kde je počet odchylek na 100 měření. Zajímavé je, že Leap Motion stále vracel informace o všech prstech, ač někdy zcela nepřesné.



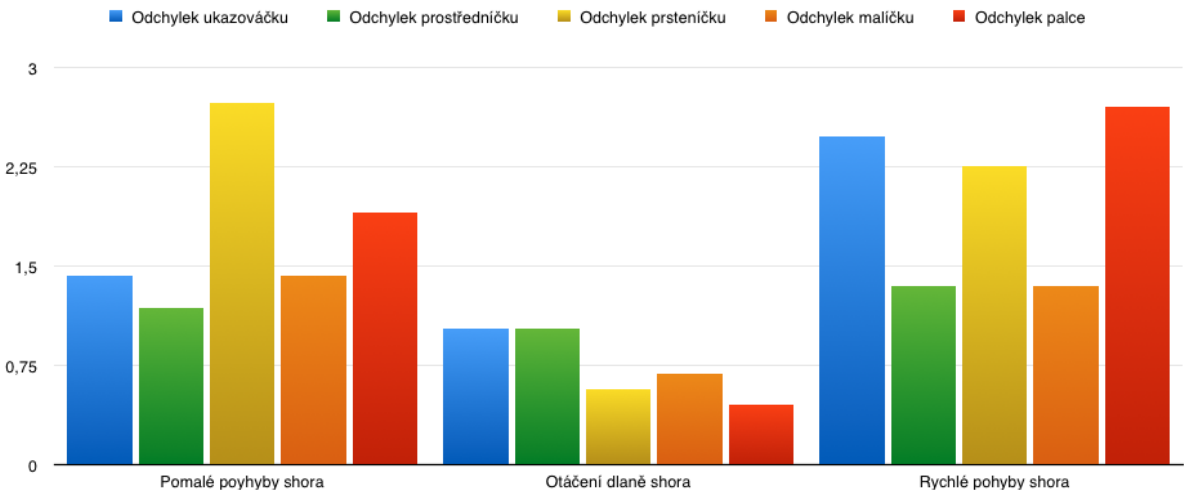
Obrázek 5.2: počet odchylek jednotlivých prstů na 100 měření u Leap Motion snímajícím dlaň zespod

Z výsledků je patrné, že snímání přees fólii přesnost snímání příliš neovlivňuje. Také je zřejmé, že přesnost klesá se složitostí a rychlostí pohybů stejně jako je z grafů patrné extrémní zhoršení přesnosti při snímání shora.

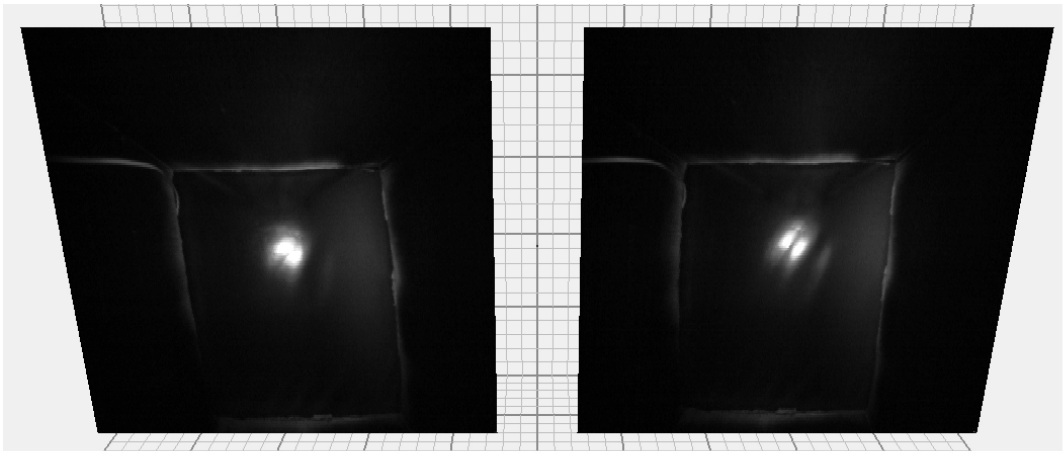
Během vývoje a testování her jsem přišel s několika vylepšeními instalace kalibrátoru a dataprojektoru. Mírného nárustu přesnosti a velkého zlepšení kvality obrazu jsem dosáhl vybarvením vnitřní části krabice matnou černou barvou, která neodráží světlo. Díky tomu se podařilo odfiltrovat většinu infračerveného světla z Leap Motion (obr. 5.5). Obraz z dataprojektoru navíc není zdvojený jako předtím, kdy část zachytila fólie a část se promítala na dno krabice.



