FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Univerzální systém gest pro virtuální realitu
Student:	Bc. Vojtěch Harašta
Vedoucí:	Ing. Petr Pauš, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je návrh a impementace řešení pro rozšíření možností klasického ovládání aplikací ve virtuální realitě (VR) zavedením univerzálního systému gest pro vybraný editor VR. Uživatel by měl mít možnost gesta jednoduše vytvářet, přidávat a napojovat na funkce aplikace. Výsledek práce poskytne odpověď na malý a různorodý počet tlačítek u headsetů a stejně tak možnost lehce přidat funkcionalitu gest do libovolného VR projektu vybraného editoru.

Cíle práce:
1) Analyzovat současné editory pro VR a možnosti dostupných komerčních headsetů.
2) Analyzovat systémy využívající gesta i mimo VR.
3) Navrhnout systém gest pro VR a jeho testování dle analýzy.
4) Implementovat navržený systém gest ve vybraném editoru VR.
5) Implementovat rozhraní testující funkčnost systému po každé změně skladby gest.
6) Vytvořit demo pro využití systému gest.
7) Zhodnotit náklady ušetřené díky využití výsledku práce oproti samostatné impementaci funkcionality gest.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 23. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Univerzální systém gest pro Virtuální Realitu

Bc. Vojtěch Harašta

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petr Pauš, Ph.D.

7. ledna 2021

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Petru Paušovi, Ph.D. za konzultace a cenné rady při realizaci této diplomové práce a dále testerům za jejich ochotu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelum. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 7. ledna 2021 . . . . . .

Odkaz na tuto práci

Harašta, Vojtěch. *Univerzální systém gest pro Virtuální Realitu.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato diplomová práce se zabývá tvorbou alternativního řešení pro ovládání VR aplikací, a to pomocí zavedení univerzálního systému gest. Tento systém gest je oproštěn od fyzických prvků pro ovládání, jako jsou tlačítka na ovladači, takže je použitelný na většině VR hardware, ať už se jedná o mobilní, konzolové či počítačové VR. Umožňuje vývojářům tvořit libovolná gesta pomocí VR hardware se sledováním pohybu a napojit je na funkce aplikace, ať už je gesto tvořeno jednou rukou nebo třeba všemi pěti prsty. V práci je nejdříve provedena analýza současných VR produktů a systémů využívajících gesta. Na základě výsledků analýzy je navrhnut a implementován systém gest, který umožňuje přidat funkcionalitu gest do libovolného projektu v Unreal Engine a zároveň je použitelný na většině VR produktech. Díky tomu vývojář nemusí nastavovat aplikaci pro každý produkt zvlášť.

Klíčová slova Virtuální realita, Gesta, Univerzální ovládání, Unreal Engine, Vstup

Abstract

This master's thesis is focused on creating an alternative to traditional input handling in VR applications, by using a universal gesture system. The gesture system is not using any physical inputs, such as buttons on a controller, so it can be used on basically any existing VR hardware, be it mobile, console or PC VR. The system enables developers to create their own gestures, using VR hardware with motion tracking, and connect that gesture to a function from the application, be it gesture made with just one hand or, for example, all five fingers. The thesis contains research of current VR products and other systems using gestures, and based on this research, the gesture system is designed and implemented. The gesture system adds gesture functionality to any Unreal Engine project and makes that application usable on most VR products, so that the developer doesn't have to worry about making product-specific changes to the application.

Keywords Virtual reality, Gestures, Universal control, Unreal Engine, Input

Obsah

Seznam obrázků

Úvod

Virtuální realita se stává každým dnem populárnější, jak například vykazují data platformy Steam [50], která hlásí 0.20 % nárůst vlastníků VR hardware z řad uživatelů oproti předchozímu měsíci. Mimo platformy Steam se dá s VR setkat v dalších oblastech, jako jsou mobilní telefony či konzole, které se v této statistice ani neprojeví, ale i zde roste zájem. Pokud tento trend, tedy růst uživatelů VR, zejména díky snižující se vstupní ceně a vyššímu počtu aplikací využívající VR, bude pokračovat, je možné očekávat, že se s VR začne více a více lidí setkávat na denní bázi. Je tedy vhodné se tímto tématem zabývat.

S virtuální realitou je možné se setkat v nejrůznějších podobách od řešení pro mobilní telefony přes VR obsahující vlastní procesor až po VR, které potřebuje připojený počítač pro výpočty. Také v jednotlivých oblastech se VR hardware velmi liší, ať už použitými ovládacími prvky, rozlišením obrazovky či způsobem sledování pohybu. Jeden VR set obsahuje například dva párové ovladače, druhý jen jeden ovladač ve formě zbraně s jinými tlačítky a třetí může mít pro ovládání celého VR pouze jedno tlačítko.

A právě zde nastává problém pro tvůrce VR aplikací, a to ten, že různé VR hardware mají různé možnosti a často je nutné pro jednotlivé VR produkty tvořit specifická mapování akcí kvůli odlišným možnostem daného VR produktu. Aplikace pro VR pak často bývají pouze pro specifické VR produkty nebo jejich údržba zahrnuje kontrolu kompatibility s obrovským množstvím rozdílného VR hardware, což je nepraktické a časově náročné.

Tento problém se bude tato diplomová práce snažit vyřešit tak, že poskytne alternativu ke klasickému ovládání aplikace pomocí mapování tlačítek na akce, a to zavedením univerzálního systému gest, jehož cílem bude co nejvíce oprostit ovládání VR aplikace od fyzických prvků, jako jsou tlačítka či touchpady, a místo toho umožní reakce aplikace vyvolávat gesty, kterými bude možné aplikaci plnohodnotně ovládat.

V první části práce tak dojde k analýze současných VR produktů a jiných systémů využívajících gesta. Při analýze VR produktů se budou hledat

zejména ovládací prvky, které se nachází na ideálně všech analyzovaných produktech, aby sloužila jako základ pro systém gest a zároveň nijak nesnížila univerzálnost budoucího systému. Cílem analýzy systémů využívajících gesta je najít doporučení a best practices, která se využijí při tvorbě gest pro tento systém.

Ze zjištěných informací z analýzy dojde k návrhu systému gest tak, aby byl co nejuniverzálnější, tedy aby jednak byl použitelný na ideálně všech VR produktech, a dále aby jeho užití samotné bylo co nejflexibilnější. Navržený systém by tedy měl využívat pouze ty prvky, které se budou nacházet na všech analyzovaných VR produktech, a jeho implementace ve vybraném editoru by měla být lehce ovladatelná a přenositelná.

Nápad vytvořit tento systém vznikl, když autor práce nalézal v populárních aplikacích pro VR rozlišné mapování ovládání, což vedlo ke zbytečnému matení autora i dalších uživatelů aplikace. Autor práce sám již pár VR aplikací vyvíjel. Při mapování tlačítek na funkce aplikace narážel na problém, že i jemu dostupné VR produkty měly rozlišná tlačítka, a tedy bylo nutné řešit mapování pro jednotlivé produkty zvlášť.

Výsledný systém gest by tak poskytnul alternativní řešení pro ovládání VR aplikací, které by nebylo závislé na použitém hardware, a tak by aplikace využívající tento systém bylo možné spustit na nejrůznějších VR produktech, nehledě na jejich fyzické zpracování. Vývojářům aplikací by tak systém gest ušetřil nejen čas s implementací vlastního řešení ovládání aplikace, ale i čas strávený následnou údržbou aplikace, a zároveň by mu zajistil, že aplikace bude spustitelná na většině VR produktech.

Kapitola 1

Cíl práce

Cílem práce je navrhnout a vytvořit univerzální systém gest pro VR, který poskytne alternativní řešení pro ovládání aplikace ve VR ke klasickému mapování tlačítek na funkce aplikace, a to tak, že nabídne možnost aplikaci ovládat pomocí gest. Tato gesta bude možné libovolně tvořit, mazat a napojovat na ně funkce aplikace, a navíc bude možné stejná gesta dělat na různých VR hardware. Systém gest by měl být, kromě univerzálnosti, co nejflexibilnější, aby ho mohli uživatelé do svých VR aplikací zakomponovat dle svých potřeb.

Práce zahrnuje analýzu současných produktů pro VR, jako je třeba HTC Vive nebo Oculus Rift, a jejím cílem je najít ovládací prvky, které se nachází na všech analyzovaných produktech, a je tedy možné je využít pro systém gest, pakliže má být co nejuniverzálnější. Budou se tedy porovnávat prvky, jako jsou tlačítka, touchpady, joysticky a dále sledované prvky, tedy ty prvky, jejichž pohyb daný VR produkt sleduje a pomocí kterých by bylo možné gesta tvořit. Součástí analýzy je i analýza editorů pro VR, primárně které VR produkty podporují, a současných systémů využívajících gesta, jejíž cílem je stanovit požadavky a najít best practices pro tvorbu gest samotných.

Po analýze bude následovat kapitola Návrh, jejímž cílem bude navrhnout systém gest na základě informací zjištěných při analýze tak, aby využíval pouze ty ovládací prvky, na jejichž existenci se dá spolehnout na všech zkoumaných VR produktech. Pro lepší kontrolu kvality systému budou v této kapitole stanoveny funkční i nefunkční požadavky, vycházející ze zadání a výsledků analýzy, pomocí kterých bude snazší hlídat splnění požadovaných vlastností systému. Dále dojde k návrhu určitých částí systému, formátu dat a použitého algoritmu pro klasifikaci gest. V poslední řadě dojde k tvorbě konceptuálního modelu v UML a jeho popisu, který bude využit pro následnou implementaci.

Cílem kapitoly Realizace bude implementovat systém gest dle návrhu a ke každému z požadavků stanovených v předešlé kapitole bude napsáno, jak ho bylo při implementaci dosaženo. Dále dojde k představení využitých funkcí a prvků zvoleného editoru a také implementace algoritmu pro klasifikaci. Na konci kapitoly bude zobrazen UML, který bude přesně reflektovat

implementovaný systém.

Cílem testování bude ověřit, že je možné lehce následovat uvedený návod k instalaci, a tedy že je systém gest možné přidat do libovolného projektu vybraného editoru bez větších potíží. Dále dojde k otestování systému gest na lidech jiných, než je autor sám, a ze zjištěných informací, jako je lehkost užití a čas pro instalaci, budou vyvozeny závěry pro potenciální uživatele systému, jež kromě jednotlivců mohou být i podniky.

Kapitola 2

Analýza

V této kapitole je provedena analýza enginů pro tvorbu aplikací ve virtuální realitě, současné nabídky zařízení pro simulování virtuální reality a jejich možnosti vzhledem k podpoře systému gest, systémů, které využívají nebo využívali gesta a různých přístupů a algoritmů pro detekci gest.

2.1 VR Engine

Tato sekce se zabývá analýzou vhodných enginů pro tvorbu aplikací ve virtuální realitě, konkrétně porovnání v současné době nejpoužívanějších komerčních enginů Unity, Unreal Engine a CryEngine a zběžné popsání dalších, méně používaných enginů. Dále v této části práce proběhne výběr jednoho z enginů pro praktickou část této práce s odůvodněním, které bude založeno na zhodnocení následujících kritérií:

- **Podporované VR platformy**: Jelikož má být výsledek této práce co nejuniverzálnější, nejdůležitějším kritériem je počet podporovaných VR platforem.
- **Výkon**: Pro aplikace pro VR je velmi důležité, aby fungovaly plně plynule a pokud možno neměly žádné záseky, protože ty mohou vést k nevolnosti uživatele.
- **Programovací jazyk**: Je nutné vzít v potaz možnosti dostupných programovacích jazyků vůči optimalizaci, podpoře a také znalosti jazyka autorem práce.
- **Platební plán**: Vybraný engine musí umožňovat tvorbu zdarma pro účely této práce.
- **Assety**: Ačkoliv pro účely této práce externí assety nejspíše nebudou potřeba, je dobré porovnat možnosti jednotlivých enginů.

- **Dokumentace a kurzy**: Je vhodné, aby měl vybraný engine detailní dokumentaci a dostupné kurzy.

- **Komunita**: Vybraný engine by měl mít co největší komunitu pro případné hledání pomoci.

2.1.1 Unity

Herní engine Unity byl vydán v roce 2005, ale v té době složil pro tvorbu pouze pro OS X a až později získal podporu pro Windows a webové prohlížeče. Jedním z hlavních cílů Unity bylo tzv. demokratizovat vývoj her a usnadnit tvorbu pro co nejvíce lidí. Tuto myšlenku si Unity uchovává do dneška a je naprosto zdarma pro osobní i komerční užití, pokud subjekt nevydělá více než 100 000 $ ročně, pro ty Unity nabízí placené programy. [10]

Kromě tvorby 2D a 3D aplikací a vytváření scén Unity nabízí svoje servery pro multiplayerové účely zdarma skrz platformu nazvanou UNet, ale ta v současné době probíhá změnami a stává se obsolete (zastaralá). Unity momentálně připravuje náhradní službu pro multiplayerové služby, která je plánovaná být plně nasazena na jaře 2021. Kompletní odstavení služby UNet je plánováno na rok 2022 [52]. Na oficiálním fóru Unity v je současné době 140 000+ příspěvků na téma skriptování a 7 000+ příspěvků na téma VR/AR tvorba. Unity Asset Store se může chlubit 64 000+ položkami od uživatelů, z čehož alespoň 11 000 je ohodnoceno pěti hvězdami z pěti. Díky své dlouhé existenci má propracovanou dokumentaci, která je uživateli velmi chválená díky svému obsahu a lehkou navigací včetně video tutoriálů. Na Udemy, tedy jedné z nejznámějších stránek pro online kurzy, má přes 6 600 dostupných kurzů.

Ačkoliv se pro veškeré programování v Unity používá C#, engine samotný je napsaný v C++, ale možnost v něm programovat není. Vyjma C# Unity podporovalo Boo a UnityScript (verzi JavaScriptu), ale od roku 2017 Unity používá výhradně C# [20]. Nově od roku 2020 Unity podporuje vizuální skriptovací jazyk Bolt (dále jen Bolt), jenž získalo od výrobce Ludiq. Bolt je obsažen ve všech platebních plánech Unity, tedy i toho zdarma, a měl být dočasnou odpovědí na již dlouho fungující systém Blueprints v Unreal Engine (více v sekci 2.1.3) do té doby, než Unity dokončí tvorbu vlastního systému pro vizuální skriptování.[9] Plánovaný na rok 2021 byl Bolt 2, ale místo toho se tvůrci Unity rozhodli Bolt 1 dát přímo do jádra Unity a vylepšovat jej v dalších letech. Od roku 2021 by tak uživatelé Unity měli očekávat ne jen podporu, ale přímou integraci Boltu do jádra enginu a jeho následné vylepšování. [25]

Unity kromě běžných platforem jako je Windows, Linux a macOS podporuje i iOS, Android, tvOS, WebGl, Playstation 4 a Xbox One, tedy platformy, na kterých se můžeme potkat s virtuální realitou (vyjma tvOS a Xbox One). Unity přímo podporuje headsety od Facebooku a Microsoftu, tedy pro sérii Oculus a Windows Mixed Reality stačí načíst adekvátní plugin od Unity. Pod-

poru pro headsety jiných tvůrců Unity samozřejmě má, ale je třeba stáhnout jejich specifické pluginy. Například pro HTC Vive a ostatní headsety využívající SteamVR firma Valve udržuje plugin s názvem OpenVR Unity XR Plugin. Aktivaci správného pluginu dle cílové platformy uživatel nemusí dělat manuálně, ale místo toho může využít funkcionalitu Unity s názvem XR. [24]

2.1.2 CryEngine

Engine německého původu byl vydán v roce 2002 firmou Crytek a jeho první využití bylo na hru Far Cry, která byla původně vytvořena jako demo pro firmu Nvidia, ale díky svému potenciálu z ní vznikla plnohodnotná hra. Od té doby byl engine použit na další známe tituly od firmy Crytek, ale samozřejmě i od jiných developerů, jako je například české studio Warhorse Studios s.r.o. a jejich hru Kindom Come: Deliverance. V roce 2015 Amazon koupil k enginu licenci za 50–70 miliónů dolarů a v následujícím roce Amazon publikoval svoje rozšíření CryEngine pod názvem Amazon Lumberyard. [49]

CryEngine kromě Windows a Linux podporuje Playstation 4, Xbox One, Oculus Rift, OSVR a HTC Vive. Tedy nejen že nepodporuje velkou část VR headsetů, zatím nemá ani možnost pro mobilní development, na kterém se ovšem v současné době pracuje. Od verze 5.6 Cryengine nepodporuje 32bitové systémy [3]. Narozdíl od ostatních zmíněných enginů umožňuje vývoj v jazycích C#, C++ a Lua, ve kterých je sám napsán, a navíc má podporu pro dva vizuální skriptovací jazyky, a to Flowgraphs a Schematyc [12]. Na oficiálním fóru se nyní nachází přes 600 příspěvků na téma C#/C++ programování a 1 200+ příspěvků, týkajících se obecných otázek na engine, z čehož pouze 36 je z kategorie VR/AR. Na CryEngine Marketplace je možné najít přes 1400 položek k užití, z čehož 135 je zdarma. K enginu existuje propracovaná dokumentace včetně tutoriálů, které ale nejsou formou videa. Na Udemy je k nalezení ovšem pouze 12 kurzů.

I v dnešní době je možné se setkat s informací, že CryEngine je pouze placený engine, a ačkoliv je pravda, že tomu tak dříve bylo, v roce 2016 přešel na formu platby typu „zaplať, kolik chceš“ za použití a přístup ke zdrojovému kódu enginu [13]. Tento model již však také není platný, jelikož v roce 2018 přešel engine po vzoru Unreal Engine na model „zaplať 5 % výdělku“ a to pouze pokud subjekt na svém produktu vydělá více než 5 000 $ za rok. Všechny projekty vyvinuté v CryEngine musí mít po spuštění viditelné jméno, logo a animované logo enginu. [11]

2.1.3 Unreal Engine

S nejstarším enginem ze zatím zmíněných se mohla veřejnost poprvé setkat v roce 1998, kdy debutoval se hrou Unreal. Od té doby engine prošel již čtyřmi iteracemi, kde současná verze, tedy Unreal Engine 4, vyšla roku 2014 s novým modelem placení, který byl založen na předplatném, a dále existovala verze

zdarma pro akademickou sféru. Tento model se ale dlouho neudržel a v roce 2015 přešel na model „zaplať 5 % výdělku“, a to pouze až čtvrtletní výdělek produktu překročí 3 000 $ [56]. Tohoto modelu se Epic Games, vlastník Unreal Engine, drží dodnes, ale podmínku placení změnil od počátku roku 2020 tak, že 5 % je třeba platit, až co celkový zisk za produkt vytvořený v Unreal Engine překročí 1 000 000 $. Pro jakékoliv nekomerční užití je Unreal Engine kompletně zdarma.[53]

Unreal Engine podporuje zatím nejširší spektrum platforem, a to Windows, Linux, MacOS, SteamOS, AR, VR, iOS, Android, HTML5, Google Stadia, Playstation 4 a 5, Xbox One a Series X a Nintendo Switch. Všechny ze zmíněných platforem jsou podporovány nativně enginem, akorát u konzolí je nutné se nejdříve zaregistrovat jako vývojář u adekvátních firem. Unreal Engine tedy nabízí podporu všech VR a AR headsetů a od verze 4.25 je i připraven na konzole další generace. [53]

Samotný engine je napsaný v C++ a jeho zdrojový kód je plně dostupný. V předešlých verzích se v Unreal Engine programovalo pomocí UnrealScript, což byl jazyk vytvořen pro jednoduché programování vyššího stupně a byl objektově orientovaný. Kromě UnrealScript byla možnost lehčích úkonů dosáhnout ve vizuálním skriptovacím systému Kismet. Od čtvrté verze byl UnrealScript odstraněn a veškerá tvorba v enginu přešla na C++ a k němu byl zaveden vizuální skriptovací systém Blueprints (dále jen Blueprints), což je nadstavba nad C++ funkcemi a umožňuje i vizuální debuggování [21]. Projekty v Unreal Engine je možné tvořit i čistě za pomocí Blueprints bez větších ztrát na výkonu, protože je to jen nadstavba nad existujícími C++ funkcemi, ale složitější úkony se v nich stávají dosti nepřehledné. Kromě možnosti programovat projekt čistě v C++ není problém obě možnosti spojit a do Blueprints zapojovat vlastní funkce napsané v C++ [54].

Na oficiálním fóru Unreal Engine se momentálně nachází 14 000+ příspěvků na téma C++ vývoje, 28 000+ příspěvků na téma Blueprints a 5 000+ příspěvků na téma VR/AR vývoje. Na Unreal Engine Marketplace je možné najít přes 14 000 položek od uživatelů, ale také od Epic Games, jenž na Unreal Engine Marketplace dává assety k projektům, které nadále nepodporuje, což jsou často velmi kvalitní a propracované assety zcela zdarma. Za svou dlouho existenci k enginu vznikla kvalitní dokumentace, včetně tutoriálů, které zahrnují i online streamy na platformě Twitch.tv, a ukázkových projektů, ovšem některé projekty již nejde zkompilovat na novějších verzí. Na Udemy je možné k Unreal Engine najít přes 2 900 dostupných kurzů.

Pátá verze enginu se nyní připravuje na svoje vydání, které je naplánované na začátek roku 2021, konkrétně verze Preview, která se dá považovat za veřejnou betu. Oficiální vydání je plánováno na druhou půlku roku 2021. Nová verze nepřinese žádné další změny na platební metodě, takže stále bude nutno za Unreal Engine platit pouze až při úspěchu projektu. Verze je zpětně kompatibilní a bude možné do ní přenést projekty ze čtvrté verze. K nové verzi společnost Epic Games představila ještě své vlastní služby pro multipla-

yer, které zdokonalila díky hře Fortnite, a nabízí je už nyní zdarma k využití včetně podpory cross-play mezi platformami Playstation, Xbox, Switch, PC, Mac, iOS i Andriod. Služba nenabízí jenom matchmaking (vytvoření spojení mezi hráči pro online hraní), ale i služby přátel, ocenění, žebříčků a správu účtů. [1]

2.1.4 Další možnosti

V této části budou zběžně představeny tři další enginy jakožto zajímavé alternativy, ale nebudou brány v potaz při výběru vhodného enginu pro tuto práci. Jedná se o enginy Godot, AppGameKit a Amazon Lumberyard:

- **Godot** je plně open-source engine za který se nic neplatí a spadá pod MIT licenci. Je psaný v C/C++, ale programovat se v něm dá v řadě jiných jazyků, jako je například C#, Rust či D a dále nabízí svůj vlastní skriptovací jazyk GDScript, který je podobný Pythonu. Architektura enginu je tvořena okolo konceptu stromu a jeho listů, kde listy jsou jednotlivé prvky scény. Godot podporuje cross-platformní tvorbu pro platformy Windows, macOS, Linux, Android, iOS, BDS, HTML5, WebAssembly i konzole po registraci u adekvátních firem. Má podporu pro tvorbu ve VR díky modulům OpenVR, OpenXR a Oculus SDK, které se dají stáhnout z jeho knihovny assetů. [26]

- **AppGameKit** je engine, který je schován za jednorázovou platbou v rámci pár desítek euro a dá se koupit například na platformě Steam, do kterého se dál dají kupovat různá rozšíření, jako je například AppGameKit VR, který přidá podporu pro vybrané VR headsety. Podporových platforem je ovšem méně než například u zmíněného Godotu, a to konkrétně Windows, Linux, Andriod, iOs a Raspberry Pi. Vývoj v tomto enginu probíhá za využití skriptovacího jazyku enginu AppGameKit Script nebo dostupných C++ knihoven od tvůrců enginu. [2]

- **Amazon Lumberyard**, jehož vznik je popsán v sekci 2.1.2, oproti CryEngine či i ostatním zde zmíněným zavádí zajímavý platební plán. Engine jako takový a produkty v něm vytvořené je možné prodávat bez jakékoliv nutnosti platit procenta nebo podobné, ale místo toho se platí za služby Amazonu, které produkt v něm vytvořený využívá, jako jsou například multiplayerové servery. Podporované platformy jsou PC, Xbox One, PlayStation 4, iOS či Android a na podpoře MacOS se nyní pracuje. Co se VR headsetů týče, Amazon Lumberyard podporuje Oculus Rift, HTC Vive a PlayStation VR, ale uvádí, že podpora VR je stavěna modulárně a není problém přidat nový modul s podporou pro nové headsety. Jelikož vychází z CryEngine, je stejně tak napsaný v C++ a Lua. Ze všech představených enginů je oficiálně nejmladší, protože v tomto

okamžiku je stále v betě, ale je založen na již dlouho existujícím CryEngine. [7]

2.1.5 Výběr enginu

Před výběrem samotným je vhodné zobrazit zjištěné informace pro daná kritéria u jednotlivých enginů v přehledné a krátké formě:

- **Podporované VR platformy**
  - Unity: Nativně Oculus a WMR, pomocí pluginů ostatní
  - CryEngine: PSVR, OSVR, Oculus a HTC Vive
  - UnrealEngine: Nativní podpora VR
- **Výkon** (1-10 autorovo hodnocení, velmi záleží na programátorovi)
  - Unity: 8 (Nebývá první volbou při tvorbě náročných titulů)
  - CryEngine: 9 (Tituly jsou často chválené díky skvělé optimalizaci)
  - UnrealEngine: 9 (Častá volba větších herních studií pro AAA tituly)
- **Programovací jazyk**
  - Unity: C#, Bolt
  - CryEngine: C#, C++, Lua
  - UnrealEngine: C++, Blueprints
- **Platební plán**
  - Unity: Do výnosu 100 000 $ ročně zdarma, poté předplatné
  - CryEngine: Do výnosu 5 000 $ ročně zdarma, poté 5 % z výdělku
  - UnrealEngine: Do výnosu 1 000 000 $ zdarma, poté 5 % z výdělku
- **Assety**
  - Unity: Přes 64 000 položek na Unity Asset Store
  - CryEngine: Přes 14 000 položek na Unreal Engine Marketplace
  - UnrealEngine: Přes 1400 položek na CryEngine Marketplace
- **Dokumentace a kurzy**
  - Unity: Dokumentace, vid. tutoriály, 6 600+ kurzů na Udemy
  - CryEngine: Dokumentace, tutoriály, jen 12 kurzů na Udemy
  - UnrealEngine: Dokumentace, vid. tutoriály, 2 900+ kurzů na Udemy
- **Komunita**
  - Unity: Na fóru 140 000+ témat na vývoj a 7 000+ na VR
  - CryEngine: Na fóru 1 800+ témat na vývoj a 36+ na VR
  - UnrealEngine: Na fóru 42 000+ témat na vývoj a 5 000+ na VR

Všechny uvedené hlavní enginy je možné používat zdarma pro nekomerční projekty, a tedy toto kritérium dále nehraje roli. Při výběru enginu je nutné se soustředit na vysokou podporu různorodých headsetů, jelikož má být výsledek této práce co nejvíce univerzální, a dále na co nejlepší výkonnost, jelikož ve světě VR je každý menší zásek znát a rozpoznávání gest může být dosti

náročná operace. V první řadě ze třech představených enginů bude odebrán CryEngine, který se sice může chlubit excelentním výkonem, ale jeho oficiální podpora je pouze pro 4 typy headsetů, konkrétně PSVR, Oculus Rift, OSVR a HTC Vive, a také má podstatně menší komunitu a znatelně méně položek na CryEngine Marketplace oproti Unity a Unreal Engine.

Zbýva tedy vybrat vhodnější engine mezi Unity a Unreal Engine, kde oba enginy se mohou pyšnit excelentní podporou VR platforem, ale Unreal Engine nativně podporuje VR platforem více, dle dokumentace všechny, kdežto Unity má nativní podporu pouze pro Oculus a Windows Mixed Reality. Co se výkonnosti týče, oba dva enginy jsou v dnešní době častou volbou při tvorbě VR projektů, ale větší projekty, kde rychlost a výkonnost hraje velkou roli, využívají spíše Unreal Engine, pravděpodobně také díky tomu, že celý engine funguje v C++ a v něm je možné více optimalizujících, nízkoúrovňových věcí než v C#, které používá Unity na programování.

Unity ovšem oproti Unreal Engine nabízí větší komunitu, více online kurzů a obecně edukačních materiálů a stejně tak má k dispozici více assetů v Unity Asset Store, ale je nutné podotknout, že Unreal Engine má na Unreal Engine Marketplace assety nejen od uživatelů, ale také ze zrušených projektů Epic Games, tedy assety z AAA her vysoké kvality. V poslední řadě je také třeba vzít v potaz zkušenosti autora, který své znalosti C++ hodnotí jako lepší, než znalosti C# a stejně tak má větší zkušenosti s Unreal Engine než s Unity.

Ze všech uvedených bodů tak plyne závěr, že práce bude implementována v Unreal Engine, zejména díky obrovské podpoře nejrůznějších VR headsetů a splnění tak univerzálnosti systému gest. Rozpoznání gest je náročná operace, a proto bude nutné programovat za využití C++, s možnou kombinací s Blueprints pro triviální úkony, pro zajištění co největší plynulosti.

2.2 VR hardware

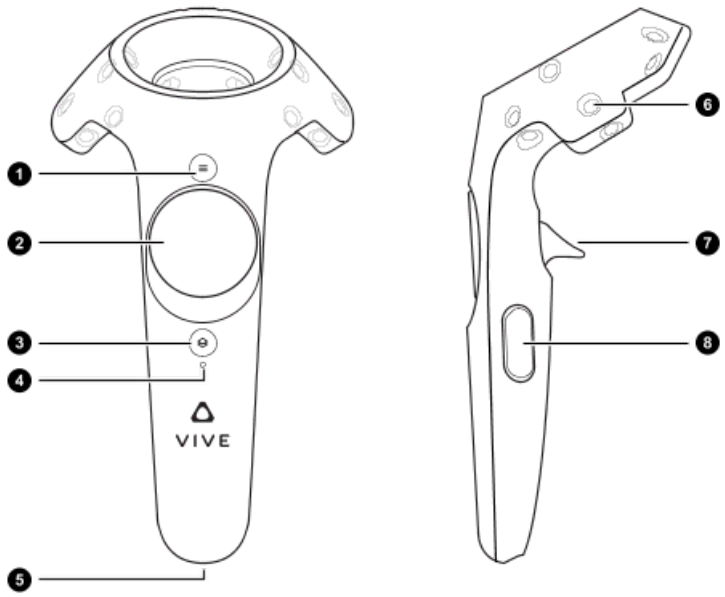
V této části práce je provedena analýza současně nejpoužívanějších VR headsetů pro domácí užití a porovnání jejich možností vzhledem k účelům této práce, tedy najití co nejuniverzálnějšího řešení pro zavedení systému gest, které bude použitelné ideálně na všechny uvedené i budoucí headsety. Pro praktickou část této práce bude použit headset HTC Vive, poněvadž k jiným headsetům autor práce za současné situace přístup nemá, a tím pádem bude toto zařízení použito jako základ při hledání univerzálně použitelných řešení.

Headset pro virtuální realitu je hardware starající se o zobrazování, ovládání a snímání uživatele ve virtuální realitě. Různé typy headsetů dále mohou nabízet funkce jako nahrávání zvuku, sluchátka, snímání pohybu očí nebo například vlastní baterii, a tedy možnost použití bez kabelů či dokonce samotného PC. Obnovovací frekvence obrazovky by neměla být o moc nižší než 90 Hz, protože pod touto hodnotou již začínají být časté problémy s nevolností způsobené nesouhlasem v mozku mezi pohybem hlavy a informacemi z očí.

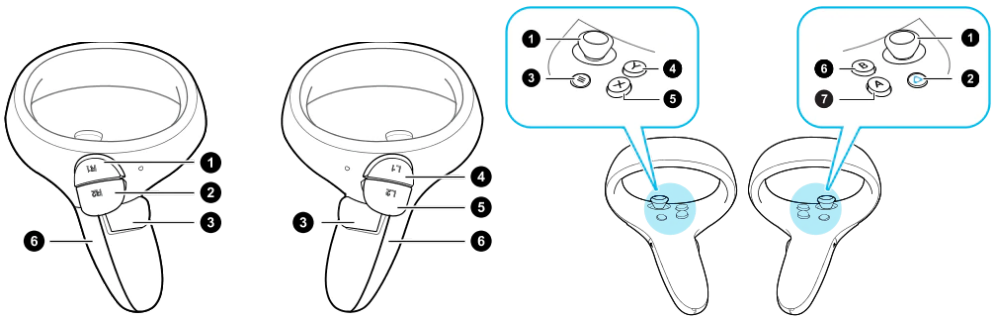
2.2.1 HTC Vive

Tato linie headsetů vznikla ze spolupráce firem HTC a Valve, vlastníka platformy Steam, které chtělo vytvořit headset pod vlastní správou pro použití na SteamVR, a tak vzniklo toho strategické partnerství s HTC [15]. První typ z této linie byl prvnímu zákazníkovi prodán v roce 2016 a od té doby přibyly ještě verze Pro, která přinesla lepší obrazovku, snímače pohybu a sluchátka, Pro Eye, jenž verzi Pro rozšiřuje o snímač pohybu očí, dále Focus, která nepotřebuje připojení k počítači ani externí snímače pohybu a obsahuje vlastní mobilní procesor, a nejnovější verzi Cosmos, kterou je možné pořídit buď s externím snímáním nebo bez něj. O externí snímání pohybu se starají tzv. majáčky, které jsou navzájem kompatibilní mezi verzemi i headsety. [16]

V každé z možných kombinací verzí HTC Vive je vždy nutně headset samotný a ovladače. Headset může být typu základní, Pro, Cosmos nebo Focus, ale jejich rozdíly jsou pro účely této práce bezvýznamné, jelikož se jedná především o rozdíly v kvalitě obrazovky a možnosti snímat ovladače bez použití externího snímání. Během 4 let existence série Vive firma HTC vyrobila již tři různé typy ovladačů, kde jedny slouží pro užití s externím snímáním pohybu a druhé pro snímání verzí Cosmos. Ovladače pro externí snímání existují ve dvou verzích, kde novější verze z roku 2018 podporuje snímání od externích snímacích stanic verze 2, ale naštěstí se skladbou tlačítek navzájem neliší. Již zde narážíme na problém zmíněný v zadání práce, tedy že ovladače mají částečně rozdílnou skladbu tlačítek, jak je vidět na obrázcích 2.2 a 2.1. Je možné



Obrázek 2.1: Schéma ovladače Vive [5].



Obrázek 2.2: Schéma ovladače Vive Cosmos [6].

vidět, že se neshoduje jak počet tlačítek (čtyři tlačítka na Vive ovladači oproti šesti tlačítkům na Vive Cosmos ovladači), tak ani typ použitého ovládání (touchpad na Vive ovladači a joystick na Vive Cosmos ovladači). Pro namapování ovládání VR aplikace na jednotlivá tlačítka tedy bude potřeba udělat nejspíše dvě verze, a to ačkoliv jsou ovladače od stejného výrobce. Je tedy vhodné najít způsob ovládání, který bude použitelný a dávat stejné výsledky na obou ovladačích.

Zařízení z linie HTC Vive tedy existují ve verzích s externím snímáním pohybu i bez něj, se sledováním pohybu očí i bez něj, se sluchátky v headsetu i bez nich a s nutností připojit počítač i bez ní. V každé možné kombinaci vždy najdeme headset a ovladače, u kterých se dají namapovat maximálně čtyři tlačítka v poměru 1:1 skrz linii Vive.

2.2.2 Valve Index

Po spolupráci s firmou HTC se společnost Valve rozhodla vytvořit headset, který bude čistě jejich výtvor, a tak v roce 2019 byl představen a vydán jejich vlastní headset nesoucí název Valve Index. Kromě běžných zlepšení, které přináší nové verze VR headsetů, jako je kvalitnější display nebo jeho vyšší obnovovací frekvence, Valve Index zejména láká svými ovladači, které umí rozpoznat pohyb prstů a přináší tak nové způsoby, jak interagovat s virtuálním světem. Používá externí snímání pohybu a je kompatibilní se senzory, které se používají pro sérii Vive, a stejně tak je možné ovladače od Valve Index používat pro headsety Vive a obráceně. Za zmínku stojí i odpověď tvůrců headsetu na otázku sluchátek, kterou vyřešili pomocí přidání reproduktorů, které jsou vzdálené kousek od ucha ale nedotýkají se. [58]

Na obrázku 2.3 je možné vidět schéma ovladače, které zde bude lehce popsáno. V přední části (třetí obrázek) lze vidět tři jednoduchá tlačítka, jedno velké tlačítko s detekcí tlaku a joystick. Dále má ovladač na svém těle senzor uchycení spolu s detekcí tlaku, který využívá na detekci pozic jednotlivých prstů. Nakonec má stejně jako ovladače Vive zespodu tlačítko pedálového

Obrázek 2.3: Schéma ovladače Valve Index [28].

typu. Valve dále uvádí [57], že jsou ovladače tvarovány s vysokým důrazem na pohodlí a dlouhé používání. Narozdíl od novějších ovladačů Vive jsou ovladače Valve Index kompatibilní i se starší verzí stanic pro externí snímání pohybu. Jejich výdrž má být minimálně 7 hodin na jedno nabití. Ovladače Valve Index tedy nabízí 3 klasická tlačítka, joystick, tlačítko s detekcí tlaku, pedálové tlačítko a plochu detekující stisk a tlak jednotlivých prstů, tedy opět jinou skladbu dostupných tlačítek a ovládacích prvků, než mají ovladače Vive.

2.2.3 Oculus

Počátky headsetů Oculus začínají v roce 2012, kdy byla založena firma Oculus VR a následně oznámena první verze Oculus headsetů, a to Oculus Rift. Ve stejném roce firma spustila crowdfunding skrz platformu Kickstarter a přes ní vybrala desetinásobek žádané částky a během vývoje se k vývojářům dostaly hned dva development sety. Před tím, než byl Oculus Rift vůbec prodáván veřejnosti, se firmě staly ještě dvě důležité věci. V roce 2014 došlo k jejímu odkoupení firmou Facebook, která za ni zaplatila astronomických 2,3 miliard amerických dolarů, kde 1,6 miliard bylo formou akcií Facebooku, 400 000 $ na ruku a dalších 300 000 $ po splnění jistých milníků [19]. V roce 2015 se firma již pod správou Facebooku spojila s firmou Samsung na výrobě Samsung Gear VR pro smartphony téže značky, který je blíže popsán v sekci 2.2.6.

V roce 2016 se k zákazníkům dostal první komerční headset Oculus Rift, který vyžaduje připojení k počítači a umožnuje sledování ovladačů bez externích snímačů. Ovladače pro Oculus Rift byly veřejnosti uvolněny až osm měsíců po vydání headsetu s názvem Oculus Touch. Roku 2017 se veřejnost mohla poprvé setkat s dalším typem headsetu, a to Oculus Go, který již obsahuje mobilní procesor a paměť a nevyžaduje připojení k počítači. Oculus Go funguje na software, který je založen na Androidu a aplikace se do něj stahují z obchodu Oculus Go Store. Ovladač pro Oculus Go je nepárový a nepou-

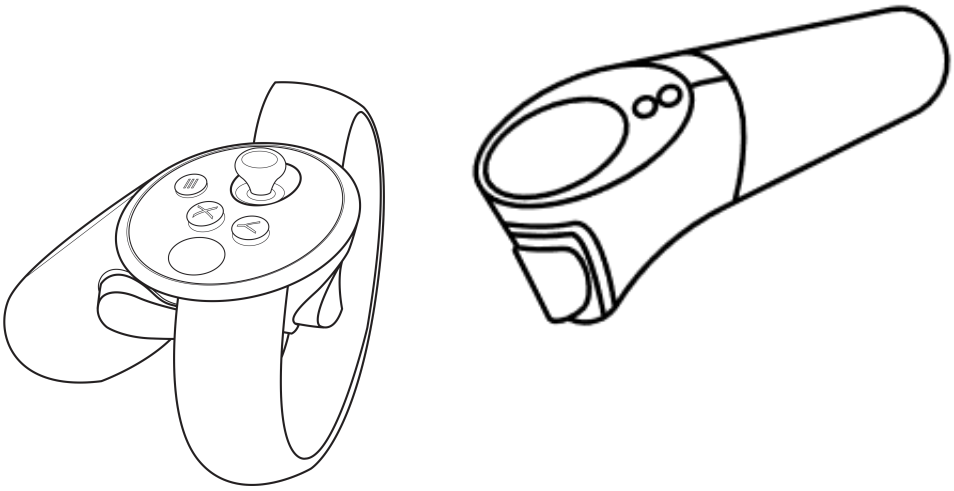
žívá sledování vůči headsetu, pouze sleduje pohyb v prostoru relativně k sobě samému. V roce 2018 pak Facebook představil Oculus Quest, jakožto výkonnější verzi Oculus Go, která má stejně jako Oculus Go operační systém na bázi Androidu a je určen pro mobilní používání, ale narozdíl od Oculus Go již má sledování ovladačů vůči headsetu, větší paměť, silnější procesor a lepší obrazovku [39]. V roce 2019 Facebook představil funkci Oculus Link, která umožnuje mobilní headsety jako Oculus Quest připojit k počítači přes USB port a využívat stejný software jako Oculus Rift. [43]

Všechny doposud zmíněné headsety z linie Oculus, ačkoliv poslední je teprve rok starý, jsou od nedávna oficiálně vyřazeny z výroby a Facebook nyní propaguje jen své dvě nejnovější varianty, Oculus Quest 2, který vyšel 13. 10. 2020 a Oculus Rift S, který vyšel zároveň s Oculus Quest, ale již nyní je na jaro 2021 plánováno jeho ukončení výroby, protože Oculus Quest 2 se svou možností jak mobilního, tak počítačového použití nahrazuje Oculus Quest i Oculus Rift S [48]. Oculus Rift S vyšel v roce 2019 jakožto následník Oculus Rift, a stejně jako on vyžaduje připojení k počítači a nevyžaduje externí sledování pohybu. Má lepší rozlišení, ale kupodivu menší obnovovací frekvenci, a to jen 80 Hz. Všechny headsety vyjma Oculus Go používají Oculus Touch ovladače. Oculus Quest 2 má dosud největší rozlišení (1832 x 1920 na oko) a podporu obnovovací frekvence 90 Hz. Dá se použít mobilně, s nutností využití aplikace na mobilu, nebo připojit k počítači. Oculus Quest 2 ale spolu s inovacemi přinesl i lehce kontroverzní problém, a to fakt, že jeho použití nutně vyžaduje přihlášení přes Facebook, což mnohé zákazníky odradilo. K Oculus Quest 2 vznikla nová verze ovladačů Oculus Touch v3, která naštěstí nemění skladbu tlačítek a přináší jen ergonomické vylepšení. [39]

V poslední části je třeba zanalyzovat ovladače linie Oculus, tedy tři verze Oculus Touch pro většinu headsetů Oculus a ovladač Oculus Go, který není párový a nemá sledování vůči headsetu. Jelikož tři různé verze ovladačů Oculus Touch mění jen ergonomickou stavbu ovladače a funkce sledování, ale ne dostupné možnosti ovládání, na analýzu se stačí podívat na jeden z nich. Jak lze vidět na obrázku 2.4, ovladač Oculus Touch má joystick, tři tlačítka a dvě tlačítka pedálového typu. Naopak ovladač pro Oculus Go nejen že není párový, ale má pouze touchpad, dvě tlačítka a jedno tlačítko pedálového typu. V linii Oculus se tedy dá spolehnout na to, že libovolný uživatel má k dispozici alespoň dvě klasická a jedno pedálové tlačítko. Nedá se ovšem spolehnout na existenci párového ovladače.

2.2.4 Windows Mixed Reality

Windows Mixed Reality je platforma od společnosti Microsoft, která je součástí jejich produktu Windows 10, a spojuje do jedné možnosti virtuální i rozšířené reality. Momentálně se rozšířená realita na této platformě týká výhradně výrobku z dílny Microsoft, a to brýle Microsoft HoloLens, které využívají pro ovládání mimo jiné gesta a jsou tedy blíže popsané v sekci 2.3.1 analyzující

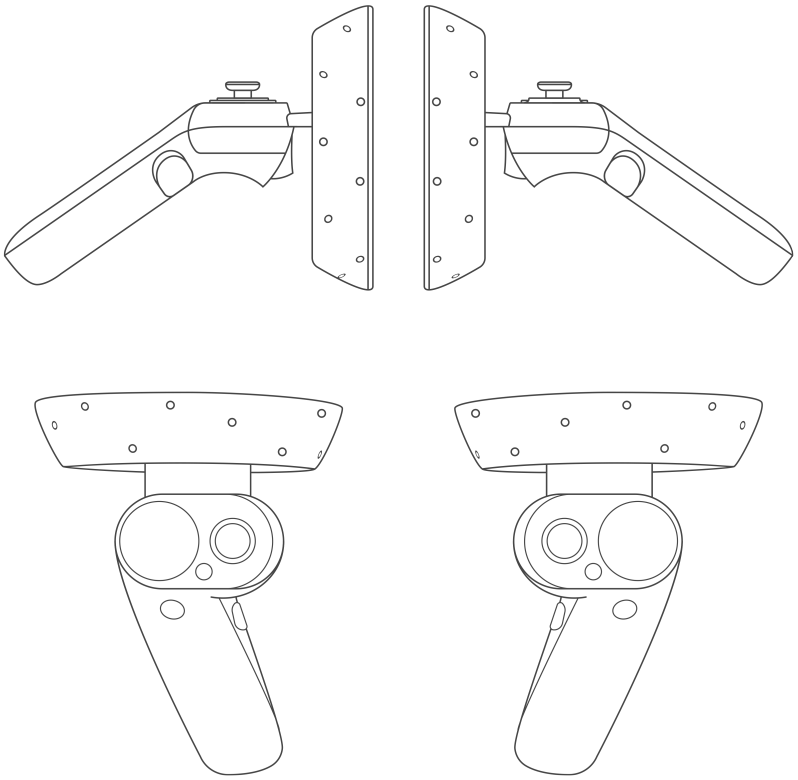


Obrázek 2.4: Schémata ovladačů Oculus Touch [40] a Oculus Go (vpravo).

systémy využívající gesta. Všechny ostatní headsety z platformy Windows Mixed Reality jsou určené pro virtuální realitu a jsou od různých výrobců, kteří jen musí dodržet jisté standarty platformy. Všechny headsety pro virtuální realitu na platformě Windows Mixed Reality fungují i na platformě SteamVR, dokonce v současné době probíhá vývoj headsetu HP Reverb G2 spoluprací firem Valve, Windows a HP, který má dle slov výrobce zavést nový standard ve VR [32]. Jeho rozlišení bude 2160 x 2160 na oko, součástí bude nový design ovladačů a jako všechny headsety z této platformy bude vyžadovat připojení k počítači.

Za pár let existence platformy pro ni vzniklo několik headsetů od různých výrobců, jako je Acer, HP, Samsung, Dell či Lenovo, ale naštěstí byly ovladače pro každý z nich totožné [17] a jejich schéma je možné vidět na obrázku 2.5, kde lze vidět dvě tlačítka, jeden joystick, jeden touchpad a jedno pedálové tlačítko. Ovladače jsou pro všechny headsety stejné, vyjma headsetů od Samsungu, u kterých došlo k drobné změně ve tvaru ovladače, ale skladba ovládacích prvků zůstala stejná, takže je netřeba vkládat jejich schéma. Ke změně ve stylu ovladače dojde právě s nadcházejícím headsetem HP Reverb G2, který navíc přidá dvě tlačítka na každém ovladači, ale odebere touchpad [32]. Všechny headsety z Windows Mixed Reality je možné ovládat také ovladači pro konzoli Xbox One, ale ty nemají sledování pozice a nejsou tak vhodnou volbou pro potenciální systém gest. U ovladačů pro Windows Mixed Reality se tedy dá

spolehnout na existenci alespoň dvou tlačítek, jednoho pedálového tlačítka a jednoho joysticku.

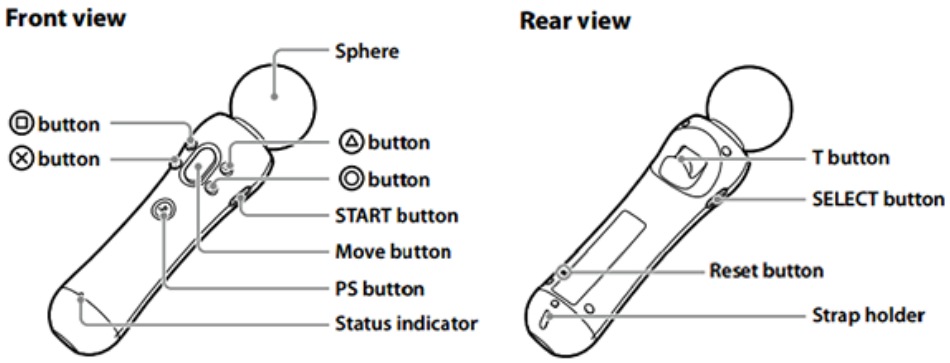


Obrázek 2.5: Schéma ovladačů pro Windows Mixed Reality [17].

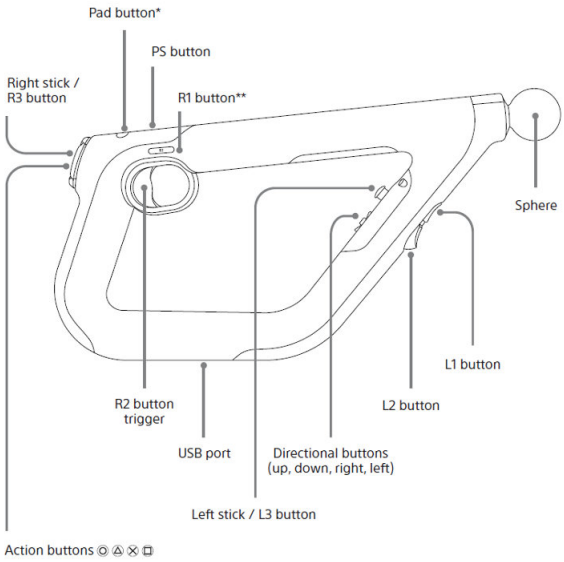
2.2.5 PlayStation VR

PlayStation VR (PSVR) je headset pro virtuální realitu vydaný v roce 2016 pro konzoli PlayStation 4 a pro nadcházející konzoli PlayStation 5 byla potvrzena jeho kompatibilita i s ní, stačí připojit kameru určenou pro PlayStation 4 pomocí adaptéru ke konzoli PlayStation 5 [44]. Headset obsahuje jednu OLED obrazovku s FullHD rozlišením, tedy na jedno oko připadá 960 x 1080 pixelů. Headset je nutné připojit ke konzoli, ale sám obsahuje slabší procesor, který se

stará o 3D zvuk. Headset má na sobě 9 LED světel, které jsou snímány speciální kamerou pro PlayStation sloužící pro externí sledování pohybu headsetu a ovladačů. [45]



Obrázek 2.6: Schéma ovladače PlayStation Move.



Obrázek 2.7: Schéma ovladače PlayStation VR Aim [46].

PSVR se dá ovládat pomocí klasického ovladače DualShock 4 pro PlayStation 4, ovladače pro virtuální realitu PlayStation Move a ovladače uzpůsobeného pro střelbu ve virtuální realitě PlayStation VR Aim. DualShock 4 je typem ovladače pro držení v obou rukách a má na sobě pouze jedno LED světlo v přední části, takže jeho sledování ve VR je poněkud nepřesné a musí

být natočen jednou stranou směrem ke kameře pro externí snímání, což může být při nasazeném headsetu obtížné. Mnohem lepších výsledků se docílí při použití ovladačů určených pro VR PlayStation Move nebo VR Aim. Jak lze vidět na schématech ovladačů, všechna mají stejné složení tlačítek, stejné jako DualShock 4 a jediným rozdílem je, že ovladače PlayStation Move jsou párové, a proto mají každý jen jedno pedálové tlačítko (T button na schématu 2.6) narozdíl od ovladače PlayStation VR Aim, který není párový a má pedálová tlačítka dvě (L2 a R2 na schématu 2.7), stejné jako DualShock 4. Na žádném z ovladačů pro PSVR nenalezneme touchpad a na ovladači PlayStation Move dokonce ani joystick. U ovladačů pro PSVR se tedy dá spolehnout na existenci alespoň osmi tlačítek klasických a jednoho pedálového, ovšem bez jistoty existence touchpadu či joysticku.