Obrázek 5.3: počet odchylek jednotlivých prstů na 100 měření u Leap Motion snímajícím dlaň zespod přes fólii



Obrázek 5.4: počet odchylek jednotlivých prstů na 100 měření u Leap Motion snímajícím dlaň shora



Obrázek 5.5: krabice vybarvená černou matnou barvu neodráží žádné infračervené světlo

Další testování probíhalo formou uživatelského testování.

5.2 Uživatelské testování

Uživatelské testování bylo rozděleno do dvou částí. Napřed jsem testoval na menším vzorku uživatelů a pozoroval, jaké mají s kalibrováním a hraním her problémy. Na základě těchto poznatků jsem poté aplikace průběžně upravoval. Další, finální testování, probíhalo na skupině deseti uživatelů, kteří na konci vyplnili krátký dotazník, kde známkami jako ve škole hodnotili různé aspekty rozhraní a testovacích her.

5.2.1 Otázky v dotazníku

- kvalita obrazu promítaného na fólii
- obtížnost kalibrace
- přenost ovládání her
- pohodlnost ovládání her
- originalita her vzkledem ke způsobu ovládání

Výsledky dotazníku jsou v obrázku 5.6.

5.3 Instalace dataprojektoru a Leap Motion

Hodnocení kvality obrazu se značně odvíjelo od denní doby. Uživatelé, kteří rozhraní zkoušeli ve večerních hodinách logicky kvalitu hodnotili lépe než ti, kteří testovali za velkého světelného znečištění přes den. Průměrně pak uživatelé hodnotili kvalitu obrazu známkou 2.

5.4 Kalibrátor

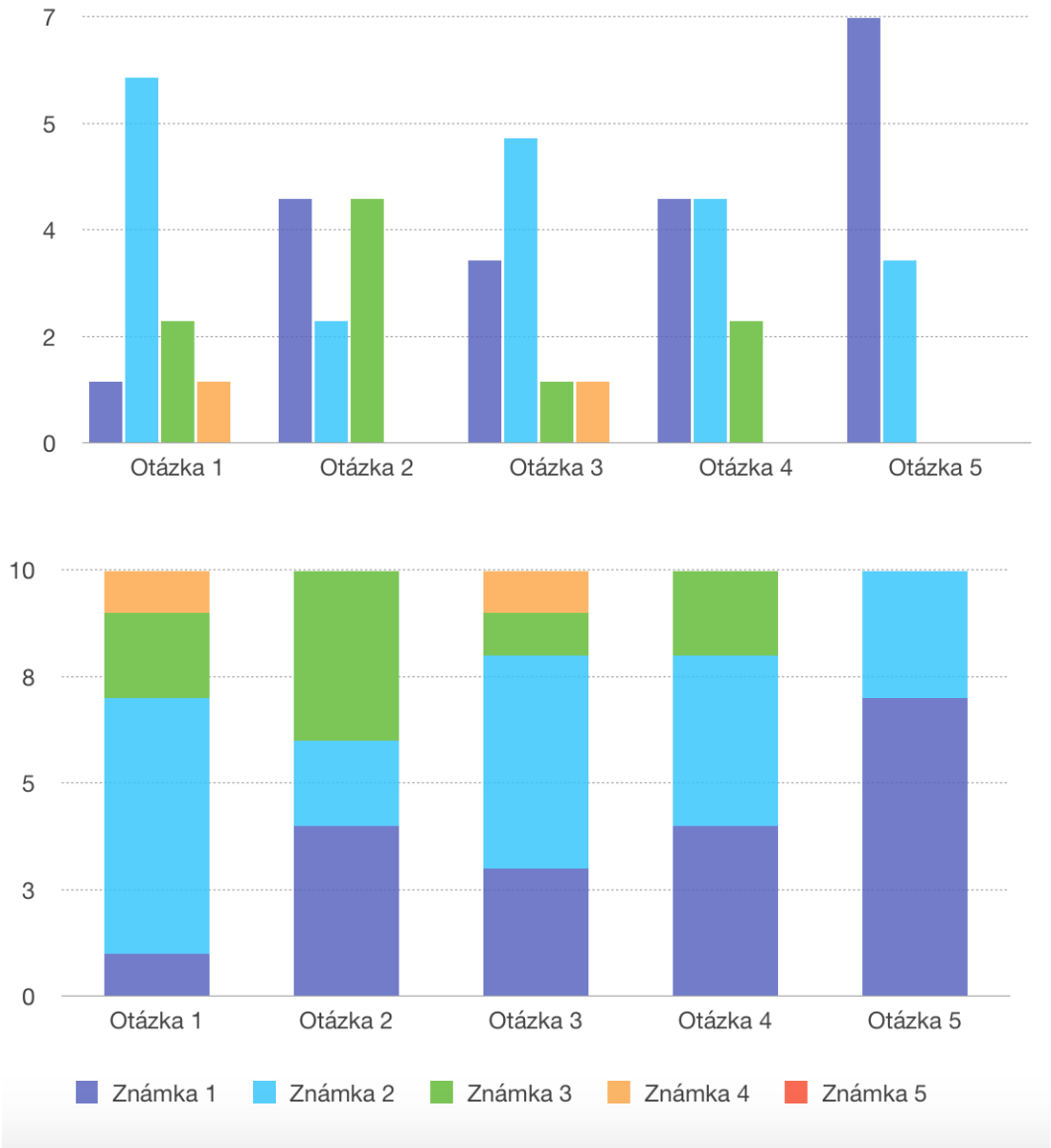
Všem uživatelům jsem napřed vysvětlil princip kalibrátoru a kroky, které budou muset pro správnou kalibraci udělat. S postupem po vysvětlení neměli uživatelé problém větší problém projít kalibračním procesem, i když jim občas dělalo problémy zapamatovat si, která barva snímaného objektu má jaký význam.