2.2.6 VR pro chytré telefony

V této části bude představen poslední způsob z možností virtuální reality, a to za využití chytrých telefonů. Do této kategorie spadají tisíce různých výrobků od rozdílných prodejců, ale v této práci bude výběr zúžen na 3 příklady, a to nejspíše nejznámějšího zástupce této kategorie Google Cardboard, jehož vybrané modely je možné si poskládat z kartonu, dále Samsung Gear VR, které vzniklo spoluprací Samsungu s Oculus VR popsanou v sekci 2.2.3, a Merge VR, které z množství podobných produktů vystupuje tím, že je tvořeno pro užití dětmi. Všechny headsety z této sekce jsou pouze držáky pro telefon a o zobrazení, vypočítání, zjištění polohy i zvuk se stará telefon sám. Rozdílné headsety pak mohou nabízet funkce navíc jako jsou třeba sluchátka nebo tlačítka navíc. Mobilní telefon je držen headsetem před očima uživatele a jeho obrazovka se rozdělí v půlce a každé oko tak vidí něco lehce jiného.

- **Google Cardboard** je dle slov výrobce nejlevnější a nejdostupnější VR platforma, která podporuje Android i iOS [27]. Nejznámější headset z této platformy je výrobek přímo od Google, jenž je z kartónu (odtud původ jména) a který si uživatel doma sám poskládá, ale headsetů zde nalezneme více, a to i od jiných firem, než je Google. Tyto headsety se ale musí držet jistých standardů pro platformu Cardboard, a to je kromě levnosti a lehkosti použití i přítomnost jednoho tlačítka pro vstup, které fungovalo na magnetickém základě a interagovalo s kompasem v mobilu, ale novější verze již používají vodivou páčku, která interaguje s dotykovou obrazovkou telefonu. Headset by měl na sobě obsahovat QR kód, jehož přečtení v aplikaci Google Cardboard způsobí správné nastavení aplikace pro fungování s daným headsetem [41].
- **Samsung Gear VR** je řada headsetů pro telefony série Galaxy výrobce Samsung, která vznikla jako jedna z prvních z řad VR headsetů pro mobily. Narozdíl od většiny VR headsetů pro chytré telefony je součástí balení i nepárový ovladač, který má touchpad, dvě tlačítka klasická

a jedno pedálové. Má zorné pole 101° a headset sám o sobě obsahuje gyroskop a akcelerometr a je možné se s ním napojit i do místností pro uživatele Oculus headsetů. [22]

- **Merge VR** headset je zaměřený na vzdělávání a výrobci ho propagují jakožto skvělou možnost pro učení, uvádějíc důvody jako odolnost, lepší učení díky ponoření se plně do výučného programu podpořené studiemi a nabízí také různé virtuální prohlídky. Je kompatibilní s Androidem i iOS a pro ovládání využívá dvě tlačítka na každé straně, které interagují s levou a pravou stranou dotykové obrazovky chytrého telefonu. Headset je tvořen z odolného materiálu, aby vydržel hrubší zacházení dětmi, je vodě i nárazu odolný a nabízí zorné pole o 96°. [37]

Headsetů pro chytré telefony je obrovské množství a neexistuje pro ně standard, ale většina z nich obsahuje alespoň jedno tlačítko, které se stará o interakci s dotykovou obrazovkou chytrého telefonu, pokud headset nemá dedikovaný ovladač. Pokud headset k sobě ovladač má, tak většinou nemá sledování jeho pohybu, tedy by nebylo možné jej využít pro univerzální systém gest, a tedy je třeba vymyslet řešení, které by fungovalo i za podmínky, že jediná věc, jejíž pohyb je sledovaný, je headset samotný.

2.2.7 Závěr analýzy VR hardware

V předešlých částích byly představeny nejznámější a nejpoužívanější headsety, ale existuje velké množství dalších, zde nepředstavených, které mohou přinést další rozdíly mezi ty již uvedené, ale když se vezmou v potaz všechny headsety zde představené, je možné vytipovat následující fakta:

- Nedá se spolehnout na existenci joysticku.
- Nedá se spolehnout na existenci touchpadu.
- Dá se spolehnout na existenci alespoň jednoho klasického tlačítka.
- Nedá se spolehnout na existenci pedálového tlačítka.
- Dá se spolehnout na sledování pohybu alespoň jednoho prvku.
- Nedá se spolehnout na sledování jednotlivých prstů.
- Nedá se spolehnout na existenci párových ovladačů.

Kvůli nemožnosti se spolehnout na existenci ani jednoho z ovládacích prvků joystick nebo touchpad nebude možné systém gest vymyslet za pomoci zadání vstupu z těchto prvků, tedy například nakreslení tvaru gesta na touchpad, a proto musí výsledné řešení využít jiných prvků, pakliže má systém být co nejuniverzálnější. Většina představených headsetů nemá sledování jednotlivých prstů, takže na nich není možnost zadávat gesta pomocí doslova ručních

gest, ale bylo by vhodné, kdyby výsledný systém tuto možnost měl, protože se dá očekávat, že funkčnost sledování jednotlivých prstů bude mít více a více novějších headsetů. Dále není možné se spolehnout na existenci dvou párových ovladačů, či obecně dvou sledovaných prvků, a proto výsledný systém musí být schopný rozeznávat gesta i za použití pouze jednoho sledovaného prvku. V poslední řadě není možné se spolehnout na existenci pedálového tlačítka, či více než jednoho klasického tlačítka (vyjma pár headsetů pro mobilní VR sloužící pouze pro 360° videa, které nemusí mít žádnou možnost pro ovládání, ale těch je minimum a jistě nevyužijí systém gest).

Všechna zjištěná fakta vedou k závěru, že co nejuniverzálnější systém gest by měl fungovat za použití maximálně jednoho tlačítka a za využití dat o poloze jednoho sledovaného prvku, tedy ovladače či headsetu, ale ideálně podporoval i data z více sledovaných prvků, například z párových ovladačů nebo jednotlivých prstů, aby se tak docílilo co největší volnosti v možnostech tvorby gest.

2.3 Systémy využívající gesta

Tato sekce se bude zabývat analýzou systémů využívající gesta, zejména jak vůbec fungují, jak gesta detekují, jaká gesta využívají pro jaké akce a jejich využití v životě. Rozpoznání gest je typ percepčního uživatelského rozhraní, které umožnuje počítačům zachytit a vyhodnotit lidská gesta jakožto příkazy, tedy definice rozpoznání gest je schopnost počítače porozumět gestům a vykonat adekvátní výkony dle daného gesta.

2.3.1 HoloLens

HoloLens jsou brýle pro smíšenou realitu od společnosti Microsoft, jejichž vývojářská verze byla vydána již v roce 2016, ale ačkoliv již byla v roce 2019 ohlášena nová verze HoloLens 2 s plánovanou cenou 3 500 $ [33], zatím neexistuje verze pro běžného uživatele, nýbrž pouze verze pro vývojáře a podniky. Slouží pro skloubení virtuálního a reálného světa, což se liší od rozšířené reality, kde je jen nad reálným světem zobrazeno virtuální překrytí. Mixovaná realita se tedy liší tím, že zde dochází k interakci virtuálních prvků s prvky reálnými, a tedy je například možné na zeď umístit prohlížeč obrázků či Skype hovor. HoloLens je možné využít například jakožto modelovací pomůcku pro programy Autodesk Maya nebo HoloStudio, což jsou programy pro 3D tvorbu, dále pro virtuální prohlídky, hraní Minecraftu pro smíšenou realitu, prozkoumání lidské anatomie a mnohé další.

HoloLens 2 mají rozlišení 2k na holografických čočkách, přes které lze vidět, a které vykreslují obsah dle pozice očí. Obsahuje senzory pro sledování hlavy, očí, hloubky, pozice a FullHD kameru. Pro navigaci v prostředí využívá svých senzorů o poloze a zmapování okolí, které slouží pro umístění virtuálních objektů na objekty v reálném světe. Má přímo v sobě procesor, baterii,

Wi-Fi i Bluetooth kartu, operační paměť i 64 GB úložného prostoru, a tedy nepotřebuje žádné připojení přes kabely pro fungování a na baterii vydrží 2-3 hodiny. Má na sobě jak mikrofon, tak reproduktor a pro interakci s uživatelem využívá 3 hlavní způsoby [29]:

- **Sledování rukou**: HoloLens je možné ovládat pomocí gest na obou rukách a přímou interakcí s objekty.
- **Sledování očí**: Kontrola, na co se uživatel dívá a dle toho adekvátní odezva, například navigace v menu.
- **Hlas**: Brýle je možné ovládat pomocí hlasových příkazů, při internetu i přirozená řeč.

Brýle mají určitou oblast, ve které dokážou snímat ruce a jejich gesta. Pokud v oblasti HoloLens nedetekují ruce, přepne se ovládání na ovládání pomocí pohledu, jinak lze vidět virtuální paprsek vycházející z dlaně ruky. Pokud je ruka ve zmíněné oblasti, je možné použít jednu ze základních akcí a gest, a to [30]:

- **Dotek**: U HoloLens 2 je možné se virtuálních objektů, nazývaných hologramy, přímo dotknout a následně s nimi manipulovat. Při držení ruky ve snímané oblasti se objeví kurzor před špičkou ukazováčku pro snazší výběr hologramu. Při doteku virtuálního objektu se hologram vybere a zobrazí se kolem něj další možnosti manipulace, jako například rotace pomocí chycení (reálnou rukou virtuální objekt) kuličky reprezentující danou osu a následné rotace kolem ní nebo chycení hologramu samotného a jeho přetažení v prostoru. Dále je po vybrání hologramu možné měnit jeho velikost pomocí chycení a táhnutí jezdce reprezentujícího velikost či výběr z pár možností editace, jako je i duplikace nebo smazání.
- **Dlaňový paprsek**: Při podržení ruky ve sledované oblasti s dlaní směřující pryč se objeví paprsek počínající v dlani, kterým je možné vybírat a následně manipulovat s objekty. Po výběru hologramu se u něj objeví stejné ovládací elementy jako při doteku a je možné provádět stejné úkony akorát na dálku, akorát navigace a výběr možnosti v menu například editace hologramu není realizováno přímým dotekem, ale vzdušným poklepáním, které je představenu v následujícím bodě.
- **Vzdušné poklepání**: Toto gesto je definované jako držení ruky ve volnější pěsti s palcem venku a ukazováčkem směrujícím ke stropu a následným klepnutím ukazováčku na palec, což slouží jakožto gesto výběru, případně podržení ukazováčku na palci, což je gesto pro zvolení a podržení objektu. Například navigace v menu probíhá tak, že si uživatel paprsek z dlaně vybere položku a výběr potvrdí vzdušným poklepáním. Příklad pro využití podržení je změna velikosti hologramu, tedy uživatel

nasměruje paprsek na jezdec pro měnění velikosti a podrží ukazováček na palci, během čehož je možné s jezdcem hýbat.

- **Start menu**: Pro otevření menu Start je dedikované gesto, které je vymyšleno tak, že uživatel dá do snímaného prostoru otevřenou ruku dlaní směrem k sobě a na zápěstí se mu objeví holografické logo Windows. Pokud se následně na této ruce dotkne ukazováčkem palce, otevře se Start menu.

2.3.2 Kinect

Další výrobek z dílny Microsoft byl v roce 2010 poprvé vydán pro veřejnost jako doplněk pro konzoli Xbox 360 s cílem poskytnout alternativu ke klasickému ovladači pro ovládání konzole. Kinect přinesl tři nové způsoby, jak se dalo s konzolí interagovat, a to pomocí jednoduchých gest, hlasového ovládání a sledováním celého těla, například pro ovládání postavy ve hře. Kinect ale neměl sloužit pouze jako pomůcka pro hraní her, jednou z jeho dalších avizovaných funkcionalit byl videohovor, který využíval sledovací schopnosti Kinectu a jeho mikrofon pro sledování hlavy člověka při volání a její centrování ve videu. Kinect se ale jakožto doplněk konzole neuchytil, zejména kvůli malé podpoře od tvůrců, ale získal nečekané uplatnění v akademické sféře, zejména protože jeho cena byla nižší než jiné senzory pro sledování těla. V roce 2017 tak Microsoft oznámil konec výroby Kinectu pro konzole a přestane ho prodávat po vypotřebování zásob [60], ale to nebyl konec Kinectu, jelikož v roce 2019 byl představen a vydán Azure Kinect, který bude blíže popsán v posledním odstavci této části.

Za desetiletou existenci Kinectu vzniklo 5 verzí, počínaje původním Kinectem pro konzoli Xbox 360 a jeho lehce vylepšenou verzí (tišší, USB kabel) pro použití s PC. V jádru sledování pohybu je senzor na infračervené světlo a jeho vysílač, který vysílá tisíce infračervených teček na scénu a čím dál je daný objekt, tím větší bude jednotlivá tečka a dle jejich velikostí Kinect rozpozná vzdálenost. Původní Kinect toto dělal třicetkrát za sekundu. Dále se algoritmus zaměří na pohyblivé části, pravděpodobně lidi, a izoluje tyto segmenty spolu se záznamem z kamery. Tyto segmenty jsou následně porovnány algoritmy využívající strojové učení, které určí, o jakou část těla se nejspíše jedná, a z těchto výstupů je následně poskládána kostra reprezentující člověka. První generace Kinectu byla schopná sledovat dvě kostry a polohy až čtyř hráčů [31]. Jelikož tato verze Kinectu ještě neuměla rozeznat prsty, ovládání se dělo pomocí celého těla. Na Kinect bylo třeba zamávat, což způsobilo sledování pozice ruky, kterou následně šlo pohybovat kurzorem po obrazovce a volba se potvrdila podržením kurzoru nad danou volbou.

Další verzí byl Kinect určený pro konzoli Xbox One, která vyšla v roce 2013, a jeho verze upravená pro užití s PC, která ale Kinect nijak neměnila vyjma použitého kabelu pro připojení. Tento Kinect již měl ToF (time-of-

flight) senzor, tedy senzor, který je schopný poznat vzdálenost objektů podle toho, jak dlouho trvá vyslanému světlu se vrátit. Oproti svému předchůdci má zhruba trojnásobnou přesnost a je schopen sledovat i jednotlivé prsty a stejně jako předchůdce využívá infračerveného světla pro fungování i bez viditelného světla. Dokáže sledovat až 6 koster lidí, jejich tep, výraz v tváři, váhu na jednotlivých končetinách, rychlost pohybu a jiné. Obsahuje již FullHD kameru a dokáže zpracovat 2 gigabity dat za sekundu [38]. Schopnost rozeznat prsty druhé verze Kinectu umožnila přidat další prvky ovládání menu konzole, například pohybovat se v menu konzole chycením menu a tažením na stranu (sevření pěsti), zavírat aplikace pomocí udělání gesta jako by člověk chytl obě strany obrazovky a sevřené ruce dal k sobě, a naopak od sebe pro otevření. Zvolení volby v menu nebylo nutné dělat podržením kurzoru nad volbou, ale přímo „zmáčknutím“ dané volby rukou ve vzduchu. V roce 2017 byl oznámen konec produkce a mělo se za to, že je to konec Kinectu.

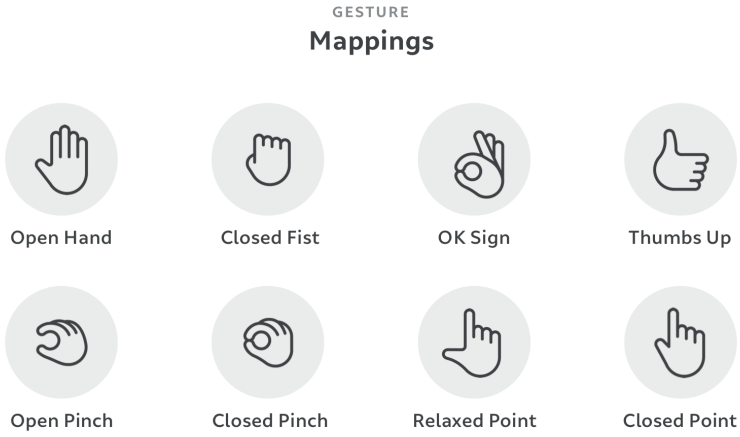
V roce 2018 však Microsoft představil Azure Kinect, verzi Kinectu určenou pro podniky a umělou inteligenci, využívající platformu Microsoft Azure, která nabízí širokou nabídku funkcí týkající se umělé inteligence. Má 12MPx kameru, stejný ToF senzor jako mají HoloLens 2 a 7 mikrofonů pro prostorové vnímání hlasu. Azure Kinect má na sobě speciální pin (kolík), pomocí kterého je možné synchronizovat data z dalších Kinectů. Je o polovinu menší než Kinect druhé generace, a výrobce uvádí příklady užití jako třeba přidání smart funkcionalit všedním předmětům, asistenta při tréningu či hlídače zásobování v obchodě [8].

2.3.3 Magic Leap

V roce 2018 vydala firma Magic Leap výrobek pro prostorovou práci s počítačem (spatial computing), který nese stejné jméno jako firma sama. K pojmům virtuální, rozšířená a mixovaná realita tak přibyl další pojem, tedy spatial computing, který popisuje interakci s reálným světem na rozdíl od mixované reality tak, že je nejen možné interagovat s reálným světem, ale i reálné věci přenášet do programu a dále s nimi interagovat, například je možné vzít ze stolu reálný nůž, který se naskenuje a vytvoří virtuální model, kterým je možné hýbat a interagovat s virtuálním světem. A právě na tuhle technologii je zaměřený výrobek Magic Leap, který si staví za cíl přinést lidstvu přirozenější interakci s technologií, která se má primárně přizpůsobovat lidem, a ne lidé technologii. Firma výrobek nazývá počítačem, který ví kde je v prostoru a pomocí senzorů a kamer chápe fyzický svět kolem sebe, který následně mixuje se světem digitálním [62].

Magic Leap 1, zatím jediný výrobek z této řady, se skládá ze tří částí nazvané Lightwear, Lightpack a Control. Lightwear je část, která je umístěná na hlavě a obsahuje brýle, přes které jde vidět a na kterých se vykresluje digitální obsah s obnovovací frekvencí 120 Hz, dále senzory pro rozpoznání okolí a pozice výrobku, mikrofon i reproduktor. Výrobci si dali za cíl udělat tuto část

zařízení co nejlehčí pro co největší komfort, a to se jim jistě povedlo, jelikož Lightwear váží pouze 316 g, oproti například HoloLens (popsaných v sekci 2.3.1) vážících 556 g. Lightpack je malý počítač, který má člověk kolem pasu a je k němu připojen Lightwear, obsahuje veškerý výpočetní hardware, tedy CPU i GPU od Nvidie, 8 GB RAM, 128 GB úložiště, karty pro připojení Wi-Fi i Bluetooth a baterii, která vydrží až 3 hodiny. Control je název pro ovladač, který má touchpad, dvě tlačítka klasická a jedno pedálové. Má senzory pro rozpoznání své pozice i rotace, baterii, která vydrží až 7 hodin, a kolem touchpadu má kruh z LED světel používaný pro některé notifikace uživatele. [36] Magic Leap nativně podporuje gesta rukou podobně jako HoloLens, tedy



Obrázek 2.8: Gesta podporovaná zařízením Magic Leap [35].

dokáže sledovat gesta v dané oblasti snímání, ale tato funkcionalita je zatím v betě, a i přímo výrobce vybízí k využití primárně ovladače Control, sekundárně mobilní aplikace a až terciárně využívat gesta. Výrobce uvádí, že zákazníci jsou prvně z funkcionality gest nadšení, ale po chvíli se stane jejich užívání dosti únavné a následně je skoro přestanou používat, což jde proti vizi výrobce o co největší pohodlí a komfort. Podporovaná gesta je možné vidět na obrázku 2.8, kde Open Hand se využívá pro výběr z blízka, Open Pinch pro výběr z dálky, Closed Fist pro chycení a manipulaci z blízka a Closed Pinch z dálky, dále OK Sign či Thumbs Up pro potvrzení, Closed Point pro aktivaci z dálky a Relaxed Point pro výběr, aktivaci z blízka a scrollování (pomocí následného tažení nahoru či dolů).

Výrobce dále pro vývojáře uvádí Best Practices při práci s gesty, konkrétně nabádá, aby gesta měla vždy nějakou odezvu (fyzickou, vizuální či audio), aby

si vývojáři dali pozor na rozdíly mezi levou a pravou rukou, aby gesta pro přímou interakci, jako je třeba výběr položky v menu, byla více odpouštějící a aby vývojáři vzali v potaz, že čím více gest bude aplikace podporovat, tím déle bude trvat rozpoznání gesta [35].

2.3.4 CyberGlove

CyberGlove Systems, s historií počínající roku 1990, je jedna z celosvětově nejznámějších firem zabývající se datovými rukavicemi, což je vstupní zařízení pro interakci s počítačem používané jako rukavice, které mají senzory pro snímání polohy a tvaru ruky, umožňující uživateli interagovat s digitálními objekty ve virtuálním prostředí. Firma současně nabízí dva typy rukavic, dvě možnosti přidání hmatové zpětné vazby rukavicím, a navíc jejich výrobky pro zavedení silové odezvy při interakci s objekty, tvořící iluzi odporu při doteku virtuálního objektu [4]. Rukavice nativně gesta rozpoznávat neumí, ale firma nabízí propracovaná API v jazyce C++ pro jednoduchou manipulaci s daty z rukavic a stejně tak ovládání jejich produktů pro imitaci odporu, jejich portfolio tedy tvoří [14]:

- **CyberGlove II** je datová rukavice sledující 18 či 22 bodů a úhlů mezi nimi s vysokou přesností. Verze s 18 senzory má dva senzory ohnutí na každém prstu, 4 senzory polohy, senzor pro měření překrytí palcem, prohnutí dlaně a natočení a polohu zápěstí. Verze s 22 senzory má na každém prstu 3 senzory ohnutí, 4 senzory polohy, senzor pro měření prohnutí dlaně, ohnutí zápěstí a jeho polohu. Rukavice jsou tvořeny z pružné tkaniny pro pohodlí a prodyšné části na dlani pro ventilaci, a navíc verze s 18 senzory má otevřené špičky prstů.

- **CyberGlove III** je novější generace datových rukavic a stejně jako CyberGlove II existuje ve verzích s 18 a 22 senzory. Tato verze má navíc přímo na sobě datové úložiště, možnost synchronizovat se s dalšími systémy zachycující pohyb, úsporný režim, nový design přinášející větší pohodlí, možnost připojení dalších senzorů na rukavici, aktivní senzor prohnutí dlaně a zpětnou podporu pro software využívající CyberGlove II. Kromě uložení dat na SD kartu nebo přes USB přidává možnost data přenášet přes Wi-Fi.

- **CyberTouch** je systém umožňující přidat hmatovou zpětnou vazbu pro datové rukavice CyberGlove. Jedná se o malé, vibrotaktilní stimulátory na každém prstu a na dlani, které je možné každý individuálně naprogramovat tak, aby simulovali různou sílu dotykových vjemů. Simulátory mohou generovat jednoduché vjemy, jako jsou pulsy nebo trvalé vibrace, a lze je použít v kombinaci k vytvoření komplexních hmatových zpětných vazeb.

- **CyberTouch II** je novější iterací systému umožňující přidat hmatovou zpětnou vazbu datovým rukavicím a narozdíl od svého předchůdce se senzory umisťují na vnitřek špiček prstů, jelikož jsou o dost menší a lehčí a zároveň tak přináší více realistické interakce s virtuálním prostředím, jelikož je možné zpětnou vazbu cítit na konečcích prstů. Šestý senzor je stejně jako u předchůdce umístěn na dlaň.

- **CyberGrasp** je systém silové zpětné vazby pro prsty a ruce. Umožňuje tak tvořit iluzi uchopení digitálních objektů ve virtuálním prostředí a například tak zabránit uživateli objekt v ruce, jelikož systém zabrání prstům se sevřít. Zařízení je lehký exoskelet, který se nasadí na datovou rukavici CyberGlove, a skládá se ze sítí šlach směrovaných do konečků prstů. Existuje pět akčních členů, jeden pro každý prst, které lze individuálně naprogramovat, aby se zabránilo prstům uživatele v proniknutí nebo rozdrcení virtuálního pevného předmětu. Zařízení vyvíjí síly, které působí zhruba kolmo na konečky prstů v celém rozsahu pohybu, a lze je naprogramovat jednotlivě. Systém CyberGrasp umožňuje plný rozsah pohybu ruky a nebrání pohybům nositele.

- **CyberForce** je systém rozšiřující možnosti CyberGrasp a to uzemněním celého exoskeletu a možnosti tak vytvářet protichůdné síly na celou ruku a ne jen prsty. S pomocí CyberForce je tedy například možné ve virtuálním prostředí položit ruku na volant či jiný povrch a systém CyberForce znemožní pohybu ruky dolů a vytvoří tak velmi přesvědčivý vjem reálnosti. Systém je již ovšem prostornější a váží necelých 9 kg, s možností vyvinutí síly o velikosti 8,8 N.

- **Haptic Workstation** je simulační systém zahrnující všechny představené produkty, tedy se jedná o stanici i s židlí a dvěma rukavicemi, které na sobě mají napojené CyberGrasp i CyberForce, dále headsetem se sledováním pozice, počítač pro 3D simulace a adekvátní software umožňující načítat 3D CAD modely pro okamžitou a intuitivní manipulaci s daným modelem. Výrobce jako příklad užití uvádí, že pomocí Haptic Workstation je možné „pocítit“ jaké to je sedět uvnitř designu automobilu a „pověsit“ obě ruce na virtuální volant před tím, než se postaví nákladný prototyp.

2.3.5 Další příklady

V této podsekci budou zběžně představeny další zajímavé systémy, produkty a firmy využívající gesta z různých odvětví, konkrétně:

- **GestureTek**, což je firma založená roku 1986, která se zaměřuje na technologie pro ovládání videa pomocí gest. Za dlouhá léta působení firma

dosti rozšířila své portfolio, které se skládá z výrobků nabízejících interaktivní povrchy promítané na jakoukoli podlahu, zeď, stůl nebo zobrazené pomocí LCD, ovládané jednoduchými gesty rukou, nohou a těla. Produkty jsou škálovatelné na jakoukoli velikost na jakémkoli povrchu a je dle výrobce [23] perfektní pro digitální značení, zábavu v obchodě, informace o produktu a pohlcující zážitková prostředí. Za svou existenci firma nainstalovala přes 8 000 interaktivních multimediálních displejů, kiosků, exponátů, digitálních značek a reklam, virtuálních herních systémů a dalších interaktivní řešení pro povrchovou výpočetní techniku, mnoho s více dotykovou (multitouch) interaktivitou. Příkladem výrobků je třeba GroundFX, u kterého je na zem promítáno video, s kterým jde interagovat pohyby těla, podobně tak WallFX, ScreenFX a TableFX, přinášející stejnou funkcionalitu na zeď, obrazovku a stůl respektive.

- **Eyesight Technologies**, vyvíjející přední systémy pro monitorování řidičů a obsazenosti prostředí ve voze za využití počítačového vidění a umělé inteligence (AI) k vytváření bezpečnějších a lepších zážitků z jízdy. Mezi jejich řešení pro prostředí automobilu patří Driver Senes – pokročilý systém sledování řidiče, který sleduje řidiče a jejich stav v reálném čase, Cabin Sense – systém sledování obsazenosti a interiéru, který sleduje celé prostředí v automobilu a Fleet Sense – řešení pro monitorování řidičů flotily a poskytovatele telematických systémů. Tato tři řešení byla navržena s cílem zvýšit bezpečnost a zlepšit zážitek z jízdy [18]. Firma dále využívá získanou technologii pro tvorbu produktu, jenž umožní sledovat angažovanost diváků při sledování televize, nazvanou Viewer Sense, která se snaží diváka nalákat pomocí umožnění ovládání zařízení (třeba televize) gesty, ale primární účel této technologie je poskytnout informace o divácích, jako je i věk či pohlaví. V poslední řadě firma umožňuje svoji technologii – tedy analýzu uživatele a ovládání gesty – uplatnit na každodenní zařízení jako je notebook nebo tablet a nazývá ji Device Sense.

- **BMW Gesture Control**, tedy jeden z příkladů využití gest pro ovládání určitých funkcionalit auta, který se nachází ve vybraných modelech aut značky BMW. Uvedené důvody zavedení tohoto systému jsou snížení roztržitosti při užití fyzických ovládacích prvků a větší pohodlí uživatelů. BMW Gesture Control nyní umožňuje přijmout či odmítnout hovor, a to ukázáním na dotykovou obrazovku, resp. mávnutí rukou doprava, dále upravovat hlasitost točením ukazováčku po či proti směru hodinových ručiček, změnit úhel přední kamery uděláním gesta chycení a táhnutí doprava či doleva a vybrat navigaci ukázáním dvou prstů na dotykovou obrazovku. [61]

- **Gestix**, což je sterilní rozhraní využívající gesta určené pro doktory, umožňující prohlížet lékařské obrázky v dynamickém lékařském pro-

středí. Systém interpretuje gesta uživatele a následně naviguje či manipuluje s prostředím s daty a obrázky pro daný lékařský úkon. Gesta rozeznává pomocí analýzy barev a jejich pohybu a gesta vyhodnocuje stavový automat. Prototyp byl otestován neurochirurgem při operaci a jeho zpětná vazba byla velmi pozitivní. [59]

- **Wii Remote**, které si zde jistě zaslouží zmínku, jelikož se jedná o první ovladač svého typu, tedy ovladač se snímáním pohybu, volně dostupný veřejnosti jakožto součást konzole již v roce 2006. Ovladač sám o sobě byl schopný měřit akceleraci na všech třech osách, což stačilo při užití například při imitaci máchání raketou při tenise nebo holí u golfu. Pro přesnější akce, jako například pohyb kurzoru na obrazovce, bylo zapotřebí připojit i Sensor Bar, což je zařízení vysílající infračervené záření, které je Wii Remote schopen vidět a dle něj rozpoznat vzdálenost sebe sama od Sensor Baru. Wii Remote bylo kompatibilní i s následující konzolí Wii U, ale nyní se již nevyrábí, jelikož současná konzole Nintendo Switch má jiné ovladače (Joy-Con) s vlastní technologií pro snímání pohybu. Zajímavostí je, že teprve až v roce 2020 Nintendo vyhrálo soudní spor trvající 7 let, ve kterém bylo obviněno za použití technologie pro sledování pohybu ve Wii Remote na základě patentu firmy iLife [47].

- **Ultraleap**, což je firma, která vznikla v roce 2019 odkoupení firmy Leap Motion firmou Ultrahaptics a dále nabízí produkty firmy Leap Motion, tedy zejména senzor stejného jména a senzor Stereo IR 170. Leap Motion Controller je malá krabička, která je pomocí infračerveného světla schopná snímat ruce v jejím snímatelném prostoru a je možné ji připojit přes USB například k notebooku. Tento produkt je určený pro veřejnost, ale dále firma nabízí Stereo IR 170, delší ale užší senzor, který ještě není určen pro veřejnost, ale má lehce větší prostor pro sledování rukou a mnohem větší zorné pole, konkrétně 170°x170° oproti 140°x120° u Leap Motion Controller. Firma Ultraleap dále prodává nástavce pro VR headsety, na které je možné umístit jejich Leap Motion Controller senzor a přidat tak možnost sledování rukou do zařízení, které je nepodporují. V současné době se firma primárně zabývá Gemini – platformou pro sledování pohybu rukou páté generace, která má přinést zatím nejvyšší rychlost a přesnost a umožní sledovat ruce jak při užití headsetu, tak i při klasickém užití počítače. [51]

- **Just Dance**, tedy hru, ve které se hráči snaží napodobit taneční pohyby postav ve hře. Proč je tedy v systémech využívající gesta zmíněná hra? Protože ačkoliv se hra ovládá pohyby těla člověka, zejména jedné ruky, je dostupná na všech aktuálních konzolích, tedy i těch, které nemají možnosti sledování pohybu hráče, jako je třeba Google Stadia. Na konzolích, kde je takový hardware přítomen (například Kinect), jej hra plně využije, pokud ale takový hardware přístupný není, je možné v ruce

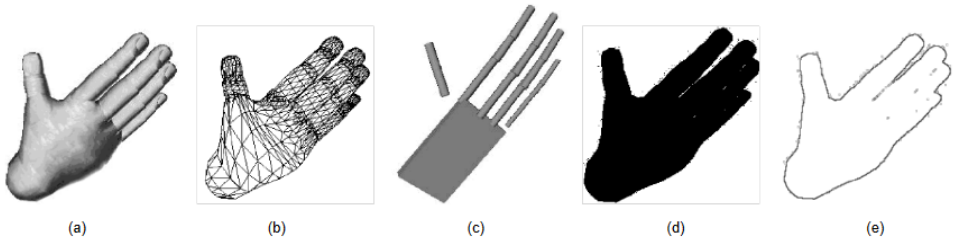
držet ovladač a hra sleduje jeho data o poloze a porovnává s referencí a pokud nejsou přítomny ani ovladače, je hru možné ovládat pomocí držení mobilního telefonu v ruce a hra stejně jako u ovladače z polohových dat telefonu vyhodnocuje přesnost pohybů. Navíc je možné všechny tyto možnosti míchat dohromady, je to tedy perfektní příklad univerzálnosti použitého systému.

2.4 Přístupy pro rozpoznání gest

V této sekci budou představeny čtyři různé přístupy pro detekci a rozpoznání gest, které je možné rozdělit na přístupy využívající algoritmy analyzující 3D model objektu a ty analyzující vzhled neboli 2D reprezentaci [42]. Přístupy, jenž využívají 3D modely je možné dále rozdělit dle toho, jestli se zabývají složitým 3D modelem nebo jeho zjednodušenou reprezentací ve formě kostry modelu. Dále zde bude zběžně představen poněkud nový přístup k rozpoznání gest, analyzující elektrické potenciály vzniklé v důsledku činností svalů, který ale nebude možné využít pro účely této práce.

2.4.1 Přístupy založené na 3D modelech

Tento přístup využívá volumetrické modely, které jsou vytvořené ze složitých 3D ploch jako jsou NURBS nebo polygony. Výpočet při užití těchto modelů je ale velmi náročný a zatím moc nepoužitelný pro rychlou analýzu v reálném čase, a je vhodné modely simplifikovat za užití primitivních tvarů, jako je třeba koule pro hlavu a podobně, čímž se ovšem ztrácí přesnost. Algoritmy z této části se zabývají analýzou reprezentace objektu například v podobě (a) či (b) z obrázku 2.9.



Obrázek 2.9: Různé interpretace modelu ruky [42].

2.4.2 Přístupy založené na kostře modelu

Místo využití 3D modelů a jejich časově náročné analýze se algoritmy v této části zabývají analýzou kostry modelu, viditelnou například pod ukázkou (c) obrázku 2.9, tedy sérii úhlů a vzdáleností mezi jednotlivými segmenty, které

v nejvíce simplikované verzi jsou pouze úsečky, což je plně dostačující pro například reprezentaci ruky, kde úsečky mohou reprezentovat jeden článek prstu a úhly mezi nimi odpovídají úhlům v kloubech. Analýza se zde tedy zabývá pozicí, orientací, délkou a úhly mezi jednotlivými segmenty. Výhodami využití tohoto přístupu je rychlý běh algoritmu, jelikož se zabývá pouze klíčovými parametry, dále možnost porovnávat jednoduše data s referencí a zaměření se pouze na důležité části těla pro rozpoznání gesta, jelikož například při využití celého 3D modelu dojde k analýze i hřbetu ruky, která jistě informace o použitém gestu nenese.

2.4.3 Přístupy založené na vzhledu

Algoritmy z této části se nezabývají analýzou různých prostorových reprezentací těla či ruky, ale přímo obrázku nebo videa, ze kterého se izoluje objekt zájmu a je porovnán s databází. Po lokalizaci předmětu zájmu v obrázku (případně série obrázků u videa) je vytvořena jeho zjednodušená 2D podoba, například binární reprezentací (viz ukázka (d) na obrázku 2.9) nebo za vyžití odhadu obrysu (viz ukázka (e) na obrázku 2.9), a tato podoba je následně porovnána s databází. Další možností je porovnávat přímo sekvenci obrázků, nebo jejich výsek, bez předešlého zjednodušení předmětu zájmu na siluetu, a určení použitého gesta v nich. Reálné systémy pak mohou používat různé kombinace zmíněných algoritmů, například Kinect využívá algoritmy založené na vzhledu k lokalizaci končetin ve videu (sekvenci obrázků) a z výsledků těchto algoritmů poskládá kostru pro reprezentaci těla, se kterou dále pracuje. Tento proces je popsán v druhém odstavci sekce 2.3.2.

2.4.4 Přístupy založené na EMG

Elektromyografie (EMG) je elektrodiagnostická metoda, která slouží především k diagnostice poruch nervosvalového aparátu. Podstatou je měření elektrických potenciálů vzniklých v důsledku činnosti kosterní svaloviny. V dnešní době již existují zařízení EMG umožňující neinvazivní přístup (jednou z možností měření je pomocí jehly) a dají se připojit přes Bluetooth. Získaná data při dělání určitého gesta se adekvátně klasifikují a jsou dále využita ve vybraném algoritmu pro strojové učení. Provedená studie [34], testující EMG pro rozpoznání gest „palec nahoru“ a „palec dolů“ byla velmi úspěšná se slušnou přesností, konkrétně při užití kombinované učící metody náhodný les byla přesnost 91.3 %, nekombinované metody podpůrných vektorů 87.14% a při použití obou metod v kombinaci dokonce 91.71 %.

Kapitola 3

Návrh

Tato kapitola se bude zabývat návrhem systému gest na základě výsledků předešlé analýzy. Jako první budou stanoveny funkční i nefunkční požadavky, dále budou navrhnuty jednotlivé prvky systému, a nakonec bude představen a důkladně popsán konceptuální model systému gest v UML.

3.1 Stanovení požadavků

V této sekci budou stanoveny funkční i nefunkční požadavky, které budou vycházet ze zadání práce a provedené analýzy a definují, co vše by měl výsledný systém umět a nabízet za funkcionality.

3.1.1 Funkční požadavky

1. **Jednoduchost vytváření, přidávání a mazání gest**
   Tento požadavek vychází přímo ze zadání, které říká, že uživatel by měl mít možnost lehkého vytváření, mazání a přidávání gest do systému tak, aby ho mohl využít i laik bez větších programátorských schopností.

2. **Lehkost napojení gest na funkce aplikace**
   Požadavek vycházející ze zadání. Je důležité, aby uživatel mohl napojovat nově vytvořená gesta na funkcionality aplikace a stejně tak měnit funkcionality již existujících gest dle libosti.

3. **Zpětná vazba při vyhodnocení gest**
   Dle analýzy v sekci 2.3.3 bylo zjištěno doporučení od výrobců analyzovaného výrobku, aby vyhodnocení gest vždy mělo nějakou zpětnou vazbu, ať už je to audiální, vizuální či haptická. Minimálně by měl systém dávat různé zpětné vazby při úspěšném vyhodnocení gesta a při neúspěchu.

4. **Kontrola při každé změně skladby gest**
   Je třeba, aby systém detekoval, pokud se uživatel pokusí přidat gesto,

které je moc podobné některému jinému, a docházelo by tak často ke špatné klasifikaci. Vzniká tak požadavek, který je již v samotném zadání, aby systém při pokusu uživatele o přidání nového gesta sám sebe otestoval a uživatele informoval o případné podobnosti a nutnosti změnit přidávané gesto.

5. **Sada výchozích gest**
   Je vhodné, aby výsledný systém gest měl pár výchozích gest pro základní ovládání systému gest a zejména jeho menu. Základní skladba gest by tedy měla obsahovat gesto pro otevření menu, navigace v menu a gesta pro potvrzení a odmítnutí.

3.1.2 Nefunkční požadavky

1. **Jednoduchost přidání systému do projektu**
   Vycházející ze zadání je požadavek na to, aby bylo možné výsledný systém přidat do libovolného projektu daného editoru, tedy zde Unreal Engine, bez větších potíží. Na zprovoznění systému by tedy mělo stačit přidat soubory systému a provést jednoduché prvotní nastavení.

2. **Menší počet gest**
   Z analýzy provedené v sekci 2.3.3 bylo zjištěno, že je důležité, aby systém nebyl přehlcen gesty, jelikož jejich rozpoznání je časově náročná operace jejíž komplexnost roste s počtem používaných gest. Je tedy vhodné, aby systém měl v základu pouze nejnutnější gesta pro jeho funkčnost a aby uživatelé dbali na tento požadavek a nevyužitá gesta mazali.

3. **Jedno tlačítko**
   Z analýzy VR hardware v sekci 2.2.7 se došlo k závěru, že pro splnění univerzálnosti výsledného systému, a tedy jeho použitelnosti na většině dostupného VR hardware, je nutné, aby využíval maximálně jedno tlačítko pro svoje fungování, jelikož zejména u VR pro mobilní telefony není možné se spolehnout na existenci více vstupních prvků.

4. **Maximální přesnost**
   Je zřejmé, že systém by měl gesta snímat s co největší přesností, pro co je naprosto nutné, aby snímání dat pro rozpoznání gesta bylo provedeno v co nejmenším možném časovém rozestupu, tedy při každé aktualizaci stavu hry.

5. **Variabilní počet sledovaných prvků**
   Ze závěru analýzy VR hardware v sekci 2.2.7 vyplývá, že polohová data je možné získat vždy alespoň z jednoho sledovaného prvku, ale pro podporu snímání gest pomocí například dvou párových ovladačů nebo všech prstů (při užití hardware podporující sledování prstů) je nutné, aby výsledný

systém podporoval snímání gest z variabilního počtu sledovaných prvků a byl tak co nejuniverzálnější.

6. **Data ve formě kostry modelu**
   Při analýze různých přístupů pro rozpoznání gest se jako nejvhodnější jeví přístup analyzující kostru modelu, popsaný v sekci 2.4.2, který pracuje s daty typu velikost segmentů a úhly mezi nimi, jelikož práce s plnohodnotnými 3D modely je časově velmi náročná a možnost využít vzhled nebo EMG zde nepřipadá v úvahu.

7. **Perzistentní uložení gest**
   Je jistě vhodné, aby se všechny změny a nově vytvořená gesta perzistentně uložila mezi jednotlivými spuštěními aplikace se systémem gest. Je tedy třeba, aby docházelo k ukládání současné skladby gest a jejich konkrétních dat, a to navíc zvlášť vzhledem k počtu použitých sledovaných prvků, aby nedošlo k pokusu o rozpoznání gesta určeného pro použití s jedním prvkem vůči referenci určené pro prvků více.

3.2 Návrh na splnění požadavků

Ke stanoveným požadavkům bude v této sekci navrhnuto, jak dané požadavky naplnit, a jak jich docílit již přímo ve vybraném editoru a vzhledem k cílovému hardware.

3.2.1 Funkční požadavky

- **FP1: Jednoduchost vytváření, přidávání a mazání gest**
  Pro co největší komfort uživatele při tvorbě nových gest je vhodné implementovat administrátorské menu, kterým by uživatel s adekvátními právy mohl přímo z aplikace inicializovat nahrávání gest, gesto nahrát i uložit, pokud projde automatickou kontrolou. Toto menu by mělo být vytvořeno jakožto Widget v Unreal Engine, což umožní jeho snadné umístění uživatelem na vybrané místo, ať už to je například nad rukou nebo před kamerou.

- **FP2: Lehkost napojení gest na funkce aplikace**
  Systém gest by měl být implementovaný tak, aby šel lehce napojit na funkce aplikace, tedy jednou možností je, aby při použití Blueprints umožňoval uživateli spojit adekvátní výstupní funkci systému s libovolnou funkcí uživatele. Druhou možností je implementovat napojování funkcí přímo do zmíněného administrátorského menu, ale to by stále vyžadovalo, aby nejprve uživatel v Blueprints napojil svoje funkce do systému gest, ovšem funkcionalita by tak šla měnit i během běhu aplikace.

- **FP3: Zpětná vazba při vyhodnocení gest**
  Většina ovladačů, a to i ty, které nejsou určené pro VR, má schopnost vibrovat, a proto se jako primární zpětná vazba jeví nejvhodnější haptická, ovšem jelikož systém gest má být možné používat i bez ovladačů, tak by sekundární zpětná vazba měla být vizuální, jelikož obrazovka jistě přítomna je, a naopak zvukový výstup garantovaný není. Pro haptickou zpětnou vazbu se jeví vhodná jedna delší vibrace pro úspěch a dvě kratší vibrace pro neúspěch.

- **FP4: Kontrola při každé změně skladby gest**
  Je třeba zamezit existenci dvou velmi podobných gest, a toho je možné docílit pouze kontrolou při každém pokusu o přidání nového gesta. Je tedy nutné vytvořit testovací rozhraní, které nové gesto otestuje vůči již existujícím, a pokud bude moc podobné, zamezí jeho přidání. Testování může probíhat tak, že budou otestována všechna existující gesta vůči novému a pokud bude některé vracet vysokou hodnotu podobnosti, která překročí daný limit, tak bude nové gesto zamítnuto.

- **FP5: Sada výchozích gest**
  Skladbu výchozích gest bude nutné buď se systémem gest distribuovat jakožto externí soubor, ze kterého dojde k načtení potřebných dat, nebo data výchozích gest zakomponovat přímo do funkce systému gest, která načte výchozí data jinak než z externího souboru.

3.2.2 Nefunkční požadavky

1. **NP1: Jednoduchost přidání systému do projektu**
   Systém gest by mělo být možné zavést do projektu bez větších potíží, což by v Unreal Engine neměl být problém, jelikož je možné externí soubory přímo přetáhnout do projektu, ale kromě zavedení souborů je třeba, aby byla co nejlehčí i samotná prvotní instalace systému gest pro snadné užití méně zkušenými uživateli enginu. Jediná věc, kterou by měl uživatel při instalaci udělat, je rozmístit objekty pro snímání a menu na adekvátní místa a napojit je v Blueprints na objekt zastřešující funkčnost systému.

2. **NP2: Menší počet gest**
   Jelikož čas výpočtu pro určení gesta není závislý čistě na počtu gest, ale na jejich (ne)podobnosti a jiných faktorech, není vhodné zavést pevný limit na maximální počet gest. Místo toho by testovací rozhraní, které bude sloužit pro kontrolu podobnosti gest, mohlo kontrolovat i čas strávený výpočtem a na něj zavést limit. Dále by měl systém gest být distribuovaný pouze s minimálním počtem základních gest, nutných pro navigaci v menu.

3. **NP3: Jedno tlačítko**
   Systém je možné navrhnout tak, aby snímání gest bylo inicializováno

kliknutím tlačítka a ukončení snímání by nastalo při opětovném stisku stejného tlačítka, nebo je možné snímání gest provádět při držení daného tlačítka a ukončit po jeho puštění. Po ukončení snímání proběhne rozpoznání gesta a adekvátní akce dle gesta. Jelikož bude možné pomocí gest menu otevřít a navigovat v něm, stačí jedno tlačítko pro plné využití systému gest i bez dalších vstupních prvků.

4. **NP4: Maximální přesnost**
   Pro maximální přesnost při snímání gest je nutné nedělat žádné zbytečné výpočty, aby každá aktualizace herního stavu proběhla v co nejkratším čase a dále je třeba využít event EventTick, který se v Unreal Engine nachází v každém Blueprintu a je zavolán při každé aktualizaci hry. Výsledný systém by tedy měl při každém zavolání EventTick uložit data o sledovaném prvku a výpočty pro rozpoznání začít až snímání skončí. EventTick také vrací čas uběhlý od poslední aktualizace hry.

5. **NP5: Variabilní počet sledovaných prvků**
   Výsledný systém je vhodné rozdělit na více objektů, každý vykonávající jinou funkci, kde jeden typ objektu bude sloužit pro získání polohových dat pro zaznamenání gesta. Těchto objektů se bude dát na završující objekt napojit jeden až nekonečno, kde jeden stačí pro sledování gest tvořené headsetem nebo i klasickým ovladačem. Pro snímání například jednotlivých prstů bude možné umístit tyto objekty na jednotlivé prsty v modelu ruky, ať už jeden nebo třeba tři na prst, a snímat tak data z více prvků. Stejně tak bude možné tvořit kombinace, tedy že například se jeden objekt umístí na headset a druhý na ovladač a gesta se pak budou dělat pohyby hlavy a ovladače zároveň. Základní gesta pro ovládání menu budou minimálně utvořena pro použití s jedním sledovaným prvkem.

6. **NP6: Data ve formě kostry modelu**
   Při každé aktualizaci herního stavu dojde k uložení pozice sledovaných prvků, tedy celé gesto je možné reprezentovat jako sérii bodů nebo vektorů. Z takto reprezentovaných dat je jistě možné zjistit jak úhly mezi jednotlivými vektory užitím goniometrických funkcí, tak jejich vzdálenosti (velikost vektorů/vzdálenost bodů) a tedy získat stejná data, která jsou užita při výpočtech s kostrou modelu.

7. **NP7: Perzistentní uložení gest**
   Pro splnění tohoto požadavku bude potřeba využít buď C++ práce se soubory, tedy data gest serializovat a ukládat/načítat ze souboru, nebo využít možnosti Unreal Engine pro perzistentní uložení dat, která bude pravděpodobněji vhodnější, jelikož navíc zajistí správné uložení na všech podporovaných platformách a tedy i podmínku univerzálnosti.

3.3 Návrh jednotlivých prvků

Tato sekce obsahuje návrhy jednotlivých součástí a prvků systému gest, jako je administrátorské menu, formát dat, použitý algoritmus a testovací rozhraní.

3.3.1 Administrátorské menu

Součástí systému gest bude administrátorské menu, přes které bude umožněno uživatelům s nižšími znalostmi programování systém gest ovládat, mazat a přidávat gesta a aplikaci ukončit. Systém gest bude ovšem samozřejmě možné ovládat a nastavovat čistě ve zdrojových souborech a nevyužít tak menu, které tedy bude sloužit zejména pro uživatele, jenž nechtějí nebo se nevyznají v kódu, a zvýší se tak využití a univerzálnost systému. Mělo by obsahovat tlačítka:

- **Záznam** pro zaznamenání libovolného gesta uživatelem, kterému bude přiděleno nejmenší volné ID, které uživatel následně může napojovat na libovolné akce v systému Blueprints. Po zahájení záznamu nebude vytvořené gesto vyhodnoceno a místo toho bude možné jej uložit, ale pouze pokud projde nové gesto automatickou kontrolou a nebude tedy dostatečně podobné gestu jinému, na což by systém upozornil a nahrávání přerušil.
- **Seznam gest** umožňující zobrazení seznamu všech současně aktivních gest, jejich ID i jmen a dále nabídne možnost libovolné gesto smazat, ovšem až po opětovném potvrzení, aby nedošlo k náhodnému smazání.
- **Nastavení** obsahující možnosti pro ovládání zpětné vazby, tedy jaký typ (haptický, vizuální) zpětné vazby má systém dávat a jestli vůbec nějaký, dále možnost zobrazení gesta v prostoru, zapnutí výpisu dodatečných informací o úspěšnosti rozpoznání gesta a v neposlední řadě možnost resetovat data gest na výchozí hodnoty.
- **Zavřít a Ukončit**, jenž jsou samozřejmou součástí snad každého menu aplikace, a umožnují menu zavřít či rovnou ukončit celou aplikaci. Při uzavření menu dojde také k uložení skladby gest, ke které nebude docházet ihned po změně skladby gest, aby měl uživatel možnost změny zrušit při udělání chyby.

3.3.2 Data a jejich formát

Z požadavků vyplývá, že budou data o poloze jednotlivých sledovaných prvků při snímání gesta zaznamenána každou herní aktualizaci pro maximální přesnost. Pro polohu bude užita funkce Unreal Engine, která vrací souřadnice a rotaci vůči středu dané mapy, a tedy bude nutné nejdříve data normalizovat.

V prvé řadě je třeba od každé souřadnice v souboru odečíst souřadnici první, a ze souboru souřadnic v prostoru získat soubor vektorů, který popisuje polohu sledovaného prvku oproti počátku gesta v každém momentě udávající jak směr, tak vzdálenost díky velikosti vektoru. Takto upravená data již popisují tvar gesta s počátkem v bodě [0,0,0], ale neberou v potaz rotaci, a tak stejné gesto udělané po otočení by mělo rozdílná data, a tedy je třeba ještě soubor vektorů rotovat o původní rotaci sledovaného prvku, čímž se již získají stejná data při udělání stejného gesta nehledě na rotaci osoby v prostoru.

V poslední řadě je vhodné data upravit na stejnou délku, aby porovnání samotné byla co nejjednodušší operace bez zbytečných výpočtů navíc, jelikož by se data stejně musela upravit na stejnou délku pro porovnání. Je třeba zvolit vhodnou konstantu, která nebude ani moc dlouhá ani moc krátká, jelikož by docházelo k velkým ztrátám na informacích.