Na základě uživatelských testování jsem zpomalil odpočítávání v kalibrátoru. Většina uživatelů kalibraci nestíhala a úspěšné zvládla kalibraci až na několikáté opakování. Po úpravách kalibrátoru hodnotili uživatelé obtížnost kalibrace průměrnou známkou 2.

5.5 Testovací hry

Při testování hry 4kanoid s uživateli jsem pozoroval velké odchylky v obtížnosti pro pravidelné hráče a nehráče. Po několika experimentech jsem našel ideální rychlost míčku tak, aby byla hra zábavná pro obě skupiny a zbytečně nefrustrovala nezkušené hráče.

Další úpravou během testování s uživateli bylo omezení pohybu pálek do spodní třetiny obrazu. Spousta hráčů pohybovala pálkami příliš vysoko a tak si velmi usnadnili hraní.



Obrázek 5.6: výsledky dotazníků od testovacích uživatelů

Stejně tak u hry LaserPopper bylo potřeba upravit obtížnost. První verze zvyšovaly obtížnost příliš rychle a někteří hráči prohrávali během krátké doby hraní.

Díky vhodně zvoleným hrám a způsobu ovládání hodnotili uživatelé přesnost ovládání průměrnou známkou 2. Ovládání formou natáhnuté dlaně s roztaženými prsty bez prudkých pohubů a pohybů skrývajících část dlaně zvládá Leap Motion evidentně velmi dobře.

Protože jsem si dal mimo jiné za cíl vymyslet i co nejoriginálnější hry a způsob jejich ovládání, ptal jsem se uživatelů také na to, jak jim připadají hry originální v návaznosti na způsob ovládání. K mému potěšení hodnotila originalitu většina uživatelů známkou 1.

Dalším cílem hned po přesnosti a originalitě her bylo, aby se hry ovládaly co nejpohodlněji a nejpřirozeněji, což je cílem každého herního ovladače nebo rozhraní. I tady uživatelé hodnotili lepší dvojkou.

Z výsledků uživatelského testování vyvozuji, že hlavní cíl vytvořit pohodlné a přesné

herní rozhraní pro dataprojektor a Leap Motion jsem splnil. Stejně tak mnou vytyčené cíle na originální hry, které budou rozhraní na plno využívata budou hráče bavit.

Jako zajímavou situaci z uživatelského testování bych chtěl vypíchnout problém Leap Motion snímat drobnou ženskou ruku s dlouhými umělými nehty. Testování se zúčastnila slečna těsně po návštěvě nehtového studia a Leap Motion často přepínal mezi levou a pravou rukou, ač slečna pro ovládání používala jen jednu z nich. Jinak nebyla přesnost snímání ovlivněna.

5.6 Možná vylepšení

5.6.1 Rozhraní Leap Motion a dataprojektoru

U samotného rozhraní by se dalo dlouze experimentovat s výběrem správného materiálu sloužícího jako promítací plocha. Ideální by byl materiál propouštějící světlo pouze z jedné strany. Tak by se zvýšila kvalita promítaného obrazu a snížily odlesky infračerveného světla vycházejícího ze snímače, což bude mít za následek zvýšení přesnosti snímání pohybu.