Gesta, jež jsou využita systémy analyzovanými v sekci 2.3, jsou převážně krátká gesta, která je možné vykonat během zhruba jedné sekundy. Jelikož pro VR je nejpoužívanější obnovovací frekvence 90 Hz, která udává horní limit pro obnovovací frekvenci aplikace, tak tedy sekundové gesto vytvoří data až o 90 vektorech. Jako vhodná konstanta pro velikost souboru vektorů se jeví číslo 100, jelikož by bez ztráty informací mělo být možné reprezentovat většinu gest a delší gesta je možné s malou ztrátou informací zredukovat.

3.3.3 Algoritmus rozpoznání

O datech je již známo, že budou mít tvar souboru 3D vektorů o velikosti 100, kde jejich pořadí odpovídá časovému průběhu gesta (dá se na ně dívat i jako 4D vektory s dimenzí času, tedy na Minkovského prostoru), tedy například padesátý vektor udává jakým směrem a jak daleko od počátečního bodu má být sledovaný prvek zhruba v půlce gesta.

Jelikož se jedná o výpočty v normovaném vektorovém prostoru, jako nejvhodnější kandidát pro výpočet podobnosti se jeví Minkowského vzdálenost, která je definovaná jako:

$$d\left(x, y\right) = \left(\sum_{i=1}^{n} \left|x_i - y_i\right|^p\right)^{\frac{1}{p}}.$$

Při volbě p rovné 1 nebo 2 získáme Manhattanskou, resp. Euklidovskou vzdálenost, která se jeví jak nejvhodnější volba, jelikož při výpočtu je důležitá ta vzdálenost mezi jednotlivými vektory, která je nejbližší možná, a tedy bude použita Euklidovská vzdálenost při výpočtu.

Samotný výpočet bude probíhat tak, že se soubor vektorů zkoumaného gesta porovná s referencí, tedy souborem gest, a pro každý vektor referenčního gesta bude vypočítaná vzdálenost dle zvolené metriky, která se pro celé gesto nasčítá. Referenční gesto s nejmenší výslednou vzdáleností bude zvoleno jako výsledek, pokud nepřekročí maximální povolenou vzdálenost danou konstantou, jinak by byl libovolný soubor vektorů vždy klasifikován jako některé gesto z reference.

3.3.4 Testování skladby gest

Jelikož je již známá forma dat a použitý algoritmus pro rozpoznání, je možné navrhnout, jak bude probíhat kontrola skladby gest, konkrétně aby ve skladbě nebyla dvě a více velmi podobná gesta, což by vedlo k časté špatné klasifikaci. Toto testování může proběhnout v jediném okamžiku, a to mezi zaznamenáním nového gesta uživatelem a jeho přidáním do reference.

Poté, co uživatel ukončí nahrávání nového gesta, proběhne analýza tohoto potenciálně nového gesta vůči referenci a výsledkem tohoto porovnání bude celková vzdálenost od nejpodobnějšího gesta. Pokud tato vzdálenost bude nižší než daná konstanta, tak dojde k zamítnutí nového gesta a uživatel bude muset zaznamenat jiné, nebo smazat konfliktní gesto, o kterém bude informován.

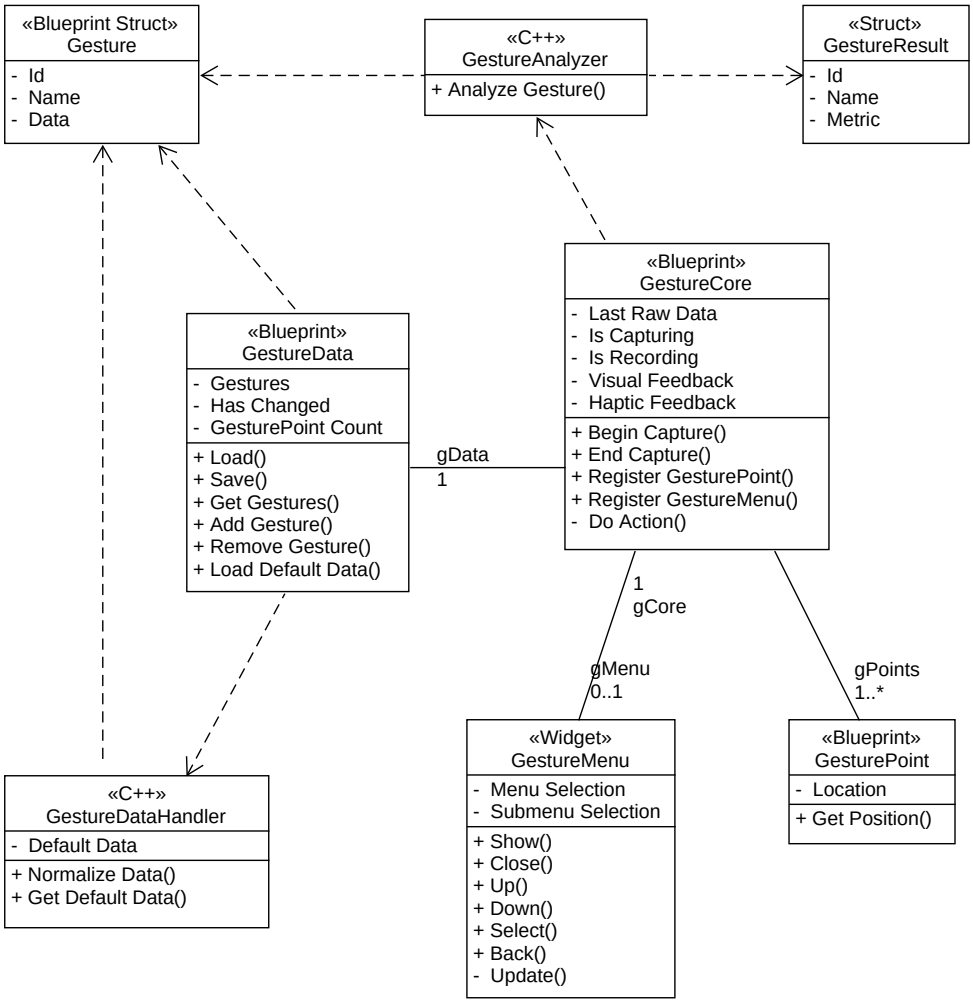
3.4 Konceptuální model

V této sekci je možné vidět konceptuální model (obrázek 3.1) systému gest v UML a následné popsání jednotlivých částí viditelných na modelu.

3.4.1 GestureCore

Jádro systému gest, které systém završuje a stará se o správnou komunikaci všech jeho komponent a také o adekvátní reakci při udělání určitého gesta. Do něj se registrují všechny volitelné komponenty a udržuje nastavení systému gest. Mezi jeho proměnnými a metodami nalezneme:

- **Last Raw Data** obsahující surová data z posledního udělaného gesta, které se dále zpracují.
- **Is Capturing**, což je boolean, který indikuje, jestli probíhá snímání gesta nebo ne.
- **Is Recording** je boolean indikující, jestli se natočené gesto má vyhodnotit nebo jestli se ho má systém pokusit uložit (pokud projde testem).
- **Visual Feedback**, tedy boolean, jež indikuje, jestli má systém poskytovat vizuální zpětnou vazbu po vyhodnocení gesta (úspěch/neúspěch).
- **Haptic Feedback** je poslední boolean indikující, zdali má systém poskytovat haptickou zpětnou vazbu po vyhodnocení gesta.
- **Begin Capture()**, což je metoda, která se zavolá při zmáčknutí tlačítka, a indikuje začátek snímání gesta, tedy nastaví a vynuluje adekvátní proměnné.
- **End Capture()** je metoda, která je zavolána při ukončení snímání gesta (například puštění tlačítka) a následně data předá dále na normalizaci a kategorizaci, po které se výstup předá do metody Do Action().



Obrázek 3.1: Konceptuální model systému gest v UML.

- **Register GesturePoint()** je metoda sloužící pro registraci jednoho a více GesturePoint do jádra, ze kterých jsou při snímání získávána polohová data gesta.
- **Register GestureMenu()** je další metoda sloužící pro registraci, ale tentokrát administrátorského menu, pomocí kterého je možné (ale ne nutné) systém ovládat.
- **Do Action()** je snad druhá nejdůležitější metoda hned po klasifikaci samotné, a určuje jaké gesto bude mít kterou odezvu. Je to jediná metoda, kterou musí uživatel aktivně upravovat (pakliže chce měnit chování

systému). Vstupem této metody bude nalezeného gesta spolu s dalšími informacemi o gestě a dle daného ID je možné na gesto napojit libovolnou funkčnost aplikace. Bude možné akce vybírat na základě jména gesta, ale upřednostňována je volba dle ID.

3.4.2 GestureData

Objekt starající se o zacházení s daty gest, tedy jejich perzistentní uložení a načtení, přidání nového gesta, smazání gesta a resetování dat na data výchozí. Mezi jeho proměnné a funkce bude patřit:

- **Gestures**, což je pole obsahující aktuálně používaný referenční soubor gest.
- **Has Changed** je boolean proměnná indikující, jestli došlo ke změně souboru gest.
- **GesturePoint Count** udává počet sledovaných prvků.
- **Load()** je funkce pro načtení dat z uloženého souboru. Pokud načtení selže, bude načten výchozí soubor gest.
- **Save()** je funkce starající se o uložení aktuálního používaného souboru gest na perzistentní úložiště. Uložení proběhne pouze pokud Has Changed je pravda.
- **Get Gestures()**, tedy funkce, která vrátí aktuální referenční soubor gest.
- **Add Gesture()** je funkce pro přidání nového gesta do souboru gest, která při úspěchu upraví proměnnou Has Changed.
- **Remove Gesture()** slouží pro smazání gesta z referenčního souboru a při úspěchu také upraví proměnnou Has Changed.
- **Load Default Data()** je funkce zabývající se načtením adekvátních výchozích dat z GestureDataHandler. Bude volána buď při selhání načtení uložených dat nebo pokud uživatel vyresetuje data v menu.

3.4.3 GestureMenu

Administrátorské menu, pomocí kterého bude umožněno uživateli gesta přímo z aplikace mazat, tvořit a obecně upravovat chování systému. Není nutnou součástí systému, do jádra se registruje pomocí adekvátní funkce. Pokud ho uživatel bude chtít využít, stačí ho umístit na libovolné místo a zaregistrovat. Funkce by neměly nic dělat, pokud bude menu schované, a jejich výčet spolu s proměnnými je následující:

- **Menu** a **Submenu Selection** jsou celočíselné proměnné udávající pozici navigace v menu, tedy v hlavním menu a v případných vedlejších menu.
- **Show()** a **Close()** jsou funkce pro zobrazení a zavření menu, kde Show() menu vždy otevře na úvodním menu a Close() vyresetuje proměnné.
- **Up()** a **Down()** jsou funkce pro vertikální pohyb v menu, upravují proměnné výběru.
- **Select()** je funkce pro výběr, která dle aktuálních hodnot proměnných vykoná adekvátní odezvu.
- **Back()** je funkce, kterou se indikuje volba vrátit se v menu zpět. Pokud se menu již nachází v hlavním menu, tak se zavolá funkce Close().
- **Update()** je zavolána automaticky po každé změně v menu, tedy po zavolání všech předešlých funkcí vyjma Close(), a stará se o korektní vykreslení menu dle proměnných.

3.4.4 GestureDataHandler

C++ třída starající se o přípravu dat a také načtení výchozího souboru gest, kterým je umožněno ovládat administrátorské menu, a skládá se z:

- **Default Data** jsou data výchozího souboru gest v blíže neurčené formě.
- **Normalize Data()** je funkce, která data připraví a upraví pro použití analyzátorem. Surová data získaná z GesturePoints je třeba upravit z bodů v prostoru na vektory s počátkem v prvním bodě, rotovat je dle rotace uživatele a upravit jejich délku na stanovenou konstantu.
- **Get Default Data()** je funkce, která výchozí data uvede do formátu souboru struktur Gesture a vrátí je.

3.4.5 GestureAnalyzer

C++ třída s jediným, ovšem nejdůležitějším úkolem, a to rozpoznat dané gesto. Pro vysokou časovou náročnost této operace je třída psaná v C++, kde je možné při správném programování dosáhnout lepších výkonnostních výsledků, ale výsledek rozpoznání předá zpět do Blueprints, tedy bude obsahovat pouze jednu funkci:

- **Analyze Gesture()**, která dostane jak referenční soubor gest, tak data zkoumaného gesta a na nich provede analýzu. Funkce bude počítat celkovou vzdálenost zkoumaného gesta od referenčních gest za využití Euklidovské vzdálenosti, jak bylo určeno v sekci 3.3.3.

3.4.6 GesturePoint

Bude sloužit k získání polohových dat sledovaných prvků a do GestureCore se zaregistruje pomocí adekvátní funkce. Alespoň jeden GesturePoint musí být přidaný pro funkci systému. Například je možné GesturePoint připojit na prst modelu ruky a tím umožnit dělání gest daným prstem. GesturePoint bude obsahovat:

- **Location**, tedy poziční data vůči počátku mapy, která jsou aktualizována editorem.
- **Get Position()**, což je funkce pro získání pozičních dat z jednoho bodu.

3.4.7 Gesture a GestureResult

Pomocné struktury pro práci s gesty. Struktura Gesture bude Blueprints čitelná, tedy bude přístupna jak v C++ kódu, tak v systému Blueprints a popisuje jedno gesto, tedy jejími složkami jsou:

- **ID** unikátní mezi ostatními gesty,
- **Name** pro snazší poznání gesta uživatelem, nemá vliv na algoritmus,
- **Data** uchovávající soubor vektorů popisující dané gesto.

Struktura GestureResult bude sloužit pro zapouzdření výsledku algoritmu pro rozpoznání, a tedy bude obsahovat:

- **ID** nalezeného gesta, případně 0 při nenalezení adekvátního gesta,
- **Name** nalezeného gesta pro zobrazení uživateli,
- **Metric** udávající hodnotu metriky, tedy celkové vzdálenosti gesta od reference.

Kapitola 4

Realizace

Tato kapitola se bude zabývat implementací systému gest dle návrhu stanoveném v kapitole předešlé. Jako vývojové prostředí bude využit Unreal Engine verze 4.25.4, který nativně využívá Visual Studio pro psaní C++ kódu, tedy konkrétně na psaní C++ kódu bude využito Visual Studio 2019. Implementace systému se bude držet stanoveného návrhu, a případné odchylky od návrhu budou v této kapitole popsány a zdůvodněny.

V první části dojde k představení využitých prvků Unreal Engine, jako jsou funkce, třídy a typy Blueprints a popis jejich užití na konkrétních částech systému. Dále zde bude popsáno, s případnými ukázkami, jak bylo dosaženo splnění jednotlivých požadavků. Všechny funkce a proměnné, které se týkají autorovi vlastní implementace, budou psány rozdílným fontem od funkcí a proměnných editoru.

V poslední části kapitoly bude představen finální model v UML, přesně reflektující implementovaný systém gest, spolu s případným popisem částí lišících se od konceptuálního modelu z obrázku 3.1.

4.1 Využité prvky editoru

V této sekci budou představeny prvky z Unreal Engine využité pro realizaci této diplomové práce spolu s popisem a konkrétními příklady užití na práci samotné.

4.1.1 Použité typy Blueprints

Blueprints, stejně jako C++ třídy, jichž jsou nadstavbou, je možné rozšiřovat pomocí dalších Blueprints jej dědící a přidávající funkčnost, která umožňuje s těmito variantami dělat různé úkony. Varianty Blueprints, které jsou použity při implementaci systému gest, jsou následující:

- **Actor** je základní objekt, které je možné vložit do světa nebo ho během chodu aplikace ve světě vytvořit. Jeho chování je definované pomocí kolekce komponent, které upravují, jak se Actor vykresluje, pohybuje atd. Dále obsahuje funkcionality pro správnou replikaci při využití sítě. Actor v systému gest je například GesturePoint nebo GestureVisual. Speciálním příkladem pro vložení jednoduchého textu na scénu je Text Render Actor, který je využit v systému gest jako řádka pro debuggování, konkrétně výpis detailních informací o klasifikaci.

- **Pawn** je Actor, kterého je možné ovládat a má schopnost dostávat vstup z ovladače. Není součástí systému gest, ale pravděpodobně právě na Pawn je nejvhodnější umístit GesturePoint v aplikacích uživatelů, jelikož často obsahuje modely ruky a podobné prvky, které uživatel ovládá.

- **Player Controller** je Actor odpovídající za ovládání Pawn, který právě používá uživatel.

- **Widget** Blueprint umožňuje rozšířit UUserWidget, což je uživatelsky rozšiřitelný UWidget, což je základní třída pro všechny zabalené Slate prvky, kde Slate UI je platformě nezávislý framework pro tvorbu uživatelského rozhraní. Slouží tedy pro tvorbu UI a je využit v systému gest pro GestureMenu.

- **Scene Component** je komponenta, kterou je možné umístit na scénu nebo připojit k jiné komponentě, ale neumí se vykreslit ani detekovat kolizi. Využívá se taky jako „dummy“ pro vytvoření odstupů v hierarchii scény.

- **SaveGame** Blueprint implementuje třídu, která slouží jako základ pro Save Game Object, což je objekt, který se využívá pro uložení stavu hry. Do nových Blueprints tohoto typu je možné vložit proměnné, do kterých je adekvátně uložen stav hry, a následně je možné instanci těchto Blueprints uložit na místo dle uvedeného jména, odkud je poté možné analogicky data získat. V této práci tuto funkcionalitu využívá GestureData.

4.1.2 Využité funkce Unreal Engine

Jelikož je jednou z hlavních podmínek maximální univerzálnost systému, většina použitých funkcí je vlastnoručně napsána, samozřejmě vyjma triviálních funkcí typu get, set, počet prvků a jiné, ale ve třech případech došlo k využití složitějších funkcí engine, konkrétně:

- **Save Game to Slot** je funkce, jejímž vstupem je ID uživatele, jméno místa pro uložení a samotný Save Game Object, tedy objekt k uložení typu představeného v předešlé sekci. Funkce uloží předaný Save Game

Object na místo dle uvedeného jména a vrátí boolean indikující úspěch či neúspěch. Engine se sám postará o perzistentní uložení dat nehledě na použitou platformu.

- **Load Game from Slot** je funkce, která na vstupu přijímá jméno místa uložení a ID uživatele a vrací Save Game Object, ať už platný nebo ne, a následujícím testem se zjistí, jestli se načtení povedlo, a pokud ano, je možné z objektu získat načtená data. Funkce Save Game to Slot i Load Game from Slot jsou využity v GestureData.

- **Play Steam VR Haptic Feedback** není samostatnou funkcí engine, ale konkrétně Steam VRInput Device Function Library, což je knihovna dostupná nativně v engine, která se stará o vytvoření haptické zpětné vazby na libovolném použitém ovladači podporovaným platformou SteamVR, která se může pyšnit obrovskou kolekcí podporovaných VR hardware. Funkce přijímá čas do startu, délku, frekvenci a amplitudu haptického efektu a její užití je možné vidět na obrázku 4.3.

- **Play Force Feedback** je funkce, která se postará o vibraci ovladačů, které přímo nejsou spojené s VR, ale dají se využít pro ovládání VR, jako jsou třeba ovladače PlayStationu nebo Xboxu. Funkce přijímá referenci na ovladač a ForceFeedbackEffect, což je asset, který popisuje křivku pro vibraci. Spolu s předešlou funkcí tak zajistí haptickou zpětnou vazbu na většině ovladačích.

4.1.3 C++ programování v Blueprints

Pro integraci C++ tříd do systému Blueprints je třeba nové C++ třídy odvodit z UBlueprintFunctionLibrary, což je základní třída pro jakékoliv funkční knihovny viditelné v Blueprints. Všechny metody v této odvozené třídě musí být statické. Jednotlivé části třídy ovšem musí být doplněny speciálními makry pro správné spojení s Blueprints, konkrétně [55]:

- **UCLASS()** je makro, které se uvádí před deklarací všech tříd, které dědí ze základního objektu UObject, tedy i UBlurprintFunctionLibrary, a dále se k tomuto makru váže makro **GENERATED_BODY()**, které se umisťuje do těla funkce.

- **UFUNCTION()** je makro předcházející funkci, která je rozeznána speciálním systémem Unreal Engine, a přijímá různé specifikace (v závorkách), které určují kde a jak bude funkce použita, například:

  - **BlueprintAuthorityOnly**
    Funkce se v Blueprints provede pouze na stroji s autoritou (server, single-player).

  - **BlueprintCallable**
    Funkci je možné zavolat z Blueprints.
  - **BlueprintCosmetic**
    Funkce je pouze kosmetická a nebude spuštěna na serverech.
  - **CallInEditor**
    Funkci je možné zavolat přímo z editoru.
  - **Client**
    Funkce je provedena pouze na klientovi, který vlastní objekt, na kterém je zavolána.
  - **Server**
    Funkce je provedena pouze na serveru.
  - **Exec**
    Funkci je možné zavolat z konzole v aplikaci.
  - **Category = "..."**
    Určuje kategorii funkce, pouze kosmetické, viditelné v Blueprints.

  Aby tedy C++ funkce byla viditelná a proveditelná v Blueprints, musí mít danou specifikaci. Funkce v této práci mají specifikaci BlueprintCallable.

- **UPROPERTY()** je makro předcházející deklaraci proměnné, a stejně jako u funkcí je možné v něm určit vlastnosti proměnné, jako je:

  - **BlueprintReadWrite**
    Proměnnou jde číst i měnit v Blueprints.
  - **BlueprintReadOnly**
    Proměnnou jde v Blueprints pouze číst ale ne měnit.
  - **BlueprintAssignable**
    Podporované proměnné je možné přiřadit v Blueprints (pouze multicast delegáti).
  - **EditAnywhere**
    Proměnná je měnitelná odkudkoliv, i v editoru nebo na instancích.
  - **Replicated**
    Jestli se má proměnná replikovat přes síť.
  - **VisibleAnywhere**
    Jestli je proměnná všude viditelná, ale ne upravitelná.
  - **Localized**
    Určuje že proměnná má lokalizované hodnoty (primárně se používá pro String).

  A dále je možné nastavit specifikace pro metadata proměnné, například:

  - **DisplayName="..."**
    Jméno proměnné pro zobrazení místo automaticky vygenerovaného.
  - **AllowPreserveRatio**
    Používá se pro FVector a určuje, jestli je možné v editoru proměnnou měnit se zachováním poměru.
  - **GetByRef**
    Zaručí, že funkce Get v Blueprints navrátí konstantní referenci místo kopie hodnoty.
  - **NoGetter**
    Zaručí, že nedojde k vygenerování funkce Get v Blueprints.

  Proměnné z této práce tedy minimálně musí mít vlastnost BlueprintReadWrite, případně BlueprintReadOnly a dále mají specifikované kategorie.

- **USTRUCT()** je makro předcházející deklaraci struktury a má pouze tři možné specifikace:
  - **Atomic**
    Indikuje, že struktura by vždy měla být serializována jako jeden celek.
  - **BlueprintType**
    Vystaví strukturu pro užití v Blueprints.
  - **NoExport**
    Nevygeneruje se žádný automatický kód.

  Dále má široké možnosti pro specifikaci metadat, například:

  - **BlueprintSpawnableComponent**
    Pokud se jedná o komponentu je možné ji vytvořit v Blueprints.
  - **ChildCannotTick**
    Dědičné třídy se nemůžou aktualizovat s herním stavem.
  - **BlueprintThreadSafe**
    Je volatelná v animačních Blueprints i z neherních vláken.
  - **DisplayName="..."**
    Upraví jméno viditelné v Blueprints.

  V této práci je jedna struktura viditelná v Blueprints, a to Gesture, která tedy musí mít specifikaci BlueprintType a stejně jako u tříd je třeba do těla struktury umístit makro **GENERATED_BODY()**. Následuje ukázka kódu obsahující definici struktury Gesture obsahující zde představené konstrukty.

```
USTRUCT(BlueprintType)
struct FGesture
{
  GENERATED_BODY()
public:
  FGesture(int ID, FString NAME, TArray<FVector> DATA) {
    id = ID;
    name = NAME;
    data = DATA;
  }
  FGesture() {}
  UPROPERTY(BlueprintReadWrite, Category = "id")
    int id;
  UPROPERTY(BlueprintReadWrite, Category = "name")
    FString name;
  UPROPERTY(BlueprintReadWrite, Category = "data")
    TArray<FVector> data;
};
```

4.2 Implementace požadavků

V této sekci bude ke každému ze stanovených funkčních i nefunkčních požadavků popsáno, jak bylo daných požadavků docíleno při implementaci systému gest a tam, kde je to vhodné, budou zobrazeny ukázky z implementace.

4.2.1 FP1: Jednoduchost vytváření, přidávání a mazání gest

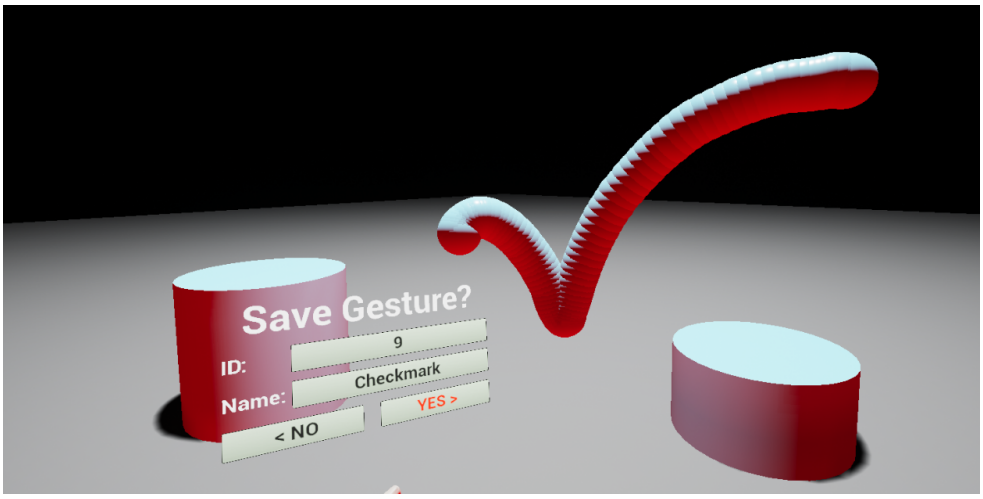
Jednoduchost z tohoto požadavku řeší administrátorské menu, které méně zkušeným uživatelům umožňuje gesta přímo z aplikace vytvářet, mazat a přidávat, což je stejně tak možné i přímo v kódu, ale to již vyžaduje jisté programátorské znalosti a nelze považovat za jednoduché. Systém gest je distribuován jak s (volitelným) administrátorským menu, tak se sadou výchozích gest umožňující dané menu plně ovládat a samotná instalace systému gest se skládá z přetažení zdrojových souborů do projektu, umístění tří objektů (jádro, sledovaný bod a menu) na vybraná místa na scéně a jejich vzájemné propojení za využití dostupných funkcí. Po této jednoduché prvotní instalaci je systém připraven a vytváření, přidávání a mazání gest je možné dosáhnout následovně:

- **Vytvořit a přidat gesto** je možné po otevření menu a zvolení možnosti Record (nahrát), po čemž menu dočasně zmizí a uživatel má možnost udělat libovolné gesto, jehož tvar ze scény po dokončení nezmizí a objeví se v menu informace o uděleném gestu, jako je jeho ID a jméno, a uživatel má následně možnost si vzniklé gesto prohlédnout a pokud bude

i spokojený, tak nové gesto v menu potvrdit a tím ho uložit do mezipaměti (ale ještě ne na perzistentní úložiště). Tento proces je možné vidět na obrázku 4.1. Pokud ovšem uživatel vytvoří nové gesto, které bude dostatečně podobné již existujícímu gestu, tak dojde k zamítnutí systémem a uživatel bude vyzván gesto nahrát znovu a informován o konfliktním gestě.

- **Smazat gesto** je také možné skrz administrátorské menu, a to tak, že uživatel přejde do seznamu gest, kde se mu zobrazí všechna aktuální gesta ve formě ID a jména, a má možnost libovolné gesto vybrat, po čemž se mu v menu zobrazí informace o daném gestě a možnost gesto smazat. Uživatel se v tomto okamžiku ještě může vrátit a zrušit tak mazání gesta. Pokud smazání uživatel odsouhlasí, dojde ke smazání gesta z mezipaměti, ale ke smazání gesta i z perzistentního úložiště dojde až při zvolení volby v menu pro uložení.

Lze tedy vidět, že vytváření, přidávání a mazání gest je možné při užití administrátorského menu (jehož instalace je otázkou pár minut) dosáhnout pomocí udělání tří gest (otevřít menu, vybrat nahrávání, potvrdit nahrávání) a tedy je jistě možné jej považovat za jednoduché a požadavek za splněný.

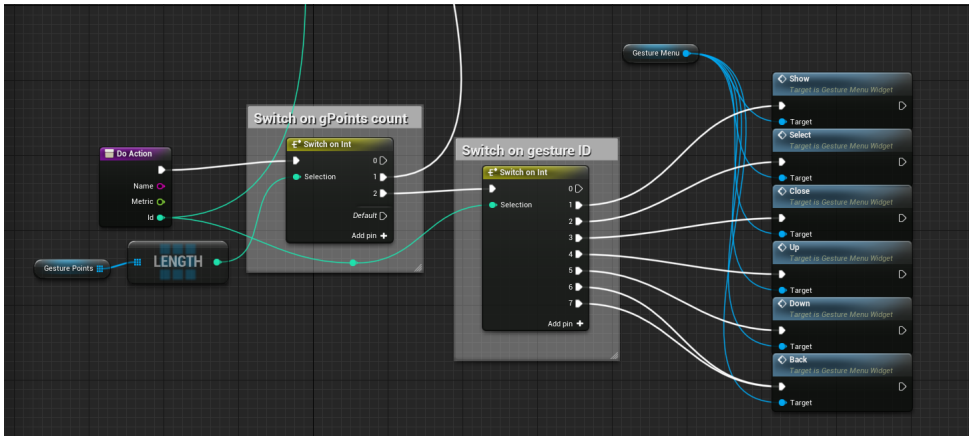


Obrázek 4.1: Tvorba nového gesta (fajfky).

4.2.2 FP2: Lehkost napojení gest na funkce aplikace

V konceptuálním modelu je možné vidět funkci DoAction() (viz sekce 3.4.1), která je avizovaná jako druhá nejdůležitější funkce celého systému, a právě zde má uživatel možnost měnit, přidávat a ubírat odpovědi aplikace na daná

gesta. Funkce na vstupu přijímá ID a jméno nalezeného gesta spolu s jeho metrikou, tedy celkovou euklidovskou vzdáleností. Jediné, co funkce obsahuje, jsou dva switche, první na počet použitých sledovaných prvků a druhý na ID gest samotných, takže je možné určovat reakce nejen dle ID gesta, ale i dle počtu současně použitých sledovaných prvků. Funkce ve výchozím stavu obsahuje napojení výchozích gest na ovládání administrátorského menu, jak je možné vidět v ukázce na obrázku 4.2, kde je stejně tak možné vidět, že jediné, co musí uživatel pro změnu reakce aplikace na dané gesta udělat je, že napojí výstup od vybraného ID k libovolně zvolené funkci aplikace, což lze jistě považovat za lehké a požadavek tak za splněný.

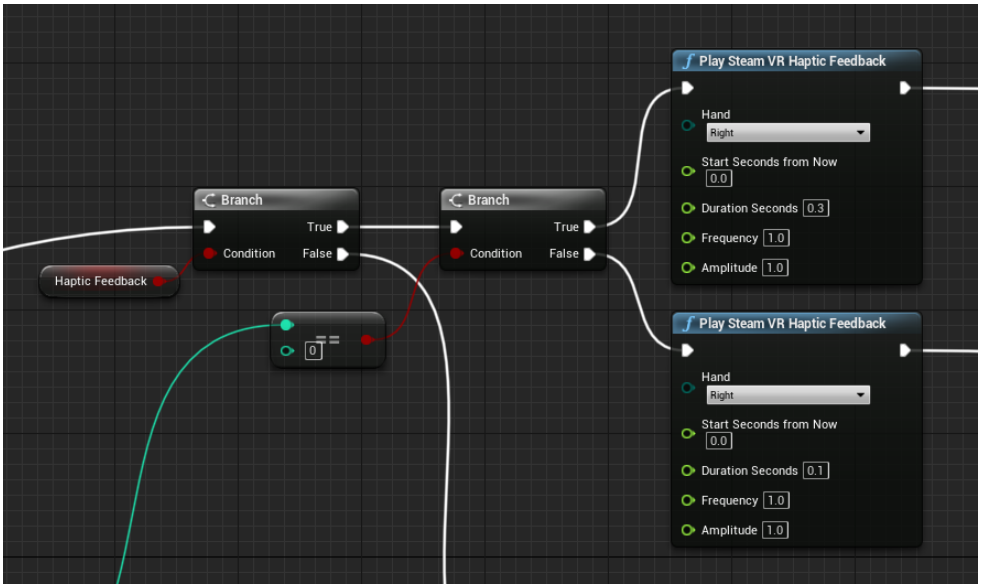


Obrázek 4.2: Funkce `doAction()` z GestureCore.

4.2.3 FP3: Zpětná vazba při vyhodnocení gest

V návrhu bylo stanoveno, že výsledný systém bude mít možnost poskytovat zpětnou vazbu haptickou a vizuální při klasifikování gesta, indikující úspěch při nalezení nějakého gesta a neúspěch při nenalezení, resp. při nalezení gesta s moc velkou celkovou vzdáleností od reference. Uživatel má možnost v administrátorském menu měnit kterou, a jestli vůbec některou, zpětnou vazbu má systém poskytovat, což adekvátně nastavuje boolean proměnné v jádru. Na obrázku 4.3 je možné vidět zpracování haptické odezvy v Blueprints, která dle hodnot proměnné `hapticFeedback` a ID výsledného gesta (kde 0 indikuje neúspěšnou klasifikaci) vyvolá adekvátní haptickou odezvu.

Pro samotný haptický efekt jsou využity funkce Unreal Engine, představené v sekci 4.1.2, které zajistí adekvátní odezvu na všech podporovaných ovladačích. Vyhodnocení pro vizuální odezvu probíhá obdobně, a proto je jeho ukázka vynechána. Systém tedy obsahuje možnosti pro jak vizuální, tak haptickou odezvu, a tedy je tento požadavek splněn.



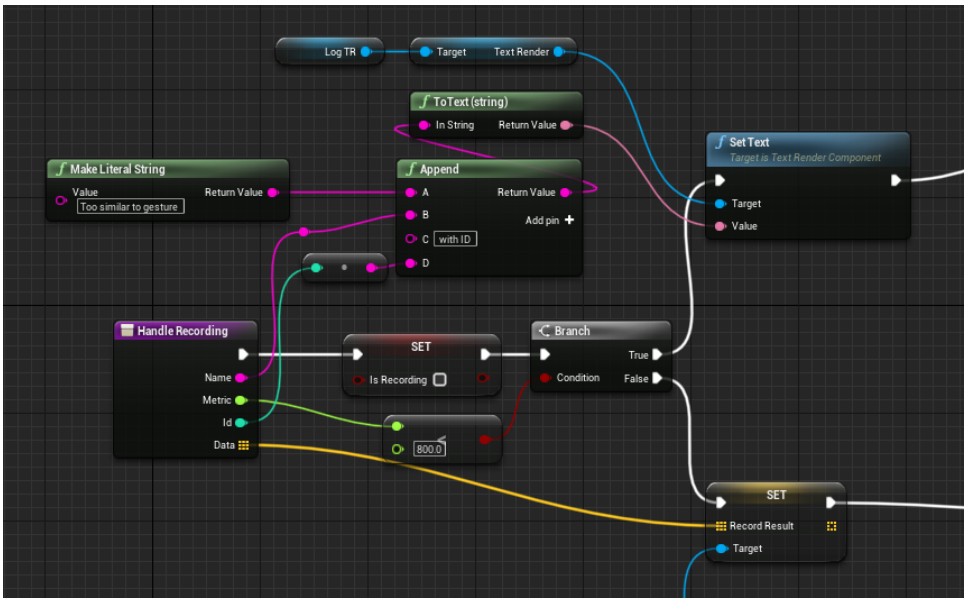
Obrázek 4.3: Zpracování haptické zpětné vazby.

4.2.4 FP4: Kontrola při každé změně skladby gest

Je nutné udržovat referenční soubor gest, který neobsahuje žádná moc podobná gesta, aby nedocházelo ke špatné klasifikaci, a proto je potřeba provést kontrolu na podobnost při každém pokusu o přidání nového gesta uživatelem. To je zajištěno daným sledem volání funkcí, které funguje následovně:

1. Uživatel inicializuje nahrávání (např. v menu), čímž se nastaví proměnná `isRecording`.
2. Uživatel nahraje potenciální nové gesto, což uloží surová data gesta.
3. Data jsou zpracována stejně jako bez nahrávání, tedy jsou nejdříve upravena, normalizována a následně klasifikována.
4. Po klasifikaci ovšem při nahrávání není předán výsledek klasifikace do funkce `doAction()`, ale do pomocné funkce `handleRecording()`, viditelné na obrázku 4.4.
5. Pokud je celková metrika, tedy vzdálenost, menší než daný limit, dojde k informování uživatele o konfliktním gestě a přerušení nahrávání.
6. Pokud gesto není dostatečně podobné, zobrazí se menu s vyzváním pro potvrzení nebo odmítnutí gesta, které lze vidět v prostoru.
7. Dle volby uživatele dojde k uložení gesta do mezipaměti nebo jeho smazaní.

V postupu lze vidět, že každé gesto projde kontrolou a pokud nesplní zadané požadavky dojde k jeho automatickému zamítnutí bez možnosti uživatele tento fakt obejít. Při nahrávání nového gesta se výsledky klasifikace místo do funce `doAction()` předají do `handleRecording()`, kde dojde k buď informování uživatele o konfliktu, nebo uložení výsledku do proměnné pro následovné uložení do mezipaměti. Dochází tak ke kontrole při každém pokusu o přidání nového gesta a zaručení tak korektnosti referenční skladby gest, čímž je splněn čtvrtý funkční požadavek.



Obrázek 4.4: Začátek funkce `handleRecording()`.

4.2.5 FP5: Sada výchozích gest

Ačkoliv je možné systém gest ovládat a gesta přidávat přímo v kódu, pro jednoduchost a umožnění užití systému méně zkušenými uživateli je součástí systému administrátorské menu, které je ovšem třeba nějak ovládat. Proto je součástí systému sada výchozích gest, sloužících primárně pro ovládání menu, které je možné použít při ovládání systému jedním či dvěma sledovanými prvky. Z nefunkčního požadavku NP2 dále plyne podmínka na co nejmenší možný počet gest. Dále je třeba vzít v potaz, že systém gest má být možné požívat na většině VR headsetů, tedy i včetně například mobilní VR, kde někdy je jediným sledovaným prvkem headset samotný, a tedy gesta je tam možné dělat pouze hlavou. Všechny uvedené důvody byly brány v potaz při tvorbě následujících výchozích gest:

1. **Přesýpací hodiny**
   Jednoduchý znak tvořený čtyřmi úsečkami tvořící dva dotýkající se trojúhelníky byl zvolen jako gesto, na které je napojené otevření administrátorského menu. Jeho použití je možné vidět na obrázku 5.2.

2. **Tah nahoru**
   Tah nahoru napojený na funkci menu `up()`.

3. **Tah dolů**
   Tah dolů napojený na funkci menu `down()`.

4. **Tah doleva**
   Tah doleva napojený na funkci menu `back()`.

5. **Tah doprava**
   Tah doprava napojený na funkci menu `select()`.

6. **Tečka**
   Gesto reprezentující tečku, ve výchozím stavu není napojené, slouží pro klasifikaci náhodných neplánovaných stisků tlačítka jakožto omyl.

7. **Přijmutí hlavou**
   Gesto reprezentující kývnutí hlavou pro užití například s mobilní VR bez ovladačů, napojené na funkci menu `select()`.

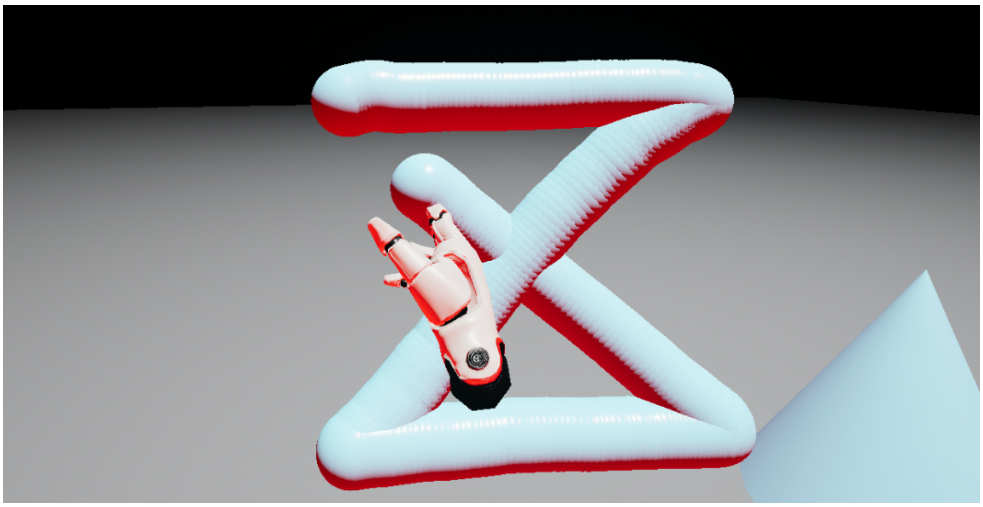
8. **Odmítnutí hlavou**
   Gesto reprezentující zavrtění hlavy pro užití bez přítomnosti ovladače, napojené na funkci menu `back()`.

Gesta tahů a pro otevření menu jde dělat i při ovládání hlavou, ale sada byla doplněna o dvě gesta pro hlavu speciálně z toho důvodu, aby poskytla přirozenější interakci, tedy doslova kývnutí pro potvrzení a zavrtění hlavy pro odmítnutí. Systém tedy bude distribuován s výchozí sadou obsahujících 8 zde představených gest potřebných pro ovládání administrátorského menu, čímž je požadavek splněn.

4.2.6 NP1: Jednoduchost přidání systému do projektu

Systém gest se skládá pouze ze zdrojových souborů Blueprints a C++, které Unreal Engine plně podporuje, takže je možné je do projektu přidat dvěma jednoduchými způsoby:

1. **Přesun** souborů systému gest přímo do souborů (složka Content) projektu cílového v libovolném systémovém prohlížeči. Není třeba spouštět žádné skripty pro import a podobné, po načtení projektu engine soubory detekuje a sám je přidá mezi dostupné soubory v projektu.



Obrázek 4.5: Použití gesta pro otevření menu s jedním GesturePoint.

2. **Migrace** souborů systému gest z jiného projektu do projektu cílového. K tomu Unreal Engine poskytuje funkčnost migrace assetů, která nakopíruje zvolený soubor spolu s veškerými jeho závislými sobory, jako může být například textura, do zvoleného projektu.

Jelikož je možné složku se soubory systému gest nakopírovat do složky libovolného projektu a dojde tak k úspěšnému přidání systému do projektu, je tento proces možné považovat za jednoduchý a požadavek tak za splněný.

4.2.7 NP2: Menší počet gest

V kapitole Návrh byly k tomuto požadavku navrhnuty dvě podmínky pro splnění, a to, aby systém gest měřil čas strávený výpočtem a při překročení limitu znemožnil přidávat gesta další a aby byl systém distribuován s co nejmenší možnou výchozí skladbou gest.

Při implementaci časového limitu ovšem docházelo k nekonzistentnímu chování, kde klasifikace trvala dlouho a vzápětí stejná klasifikace byla v mžiku hotova, jelikož čas nezávisí jen na velikosti referenčního souboru, ale i na konkrétních datech gesta pro porovnání, na hardware použitého systému a zejména na tom, co daný systém ve stejný okamžik počítá na pozadí, a z těchto důvodů byl časový limit zrušen. Místo toho budou uživatelé systému silně avizování, aby hlídali čas strávený klasifikací a při potížích referenční soubor gest zredukovali.

Co se druhé podmínky týče, jak je popsáno u funkčního požadavku FP5, systém gest bude distribuovaný pouze s osmi gesty potřebnými na ovládání administrátorského menu, které zahrnují všechny potřebné úkony plus dvě gesta

pro ovládání pomocí hlavy. Systém gest bude tedy distribuován s nejmenší možnou sadou výchozích gest a uživatel je avizován oproti nesmyslnému přidávání množství nepotřebných gest, tedy požadavek je po částečné změně oproti návrhu splněn.

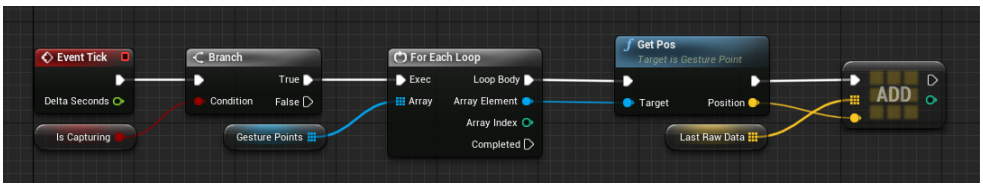
4.2.8 NP3: Jedno tlačítko

Pro plnou funkčnost systému gest stačí napojit funkce `beginCapture()` a `endCapture()` na vstupní prvek, nejpravděpodobněji právě tlačítko, a je jen na uživateli, jestli systém bude zaznamenávat gesto při držení tlačítka, nebo mezi dvěma zmáčknutími tlačítka, tedy tak jak tak je jisto jistě možné, ba i doporučené systém gest ovládat jedním tlačítkem, čímž se zajistí funkčnost i na hardware, který má jen jedno tlačítko, jako jsou vybrané mobilní VR, a tím je požadavek splněn.

4.2.9 NP4: Maximální přesnost

Pro dodržení maximální přesnosti při snímání gesta je nutné dodržet dvě věci, a to:

1. Využít **Event Tick** v GestureCore, což je event, který je zavolaný při každé aktualizaci herního stavu, tedy nejmenší možné časové prodlevy.
2. V onom eventu **nedělat žádné výpočty**, které by mohly způsobit případné zpoždění a záseky, a místo toho pouze data uložit v surové formě a výpočty započít až po skončení snímání.



Obrázek 4.6: Event Tick v GestureCore.

Tyto dvě podmínky byly dodrženy, jak je možné vidět na obrázku 4.6, a veškeré výpočty započnou až po zavolání funkce `endCapture()`, která je zavolána pro ukončení nahrávání. Obě dvě stanovené podmínky pro maximální přesnost jsou splněny, a tedy stejně tak i požadavek samotný.

4.2.10 NP5: Variabilní počet sledovaných prvků

Na systém gest je kladena podmínka, aby podporoval tvorbu gest jedním a více hardware s podporou sledování pohybu, jako jsou ovladače nebo headset

samotný, čehož je docíleno pomocí GesturePoint, který je možné v Unreal Engine připojit na libovolný prvek scény, tedy například na prsty u ruky nebo na hlavu samotnou a přidat jim tak schopnost dělat gesta.

Tyto body se do jádra systému registrují funkcí `registerGesturePoint()`, která se postará o uložení reference na ně a při snímání gesta dojde k uložení polohových dat každého zaregistrovaného bodu, jak je možné vidět na obrázku 4.6, a tato data jsou po skončení snímání dále zpracována.

Je důležité brát v potaz pořadí při registraci bodů a po vytvoření gest pro danou kombinaci bodů jejich pořadí již neměnit. Do systému je třeba registrovat minimálně jeden bod pro funkčnost, ale horní limit na počet bodů stanoven není, takže je systém schopný podporovat variabilní počet sledovaných prvků, čímž je požadavek splněn.

4.2.11 NP6: Data ve formě kostry modelu

Finální data popisující jedno gesto jsou ve formě pole o 100 vektorech, kde každý určuje vzdálenost a směr od počátku gesta v daném okamžiku, což nám dává informace o jednotlivých segmentech a jejich délce a úhlech s ostatními segmenty, což platí i pro data ve formě kostry modelu. Data prochází následující úpravou:

1. Data (souřadnice vůči počátku mapy) z každého GesturePoint jsou uložena do pole během snímání.

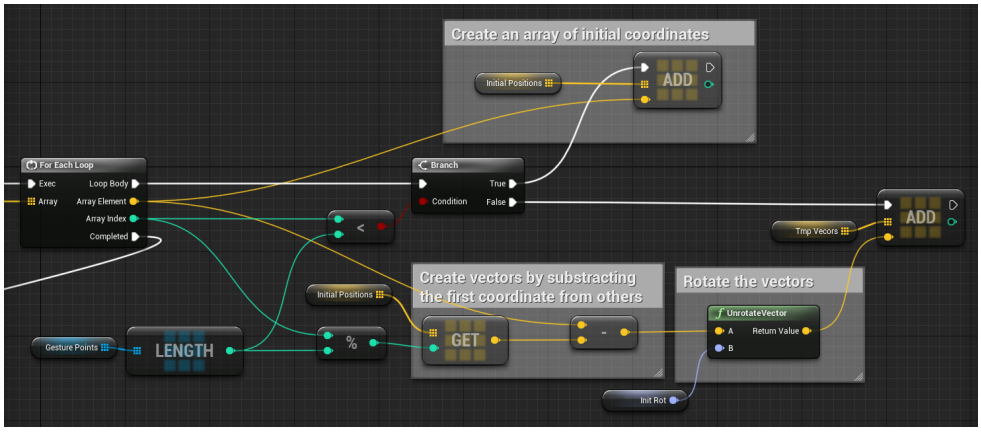
2. Po ukončení snímání jsou souřadnice převedeny na vektory tak, že se od každé odečte první souřadnice každého GesturePoint, a tak vznikne pole vektorů popisující směr a vzdálenost GesturePoint od počátku gesta v každém okamžiku. Tento proces je možné vidět na obrázku 4.7.

3. Dále je třeba data převést na stejnou délku, která byla v návrhu stanovena na 100 na jeden GesturePoint, a to co nejrovnoměrněji, o což se stará C++ funkce `normalizeData()` třídy GestureDataHandler.

Výsledná data tedy tvoří na každý použitý GesturePoint 100 vektorů na 3D prostoru, kde mezi každým vektorem je možné zjistit jak úhly, tak vzdálenosti jednotlivých segmentů, stejně jako u kostry modelu, čímž je požadavek splněn.

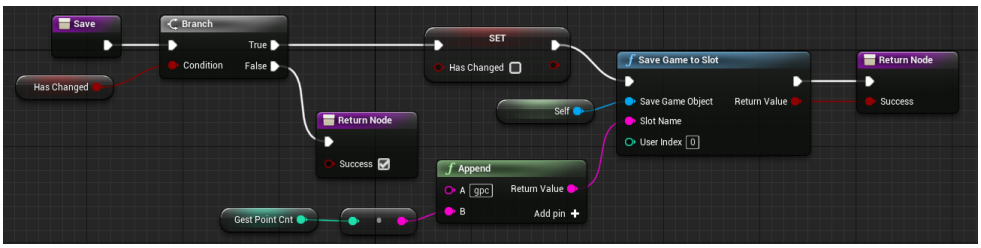
4.2.12 NP7: Perzistentní uložení gest

Pro perzistentní uložení a získání zpět z úložiště je vyžito funkcí Unreal Engine, představených v sekci 4.1, které pracují se speciálním typem Blueprints, kterým je právě GestureData, jež uchovává informace o aktuálním referenčním souboru. Gesta jsou ukládána pod názvem obsahující počet GesturePoints, takže nedojde ke smíchání gest určených pro jiný počet sledovaných bodů. Funkce `save()` je přímo funkcí GestureData, takže v ní dochází k uložení



Obrázek 4.7: Převod dat z pole souřadnic na pole vektorů a jejich rotace.

vlastní reference, a je ji možné vidět na obrázku 4.8. Načítání dat probíhá obdobně, akorát navíc při neúspěchu dojde k zavolání funkce `getDefaultData()` C++ třídy GestureDataHandler pro načtení dat výchozích. Data jsou tedy ukládána a načítána z perzistentního úložiště a požadavek tak splněn.



Obrázek 4.8: Funkce `save()` u GestureData.

4.3 Klasifikace

O klasifikaci gest se stará funkce `analyzeGesture()` C++ třídy GestureAnalyzer, kterou je možné vidět pod tímto odstavcem a jejímž vstupem je referenční soubor gest spolu s daty ke klasifikaci a výstupem je ID (0 značí nepovedenou klasifikaci), jméno klasifikovaného gesta a metrika, obsahující vzdálenost gesta od zvolené reference.