Místo pro vylepšení se také nabízí v instalaci Leap Motion do papírové krabice. Vhodnější by bylo umístit jej do krabice na pevno do ideální výšky tak, aby se s ním nedalo pohnout a zabíral co největší plochu.

5.6.2 Kalibrátor

Základní vylepšení kalibrátoru, které by šlo do budoucna implementovat, je rozšířit kalibraci o třetí rozměr. Určitě by bylo zajímavé vyzkoušet, zda by se dala vytvořit ovladatelná trojrozměrná hra, která by obraz přesně mapovala na dlaň hráče.

Vhodné by bylo implementovat variantu dotykového ovládání například pro tvorbu menu her. Zajímavé by jej bylo implementovat pomocí fyzikální knihovny, kdy by mezi sebou interagovaly ovládací objekty namapované na uživatelovy prsty a tlačítka.

Další vylepšení, které se nabízí, je udělat z kalibrátoru komponentu pro Unity Asset Store tak, aby uživatel do projektu pouze naimportoval balíček s kalibrátorem a mohl jej využívat.

5.6.3 Testovací hry

Protože jsou vytvořené hry určeny pro testování rozhraní a vytvořeny jen jako prototyp, je u nich spousta prostoru k vylepšování. U hry 4kanoid by se dalo vytvořit více levelů, přidat bonusy rozšiřující pálky, zvětšující počet míčků, zpomalující míček a podobně, stejně jako v klasickém Arkanoidu. Ve hře LaserPopper by bylo možno přidat více druhů míčků, více druhů laserů a stejně tak různé bonusy oživující hraní podobně jako ve hře 4kanoid.

Kapitola 6

Závěr

Cílem práce bylo vytvořit herní rozhraní pro ovládání počítačových her s využitím pohybového snímače Leap Motion a obrazu promítaného dataprojektorem. Na začátku práce je čtenář seznámen s pojmem mapování obrazu a ukázkami různých technických řešení se zaměřením na interaktivní mapování na pohybující se objekty, především pak lidské tělo.

Rozhraní bylo navrhováno s důrazem na přesnost ovládání a pohodlí uživatele. Při návrhu jsem objevoval omezení a slabá místa snímače Leap Motion a na základě toho pak vznikla instalace dataprojektoru s Leap Motion balancující mezi co největší eliminací nedostatků a nepřesnosti snímače a udržením kvality promítaného obrazu umístěním Leap Motion pod průhlednou promítací plochu.

Pro snadnou poloautomatickou kalibraci dataprojektoru a snímače Leap Motion byla vytvořena kalibrační scéna pro herní engine Unity, pomocí které uživatel obě zařízení snadno zkalibruje během několika sekund. S kalibračními daty pak pracují testovací hry vytvořené z důvodu ověření funkčnosti rozhraní a uživatelského testování.

Ač software Leap Motion stále vykazuje značné nedostatky ve snímání, přesnost rozhraní dosahuje ochylky 0 - 5 mm s občasnými extrémy. Testovacím programem jsem ověřil, že mnou vytvořená instalace dataprojektoru a Leap Motion výrazně neovlivňuje přesnost snímání. Uživatelské testování pak potvrdilo, že se testovací hry přes vytvořené herní rozhraní velmi přesně ovládají.

V budoucnu bych chtěl Leap Motion znovu vyzkoušet, protože se snímací software neustále vyvíjí a poslední verze vykazují značně vyšší přesnost než verze, se kterou jsem pracoval při návrhu rozhraní. Pokud bude přesnost dostačující, bylo by zajímavé vyzkoušet 3D kalibraci a vytvoření 3D hry. Také by se dal z kalibrátoru vytvořit balíček pro Unity, který by se snadno naimportoval do projektu.

Cíle práce byly splněny. Vytvořil jsem herní rozhraní i demonstrační hry a otestoval přesnost jak programově, tak uživatelským testováním. Vytvořené herní rozhraní je dostatečně přesné pro hraní jednoduchých her a dá se použít univerzálně. Navíc se dá snadno přenášet a instalace i s kalibrací je otázka několika minut. V porovnání s dotykovými displeji nabízí větší variabilitu využití díky snímání celé dlaně. Oproti snámání klasickou kamerou pak nabízí snadnější a rychlejší kalibraci.

Literatura

[1] Finger - Leap Motion Unity SDK v2.2 documentation. [online] https://developer.leapmotion.com/documentation/unity/api/Leap.Finger.html#csharpclass_leap_1_1_pointable_1aa51b84d63c25aeb0fc15554f0f61e4d5.

[2] Intro Skeleton API. [online] https://developer.leapmotion.com/documentation/csharp/devguide/Intro_Skeleton_API.html, 2014, [cit. 2014-08-09].

[3] Leap Motion API overview. [online] https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html, 2014, [cit. 2014-08-09].

[4] Leap Motion API reference. [online] https://developer.leapmotion.com/documentation/csharp/api/Leap_Classes.html, 2014, [cit. 2014-11-09].

[5] Leap Motion Support : Modes of Operation. [online] https://support.leapmotion.com/entries/39433157-Modes-of-Operation, 2014, [cit. 2014-11-09].

[6] INTRODUCING THE LEAP MOTION PLUGIN FOR UE4. [online] https://www.unrealengine.com/blog/introducing-the-leap-motion-plugin-for-ue4, 2015-02-24 [cit. 2015-04-20].

[7] C# Programming Guide. [online] https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx, 2015, [cit. 2015-04-29].

[8] JavaScript — MDN. [online] https://developer.mozilla.org/en-US/docs/Web/JavaScript, 2015, [cit. 2015-04-29].

[9] Kinect for Windows Sensor Components and Specifications. [online] https://msdn.microsoft.com/en-us/library/jj131033.aspx, 2015, [cit. 2015-04-29].

[10] THE LEADING GLOBAL GAME INDUSTRY SOFTWARE. [online] https://unity3d.com/public-relations, 2015, [cit. 2015-05-07].

[11] Unity - Get Unity. [online] http://unity3d.com/get-unity, 2015, [cit. 2015-05-07].

[12] Unity - Multiplatform - Publish your game to over 10 platforms. [online] http://unity3d.com/unity/multiplatform, 2015, [cit. 2015-05-07].

[13] Unity editor. [online] http://unity3d.com/unity/editor, 2015, [cit. 2015-05-07].

[14] Unreal Engine 4. [online] https://www.unrealengine.com/unreal-engine-4, 2015, [cit. 2015-05-07].

[15] Leap SDK Release Notes. [online] https://developer.leapmotion.com/documentation/Leap_SDK_Release_Notes.html#version-2-0-0-beta, 2015, [cit. 2015-05-12].

[16] Buckwald, M.: V2 Software Now Available for Everyone. [online] http://blog.leapmotion.com/v2-software-now-available-everyone/, 2014-10-26 [cit. 2015-04-12].

[17] Colgan, A.: How Does the Leap Motion Controller Work? [online] http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/, 2014-08-09 [cit. 2014-11-09].

[18] Jay, R.: The Sydney Opera House during the 2013 Vivid Sydney projection display. 2013.
URL http://en.wikipedia.org/wiki/Projection_mapping#/media/File:Vivid_Sydney_-_Opera_House_sails_(9002375891).jpg

[19] Jones, B.; Sodhi, R.; Murdock, M.; aj.: RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-3069-5, s. 637–644, doi:10.1145/2642918.2647383.
URL http://doi.acm.org/10.1145/2642918.2647383

[20] Kogan, G.: Kinect Projector Toolkit. [online] http://www.genekogan.com/works/kinect-projector-toolkit.html, 2015, [cit. 2015-04-29].

[21] MacCormick, J.; Isard, M.: Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking. In *Computer Vision — ECCV 2000*, editace D. Vernon, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2000, ISBN 978-3-540-67686-7, s. 3–19, doi:10.1007/3-540-45053-X˙1.
URL http://dx.doi.org/10.1007/3-540-45053-X_1

[22] Moreno, D.; Taubin, G.: Simple, Accurate, and Robust Projector-Camera Calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, Oct 2012, s. 464–471, doi:10.1109/3DIMPVT.2012.77.

Příloha A

Obsah CD

- soubor `4kanoid.app` pro spuštění hry 4kanoid na OS X
- adresář `4kanoidWin` obsahující soubory pro spuštění hry 4kanoid na OS Windows
- soubor `LaserPopper.app` pro spuštění hry LaserPopper na OS X
- adresář `LaserPopperWin` obsahující soubory pro spuštění hry LaserPopper na OS Windows
- adresář `src` obsahující zdrojové kódy projektu pro Unity
- soubor `thesis.pdf` obsahující text práce