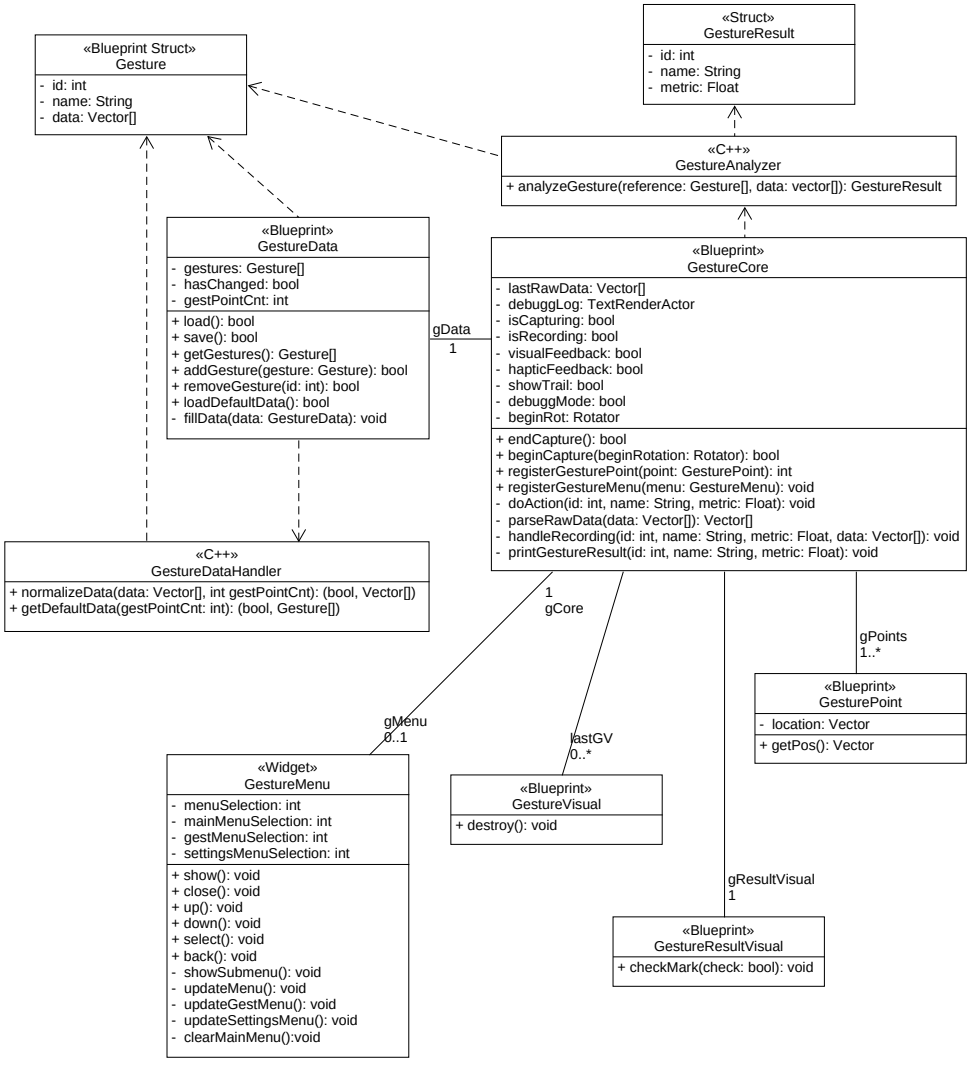
```
void UGestureAnalyzer::analyzeGesture(const TArray<FVector>...
{
float minDist = std::numeric_limits<float>::max();
// Calculate the limit based on number of used GesturePoints
int limitDist = metricLimit * (data.Num() / dataLen);

// For each reference
for (auto ptr = reference.begin(); ptr != reference.end()...
  // Pointer to data to be classified
  auto dataptr = data.begin();
  float dist = 0;

  // Calculate the sum of distances for each pair of vectors
  for (auto vectr = (*ptr).data.begin(); vectr != (*ptr).data...
    // Euclidean distance
    dist += FVector::Dist((*dataptr), (*vectr));
    dataptr.operator++();
  }

  // Update closest reference if necessary
  if (dist < minDist) {
    minDist = dist;
    id = (*ptr).id;
    name = (*ptr).name;
    metric = dist;
  }
}
// If metric exceeds the limit the classification was unsucc...
if (metric > limitDist)
  id = 0;
return;
}
```

Jak je vidět, funkce pro každé gesto z referenčního souboru vypočítá celkovou vzdálenost od dat ke klasifikaci a vybere tu referenci, která má vzdálenost nejnižší. Pokud tato vzdálenost nepřekročí daný limit, dojde k úspěšné klasifikaci a výsledek je předán zpět do Blueprints, v opačném případě dojde k nastavení `id` na 0, což indikuje neúspěšnou klasifikaci, a výsledek je také předán zpět do Blueprints.

4.4 Model v UML

V této sekci bude představen UML model finálního systému gest, který následuje hned po tomto odstavci na obrázku 4.9, a dále na něm budou popsány

změny oproti konceptuálnímu UML modelu na obrázku 3.1. Hlavní změny oproti konceptu tvoří dodatečné pomocné proměnné a funkce daných částí, přibyly pouze GestureVisual a GestureResultVisual, obě sloužící pro zobrazení určité části systému gest uživateli na scéně, důkladněji popsané níže. Zbytek modelu odpovídá tomu konceptuálnímu. Následuje výčet jednotlivých



Obrázek 4.9: UML model systému gest.

součástí systému gest a popsání proměnných a funkcí rozdílných od konceptuálního UML modelu z obrázku 3.1.

4.4.1 GestureCore

Oproti návrhu v jádru přibyly zejména pomocné proměnné, sloužící pro držení referencí a dalších nastavení. Dále v jádru proběhne prvotní zpracování dat a jejich potenciální výpis na řádku pro debuggování. Konkrétní novinky tedy jsou:

- `lastGV`
  Reference na GestureVisual, blíže popsaného níže, která se využívá při zobrazování „ocasu“ gest při kreslení.

- `gResultVisual`
  Reference na GestureResultVisual, blíže popsaného níže, která se využívá při zobrazování vizuální zpětné vazby. Není povinná.

- `debuggLog`
  Reference na volitelný TextRenderActor, které je možné využít pro výpis detailních informací o klasifikaci sloužící pro debugging, které je možné zapnout v menu nastavení.

- `showTrail`
  Pomocná proměnná indikující, jestli při kreslení gest má po nich zůstávat tzv. „ocas“, kterou lze měnit v menu nastavení.

- `debuggMode`
  Pomocná proměnná indikující, jestli se mají detailní výsledky klasifikace vypisovat na `debuggLog`.

- `parseRawData()`
  Funkce, která se stará o prvotní předzpracování dat tak, že je převede ze souboru souřadnic na soubor vektorů, a dále se předají třídě GestureDataHandler.

- `handleRecording()`
  Pomocná funkce, která slouží pro zjednodušení funkce `endCapture()` tím, že se stará o zpracování výsledků klasifikace a adekvátní zobrazení uživateli při nahrávání nového gesta.

- `printGestureResult()`
  Funkce sloužící pro výpis detailních informací o klasifikaci na `debuggLog`, pokud je zapnutý debugging režim, který je nastavitelný v menu nastavení.

4.4.2 GestureMenu

Změny v GestureMenu oproti návrhu činí primárně další proměnné pro indikaci pozice v navigaci jednotlivých podmenu, aby bylo možné se do nich

vracet bez ztracení pozice, a dále pomocných funkcí pro vykreslení jednotlivých menu, tedy:

- `mainMenuSelection`, `gestMenuSelection` a `settingsMenuSelection`
  Pomocné proměnné indikující pozici navigace v jednotlivých menu.
- `showSubmenu()`
  Pomocná funkce pro výběr adekvátního podmenu dle hodnot proměnné `menuSelection`.
- `updateMenu()`, `updateGestMenu()` a `updateSettingsMenu()`
  Pomocné funkce pro správné vykreslení daného menu dle hodnot proměnných.
- `clearMainMenu()`
  Funkce pro „vynulování“ hlavního menu.

4.4.3 GestureData

Jediná změna v GestureData je ve formě pomocné funkce, a to:

- `fillData()`
  Funkce, která se stará o načtení dat ze získaného objektu po zavolání funkce enginu pro načtení dat z perzistentního úložiště.

4.4.4 GestureDataHandler

Zde nedošlo ke změně v signatuře, pouze došlo k předání části povinností funkce `normalizeData()` na funkci `parseRawData()` z GestureCore, která data převede z souřadnic na vektory, kdežto funkce `normalizeData()` se stará o jejich správnou transformaci na data stejné stanovené délky, ať už do funkce vstupují data delší či kratší než je stanovená délka.

4.4.5 GestureVisual a GestureResultVisual

Pomocné objekty potřebné pro zobrazení určitých částí systému uživateli, konkrétně:

- **GestureVisual**
  Je užíváno pro vykreslení tzv. „ocasu“ gesta, a to pokud uživatel tuto funkčnost povolil v menu. Při kreslení gesta za sebou zanechává GesturePoint na scéně tyto objekty, které jsou po klasifikaci, případně ukončení nahrávání nového gesta, smazány funkcí `destroy()`.
- **GestureResultVisual**
  Slouží pro zobrazení vizuální zpětné vazby uživateli, pokud se v GestureCore nastaví, jinak je jádrem ignorováno. Po klasifikaci dojde k zavolání následující funkce:

– `checkMark()`
  Funkce přijímající boolean, který indikuje úspěch či neúspěch při klasifikaci, který GestureResultVisual zobrazí ve vizuální formě uživateli (znak fajfky při úspěchu, odtud název funkce).

Bez jakékoliv změny oproti konceptuálnímu modelu tak zůstaly struktury Gesture a GestureResult a stejně tak GesturePoint a GestureAnalyzer. Většina provedených změn ovšem nijak neměnila funkčnost nebo topologii systému, převážně se jedná o rozložení „superfunkcí“ na více menších funkcí s přesněji definovanými starostmi, například navrhovaná funkce `updateMenu()`, která se měla starat o správné vykreslení všech menu, byla při implementaci rozdělena na více funkcí, kde se každá stará o vykreslení jednoho (pod)menu.

Kapitola 5

Testování

Na systém gest jsou kladeny dvě hlavní podmínky, a to, aby byl co nejuniverzálnější, což je zaručeno vhodnou volbou vývojového prostředí a použitých konstrukt v něm, a dále aby byla práce s ním co nejjednodušší a intuitivní, aby mohl být využit co nejširším spektrem uživatelů. A právě tato podmínka bude v této sekci testována, tedy jestli je možné systém gest bez větších potíží nainstalovat a provozovat a jestli jej koncoví uživatelé dokážou lehce používat. Ze zjištěných faktů budou dále vyvozeny implikace pro podniky, které by mohly systém gest využít pro své produkty.

5.1 Instalace

V této sekci tedy dojde k instalaci systému dle uvedeného detailního návodu a budou konstatovány případné zádrhely. Systém gest byl tvořen a je zde testován na verzi Unreal Engine 4.25.4. Spolu s testováním jednoduchosti instalace dojde k změření času potřebného na zprovoznění systému gest, který by jistě měl být v řádu maximálně desítek minut, aby bylo možné instalaci považovat za snadnou a rychlou.

5.1.1 Zavedení souborů

Soubory je možné do nového projektu zavést dvěma způsoby, a to za využití migrace assetů, tedy funkce Unreal Engine, která vybraný asset a všechny závislé soubory překopíruje z jednoho projektu do druhého, a tedy tak zajistí všechny potřebné věci, tedy by neměl nastat žádný problém, a proto zde bude otestován způsob druhý, a to přetažení zdrojových souborů systému gest do složky cílového projektu. Pro migraci assetů je možné využít přiložené zdrojové soubory DEMO projektu.

Je třeba do projektu zavést dva typy zdrojových souborů, a to Blueprints a C++, kde jednodušší jsou jednoznačně Blueprints zdrojové soubory, které stačí přetáhnout do složky Content cílového projektu a po spuštění engine je

bude možné vidět a používat. Jelikož Blueprints vyžívají datové typy a funkce definované v C++ souborech, je vhodné Blueprints soubory přidat zároveň s C++ soubory, případně až po nich.

Pro správné zavedení C++ souborů je třeba provést pár kroků, a to:

1. **Přidat C++ funkčnost do projektu**, pokud je čistě v režimu Blueprints, lze docílit přidáním libovolné nové C++ třídy přímo z editoru, což zajistí správné nastavení prostředí pro podporu C++ souborů a vytvoří třídu, kterou je možné následně smazat.

2. **Přesunout zdrojové soubory**, tedy .cpp a .h soubory, rozdělené do složek Private a Public, do složky Source cílového projektu.

3. **Změnit signaturu tříd** u dvou hlavičkových souborů, jelikož Unreal Engine pro správné napojení C++ tříd do jádra generuje API, které má jméno dle projektu. Například výsledná signatura třídy GestureAnalyzer by měla vypadat následovně pro projekt s názvem MyProject: `class MYPROJECT_API UGestureAnalyzer`.

4. **Překompilovat projekt** ještě před spuštěním engine, pro jistotu je lepší smazat již vygenerované soubory a znovu je vygenerovat tak, že se pravým klikem na soubor projektu (.uproject) zvolí možnost „Generate Visual Studio project files“, což vytvoří mimo jiné stejnojmenný soubor s přípono .sln, kterým je již možné otevřít projekt ve Visual Studiu a provést kompilaci.

5. **Otevřít projekt** v editoru. Pokud se předešlé kroky provedly správně, projekt se otevře bez hlášek a bude již možné vidět jak Blueprints, tak C++ soubory přímo v editoru.

6. (Potenciálně) **Opravit typ proměnné**, pokud došlo k problému, který může nastat zejména při importu Blueprints před C++ třídami, jelikož je v Blueprints využita struktura definovaná v C++ zdrojových souborech (viditelná na konci sekce 4.1.3). Tuto strukturu v Blueprints editor někdy označí za nedefinovanou a je třeba instalaci opakovat nebo nedefinovanou proměnnou manuálně nastavit na typ Gesture. Toto je jediný těžší krok při instalaci, ale nenastane vždy a jedná se o problém editoru, tedy není možné ho kompletně odstranit autorem. Místo toho jsou zasažená místa (GestureData) doplněná o komentáře popisující kde přesně a jak danou proměnnou nastavit. Je pravděpodobné, že problém bude v novějších verzích Unreal Engine opraven.

Uvedené kroky byly provedeny na nově vytvořeném projektu a s přesunutím Blueprints zdrojových souborů nebyl sebemenší problém a samotný akt přesunutí trval 5 sekund, což je čas potřebný pro zavedení Blueprints souborů do projektu nového.

Trochu déle trvalo přenést C++ zdrojové soubory, jelikož bylo třeba nejdříve projekt nastavit na podporu C++ souborů (20 sekund), přesunout soubory (5 sekund), smazat vygenerované soubory (5 sekund), nechat je znovu vygenerovat (20 sekund) a v poslední řadě zkompilovat projekt ve Visual Studiu (30 sekund). Přidání C++ souborů do projektu tedy zabralo při testování 1 minutu 20 sekund.

Při testu došlo k problému zmíněnému v 6. kroku instalace, tedy engine si nespojil strukturu z C++ souborů se strukturou použitou v GestureData a došlo k jejímu manuálnímu nastavení (100 sekund) v pěti postižených funkcích, které jsou doplněné o komentáře popisující, co s proměnnou v dané funkci udělat.

Celkové zavedení všech zdrojových souborů systému gest, včetně potenciálního, však nekonzistentního problému zabralo 3 minuty. Všechny uvedené kroky je možné lehce následovat a jedná se o triviální úkony, které jistě zvládne každý. Při instalaci se může objevit jeden těžší krok, který sice nenastane pokaždé, ale vyžaduje lehkou editaci v Blueprints, což není sice triviální úkon, ale daný Blueprint je doplněn o komentáře schopné navést i laika k úspěšné opravě.

5.1.2 Inicializace GestureCore

Po zavedení zdrojových souborů je možné systém gest zapojit do scény. Jako první je třeba inicializovat GestureCore a uložit jeho referenci jako proměnnou libovolného objektu, kde vhodným kandidátem je Pawn (viz sekce 4.1.1). Je tedy potřeba vytvořit proměnnou s typem reference na GestureCore a uložit do ní referenci po vytvoření rodičovského objektu.

Přidání proměnné do zvoleného objektu (10 sekund) i její inicializace (15 sekund) je triviální úkon a při testování nenastal žádný problém. Inicializaci je vhodné provést buď v eventu BeginPlay, nebo přímo v konstrukčním skriptu, aby došlo k inicializaci co nejdříve. Pro inicializaci se používá funkce Unreal Engine SpawnActor a následně je do vytvořené proměnné uložena reference vrácená funkcí.

GestureCore je tedy možné inicializovat a uložit jeho referenci do proměnné během 30 sekund, kde GestureCore se dále postará o správnou inicializaci i užití GestureData.

5.1.3 Napojení vstupu

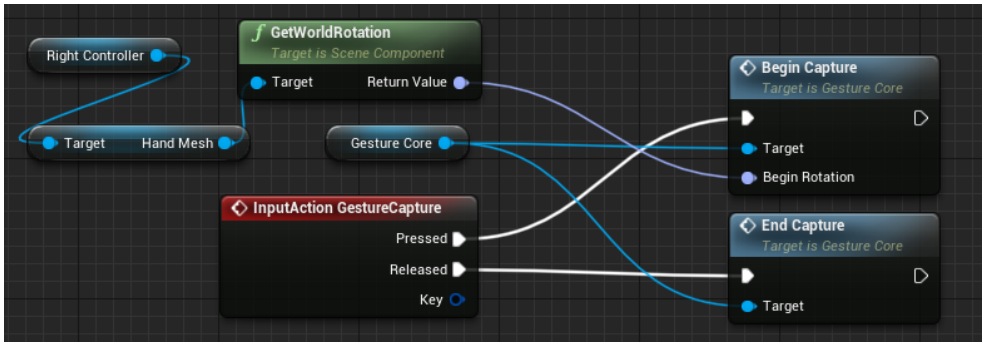
Dalším krokem je napojit dvě funkce GestureCore, indikující začátek a konec snímání, na vybraný vstupní prvek (například jedno tlačítko).

V první řadě je třeba vytvořit tzv. „Action Mapping“ (AM), což se dělá v nastavení engine, záložka Input, a zde si uživatel vytvoří AM a přiřadí si k němu libovolnou kolekci vstupních prvků, které daný AM vyvolají. Právě

zde je možné na jeden AM napojit vstupy od nejrůznějších tlačítek, jako jsou ta na VR, PS4, Xbox a Steam ovladačích, mobilních VR, klávesnicích a jiných.

V druhé řadě je třeba nově vzniklý AM napojit na funkce `beginCapture()` a `endCapture()`, což je triviální záležitost viditelná na obrázku 5.1. Do funkce `beginCapture()` se ještě předává rotace objektu pro tvorbu gest (zde pravá ruka) pro následnou práci s daty.

Při testování tvorba AM zabrala zhruba 25 sekund a následné napojení na funkce dalších 25 sekund, žádné problémy se nevyskytly, tvorba AM a napojení na funkce tak trvá zhruba 50 sekund.



Obrázek 5.1: Napojení GestureCore na vstup.

5.1.4 Registrace GesturePoint

Poslední věc, kterou je třeba udělat před tím, než bude systém gest plně funkční, je vložit GesturePoint na zvolené místo, tedy na objekt, kterým se budou tvořit gesta, a zaregistrovat jeho referenci do GestureCore. Po registraci alespoň jednoho GesturePoint je systém funkční.

Uživatel rozmístí GesturePoints na vybraná místa, například prsty modelu ruky, na které je napojí a vzniklé GesturePoints zaregistruje do jádra pomocí funkce `registerGesturePoint()`, která uloží jejich referenci a postará se o následné správné nahrání referenčních gest dle počtu použitých GesturePoints.

Při testování tohoto kroku tedy stačilo jednou funkcí bod vytvořit, druhou ho „uchytit“ na prst ruky a třetí bod zaregistrovat do jádra. Jelikož došlo při testování ještě k tvorbě socketu na modelu ruky, což pouze slouží pro přesnější určení místa GesturePoint, avšak není povinné, tento krok zabral zhruba 55 sekund.

5.1.5 Volitelné kroky

Po rozmístění alespoň jednoho GesturePoint je již systém gest funkční, ale je možné vykonat ještě dva volitelné kroky, a to:

1. **Registrace GestureMenu**
   Systém gest je distribuován se sadou výchozích gest, které slouží k ovládání administrátorského menu, kterým je možné systém gest ovládat a měnit jeho nastavení přímo z aplikace, ale stejně tak je možné chování systému měnit ve zdrojových souborech, takže toto menu není nutné pro funkčnost systému. Pokud se uživatel rozhodne menu nevyužít, je vhodné smazat napojení na funkce menu ve funkci `doAction()`.

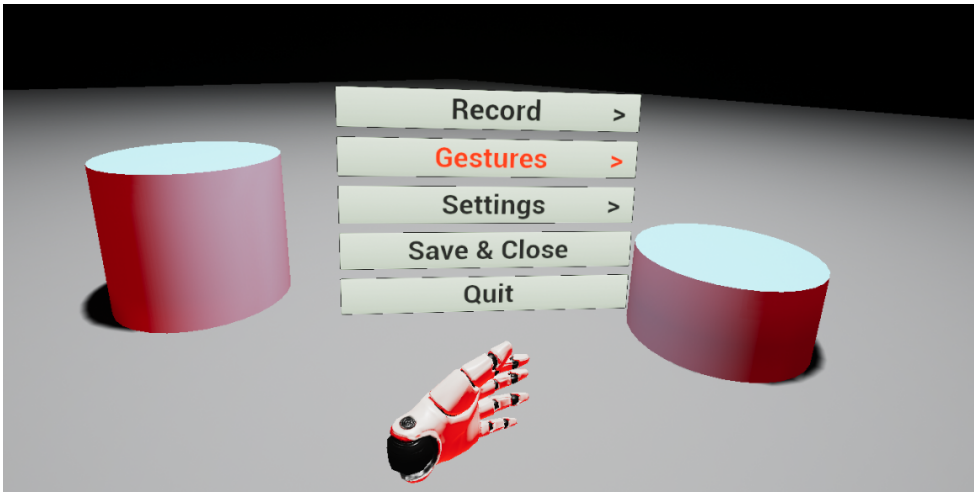
   Registrace menu probíhá stejně, jako s GesturePoint, akorát menu může být pouze jedno. Je třeba umístit Widget s GestureMenu na libovolnou pozici na scéně, ať už doprostřed obrazovky nebo třeba nad model ruky (viz obrázek 5.2), a referenci na něj do GestureCore zaregistrovat pomocí funkce `registerGestureMenu()`. Nyní je možné systém gest ovládat z aplikace pomocí administrátorského menu.

   Při testování tohoto kroku nebyl nalezen žádný problém, vskutku pouze stačilo umístit Widget na scénu a zavolat na něj zmíněnou funkci, což dohromady zabralo zhruba 50 sekund a po provedení tohoto kroku bylo možné menu otevřít adekvátním gestem přímo v aplikaci.

2. **Přidání debuggovací řádky**
   Posledním, avšak volitelným, krokem je přidání řádky pro debuggování, která do aplikace vypisuje informace o provedené klasifikaci jako je ID, jméno či hodnota metriky. Systém má dvě možnosti zpětné vazby, které uživatele informují o (ne)úspěchu klasifikace, a tak není důvod pro běžného uživatele, aby řádku využil, jelikož pouze přidává možnost vidět konkrétní hodnoty klasifikace.

   Pro výpis textu na scénu se využívá Text Render Actor, představený v sekci 4.1.1, který je třeba umístit na libovolné místo na scéně a jeho referenci uložit do GestureCore. Jelikož řádku pro debuggování využije jistě někdo, kdo se alespoň trochu v kódu vyzná, není vytvořena žádná ulehčující funkce pro uložení reference a místo toho uživatel pouze nastaví proměnnou `debuggLog` na daný Text Render Actor. Před každým užitím této proměnné systém kontroluje její platnost, takže nedojde k problémům při jejím nevyužití.

   Při testování tohoto kroku opět nedošlo k žádném problému, na scénu byl umístěn Text Render Actor (40 sekund) a jeho reference byla uložena do proměnné `debuggLog` (20 sekund). Tento volitelný krok tedy zabral zhruba 1 minutu.



Obrázek 5.2: Ukázka menu napojeného nad model ruky.

5.1.6 Závěr testování instalace

Proběhlo testování instalace za využití přesunutí zdrojových souborů dle uvedeného návodu, a celý instalační proces bylo jednoduché následovat. Při testování se objevil jeden předtím neočekávaný problém, a to fakt, že při importu zdrojových souborů do Unreal Engine došlo k nespojení C++ struktury Gesture, zobrazené na konci sekce 4.1.3, spolu s použitou strukturou v Blueprints a dojde k jejímu označení za nedefinovanou. Unreal Engine ovšem vždy strukturu Gesture rozeznal, takže stačí v postiženém Blueprints souboru změnit typ z nedefinovaného na typ Gesture. Pro větší komfort méně znalých uživatelů byla potenciální místa konfliktu doplněna o komentáře popisující, jak případnou chybu opravit.

Při testování došlo také ke změření času potřebného na jednotlivé kroky, a povinné úkony, včetně opravy vzniklého problému, zabraly při testování celkově 6 minut a 35 sekund. Nepovinné kroky, tedy registrace GestureMenu a přidání řádky pro debuggování, zabraly další minutu a 50 sekund, tedy dohromady instalace systému gest, včetně opravy problému a provedení nepovinných kroků, zabrala při testování 8 minut a 25 sekund.

Povedlo se tedy systém gest úspěšně nainstalovat do nového projektu během méně než 10 minut, a to za využití metody přesunu zdrojových souborů, ovšem instalace byla prováděna autorem, který systém gest dobře zná, takže se dá předpokládat, že by instalace uživatelem, který systém gest ještě neviděl, mohla trvat o něco málo déle.

5.2 Testování systému na lidech

Bylo otestováno, že systém gest je možné lehce a rychle zprovoznit, což je vhodné, jelikož má systém být co nejuniverzálnější a ideálně použitelný i nezkušenými uživateli. Dále byly v kapitole Realizace dokázány za splněné požadavky typu jednoduchost napojení funkcí aplikace na systém gest nebo jednoduchost manipulace se souborem aktivních gest.

Jako poslední tedy zbývá otestovat na lidech, ideálně plně nezkušených, jestli budou schopní zapojený systém gest v aplikaci využívat bez potíží a u toho i otestovat přesnost systému a jeho schopnost kontrolovat nově přidaná gesta na podobnost vůči ostatním. Bohužel kvůli současné situaci kolem nemoci COVID-19 není možné si domů zvát hosty, a tak testování proběhne pouze na 3 lidech, z toho 2 dospělí a 1 dítě, kteří s VR mají minimální až žádné zkušenosti, a tedy budou jistě vhodnými kandidáty na testování.

Nejdříve dojde k představení VR samotného a rychlý úvod k systému gest, aby zkušební lidé vůbec věděli, co se po nich chce, a následně budou každému z nich zadány následující úkoly:

1. **20 náhodných gest**
   Každému člověku bude postupně dáno za úkol vykonat 20 náhodných gest, týkajících se navigace v menu, a bude se zkoumat úspěšnost lidí a systému, tedy zejména jestli udělají správné gesto a jestli ho systém rozezná.

2. **Vytvořit gesto**
   Dalším úkolem bude vytvořit libovolné nové gesto za využití menu, kde dané gesto bude napojeno na funkci teleportu, takže cílem tohoto úkolu bude nové gesto vytvořit a následně jej použít pro teleportaci po scéně.

3. **Vytvořit stejné gesto**
   Posledním úkolem bude vytvořit nové gesto, ovšem za použití stejného tvaru jako minule, a bude zkoumáno, jestli systém detekuje duplikát a gesto odmítne.

5.2.1 První úkol

Všichni tři testeři po krátkém úvodu velmi rychle pochopili navigaci v menu, a to i včetně padesátníka s minimální technickou znalostí, a každému z nich byly dány úkoly typu „V nastavení vypni haptickou vazbu“ či „Vymaž gesto 3“, které všichni velmi intuitivně zvládali, používali vždy správná gesta a systém jejich gesta interpretoval s vysokou přesností.

Jediný zádrhel nastal, když asi ve 2 případech testera prvního a 1 případu u testera druhého bylo provedeno gesto typu „Tah“ moc krátce, a vyhodnotilo se tak místo tahu jako tečka, což ale ihned sami testeři správně napravili a gesto udělali delší. Za zmínku stojí také pozorování, že tester 2 nejprve držel

ovladač natočený zhruba o 90° a pokusil se provést gesto tah doprava, což bylo vyhodnoceno jako tah dolů, protože ovladač vskutku vykonal pohyb dolů, což je považované za správné chování, ale pokud by uživatel chtěl, aby se směr nebral vůči ovladači nýbrž vůči pozici headsetu, stačí zaměnit zdroj informace o prvotní rotaci z ovladače na HMD.

Z 60 daných úkonů tedy testeři zvládli vždy vykonat adekvátní gesta a systém je správně interpretoval v 56 případech, což dává $93.\overline{3}\%$ přesnost, kde většina nepovedených klasifikací byla z důvodu udělání gesta moc krátkého, což systém správně klasifikoval blíže k tečce než k tahu, ale to nebyl úmysl testerů a proto je klasifikace považována za špatnou v těchto pár případech.

5.2.2 Druhý úkol

V tomto úkolu byly testeři požádáni, aby zapnuli nahrávání gest, a následně si měl každý tester vytvořit libovolné gesto dle libosti. Tester 1 vytvořil gesto reprezentující číslo 8, tester 2 znak napodobující otazník a tester 3, dítě, si jako svoje gesto vybralo karikaturu hlavy koně.

Na toto nově vytvořené gesto byla napojena funkce teleportace, a testeři měli za úkol vykonat svoje gesto a teleportovat se, což se každému testerovi povedlo hned na první pokus a následně se sami ještě párkrát teleportovali. Tento krok byl tedy jak testery, tak systémem splněn na 100 %.

5.2.3 Třetí úkol

Posledním úkolem pro testery bylo vytvořit stejné gesto jako v předchozím úkolu. Tester 1 při tvorbě své „osmičky“ lehce ustoupil a vzniklé gesto bylo možné uložit, ovšem při opětovné tvorbě již systém označil gesto jako duplikát a nahrávání přerušil. U testera 2 došlo k podobné situaci, kde první pokus o napodobení gesta systém nechal uložit, ale druhý pokus již označil jako duplikát a nahrávání přerušil. U testera 3 systém gesto vytvořené za duplikát neoznačil, ovšem z pohledu autora nově vzniklá karikatura hlavy koně byla dosti rozdílná od té původní, a tedy to není možné brát jako chybu systému.

Testeři se tedy pokusili napodobit svoje gesta, ale ani jeden ho na poprvé nenapodobil dostatečně, aby jej systém označil za duplikát, což ovšem ještě neznamená nutně chybu, jelikož vše záleží na stanoveném limitu pro podobnost gest a schopností člověka gesto přesně napodobit. Pokud by ovšem bylo žádané, aby systém detekoval duplikáty přísněji, stačí upravit hodnotu udávající limit pro podobnost ve funkci `handleRecording()`.

5.2.4 Vyhodnocení

Díky současné situaci není možné provést testování na velkém počtu lidí, ale i přes to testování systému na 3 lidech, kteří se skládají ze dvou dospělých a jednoho dítěte, přineslo velmi zajímavá zjištění.

Jako první bylo velmi zřejmě vidět, že používání gest je pro všechny testery velmi přirozené a intuitivní, a po pár slovech o tom, jak systém funguje, dokázali sami v menu navigovat a používat gesta, a to včetně padesátníka, který se jinak technologii vyhýbá a poprvé vyzkoušel VR.

Druhým pozitivním zjištěním je fakt, že systém gesta klasifikoval ve většině případů správně, resp. klasifikoval je správně vždy, ale párkrát (3 z 60) testeři gesta udělali moc krátká, a tak byla klasifikována jako tečka, což ovšem není žádná chyba systému a závisí na referenčních gestech. Všem testerům se povedlo nahrát vlastní gesto a následně ho využít pro teleportaci po scéně. Zvolená gesta byla osmička, otazník a hlava koně.

Poslední zjištění proběhlo během testování schopnosti systému detekovat duplikáty, kde sice systém na první pokus duplikát nedetekoval, ale testeři gesta neúmyslně lehce pozměnili oproti svým referencím, jelikož jsou to stále pouze lidé, takže není vhodné z toho vyvodit závěr, že systém špatně detekuje duplikáty, protože při druhém pokusu již u dvou testerů došlo k detekci duplikátu. Detekce duplikátu se nepovedla pouze u testera 3, kterým bylo dítě, a neshoda gesta (hlava koně) tvořeného s tím referenčním byla znát již pouhým okem, ovšem nebyl problém toto gesto používat pro druhý úkol, jelikož detekce duplikátů má mnohem přísnější limity než klasifikace samotná.

Velmi milým zjištěním tedy bylo, že systém působil pro všechny testery velmi intuitivně, a to jak pro dítě, tak pro padesátníka, kde oba dva použili VR poprvé, a všem testerům se podařilo vytvořit si svoje gesto a s ním provést akci. Jediné, co nemělo ideální výsledky, byla detekce duplikátů, ale to bylo způsobeno primárně lidskou chybou, ovšem pokud by uživatel chtěl detekovat duplikáty s vyšší přísností, stačí upravit hodnotu maximální povolené podobnosti ve funkci `handleRecording()`.

5.3 Implikace pro podniky

Systém gest se jeví jako vhodnou volbou pro podniky, které by chtěli rozšířit svoji působnost na oblast VR, ale nechtějí pro to naložit zbytečně velký kapitál. Ať už se jedná o nějakou marketingovou kampaň, nebo přímo produkt pro VR, systém gest by danému podniku mohl značně ulehčit práci.

Z testování bylo zjištěno, že systém gest je možné nainstalovat do 10 minut i spolu s volitelnými kroky, kde instalace sice byla provedena autorem se znalostmi systému, takže asi byla rychlejší, než by byla provedena někým, kdo systém vidí poprvé, ale i tak by čas instalace neměl přesáhnout maximálně půl hodiny, což je nesrovnatelně menší čas, než by vyžadovalo vlastnoruční implementace systému zpracovávající gesta na většině VR hardware.

Vysoká univerzálnost systému dále podnikům zajistí, že jejich produkt využívající systém gest bude možné spustit a ovládat na většině VR hardware, a zároveň není třeba pro každý hardware zvlášť napojovat ovládání a podobně,

takže dojde k dalšímu velkému ušetření času, což je u podniků přímo úměrné ušetřeným penězům.

Dále bylo při testování zjištěno, že používání gest je pro koncové uživatele velmi intuitivní a přirozené, a i ti, kteří neměli žádné předešlé zkušenosti s VR, plně pochopili, jak gesty ovládat aplikaci, a tedy se dá předpokládat, že produkty podniků využívající systém gest by byli uživateli intuitivně používány.

Ať už by se jednalo o jednoduchou reklamu na mobilních VR, či plnohodnotný produkt pro VR, využití systému gest by podnikům ušetřilo obrovské náklady oproti samostatné implementaci podobné funkcionality, zejména jelikož instalace nezabere více než půl hodiny oproti týdnům stráveným implementací funkcionality podobné. Navíc bylo na systému gest otestováno, že je velmi intuitivní, a nemají ho problém užívat ani technicky méně zdatní uživatelé, což je jistě žádaná vlastnost od softwarových produktů podniků.

Závěr

Cílem této diplomové práce bylo navrhnout a implementovat univerzální systém gest pro VR, který by poskytl alternativu ke klasickému ovládání VR aplikací pomocí vstupních prvků na ovladačích, a to ve formě možnosti ovládat aplikaci pomocí dělání gest, která mohou mít libovolný tvar dle přání uživatele. Tento systém gest má být co nejuniverzálnější, aby ho bylo možné využít na většině VR produktů, a vyřešit tak hlavní problém s klasickým ovládáním VR aplikace, kterým je fakt, že různé VR produkty mají různé vstupní prvky, a často je tak třeba nastavovat aplikaci pro každý produkt zvlášť.

V první části práce došlo k analýze současných editorů pro VR, ze kterých byl vybrán ten, jenž sliboval nejširší nativní podporu VR hardware, a tedy i univerzálnost systému, což byl Unreal Engine. Dále došlo k analýze současných VR produktů a hledali se ty ovládací prvky, které se nachází na ideálně všech produktech, což se ukázalo, že je pouze jedno jediné klasické tlačítko a jeden prvek se sledováním pohybu v prostoru. V poslední části došlo k analýze dalších systémů využívajících gesta, což sloužilo pro následné stanovení požadavků a best practices pro tvorbu gest.

V následující části proběhl návrh systému gest, který se skládal z tvorby funkčních a nefunkčních požadavků vycházejících ze zadání a výsledků analýzy, které byly stanoveny tak, aby bylo možné lépe hlídat univerzálnost, flexibilitu a jednoduchost užívání systému gest. Ke každému z požadavků byl doplněn návrh na jeho splnění a dále došlo k návrhu jednotlivých prvků systému, jako je třeba pomocné menu či algoritmus pro klasifikaci gest. V poslední části byl představen konceptuální model v UML, který obsahuje popis jednotlivých částí i jejich funkcí a proměnných.

V další části práce došlo k úspěšné implementaci systému gest dle návrhu a ke každému ze stanovených požadavků je napsáno, jak ho bylo při implementaci dosaženo. Dále došlo k představení použitých funkcí a prvků z Unreal Engine a uvedení ukázek C++ kódu klíčových částí, jako je třeba algoritmus pro klasifikaci gest, který využívá euklidovskou vzdálenost. V poslední části je představen UML model, který již přesně reflektuje implementovaný systém

a má popsané změny oproti modelu konceptuálnímu.

V poslední části práce proběhlo testování systému, které se skládalo z instalace systému do nového projektu dle uvedeného návodu za současného měření času stráveného instalací, což zabralo méně než 10 minut, a systém se povedlo úspěšně nainstalovat. Dále proběhlo testování systémů na jiných lidech, než je autor, a zkoumalo se, jak se systémem interagují, jestli je intuitivní, jestli lehce dokážou vytvořit vlastní gesta a jestli systém správně klasifikuje a detekuje duplikáty. Testeři se systémem interagovali až nečekaně intuitivně a zvládli vykonat všechny zadané úkony. Ze zjištěné vysoké intuitivnosti systému a jeho lehké a rychlé instalaci došlo k vyvození implikací pro podniky, jelikož se systém jeví jako vhodný produkt pro jakýkoliv podnik, který by chtěl rozšířit svoji působnost na oblast VR a zároveň ušetřit náklady.

Systém gest je možné v budoucnu dále rozšiřovat, kde možná rozšíření jsou například přidání možnosti měnit chování aplikace přímo z pomocného menu, umožnění jednoduchého předávání vytvořených gest mezi různými projekty nebo třeba udělat proces nahrávání nového gesta komplexnější, aby například data referenčního gesta byla vytvořena z více nahrávek stejného gesta namísto z pouze jednoho nahrání.

Systém gest je vhodný i pro uživatele s menšími vývojářskými znalostmi, jelikož obsahuje i pomocné menu, pomocí kterého je možné gesta tvořit, mazat a prohlížet přímo z aplikace, ovšem není problém stejné úkony provádět i ve zdrojových souborech. Systém gest pro fungování vyžaduje pouze jedno jediné tlačítko. Je tedy možné ho využít i například na VR pro mobilní telefony, které často ani víc vstupních prvků nemá. Je ovšem možné na něm dělat stejná gesta jako například pomocí ovladače, což dále podporuje univerzálnost systému. Navíc modulární rozdělení systému gest umožňuje uživatelům systém využít přesně dle svých potřeb. Je možné jisté volitelné prvky ani nepoužít, jako je třeba zmíněné pomocné menu či řádka pro debuggování. Uživatel tak může využít ty části systému, které potřebuje, a navíc je může napojit na libovolné objekty na scéně, což zvyšuje flexibilitu systému. Systém gest byl tedy dle zadání úspěšně navrhnut, implementován i otestován.

Bibliografie

1. *A first look at Unreal Engine 5* [online]. 2020 [cit. 2020-10-06]. Dostupné z: https://www.unrealengine.com/en-US/blog/a-first-look-at-unreal-engine-5.
2. *About AppGameKit* [online]. 2020 [cit. 2020-10-06]. Dostupné z: https://www.appgamekit.com/about-agk.
3. *About CRYENGINE* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://www.cryengine.com/support/view/general.
4. *About CyberGlove Systems* [online]. 2017 [cit. 2020-10-15]. Dostupné z: http://www.cyberglovesystems.com/about-us.
5. *About the VIVE Controllers (2018)* [online]. 2020 [cit. 2020-10-07]. Dostupné z: https://www.vive.com/us/support/vive-pro/category_howto/about-the-controllers---2018.html.
6. *About the VIVE Cosmos controllers* [online]. 2020 [cit. 2020-10-07]. Dostupné z: https://www.vive.com/us/support/cosmos/category_howto/about-the-controllers.html.
7. *Amazon Lumberyard - Frequently Asked Questions* [online]. 2020 [cit. 2020-10-06]. Dostupné z: https://aws.amazon.com/lumberyard/faq.
8. *Azure Kinect DK* [online]. 2020 [cit. 2020-10-13]. Dostupné z: https://azure.microsoft.com/en-us/services/kinect-dk/#industries.
9. BOWELL, Andrew. *Bolt visual scripting is now included in all Unity plans* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://blogs.unity3d.com/2020/07/22/bolt-visual-scripting-is-now-included-in-all-unity-plans.
10. *Compare Unity plans* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://store.unity.com/compare-plans.
11. *CRYENGINE Licensing* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://www.cryengine.com/support/view/licensing.

12. *CRYENGINE Programming* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://docs.cryengine.com/display/CEPROG/CRYENGINE+Programming.

13. *CryEngine V releases today on a pay-what-you-want basis* [online]. 2016 [cit. 2020-10-05]. Dostupné z: https://www.pcgamer.com/cryengine-v-releases-today-on-a-pay-what-you-want-basis.

14. *CyberGlove Systems - Featured Hardware* [online]. 2017 [cit. 2020-10-15]. Dostupné z: http://www.cyberglovesystems.com/.

15. DANTE D'ORAZIO, Vlad Savov. *Valve's VR headset is called the Vive and it's made by HTC* [online]. 2015 [cit. 2020-10-07]. Dostupné z: https://www.theverge.com/2015/3/1/8127445/htc-vive-valve-vr-headset.

16. *Discover the Full Line-up of VIVE Systems* [online]. 2020 [cit. 2020-10-07]. Dostupné z: https://www.vive.com/us/product.

17. *Docs / Motion controllers* [online]. 2018 [cit. 2020-10-08]. Dostupné z: https://docs.microsoft.com/en-us/windows/mixed-reality/design/motion-controllers.

18. *Eyesight Technologies - Home* [online]. 2020 [cit. 2020-10-16]. Dostupné z: https://www.eyesight-tech.com/.

19. *Facebook to Acquire Oculus* [online]. 2014 [cit. 2020-10-08]. Dostupné z: https://about.fb.com/news/2014/03/facebook-to-acquire-oculus.

20. FINE, Richard. *UnityScript's long ride off into the sunset* [online]. 2017 [cit. 2020-10-05]. Dostupné z: https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset.

21. *For Tim Sweeney, advancing Epic means racing into AR and VR* [online]. 2017 [cit. 2020-10-06]. Dostupné z: https://www.gamasutra.com/view/news/292573/For_Tim_Sweeney_advancing_Epic_means_racing_into_AR_and_VR.php.

22. *Gear VR with Controller* [online]. 2020 [cit. 2020-10-11]. Dostupné z: https://www.samsung.com/global/galaxy/gear-vr/.

23. *GestureTek - Home* [online]. 2020 [cit. 2020-10-16]. Dostupné z: http://gesturetek.com/index.php.

24. *Getting started with VR development in Unity* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://docs.unity3d.com/Manual/VROverview.html.

25. GIBERT, Laurent. *Visual scripting roadmap update - August 2020* [online]. 2020 [cit. 2020-10-05]. Dostupné z: https://forum.unity.com/threads/visual-scripting-roadmap-update-august-2020.951675.

26. *Godot Features* [online]. 2020 [cit. 2020-10-06]. Dostupné z: https://godotengine.org/features.

27. *Google VR Developers - Platforms - Cardboard* [online]. 2017 [cit. 2020-10-09]. Dostupné z: https://developers.google.com/vr/discover/cardboard.

28. GRIFFAIS, Pierre-Loup A. *ValveSoftware / IndexHardware / Controller* [online]. 2019 [cit. 2020-10-07]. Dostupné z: https://github.com/ValveSoftware/IndexHardware/blob/master/Controller/3control.png.

29. *HoloLens 2* [online]. 2020 [cit. 2020-10-13]. Dostupné z: https://www.microsoft.com/en-us/hololens/hardware.

30. *HoloLens 2 gestures for authoring and navigating in Dynamics 365 Guides* [online]. 2020 [cit. 2020-10-13]. Dostupné z: https://docs.microsoft.com/en-us/dynamics365/mixed-reality/guides/authoring-gestures-hl2.

31. *How You Become the Controller (Archived)* [online]. 2010 [cit. 2020-10-13]. Dostupné z: https://web.archive.org/web/20110111115024/https://www.xbox.com/en-US/Live/EngineeringBlog/122910-HowYouBecometheController.

32. *HP Reverb G2* [online]. 2020 [cit. 2020-10-08]. Dostupné z: https://www8.hp.com/us/en/vr/reverb-g2-vr-headset.html.

33. KELLY, Heather. *Microsoft's new $3,500 HoloLens 2 headset means business* [online]. 2019 [cit. 2020-10-13]. Dostupné z: https://edition.cnn.com/2019/02/24/tech/microsoft-hololens-2/index.html.

34. KOBYLARZ, Jhonatan; BIRD, Jordan J.; FARIA, Diego R.; RIBEIRO, Eduardo Parente; EKÁRT, Anikó. Thumbs up, thumbs down: non-verbal human-robot interaction through real-time EMG classification via inductive and supervised transductive transfer learning. *Journal of Ambient Intelligence and Humanized Computing*. 2020. ISSN 1868-5145. Dostupné z DOI: 10.1007/s12652-020-01852-z.

35. *Magic Leap - Gesture* [online]. 2018 [cit. 2020-10-14]. Dostupné z: https://developer.magicleap.com/en-us/learn/guides/design-gesture.

36. *Magic Leap 1 Overview* [online]. 2019 [cit. 2020-10-14]. Dostupné z: https://developer.magicleap.com/en-us/learn/guides/magic-leap-one-overview.

37. *Merge Headset - Be Transported to Virtual Worlds* [online]. 2020 [cit. 2020-10-11]. Dostupné z: https://mergeedu.com/headset.

38. O'BRIEN, Terrence. *Microsoft's new Kinect is official: larger field of view, HD camera, wake with voice* [online]. 2013 [cit. 2020-10-13]. Dostupné z: https://www.engadget.com/2013-05-21-microsofts-new-kinect-is-official.html.

39. *Oculus - Compare Headsets* [online]. 2020 [cit. 2020-10-08]. Dostupné z: https://www.oculus.com/compare.

40. *Oculus› Development› Oculus Rift S/Rift Development› Touch line art* [online]. 2016 [cit. 2020-10-08]. Dostupné z: https://forums.oculusvr.com/developer/discussion/42522/touch-line-art.

41. *Overview of generating a QR viewer profile* [online]. 2020 [cit. 2020-10-10]. Dostupné z: https://support.google.com/cardboard/manufacturers/answer/6321873.

42. PAVLOVIC, Vladimir I.; SHARMA, Rajeev; HUANG, Thomas S. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*. 1997, s. 677–695.

43. *Play Rift Content on Quest with Oculus Link, Available Now in Beta* [online]. 2019 [cit. 2020-10-08]. Dostupné z: https://about.fb.com/news/2019/11/play-rift-content-on-quest-with-oculus-link-available-now-in-beta.

44. *PlayStation VR - Overview* [online]. 2020 [cit. 2020-10-09]. Dostupné z: https://www.playstation.com/en-us/explore/playstation-vr.

45. *PlayStation®VR - Tips & Tech Specs* [online]. 2020 [cit. 2020-10-09]. Dostupné z: https://www.playstation.com/en-us/explore/playstation-vr/tips-and-specs/#techspecs.

46. *PS VR aim controller* [online]. 2017 [cit. 2020-10-09]. Dostupné z: https://www.playstation.com/en-in/get-help/help-library/accessories/controller--headset---playstation-camera/ps-vr-aim-controller.

47. RAHMING, A.K. *Nintendo wins court case against iLife over Wii Remote tech controversy* [online]. 2020 [cit. 2020-10-16]. Dostupné z: https://www.nintendoenthusiast.com/nintendo-wins-court-case-against-ilife-over-wii-remote-tech-controversy/.

48. ROBERTSON, Adi. *Oculus' new Quest 2 VR headset starts at $299 and ships October 13th* [online]. 2020 [cit. 2020-10-08]. Dostupné z: https://www.theverge.com/2020/9/16/21427947/oculus-quest-2-facebook-vr-headset-price-release-date-preorder-connect.

49. SCHENK, Eric. *AWS Game Tech Blog* [online]. 2016 [cit. 2020-10-05]. Dostupné z: https://aws.amazon.com/blogs/gametech/welcome-to-the-amazon-gamedev-blog.

50. *Steam Hardware & Software Survey: November 2020* [online]. 2020 [cit. 2020-12-21]. Dostupné z: https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam.

51. *The world's most advanced hand tracking* [online]. 2020 [cit. 2020-10-16]. Dostupné z: https://www.ultraleap.com/tracking/.

52. *UNet Deprecation FAQ* [online]. 2019 [cit. 2020-10-05]. Dostupné z: https://support.unity3d.com/hc/en-us/articles/360001252086-UNet-Deprecation-FAQ.

53. *Unreal Engine - Frequently Asked Questions* [online]. 2020 [cit. 2020-10-06]. Dostupné z: https://www.unrealengine.com/en-US/faq.

54. *Unreal Engine 4 Documentation - Class Creation Basics* [online]. 2020 [cit. 2020-10-06]. Dostupné z: https://docs.unrealengine.com/en-US/Gameplay/ClassCreation/index.html.

55. *Unreal Engine 4 Documentation - Gameplay Architecture* [online]. 2020 [cit. 2020-12-07]. Dostupné z: https://docs.unrealengine.com/en-US/ProgrammingAndScripting/GameplayArchitecture/index.html.

56. *Unreal Engine 4 is Free for Everyone* [online]. 2017 [cit. 2020-10-06]. Dostupné z: https://www.unrealengine.com/en-US/faq.

57. *Valve Index - Controllers* [online]. 2019 [cit. 2020-10-07]. Dostupné z: https://www.valvesoftware.com/en/index/controllers.

58. *Valve Index Impressions - A Necessary But Incremental Step For PC VR* [online]. 2019 [cit. 2020-10-07]. Dostupné z: https://www.gamespot.com/images/1300-3543578.

59. WACHS, Juan; STERN, Helman; EDAN, Yael; GILLAM, Michael; FEIED, Craig; SMITH, Mark; HANDLER, Jon. Gestix: A Doctor-Computer Sterile Gesture Interface for Dynamic Environments. In: SAAD, Ashraf; DAHAL, Keshav; SARFRAZ, Muhammad; ROY, Rajkumar (ed.). *Soft Computing in Industrial Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, s. 30–39. ISBN 978-3-540-70706-6.

60. WEINBERGER, Matt. *The rise and fall of Kinect: Why Microsoft gave up on its most promising product* [online]. 2018 [cit. 2020-10-13]. Dostupné z: https://finance.yahoo.com/news/downfall-kinect-why-microsoft-gave-183900710.html.

61. *What is BMW Gesture Control?* [Online]. 2020 [cit. 2020-10-14]. Dostupné z: https://www.bmwoflittlerock.com/gesture-control/.

62. *What is Magic Leap?* [Online]. 2018 [cit. 2020-10-14]. Dostupné z: https://developer.magicleap.com/en-us/learn/guides/design-why-magic-leap.

Příloha A

Seznam použitých zkratek

AI Artificial Intelligence

AR Augmented Reality

CAD Computer-Aided Design

CPU Central Processing Unit

EMG Electromyography

GB Gigabyte

GPU Graphics Processing Unit

HMD Head-Mounted Display

LED Light-Emitting Diode

NURBS Non-Uniform Rational B-Spline

OLED Organic Light-Emitting Diode

OSVR Open-Source VR

RAM Random Access Memory

UI User Interface

UML Unified Modeling Language

VR Virtual Reality

XR eXtended Reality

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `thesis.pdf` ..... text práce ve formátu PDF
- `exe` ..... adresář se spustitelnou formou DEMO projektu
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `demo` ..... zdrojové kódy DEMO projektu pro UE 4.25.4
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX