Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Realizace jednoduché BCI hry s podporou pro eye-tracker

Plzeň 2020 Ondřej Vaic

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Ondřej VAIC**
Osobní číslo:	**A17B0385P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Realizace jednoduché BCI hry s podporou pro eyetracker**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s měřením mozkové aktivity v neuroinformatické laboratoři na KIV.
2. Prostudujte vlastnosti snímačů mozkové aktivity a snímačů očního pohybu a zvolte vhodná reprezentativní zařízení.
3. Pro zvolená zařízení navrhněte jednoduché rozhraní mozek počítač (BCI) pro zařízení uvedená v bodě 2.
4. S ohledem na možnosti zařízení navrhněte vhodný typ hry a tuto hru implementujte.
5. Otestujte BCI aplikaci na reprezentativním vzorku experimentů a zhodnoťte dosažené výsledky práce.

Rozsah bakalářské práce:	**doporuč. 30 s. původního textu**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce:	**Ing. Petr Brůha** Katedra informatiky a výpočetní techniky
Datum zadání bakalářské práce:	**7. října 2019**
Termín odevzdání bakalářské práce:	**7. května 2020**

Radová

Doc. Dr. Ing. Vlasta Radová
děkanka

L.S.

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 15. října 2019

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2020

Ondřej Vaic

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petru Brůhovi, za odbornou pomoc při realizaci této práce. Dále bych chtěl poděkovat rodičům za podporu po celou dobu studia a všem, kteří se podíleli na otestování výsledné hry.

Abstract

This bachelor thesis aims to explore the possibilities of controlling computer games using eye–tracking and brain–computer interface (BCI) and then create a game in which these technologies will be used to control it. The theoretical part describes the principles of measuring brain activity and eye movement. Subsequently, the individual devices that are suitable for this purpose are examined. In the practical part, a game will be implemented in which the player will control the snowboarder using these technologies. The game is designed primarily for cognitive training and entertainment of patients who have limited ability to operate a computer as a result of a disability

Abstrakt

Tato bakalářská práce si klade za cíl prozkoumat možnosti ovládání počítačových her pomocí eye–trackingu a rozhraní mozek-počítač (BCI) a následně vytvořit hru, ve které budou tyto technologie využity k jejímu ovládání. V teoretické části práce jsou popsány principy měření mozkové aktivity a pohybu očí. Následně jsou prozkoumána jednotlivá zařízení, která jsou pro tento účel vhodná. V praktické části bude implementována hra, v níž lze ovládat snowboardistu těmito technologiemi. Hra je určena především pro kognitivní trénink a zabavení pacientů, kteří mají v důsledku postižení omezené možnosti ovládat počítač běžným způsobem.

Obsah

1 Úvod

O vývoj počítačových her se zajímám již více než čtyři roky. Téma této bakalářské práce jsem si vybral, protože si myslím, že je to skvělá příležitost někomu pomoci a zároveň se něco dozvědět o nových technologiích.

Většina počítačových her je v dnešní době ovládána pomocí klávesnice a myši, popřípadě ovladače. V posledních letech se výrazně zlevnila některá komerční zařízení pro eye–tracking a měření mozkové aktivity. Tato zařízení umožňují alternativní způsob ovládání počítačových her. Některé hry mohou tyto technologie využívat jako doplněk ke klávesnici a myši. Cílem této práce je vytvořit hru, kterou bude možné úplně ovládat bez přímého dotyku uživatele. Taková hra by mohla být atraktivní především pro pacienty, kteří jsou kvůli svému zdravotnímu stavu připoutáni na lůžku. Těmto pacientům může posloužit jako alternativní forma zábavy. Hra bude zároveň navržena tak, aby kromě zabavení mohla být použita i pro trénování mozku. Her, které obsahují elementy kognitivního tréninku a lze je ovládat eye–trackingem, je nedostatek. Tato práce si klade za cíl vytvořit hru, která pacienta zabaví a bude ho motivovat k dalšímu hraní a zlepšování se. Pacienti si tedy budou moci zábavnou formou trénovat své kognitivní funkce.

Bakalářská práce je členěna do dvou částí - teoretické a praktické. Teoretická část se zabývá technologiemi, které se týkají rozhraní mozek-počítač a eye–trackingu. Jsou v ní popsány principy, na kterých tyto technologie fungují, a jejich využití v dnešním světě. Dále jsou představena různá zařízení, která jsou dnes na trhu, uvádím jejich porovnání a následný výběr nejvhodnějších zařízení pro implementaci hry. Následuje část zabývající se herními enginy a návrh vlastní počítačové hry. Praktická část se věnuje implementaci samotné hry. Jsou zde popsány techničtější detaily vývoje hry, která byla otestována a upravena podle připomínek získaných při testování.

2 Měření mozkové aktivity na KIV

V rámci teoretické části jsem se seznámil s měřením mozkové aktivity v neuroinformatické laboratoři na KIV[1]. KIV disponuje dobře vybavenou laboratoří pro různé experimenty založené na měření mozkové aktivity. Mezi ně patří například systém pro rozpoznání aktivity, na kterou člověk myslí. Prováděl se zde výzkum zaměřující se na pozornost řidiče, je tu postaven i model železnice s vlakem, který je možné rozpohybovat pouhým soustředěním. V laboratoři je také zvukotěsná kabina, která je navíc odstíněna od okolního elektrického pole, a tím umožňuje přesnější měření EEG[2] signálu bez vnějšího rušení. Kabina je znázorněna na obrázku 2.1.

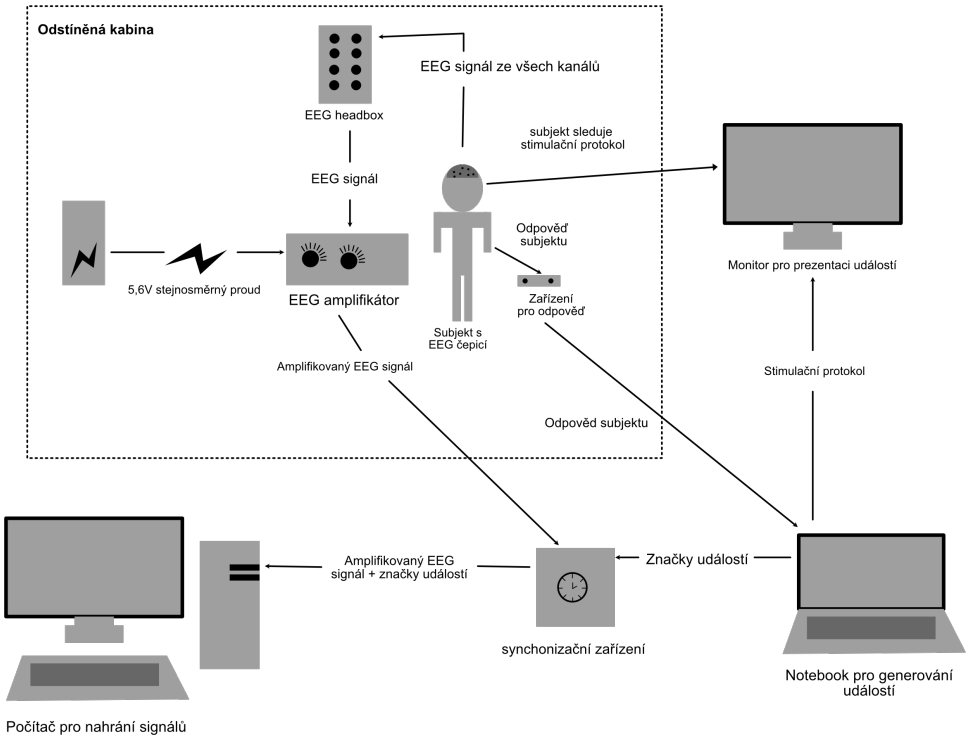
Infrastruktura EEG experimentu (obrázek 2.1), založeného na měření evokovaných potenciálů v neurolaboratoři na KIV, ukazuje sledovaný subjekt s nasazenou EEG čepicí, jak sleduje stimul[3] na monitoru. V odstíněné kabině je EEG signál zachycen, amplifikován a dále synchronizován s časovými údaji o zobrazovaném stimulu[4]. Tyto údaje jsou nakonec vyhodnoceny (například: na monitoru problikávají před subjektem různé aktivity a software vyhodnotí, na kterou aktivitu subjekt myslí[1][2]).

[1]Katedra informatiky a výpočetní techniky

[2]Elektroencefalografie

[3]Podnět pro mozkovou aktivitu, například nějaký obrázek.

[4]Kdy se co zobrazilo.



Obrázek 2.1: Diagram infrastruktury EEG experimentu v neurolaboratoři na KIV.[1]

2.1 Rozhraní mozek počítač

Rozhraní mozek-počítač (BCI)[5] označuje využití mozkové aktivity pro interakci člověka se strojem. Ovládání počítače pomocí myšlenek je velmi atraktivní koncept pro téměř kohokoliv. Kvalitní rozhraní mozku a počítače by přineslo nespočet zajímavých aplikací, které by ocenili především lidé s omezenými motorickými funkcemi. Tato zařízení jim mohou výrazně zlepšit kvalitu života. Zajímavé využití se nabízí také pro uživatele, kteří nemohou ovládat žádné svaly, a BCI by tedy byl jediný způsob, kterým by mohli rozhodovat o svých dalších aktivitách. Použití myšlenek pro interakci s počítačem by ale určitě ocenil a využil téměř každý. Přínos pro běžného uživatele může spočívat ve zjednodušení navigace v operačních systémech chytrých telefonů a desktopů, psaní pouze pomocí myšlenek, napojení na chytré spotřebiče v domácnosti a podobně. Takové aplikace by mohly výrazně zvýšit produktivitu díky snížení času stráveného ručním ovládáním daného přístroje.

Od spolehlivého zadávání rozličných příkazů počítači pomocí myšlenek

[5]Brain computer interface, dále BCI.

jsme však dnes ještě daleko a s velkou pravděpodobností ještě dlouho budeme. Zároveň se ale v posledním desetiletí objevilo hned několik relativně dostupných zařízení, která dokáží měřit základní mozkovou aktivitu. Ta poté může být interpretována a použita pro velmi omezené ovládání některého softwaru. Především právě těmito zařízeními dostupnými běžnému uživateli se tato práce bude zabývat, přičemž dostupností je míněna cena mezi 2 000 až 10 000 Kč a uživatelovy minimální technické znalosti. Tato zařízení jsou již dnes využívána pro terapeutické účely po zraněních, která ovlivnila fungování nervové soustavy, pro zlepšení koncentrace jedinců s poruchou pozornosti nebo také v herním průmyslu pro nadšence této technologie.

2.2 Měření mozkové aktivity

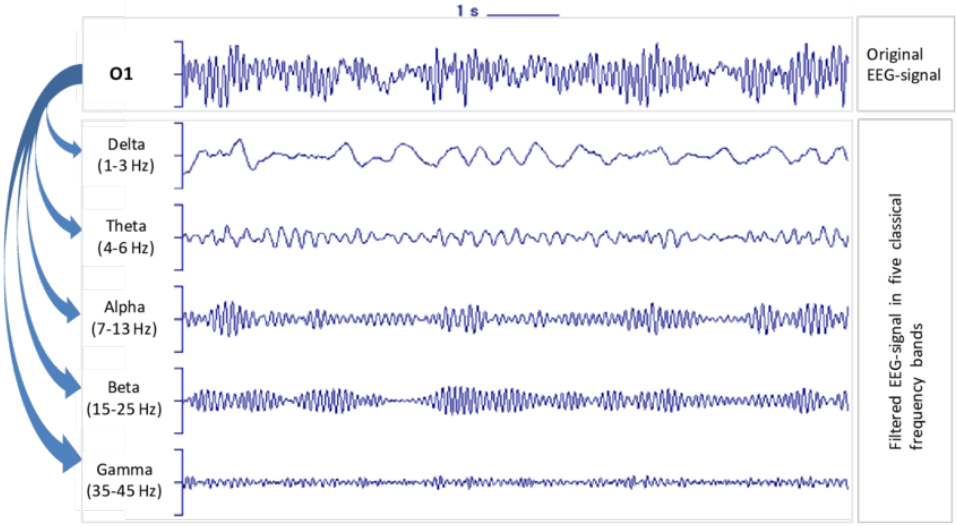
Fungování mozku je do dnes relativně málo probádaná oblast. Není tedy divu, že extrahování jeho aktivity pomocí přístrojů je náplní aktivního výzkumu. Mozek při své funkci vytváří měřitelné elektrické signály, jež mají různé amplitudy a frekvence, které je možné dále interpretovat. Toto měření se nazývá elektroencefalografie neboli EEG a bylo poprvé naměřeno německým psychiatrem Hansem Bergerem v roce 1924.[47][46]

2.2.1 Princip EEG

Elektroencefalografie funguje na principu měření proměnných elektrických potenciálů, které jsou generovány mozkovou aktivitou. Tyto potenciály jsou měřitelné elektrodami, nejlépe přímo na povrchu mozku, nebo na povrchu skalpu (viz podkapitoly Invazivní EEG a Neinvazivní EEG). Signál naměřený na povrchu skalpu je velmi slabý 10–300 μV o frekvenci 0,1–60 Hz. Jednou z výzev je tedy signál kvalitně zachytit a odfiltrovat od různých artefaktů (viz podkapitola Artefakty). Signál z mozku je vzorkován s určitou frekvencí, a tím vzniká diskrétní signál. Z toho je ale velmi těžké něco vyčíst, používá se tedy diskrétní Fourierova transformace k rozdělení hrubého EEG na jeho jednotlivé složky.[47][1]

Na obrázku 2.2 lze vidět v horní křivce hrubý signál EEG a pod ním jeho rozklad na složky delta, theta, alpha, beta, gamma (viz dále).

Signál se dělí do několika kategorií podle dané frekvence a amplitudy. Následně mohou být aplikovány klasifikační algoritmy nebo metody strojového učení pro přiřazení významu danému signálu.[1]



Obrázek 2.2: Hrubý EEG signál rozdělen na jednotlivé složky. [44]

Kategorizace vln

Vlny je možné rozdělit do několika základních kategorií. Jejich frekvence a amplitudy jsou uvedeny v tabulce 2.1.[47]

Rytmus	Frekvence(Hz)	Amplituda(μV)
Alfa	$8-13$	$30-80$
Beta	$14-40$	$10-30$
Gamma	$36-44$	≈ 10
Delta	<4	$10-300$
Theta	$4-7$	<30

Tabulka 2.1: Tabulka vln EEG.

Artefakty

Jako artefakty jsou označovány EEG záznamy, které nemají původ v mozkové aktivitě. Artefakty mohou pocházet jak z okolních elektrických spotřebičů, tak přímo z provozu EEG zařízení. V ideálním případě je tedy měření prováděno v komoře, která je odstíněna od okolního světa.[47]

Dalším zdrojem artefaktu v záznamu EEG je biologická aktivita snímaného jedince. Vzhledem k tomu, že elektrické potenciály jsou generovány také zatínáním svalů, rušení způsobuje i srdeční aktivita, mrkání nebo jakákoliv mimika. Těmito signály se zabývá elektromyografie (EMG) a speciálně

elektrookulografie (EOG), která zkoumá signály způsobené pohyby oka a mrkáním.

Některá komerční zařízení využívají artefakty spolu s EEG jako jeden ze způsobů ovládání. Oproti EMG jsou jednodušší a jednoznačnější pro klasifikaci a například mrknutí je tak možné vcelku spolehlivě používat jako kontrolní prvek v BCI.[47][10]

2.2.2 Invazivní EEG

Invazivní metody měří signál přímým položením elektrod na mozek, nebo dokonce zavedením elektrod do mozku. Je tedy potřeba chirurgického zákroku. Tato metoda se používá pro určení zón, které jsou zodpovědné za vyvolávání epileptických záchvatů. Tyto zóny jsou následně odoperovány. Elektrody jsou položeny pouze na malou část mozku, ze které je snímán velmi dobrý (v porovnání s neinvazivními metodami) signál. Část zájmu je nejprve určena na základě výsledků neinvazivního vyšetření EEG a magnetické rezonance.[50]

Tato metoda sice poskytuje velmi dobrý signál, ale její využití pro BCI je ze zřejmých důvodů nevhodné.

2.2.3 Neinvazivní EEG

Při neinvazivním snímání jsou elektrody položeny na skalp. Existují dva druhy EEG snímačů, vlhké a suché. Pro vlhké snímače je potřeba aplikovat gel na místa elektrod. Ten zvyšuje vodivost, a tím i kvalitu signálu. Zároveň je ale tato procedura nekomfortní a časově náročnější. Vlhké EEG snímače se tak používají především pro důležitá diagnostická měření, například k diagnostice epilepsie.

Vlhkými snímači jsou profesionální čepice s vyšším počtem elektrod[6]. Bývají však dražší[7], tudíž jsou nevhodné pro tuto práci, zaměřenou především na jednoduché a dostupné BCI pro terapii běžného pacienta.[42]

[6]Je nutné podotknout, že některé novější profesionální čepice umožňují suché i vlhké snímání.

[7]Epoc od firmy Emotiv je nejspíš nejlevnější a vyjde na 799 $, profesionálnější zařízení mohou stát více než 25 000 $.

Obrázek 2.3: Ukázka vlhkého EEG zařízení s větším množstvím elektrod[21]

Suché EEG snímače bývají méně přesné, zato jsou levnější a jednoduché na použití. Fungují na stejném principu jako vlhké čepice, využívají většinou méně elektrod. Často také snímají některé EMG signály, které rozšiřují možnosti ovládání. Jsou tedy ideální pro využití v této práci.

V posledních letech se na trhu objevilo hned několik takových zařízení. V další části práce bude uveden výběr z těchto zařízení a provedu jejich vzájemné porovnání.[42]

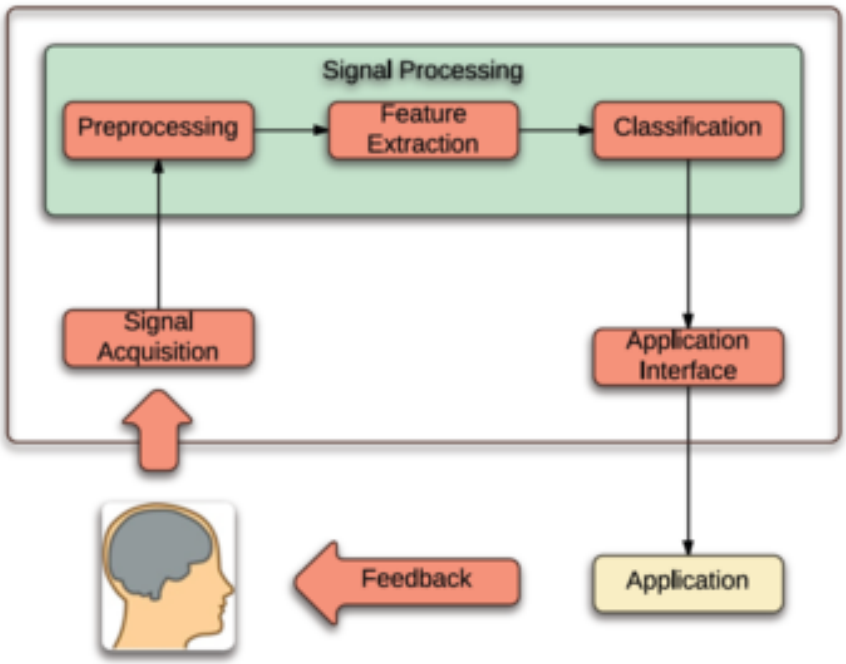
2.3 Možnosti ovládání pomocí BCI

Existuje několik způsobů, jak lze pomocí EEG signálu ovládat software. Rozpoznání toho, jakou akci chce uživatel vykonat, nebývá vždy jednoznačné. Přesnost rozpoznání závisí i na samotném jedinci a je možné ji zlepšovat tréninkem. Ten, kdo si nasadí EEG headset poprvé a pokusí se ovládat aplikaci nebo hru, může mít ze začátku velké problémy. Při vývoji je tedy důležité koncipovat hru tak, aby uživatele neodradila svou obtížností, a postupně obtížnost zvyšovat v závislosti na schopnostech jedince. Dále jsou uvedeny

čtyři způsoby, jimiž je možné ovládat software pomocí EEG signálu.

Následující diagram 2.4 ukazuje jednotlivé části BCI. Aplikace uživateli předává informace (nejčastěji z monitoru). Jejich zpracování uživatelem je zachyceno pomocí EEG signálu, který je následně zpracován a vyhodnocen. Podle toho je rozhodnuto, jak se bude aplikace chovat.

V neurolaboratoři na KIV jsou dostupná k zapůjčení tři zařízení pro realizaci BCI. Jsou to Mindwave Mobile, Muse a Emotiv. Jejich porovnání a výběr jednoho zařízení uvádím níže (kapitola Dostupná zařízení).



Obrázek 2.4: Diagram znázorňující průběh interakce s BCI.[22]

2.3.1 Soustředění

S mírou soustředění se dají vymyslet velmi zajímavé hry a hardwarové hračky. V hardwarových zařízeních může být soustředění použito například pro ovládání výšky vrtulníčku nebo rychlosti modelu vláčku, v počítačových hrách může ovládat rychlost autíčka, dva hráči se mohou přetahovat o virtuální lano a podobně. Soustředění může být využito také jako vedlejší prvek hry, například když hráč přijde k pochodni a začne se soustředit, pochodeň se zapálí.[5][6][45]

2.3.2 Motor imagery

Další možností, pro kterou je dnes tvořen velký počet výzkumů, je takzvané Motor imagery. Při přemýšlení jedince nad specifickým pohybem jsou u něj generovány podobné vlny EEG jako při samotném pohybu. Pomocí klasifikačních algoritmů je možné míru přemýšlení nad určitým pohybem zachytit a využít jako ovládací prvek. Uživatel poté může například zatáčet do stran přemýšlením o pohybu levou, či pravou rukou. Výhodou oproti následující metodě je, že není nutné kalibrovat ovládací prvky pro konkrétního uživatele.[41][4]

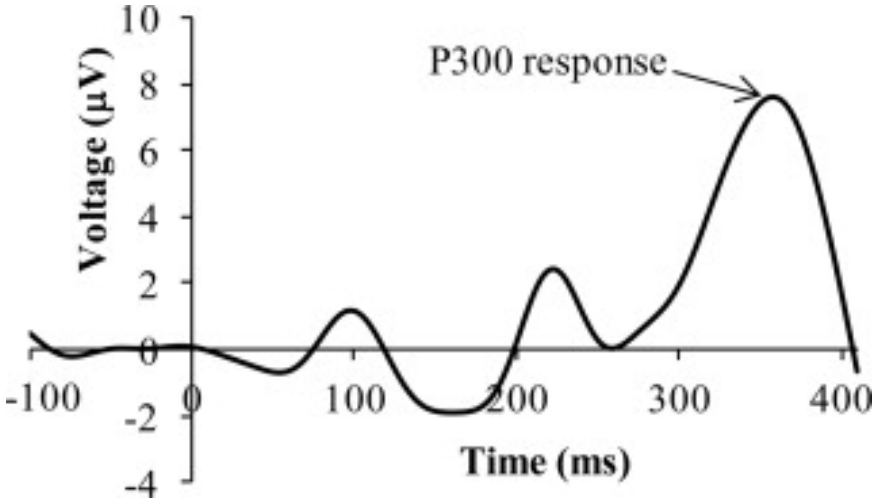
2.3.3 Klasifikace stavu mysli

Systém EPOC+ je možné naučit, jak vypadá EEG signál uživatele při nějaké myšlence. Je ale nutné systém na tuto myšlenku zkalibrovat. EPOC+ poté klasifikačními algoritmy může určit míru toho, jak moc uživatel přemýšlí nad danou myšlenkou. Jedná se o velmi slibný ovládací prvek, ale záleží na tom, jak přesný tento systém je. [3][8]

2.3.4 P300

P300 je jev, o kterém můžeme nalézt první zmínky již v první polovině 60. let 19. století. Jedná se o takzvaný evokovaný potenciál[8]. Je to signál, který se objeví přibližně 300 milisekund od stimulace spojené s kategorizací. V praxi to může vypadat například tak, že na obrazovce náhodně problikávají různá písmena. Když písmeno, které chce člověk komunikovat problikne, přibližně po 300 milisekundách se objeví jev P300 v EEG záznamu. Tato aplikace se dá použít například pro vybírání obrázku aktivity, kterou chce dělat pacient, jenž nemá jinou možnost komunikace. Zefektivnění takového výběru je možné uspořádáním písmen nebo obrázků do matice a problikáváním řádků a sloupců.[47][1]

[8]Výchylka v EEG na základě podnětu zvenčí.



Obrázek 2.5: Ukázka, jak může vypadat evokovaný potenciál P300.[51]

3 Eye–tracking

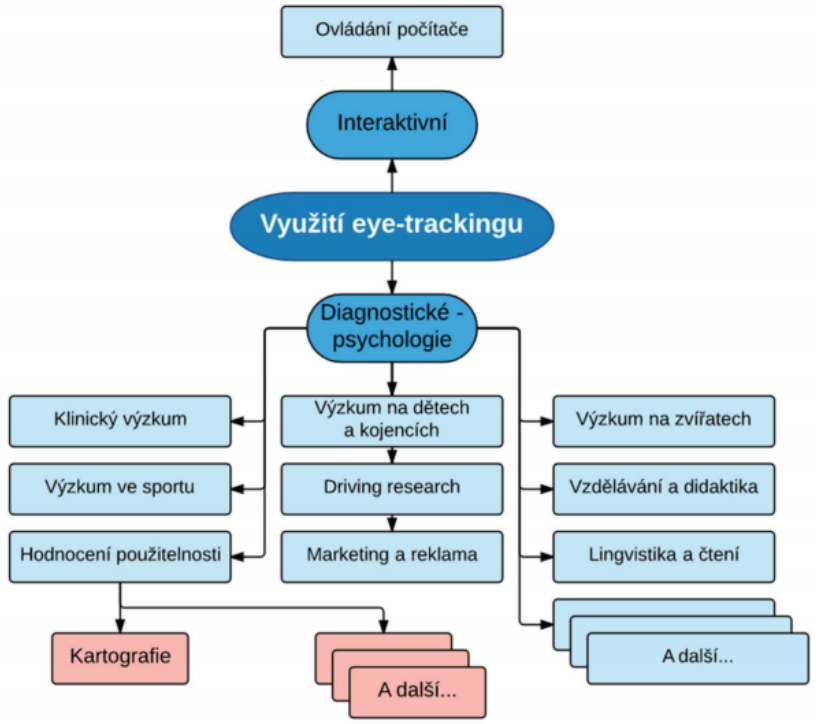
Pojmem eye–tracking se označují technologie, které snímají pohyb očí, tedy sledování místa, na něž se uživatel dívá. Spolehlivé ovládání počítače mozkem v běžném životě je dnes stále ještě daleko od reality, zato ovládání počítače očima je pro mnoho lidí každodenní činnost, neboť technologie eye–tracking umí výrazně ulehčit život lidem s omezenými motorickými schopnostmi.

Eye–tracking má ale i další využití. Je velké množství odborných studií, které se zabývají sledováním toho, kterou část webové stránky či aplikace uživatel sleduje. Tyto informace tak mohou pomoci optimalizovat software, aby byl uživatelsky co nejpřívětivější. Z pochopitelných důvodů je také zajímavé sledovat, kam se dívají zákazníci v obchodě. Eye–tracking může také pomoci diagnostikovat některé choroby (viz Elektrookulografie). Zajímavé využití spočívá i v technologii foveated rendering, která umožňuje optimalizaci renderingu her. Spočívá v tom, že nejkvalitněji jsou renderovány pouze ty části obrazovky, na které se uživatel přímo dívá. Ve sportu je možné uplatnit eye–tracking například pro sledování koordinace oka a těla. Tyto informace mohou posloužit pro trénink profesionálů nebo k porovnání profesionálů a amatérů. Dále může být zajímavé sledovat oči řidiče a zkoumat jeho pozornost, například při interakci s GPS.[49][48][40][52]

Aktivit, které je možné provádět s eye–trackingem je mnoho, jelikož zrak je nejdůležitější smyslem, který člověk používá.

Na diagramu 3.1 jsou vidět různá využití eye–trackingu. Je možné je rozdělit do dvou kategorií – první, zaměřující se na diagnostiku a výzkum, a druhé, interaktivní, tedy zabývající se ovládáním počítače[1].[49]

[1]Ovládán může být například operační systém nebo speciální software, například hry.



Obrázek 3.1: Diagram možných využití eye–tracking.[49]

3.1 Metody eye–trackingu

3.1.1 Magnetookulografie

Magnetookulografie je jedna z metod, které spoléhají na přímý kontakt s okem. Jedná se o velmi přesnou metodu měření pohybu očí. Subjektu je nasazena kontaktní čočka s cívkou. Okolo je oscilující magnetické pole, které v cívce při pohybu indukuje elektrické napětí.

Při magnetookulografii je měřena pouze relativní poloha oka vůči hlavě. Pro určení bodu pohledu musí být měřena i poloha hlavy. Celá procedura je ale nepříjemná a je zapotřebí odborná asistence a poměrně velká aparatura. Komplikaci může způsobit i to, že po delším měření může drát z čočky vypadnout.[49][48]

3.1.2 Elektrookulografie

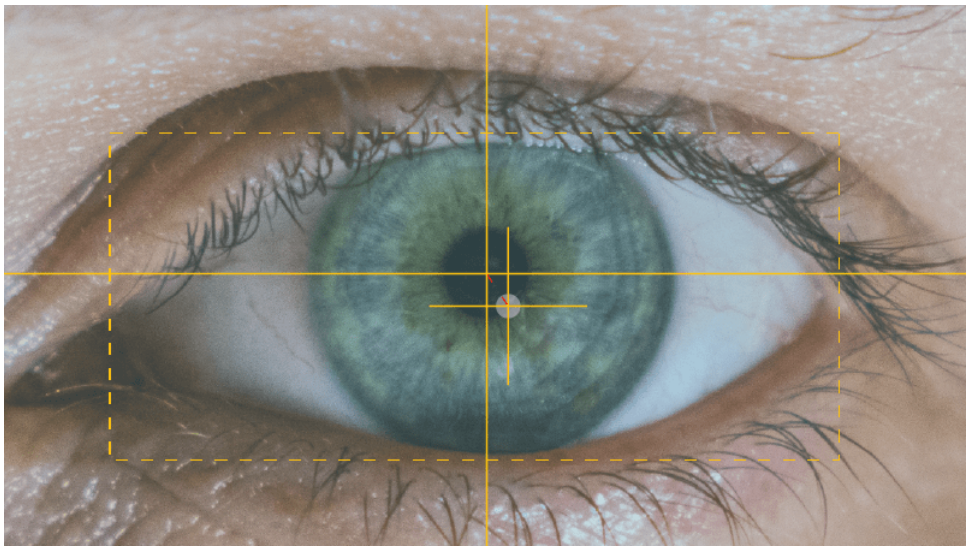
Tato technika je založena na snímání rozdílů elektrického napětí, které generují svaly oka při stažení. Kolem očí jsou rozmístěny elektrody snímající tuto aktivitu. Elektrookulografie oproti ostatním metodám však není tak přesná.

Hodí se spíše pro relativní pohyby oka. Pomocí této metody je možné diagnostikovat některé choroby, které ovlivňují pohyb očí, například Alzheimerovu chorobu, Parkinsonovu nemoc nebo schizofrenii. Výhodou metody je možnost eye–trackingu i ve spánku.[49][48]

3.1.3 Videookulografie

Pro tuto práci je nejdůležitější videookulografie. Jedná se o bezkontaktní technologii. Eye-tracker založený na videookulografii se skládá z infračervených světel, která osvětlují oči, kamery a vyhodnocovacího softwaru. Kamera zaznamenává oči. Algoritmy zpracování obrazu určí polohu zorničky a odrazu infračerveného světla. Tyto odrazy (obrázek 3.2) se nazývají Purkyněho obrázky, podle českého vědce Jana Evangelisty Purkyně. Odraz zůstává téměř na stejném místě, zatímco zornička se pohybuje. Z těchto informací je již možné vypočítat, kam se uživatel dívá. Infračervené světlo je zvoleno proto, že uživateli nevadí a je možné jej odlišit od okolního světla. Kvůli individuálním vlastnostem očí každého jedince je vhodné eye-tracker před použitím kalibrovat.[49][48][43][12][11]

Tyto eye-trackery mohou být buď připevněny pod monitorem a zachycovat, kterou část obrazovky uživatel sleduje, nebo jsou zabudovány ve speciálních brýlích. Takové brýle mají tu výhodu, že jsou přenosné, a najdou uplatnění například ve sledování toho, kam se dívá zákazník v obchodě.



Obrázek 3.2: Střed zornice, Purkyněho obrázek a vzdálenost mezi nimi. [43]

3.2 Pohyby očí

Pohyby očí můžeme rozdělit do několika kategorií. Pro zkoumání pohybu očí či implementování systému, který je ovládán zrakem, je dobré znát alespoň základní pohyby očí.[49][48]

3.2.1 Fixace

Fixace je relativní klid oka. Například při čtení jednoho slova se oko zaměří na jedno písmeno a poté skočí na další. Zajímavé je, že ani během fixace není oko úplně v klidu (viz Třes).[49][48]

3.2.2 Sakády

Mezi jednotlivými fixacemi se oko většinou pohybuje takzvanými sakádami. Jsou to velmi rychlé pohyby z jednoho místa na druhé. Vizuální vjem je během sakád mozkem potlačen. Sérií sakád a fixací například čteme nebo si prohlížíme obrázek. Mezi jednotlivými sakádami je obvykle přibližně 300 ms fixace.[49][48]

3.2.3 Sledování

Sledováním je myšlen plynulý pohyb oka, které sleduje pohybující se cíl. Neobjevuje se však při sledování statické scény (zde se spíše uplatňují sakády a fixace).[49][48]

3.2.4 Třes

Třes je jemný pohyb oka, zatímco je ve fixaci. Tento pohyb je nepatrný a má nízkou amplitudu (v řádech úhlových sekund) a vysokou frekvenci přibližně 30–80 Hz. Jednoznačný důvod třesu (tremoru) není znám. Je možné, že jde o nepřesnost svalů, které drží oko. Dalším důvodem může být neustálé dráždění receptorů sítnice, aby sledovaný obraz nevybledl.[49][48]

4 Dostupná zařízení

4.1 BCI

4.1.1 Mindwave Mobile

Mindwave Mobile je zařízení od firmy Neurosky. V současné době je na trhu již druhá verze Mindwave Mobile, jejíž cena pohybuje kolem 99 $. Tento headset snímá pouze pomocí dvou elektrod, jedné na čele a druhé, referenční, na ušním lalůčku. Z naměřených hodnot můžeme přes bluetooth získat základní EEG vlny (alpha a podobné). Zároveň je možné získat i hrubý EEG signál. Mindwave Mobile dále umožňuje získat hladinu soustředěnosti a takzvané meditace (klid uživatele). Spolehlivou možnost ovládání poskytuje indikátor mrknutí, který z artefaktů v EEG signálu pozná, jestli uživatel mrkl. Většina her využívajících Mindwave Mobile jsou hry na zlepšení pozornosti, existují však i hry na podporu meditace.[9]

4.1.2 Emotiv EPOC+

EPOC+ od firmy Emotiv oproti Mindwave Mobile má 14 senzorů mozkové aktivity, které jsou rozloženy po celém skalpu. Dá se tedy očekávat, že EEG signál bude kvalitnější. Zařízení disponuje i gyroskopy, které detekují pohyby hlavy. Na druhou stranu je EPOC+ samozřejmě dražší headset, samotný stojí 799 $. Výrobce dále nabízí dva plány pro jejich API[1], první je zdarma a druhý, který umožňuje získat navíc i hrubý EEG záznam, za 55–99 $ za měsíc. EPOC+ rozpoznává některé výrazy obličeje pomocí EMG, pohyby hlavy a základní druhy EEG vln. EPOC+ dále umožňuje specifikování vzorů myšlenek, které systém poté může rozpoznávat, a díky tomu je uživateli umožněno ovládat některé aplikace.[8]

4.1.3 Muse 2

Další dostupné zařízení, které měří mozkovou aktivitu, je Muse 2. Jedná se o střední cestu mezi výše zmíněnými zařízeními. Disponuje dvěma senzory na čele, dvěma za ušima a třemi referenčními. Muse 2 také dokáže určit tep, frekvenci dechu a pohyb hlavy pomocí gyroskopu. Objednat se dá za

[1] Application Programming Interface, rozhraní, které výrobce poskytuje programátorům.

269 $. Výrobce bohužel v současné době nepodporuje žádné API. Muse 2 upřednostňuje pohodlí při nošení. Celý produkt je cílený především na podporu meditace. Uživatel si má zařízení nasadit spolu se sluchátky a začít meditovat. Ze sluchátek poté dostává audio feedback o tom, jak dobře se mu meditování daří.[7]

Obrázek 4.1: Zprava: Muse 2, Mindwave Mobile 2, Epoc+.[7][9][8]

Zařízení	Cena	Počet elektrod	EMG	API
Mindwave Mobile	99$	1+1	Rozpoznání mrknutí	Ano, zdrama
Muse 2	229 $	4+3	Žádné	Ne
Epoc+	799 $	14+2	Rozpoznání: mrknutí, úmyslné mrknutí, překvapení, zamračení, úsměv, zatnutí zubů, smích, ušklíbnutí	Ano, 2 licence, jedna zdrama, druhá s možností hrubého EEG 55 – 95 $ za měsíc

Tabulka 4.1: Tabulka porovnání EEG zařízení

4.1.4 Shrnutí

Pro porovnání jednotlivých zařízení jsem vytvořil tabulku 4.1. Na základě této tabulky a po poradě s vedoucím práce bylo pro tuto práci zvoleno zařízení Mindwave Mobile od firmy Neurosky. Oproti Muse 2 umožňuje rozpoznání mrknutí. Epoc+ je sice s velkou pravděpodobností ve všech ohledech lepší zařízení, je ale 8krát dražší. Je tedy velmi nepravděpodobné, že by ho někdo, kdo by si chtěl hru zahrát, vlastnil nebo si ho snadno pořídil. Další výhodou Mindwave oproti Muse 2 je existence API, které je navíc zcela bezplatné. S Mindwave Mobile mám navíc praktickou zkušenost s vývojem jedné hry v Unity. Jedna výhoda oproti Epoc+ spočívá v tom, že výrobce Epoc+ si účtuje měsíční poplatky za přístup k hrubému EEG signálu. Myslím si, že i přes slabší kvalitu signálu je pro tuto práci Mindwave Mobile dostačující, a navíc jednoduše dostupný.

4.1.5 Další informace o Mindwave Mobile

Zkušenost s tímto zařízením mám už z jedné hry, realizované v rámci předmětu KIV/ZSWI.

Připojení uživatelem

Připojení tohoto zařízení pro mě jako uživatele, když jsem se s Mindwave Mobile seznamoval, nebylo nijak složité. Na Windows stačí stáhnout program, který běží jako daemon, a zapnout Bluetooth. Zařízení stačí zapnout a ono se samo připojí. Po nasazení na hlavu a umístění elektrod začne rovnou posílat data. Pro připojení k telefonu s Androidem není ani potřeba stahovat žádný další program.

Připojení programátorem

Připojení z pohledu programátora bylo taktéž jednoduché. Výrobce poskytuje API pro komunikaci přes sockety. Data přijatá ze zařízení musejí být ale nejdřív parsovaná. Pro zjednodušení a zrychlení vývoje bude použit balíček do Unity zveřejněný na github od uživatele DaCookie, který řeší připojení a parsování. S tímto balíčkem již mám zkušenosti, v mé předchozí práci fungoval dobře.[26][27]

Technické parametry

Napájení	1 AAA baterie
Výdrž baterie	8 hodin
Vzdálenost uživatele	max 10 m
Připojení	Bluetooth
Vzorkovací frekvence	512 Hz

Tabulka 4.2: Technické parametry Mindwave Mobile.[9]

Základní technické parametry popisuje tabulka 4.2. Zařízení dále disponuje dvěma elektrodami, jednou referenční, která slouží pro rozlišení artefaktů, a jednou pro záznam EEG. Referenční je připnuta na ušní lalůček a druhá elektroda je položena na čelo. Důležitou součástí zařízení je modul TGAM. Ten zpracovává hrubé EEG; ze zařízení poté dostáváme hrubý EEG signál, spektra vln (alpha, beta a podobně), kvalitu signálu a míru soustředění, meditace a mrknutí.[24][28]

4.2 Eye–tracking

4.2.1 Tobii

Tobii je dnes největší firma zabývající se eye–trackingem. Byla založena v roce 2001, sídlí ve Švédsku a zaměstnává 1200 lidí. Tobii se dělí se na tři větve: Tobii Dynavox, Tobii Pro, Tobii Tech[13]

Tobii Dynavox

Tobii Dynavox se zaměřuje na technologie, které asistují lidem s různým postižením. Jejich produkty jsou především kompletní řešení - software, eye-tracker a případně i tablet nebo notebook pro bezkontaktní interakci s počítačem. Některé jejich produkty využívají i rozpoznávání řeči.[14]

Tobii Pro

Tobii Pro dodává eye–trackingová řešení pro výzkumné potřeby. Nabízí eye-trackery podobné Tobii 4C, který je zmíněn níže, ale přesnější a dále Tobii Pro SDK[2] a Tobii Pro Lab, což je software, který zjednodušuje výzkum (umožňuje jednoduše vizualizovat data z eye–trackingu). Zajímavým produktem jsou také brýle Tobii Pro Glasses 2. Ty natáčí to, co uživatel vidí, a zároveň určují, kam se zrovna dívá. Tyto brýle je možné použít například pro výzkumy zaměřené na sport nebo, jak už bylo zmíněno, na sledování očí zákazníka supermarketu.[16]

Tobii Tech

Tobii Tech řeší integraci eye–trackingu do konzumní elektroniky jako jsou monitory, laptopy, headsety virtuální reality nebo také chytrá auta. Tobii Tech prodává též samostatné eye-trackery, které stačí zapojit do USB a položit pod monitor. Jejich současný eye-tracker 4C je jediný eye-tracker zaměřený na běžného uživatele na trhu. Tobii 4C se dá pořídit za cenu nižší než je 6 000 Kč.

Toto zařízení je určeno především pro vylepšení herního zážitku; existuje malé množství experimentálních her, které jsou ovládány pouze očima. Nemalé množství AAA[3] her ale implementovalo podporu pro eye–tracking v různých formách. Příkladem může být například zamíření zbraně na místo pohledu, asistence ovládání kamery nebo adaptivní grafické efekty.[15]

[2]Software developement kit, tedy sada vývojových nástrojů.

[3]Hry s velkým rozpočtem a vysokou kvalitou, například Assasin's Creed nebo Kingdom Come Deliverance od českého herního studia Warhorse Studios.

4.2.2 Další eye–trackingový hardware

Gaze Point

Gaze Point je kanadská společnost zabývající se eye–trackingem. Podle jejích produktů se zaměřuje především na výzkum a firmy, které chtějí optimalizovat interakci uživatele s jejich softwarem. Nabízejí různé balíčky pro výzkum a UX[4]. Tyto balíčky, které obvykle obsahují eyetracker a software pro zpracování výstupu, stojí mezi 1 600 $ a 4 000 $. Firma dále nabízí dva samotné eyetrackery podobné Tobii 4C, které uživatel připevní pod monitor. Levnější s frekvencí snímků 60 Hz vyjde na 695 $, dražší s frekvencí 150 Hz stojí 1 995 $.[18]

EyeTech Digital Systems

EyeTech Digital Systems se zabývají eye–trackingem pro mnoho využití. Podíleli se na mnoha zajímavých projektech v těchto oblastech: asistenční systémy pro postižené, výzkum, medicínská zařízení[5], zařízení pro bezpečnost a zaškolování[6], automobilový průmysl[7], zábava, eye–tracking pro chytré telefony a eye–tracking pro virtuální realitu. Vyvinuli i vlastní eye–trackingový chip, který nabízejí k zakomponování do vlastního hardwaru. Výhodou je, že všechny výpočty jsou provedeny na tomto chipu, tudíž není potřeba PC. Dále nabízejí dva eyetrackery pro výzkum a jeden systém pro jedince s postižením. Bohužel ze všech produktů je cena uvedena jen u jednoho, a to u eyetrackeru VT3 MINI OEM, který je nabízen za 3 000 $.[19]

Pupil Labs

Pupil Labs je ze tří výše uvedených nejmladší firmou založenou teprve nedávno, v roce 2014. Podle svých webových stránek se zaměřují především na výzkum a herní průmysl. Jejich produkty se dělí do tří kategorií: VR/AR, Core, Invisible. VR/AR jsou produkty pro zabudování eye–trackingu do VR a AR headsetů. Core jsou brýle, které snímají, kam se uživatel dívá, spolu s videozáznamem okolí. Jedny brýle Core jsou v neurolaboratoři na KIV. Podle vedoucího práce s nimi nejsou moc dobré zkušenosti. Jsou nepřesné a přehřívají se. Invisible jsou brýle podobné Core; hlavním rozdílem je, že vypadají jako běžné sluneční brýle a nepotřebují kalibraci. Doplňky do VR/AR

[4]User experience - zajištění dobrého uživatelského prožitku.

[5]Například diagnostika neurologických onemocnění.

[6]Autentizace pomocí očí, část detektoru lži nebo systém pro zaškolení zaměstnanců, kteří obsluhují rentgen na letištích.

[7]Software pro určení pozornosti řidiče nebo ovládání palubního displaye eyetrackerem.

stojí 1 400 € a 1 750 €, Core je možné koupit za 1 840 € a Invisible za 5 500 € nebo 4 800 € pro akademické účely.[17]

4.2.3 Shrnutí eye–trackingového hardwaru

Po prozkoumání výše uvedených zařízení a po domluvě s vedoucím práce bylo rozhodnuto, že pro implementaci hry bude použit eyetracker od firmy Tobii, typ 4C. Je to bezkonkurenčně nejlevnější řešení ze všech výše uvedených a navíc existuje oficiální SDK pro herní engine Unity zdarma. Pro většinu výše zmíněných eye-trackerů platí, že si výrobce účtuje velké sumy za zpřístupnění API, navíc nebývá přímá podpora pro Unity. Další výhodou je také, že tento eye-tracker je dostupný v neurolaboratoři na KIV. Oproti některým eye-trackerům je trackování jistě sice slabší, pro účely této práce je ale přesnost dostačující. Výhodou je i to, že s tímto eyetrackerem mám už vlastní praktické zkušenosti a vím tedy, že vývoj hry bude bez problémů.

4.2.4 Další informace o Tobii 4C

Stejně jako s Mindwave Mobile mám už s tímto zařízením zkušenost ze hry, realizované v rámci předmětu KIV/ZSWI.

Připojení uživatelem

Z pohledu uživatele je velmi jednoduché zařízení připojit. Stačí stáhnout software z internetových stránek Tobii a připojit eye-tracker přes USB port. Eyetracker musí být umístěn pod monitorem. Poté je potřeba zařízení zkalibrovat na uživatelovy oči. Kalibrace probíhá tak, že se uživateli zobrazují tečky na obrazovce a on se na ně má dívat. Poté je už vše připraveno pro běžné použití. Celý proces trvá pár minut a je velmi jednoduchý. To je pro použití běžným uživatelem zásadní. Výhodou je také to, že uživatel může oproti některým eye-trackerům hýbat volně hlavou. Dále je výhodou, že není nutné zařízení pokaždé kalibrovat.

Připojení programátorem

Integrace do herního enginu Unity je z pohledu programátora taktéž velmi jednoduchá. Na Unity Asset Store[8] je bezplatně zvěřejněn balíček, který je velmi jednoduché používat. Balíček, který je nutné naimportovat do projektu, obsahuje i množství ukázek různých využití. Dále je už čtení pozice

[8]Online obchod integrovaný do Unity.

na obrazovce, na kterou se uživatel dívá, stejně jednoduché jako čtení pozice myši přes nějakou knihovní funkci.

4.2.5 Technické parametry

Váha	95 g
Délka	33,5 cm
Vzdálenost uživatele	50 - 92 cm
Připojení	USB
Vzorkovací frekvence	90 Hz
Trackování hlavy	ano
Zatížení CPU	+-5% (Core i7, 9th Gen)

Tabulka 4.3: Technické parametry Tobii 4C.[31]

Tabulka 4.3 popisuje základní parametry tohoto eye-trackeru. Dále bych chtěl zmínit, že z mé zkušenosti má eye-tracker dostatečnou přesnost (přibližně na 3 cm). Funguje dobře, i když uživatel nosí brýle, nebo má jedno oko zavřené. Eyetracker jsem zkoušel na různých velikostech obrazovky a poměrech stran a nikdy jsem se nesetkal s žádnými problémy. Nejmenší obrazovka měla 15 palců úhlopříčku a poměr stran 16:9, největší měla 34 palců a poměr 21:9.

Obrázek 4.2: Eyetracker Tobii 4C.[16]

5 Herní enginy

S herními enginy a vývojem her mám zkušenosti, jelikož se tomu již více než čtyři roky věnuji.

Herní engine je takový balíček funkcionality, která je společná většině her. „*...Renderují svět, počítají fyziku, přehrávají zvuky a tak dále. Enginy jsou všechen kód, který není specifický vaší hře a může být potencionálně znovu použit v jiné hře.*“[20]

Herní enginy výrazně urychlují vývoj počítačových her. Dříve si herní studia vytvářela vlastní herní enginy. I přestože si některá herní studia stále vytváří vlastní enginy dodnes, častější je využití některého z komerčních enginů. Šetří to čas i peníze na vývoj, navíc překonat nějaký komerční engine je velmi obtížné vzhledem k jejich sofistikovanosti. Další výhodou pro firmy je, že nemusí tolik zaškolovat nové vývojáře v jejich enginu, protože mohou zaměstnat vývojáře, kteří s daným enginem již mají zkušenosti. Naopak výhodou vlastního enginu je, že není třeba platit žádnou licenci a vývojáři přesně vědí, s čím pracují a co jak přesně funguje.

Dále budou uvedeny hlavní komponenty většiny herních enginů a stručně popsán jejich význam. Poté budou zmíněny nejpopulárnější herní enginy a výběr jednoho z nich pro implementaci praktické části této práce.

5.1 Komponenty herních enginů

5.1.1 Grafika

Snad každý herní engine poskytuje nástroje pro vykreslování grafiky. Rendering může být pouze 2D obrázků (spritů) nebo většinou i 3D modelů. Dnešní herní enginy poskytují sofistikované renderovací pipeliny které vykreslují 3D modely velmi rychle a kvalitně. Dále do grafiky patří částicové systémy, které mohou simulovat různé přírodní jevy a efekty, jako například kouř, oheň, déšť, jiskry, efekty, kouzel, výbuchy a podobně. Některé enginy umožňují i akceleraci částicových efektů pomocí GPU, tím je možné dosáhnout milionů částic při 60 obrázcích za sekundu i na běžných PC. Systémy pro tvorbu a úpravu animací jsou dnes také běžnou součástí herních enginů.

Obrázek 5.1: Screenshot z videa, které předvádí grafické možnosti Unreal Enginu.[23]

5.1.2 Fyzika

Důležitou součástí her bývají různé fyzikální simulace. Fyzikou je myšlena především kinematika a dynamika. Někdy stačí pouze určit, kdy dochází mezi dvěma objekty ke kolizi. Dnešní enginy ale umožňují i dynamické simulace kolizí více objektů, simulace pohybů závěsů a oblečení (kusu látky), simulace pohybů lan nebo konstruování složitějších fyzikálních těles pomocí spojů, pružin a podobně.

5.1.3 Zvuky

Téměř žádná hra se neobejde bez zvuků. Enginy umožňují jednoduše přehrát zvuky v mnoha různých formátech. Dále mohou pomoci s jednoduchými úpravami zvukového záznamu a vytvořit iluzi prostorového zvuku.

5.1.4 Sítě

Vývoj multiplayerových her může být výrazně urychlen díky nástrojům pro komunikaci po síti. Enginy mohou poskytovat rozsáhlá a optimalizovaná API pro sdílení a synchronizaci stavu multiplayerové hry.

5.1.5 Grafické uživatelské rozhraní

Téměř žádný software se neobejde bez grafického uživatelského rozhraní. Herní enginy typicky zahrnují i framework pro vytvoření tlačítek, sliderů a podobných elementů.

5.1.6 Skriptování

Samotná herní logika je většinou implementována v nějakém programovacím jazyce, například C++/C#/Lua nebo skriptovacích jazycích navržených přímo pro daný engine. Napsaný kód poté využívá výše uvedené komponenty. Některé enginy podporují i takzvaný visual scripting, tedy programování logiky pomocí diagramů bez psaní kódu. Velkou výhodou enginů bývá jejich multiplatformnost, tedy schopnost zkompilovat jednu hru na více platforem, jako například Android, Windows, Playstation a podobně. Většinou je nutných pouze pár drobných změn, které se týkají vstupů.

5.2 Výběr herního enginu

Dnes je na výběr velké množství různých herních enginů. V úvahu budou brány pouze ty, které nejsou zaměřeny primárně na 2D. Nejvýznamnější trojicí je nejspíše Unreal Engine, CryEngine a Unity. Při výběru enginu je především nutné zjistit, jestli pro něj existují nástroje pro jednoduchou integraci eyetrackeru Tobii a Mindwave Mobile. Na oficiálních stránkách Tobii je SDK pouze pro Unity a Unreal Engine (momentálně v betě). Na oficiálních stránkách Neurosky je uvedena i podpora pro C++, ale především SDK se snadnou integrací přímo do Unity. Hra vytvořena v praktické části bude tedy implementována v Unity, jelikož má nejlepší podporu pro obě zařízení a navíc mám s tímto enginem dobré zkušenosti.[25][24]

6 Kognitivní trénink hrou

Úrazy hlavy v důsledku autonehod a podobně mohou zhoršit některé kognitivní funkce. Ergoterapie může využívat kognitivních her pro zlepšení ztracených schopností. Tyto hry mohou pomoci i pacientům, kteří jsou upoutáni na nemocniční lůžko, přičemž u nich může docházet ke zhoršování některých kognitivních funkcí, z důvodu nedostatečné stimulace.

Cílem této bakalářské práce je vytvořit hru, která by mohla zábavnou formou trénovat mozek. V rámci teoretické části jsem vyzkoušel několik her zaměřených na kognitivní trénink, abych zjistil, jak je možné mozek trénovat pomocí počítačových her. Po konzultaci s vedoucím práce budou varianty her, které se mi líbily nejvíce, zahrnuty do hry implementované v praktické části.

6.1 Vyzkoušené hry

Při hledání her zaměřených na kognitivní trénink na internetu jsem zjistil, že většina z nich je vytvořena jako mobilní aplikace. Výhodou je to, že je možné tyto hry hrát kdekoliv a kdykoliv, zároveň nemají vysoké nároky na hardware.

Na Play Store pro platformu Android je velké množství takto zaměřených her, které mají přes milion stažení. Velká část z nich se, bohužel, pouze tváří jako bezplatné. Po stažení chtějí po uživateli typicky měsíční placené členství. Podařilo se mi však najít pět aplikací, které obsahují sadu her zcela bezplatných. K tomu jsem si zapůjčil jednu hru pro Windows z neurolaboratoře KIV. Zde uvádím hry, ze kterých dále vycházím:

- **Happy Neuron** – Windows,
- **Brain Games** – Android,
- **Mind Games** – Android,
- **Skillz** – Android,
- **Brain Training** – Android,
- **Smarter** – Android.

6.2 Společné charakteristiky

Všechny aplikace, které jsem odzkoušel, měly společné některé vlastnosti.

- **Větší množství her** – Každá aplikace obsahovala několik malých her. To je důležité, protože uživatele po chvíli omrzí hrát stále stejnou hru. Důležité je to, že různé hry trénují rozličné kognitivní funkce.
- **Zvyšující se složitost** – Snad všechny hry byly navrženy na několik kol. Je důležité nejdříve začít pomalu a postupně přidávat na složitosti. Hry, které jsem hned ze začátku nezvládal, mě nebavily a už jsem se k nim nevrátil. Naopak hry, které začaly zvolna a já se v nich postupně zlepšoval, mě motivovaly a měl jsem radost z toho, že se zlepšuji.
- **Jednoduchý koncept** – Většina her byla velmi snadná na pochopení. Jednalo se o ne složité úkony, které lze popsat jednou nebo dvěma větami.
- **Vysvětlení pravidel** – Protože je her více, je jednoduché zapomenout, jak se daná hra hraje. Proto před každou hrou bývají jednoduše vysvětlena její pravidla. Takové poučení je vhodné i v dalších kolech stejné hry.
- **Motivace** – Je také důležité hráče po každé hře dále motivovat. Motivace může spočívat v textu, který říká, že se zlepšuje, nebo v opačném případě ho uklidňuje, že se nic neděje a že to snad příště vyjde. Motivace může být také navozena sbíráním bodů za jednotlivá kola.

Žádnou z otestovaných her není možné ovládat ochrnutým uživatelem. Právě s tím se snaží pomoci tato bakalářská práce. Další výhodou je zapojení zařízení pro měření mozkové aktivity. To pomůže s feedbackem a může sloužit i jako ovládací prvek. Věřím, že tyto hry mohou pomoci trénovat ne zcela ztracené kognitivní funkce.

6.3 Kategorie her

Hry pro kognitivní trénink se dají rozdělit do několika kategorií podle schopností, které mají procvičit. Některé hry lze zařadit současně i do více kategorií.

6.3.1 Hry na podporu soustředění

Velká část her je zaměřena na procvičení soustředění na úkol, například:

- Seřazení čísel od nejmenšího po největší za určitý čas.
- Hledání rozdílů.
- Hra, ve které se uživateli zobrazuje text s názvem barvy. Text je napsaný náhodnou barvou a uživatel má určit, jestli barva odpovídá názvu barvy.
- Hledání znaku "8"mezi znaky "B" (a podobné).
- Hra, kde se zobrazuje několik šipek v různém směru a uživatel má kliknout na tlačítko se směrem, který ukazuje prostřední šipka.

6.3.2 Hry na podporu krátkodobé paměti

Další populární kategorií jsou hry, které trénují krátkodobou paměť. Níže jsou vybrané příklady těchto her.

- Zapamatování si sérii obrázků nebo čísel.
- Hra, kde se animací nejdříve pospojují body na obrazovce a uživatel je poté má spojit ve stejném pořadí.
- Pexeso.

6.3.3 Hry na podporu prostorové orientace

- Uživateli je například předveden základní obrázek spolu s jedním otočeným a několika dalšími podobnými. Hráč má zjistit, který obrázek je ten původní, ale otočený.
- Hráč vidí trojrozměrnou scénu s několika geometrickými tělesy z jednoho pohledu. Hráč k ní má přiřadit, který snímek, z možných několika pohledů z ptačí perspektivy, odpovídá dané scéně.

6.3.4 Hry na podporu aritmetických schopností

Tyto hry nemusí sloužit pouze ke trénování základních počtů. Vyžadují vysokou úroveň soustředění, i když se jedná o jednoduché příklady.

- Vynechání členu v rovnici. Hráč má zjistit, který člen chybí.

- Výběr řešení a kliknutí na správný na správný výsledek početní operace.

6.4 Shrnutí

Z výše uvedených kategorií a po konzultaci s vedoucím práce jsem se rozhodl soustředit se spíše na trénink soustředění a krátkodobé paměti. Tyto schopnosti jsou abstraktnější a důležitější než prostorová orientace a aritmetické dovednosti. To ale neznamená, že hra vytvořená v praktické části nebude některé tyto aspekty trénovat.

Při hraní různých kognitivních her jsem zjistil mnoho informací a načerpal inspiraci pro vlastní návrh hry, která bude vytvořena v praktické části. Při návrhu hry chci uplatnit především poznatky ze sekce Společné charakteristiky.

Do své hry chci vlastním způsobem zakomponovat prvky her zmíněných v sekci Kategorie her tak, aby byly součástí zábavnějšího a vizuálně přívětivého celku, který motivuje uživatele ke hraní.

7 Návrh hry

Za hlavní téma hry jsem si zvolil snowboarding. Hráč bude očima ovládat snowboardistu, který jede po svahu. Mezi startem a cílem bude hráč muset plnit různé úkony založené na hrách popsaných v sekci Kategorie her. Má hra bude obsahovat několik kol o různé obtížnosti.

Při jízdě bude možné snímat hráčovu míru soustředění pomocí Mindwave Mobile. Míra soustředění bude v jedné části hry použita i jako ovládací prvek.

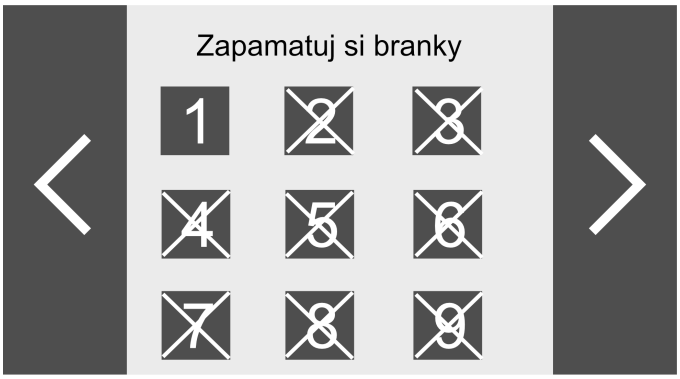
Při implementaci hry se budu snažit dodržet prvky kognitivních her, které jsem popsal v sekci Společné charakteristiky. Dále budu dbát na to, aby se celá hra včetně menu dala ovládat pouze očima. Hra bude po každém kole vyhodnocovat, jak si hráč vedl, a tyto informace ukládat, aby hráč a případně jeho ergoterapeut viděli, jak se hráč zlepšuje.

7.1 Menu

Jelikož chci, aby se menu dalo ovládat očima, je důležité, aby bylo přehledné a aby tlačítka byla velká a dostatečně daleko od sebe. Návrh úvodní obrazovky hry je zobrazen na obrázku 7.1. Šipkou doprava se uživatel dostane do výběru úrovně. Šipka doleva vede k ukončení hry.

Obrázek 7.1: Hlavní menu.

Na obrázku 7.2 je zobrazena obrazovka, na které si uživatel vybere kolo. Nahoře je informace o tom, který druh hry si uživatel vybírá (viz sekce Druhy her). Šipkou doleva se hráč vrátí na předchozí obrazovku, šipkou doprava se dostane k dalšímu druhu hry. Tlačítky označenými čísly se dostane do příslušné úrovně. Přeškrtnutí značí, že hráč ještě nedokončil předchozí úroveň.

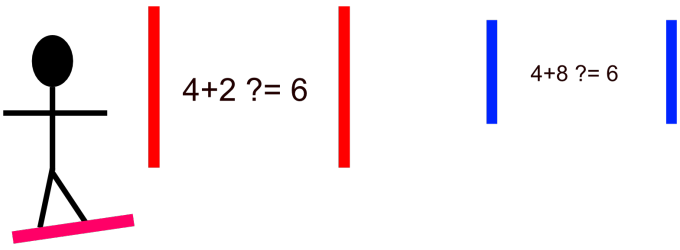


Obrázek 7.2: Menu, ve kterém si uživatel vybere úroveň.

7.2 Hra

Samotná podstata hry spočívá v simulaci snowboardové jízdy. Hráč ovládá snowboardistu, který musí projet danou trať za určitých podmínek. Na začátku každého kola mu budou na obrazovce vysvětlena pravidla, poté kolo začne.

Většina kol hry bude spočívat v tom, že mezi startem a cílem budou různě rozmístěny branky. Hráč bude mít za úkol podle nějakého pravidla projet pouze některé z nich (viz sekce Druhy her). Za správné projetí se hráči přičtou body, za špatné projetí se body odečtou. Na obrázku 7.3 je vidět jeden z druhů her, kde hráč má projet pouze brankami s korektní rovnicí. Malou část celkového bodového skóre bude tvořit také čas, za který se hráči povedlo dostat se do cíle. Bonusové body hráč může získat za to, že snowboardista skočí na skokánku. Všechny získané body se sčítají a výsledek bude viditelný v hlavním menu. Při letu snowboardisty se zpomalí čas a hráč bude muset provést nějaký jednoduchý úkon. Pokud se mu to povede, snowboardista udělá salto. Tato část slouží k ozvláštnění hry i jako určitá motivace.



Obrázek 7.3: Ilustrace principu jednoduché hry.

Na konci kola proběhne vyhodnocení, hráč se dozví výsledky tohoto i

předchozích pokusů a bude informován o tom, zda postoupil do dalšího kola. V případě, že uživatel do vyššího kola nepostoupil, bude informován o nutném skóre pro další postup.

7.3 Druhy her

Vzhledem k základnímu pojetí hry většina variací bude spočívat v určení, která branka má být projeta, a následně její projetí snowboardistou.

- **Jízda ovládaná soustředěním** – Jediný druh hry, ve které hráč nebude projíždět brankami. Snowboardista pojede jen rovně na sérii skoků. Hráč se bude muset soustředit, aby nabral dostatečnou rychlost na skok. Míra soustředění pro úspěšný skok bude s každým skokem vyšší.
- **Řady čísel** – Hráči se na začátku kola zobrazí několik konkrétních čísel. Jeho úkolem bude projet pouze těmi brankami, které mají právě tato čísla.
- **Barevný text** – Na každé brance bude právě jednou barvou napsán název jedné barvy. Hráč bude mít za úkol projet pouze ty branky, u kterých se barva shoduje s názvem barvy.
- **Správné pořadí** – Na svahu budou rozmístěny očíslované branky a úkolem hráče je nechat projet snowboardistu postupně branky s čísly od 1 do N.
- **Jednoduché počty** – Na každé brance bude zobrazena jednoduchá rovnice. Hráč bude mít za úkol projet jen ty branky, jejichž rovnice jsou pravdivé. Tato hra nemá trénovat matematické dovednosti, ale schopnost soustředění. Příklady budou proto velmi jednoduché.

7.4 Shrnutí

Myslím si, že navržená hra je dobrou kombinací zábavy, potěšení z jízdy a zároveň tréninku kognitivních funkcí jedince. Hráč se bude muset soustředit na více věcí na jednou – jak na úkol, tak na jízdu. Ta zároveň tvoří přirozený časový limit k vyřešení úkolu na příští brance.

Hra by měla zároveň dostatečně motivovat pomocí odměn – salta snowboardisty a sbíraných bodů. Hráč má na výběr z velkého množství kol, takže

když ho přestane bavit jeden druh hry, může přejít na jiný. Hra bude zároveň s každým kolem těžší, aby se hráč nenudil. Hráč bude současně dostávat feedback o tom, jak moc se soustředí.

Menu jsem se snažil navrhnout co nejpřehlednější a nejsrozumitelnější, aby se v něm kdokoli dokázal zorientovat a aby bylo možné jej přirozeně ovládat očima.

8 Architektura hry

Pro úspěšnou implementaci hry je důležité dobře navrhnout, jak spolu budou různé prvky hry spolupracovat. Hra obsahuje čtyřicet pět různých kol a devět různých menu. Jedná se o poměrně rozsáhlý projekt, a je tedy velmi důležité správně rozčlenit kód a ostatní části z kterých se hra skládá.

V této kapitole bude krátce nastíněn vývoj her v Unity, a budou vysvětleny některé pojmy, které s tím souvisí. Dále bude popsán princip událostmi řízené architektury (anglicky Event-driven architecture), její realizace v Unity a jak pomohla rozčlenit vývoj hry do malých zvládnutelných a znovupoužitelných částí.

8.1 Vývoj v Unity

8.1.1 Vybrané pojmy

Asset

`Asset` je jakýkoliv soubor uložený v projektu v Unity. Může se jednat o obrázek, script, mesh nebo jiné.

Scene

Každá hra v Unity obsahuje alespoň jednu `Scene` neboli scénu. Typicky se ale hry skládají hned z několika scén. Scéna je určitá abstrakce, která dělí hru na logické celky. V této hře je například scéna pro každé menu a pro každou hru. Po spuštění scény jsou všechny objekty z předchozí scény smazány (dají se udělat i výjimky). Do scény jsou umisťovány herní objekty, které ji tvoří.

GameObject

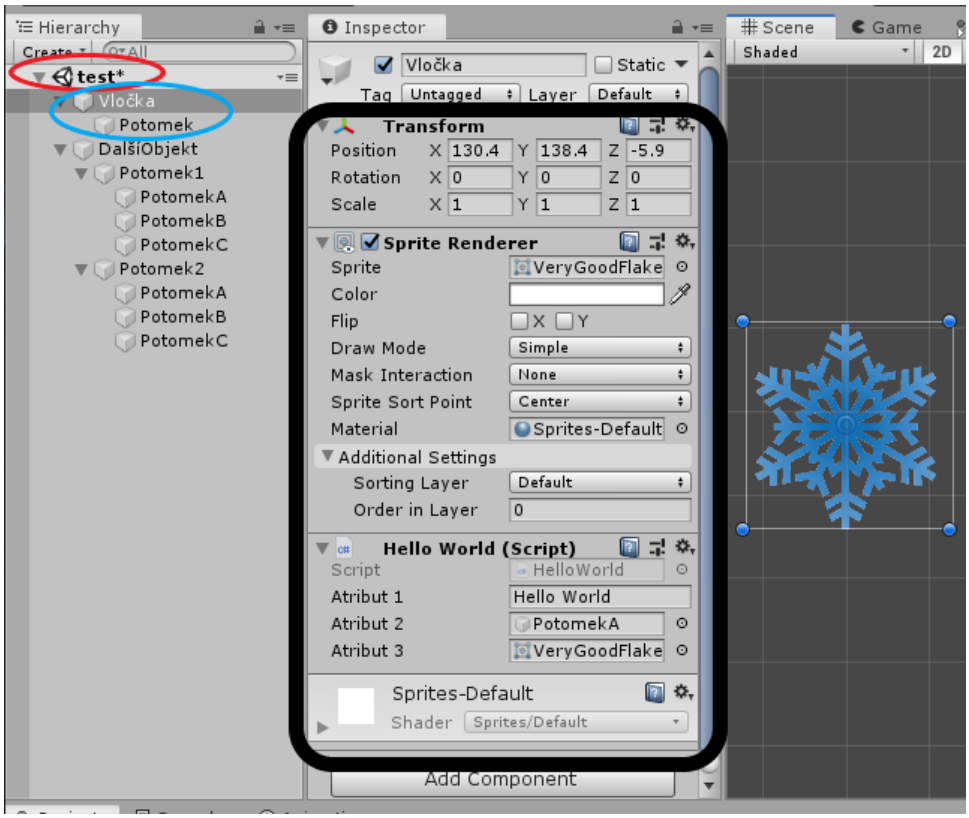
`GameObject` (na obrázku 8.1 modře, dále česky herní objekt) je abstrakce jedné části hry v rámci scény. Každý herní objekt je pojmenován a vlastní několik komponent, které určují jeho chování. Nově vytvořený herní objekt má pouze komponentu `Transform`, která určuje jeho pozici, rotaci a škálování. Herní objekt může tvořit stromovou strukturu, ve které jeden herní objekt může obsahovat potomky, které jsou také typu `GameObject`. `Transform` potomků je poté relativní k jejich rodiči.

Příkladem může být snowboardista. V kořeni se nacházejí komponenty, které umožňují jeho ovládání. Jeden z potomků může být herní objekt obsahující mesh snowboardisty, ten má jako potomka herní objekt, který obsahuje mesh snowboardu.

Component

`Component` (dále komponenta) je část jednoho herního objektu. Na obrázku 8.1 jsou černě vyznačeny 3 komponenty herního objektu Vločka. Komponenty jsou třídy v C# asociované s objektem, na který jsou umístěny. Pouze třídy, které dědí od `Monobehaviour`, mohou být komponentami. Je důležité vědět o metodách `Start` a `Update` z třídy `Monobehaviour`. `Start` je zavolána při načtení scény, `Update` je volána neustále, několikrát za sekundu.

Veřejné atributy, nebo atributy označené pomocí `[SerializeField]` jsou přístupné v Unity a je možné jim přiřadit hodnotu před samotným spuštěním, i za běhu. Atribut může být například komponenta herního objektu v té stejné scéně nebo asset. Atributy komponent jsou po vypnutí hry vráceny na jejich hodnoty před spuštěním. Na obrázku 8.1 lze vidět dvě komponenty, které poskytuje Unity: `Transform` a `Sprite Renderer`. Ten se stará o vykreslení obrázku, který je vložen do políčka (atribut) `Sprite`. Pod těmito komponenty je programátorem vytvořená třída v C# `HelloWorld.cs`.



Obrázek 8.1: Ukázka jedné scény vyznačena červeně, s vybraným herním objektem vyznačen modře a jeho komponenty vyznačeny černě.

Prefab

`Prefab` (dále prefabrikát) je užitečn,ý pokud je jeden herní objekt potřeba vícekrát v celé hře. Programátor poskládá například snowboardistu a vytvoří z něj prefabrikát, který si uloží mezi assety. Tento prefabrikát může být následně vložen do jakékoliv scény jako herní objekt. Při změně prefabrikátu se změna promítne do všech jeho instancí ve všech scénách. Unity podporuje i vnořené prefabrikáty a umožňuje přidat takzvané override pro atributy jednotlivým instancím prefabrikátů. Tyto atributy si poté při změně prefabrikátu zachovají svou hodnotu. Jako override je také možné přidat na instanci novou komponentu, nebo mu lze přidat nového potomka. Jako override však nelze odebrat potomka či komponentu.

ScriptableObject

`ScriptableObject` (dále SO jako asset instance třídy, která dědí od `ScriptableObject`) není pro vývoj v Unity nezbytná konstrukce. Pokud chceme vytvořit vlastní SO, je nejdříve nutné definovat třídu v C#, která dědí od

`ScriptableObject`. Poté je možné vytvářet její instance jako assety a jejím atributům přiřazovat hodnoty. Tyto assety nemohou existovat samostatně v scéně, ale pouze jako odkaz v atributu komponenty. Atributy SO jsou po skončení hry vráceny na jejich výchozí hodnoty.

Například může existovat třída `GameScore`, která dědí od `ScriptableObject` a má atributy `numGatesMissed` a `score`. `GameScore` lze poté v Unity vytvořit a uložit mezi ostatní assety. Několik různých komponent může držet odkaz na asset `GameScore` a inkrementovat skóre v okamžiku, kdy je to potřebné. Po ukončení kola jiná komponenta vezme atribut `score` assetu `GameScore` a jeho hodnotu zobrazí hráči.

8.1.2 Tradiční přístup

Komponenty různých herních objektů spolu často potřebují komunikovat. Buď potřebují získat hodnotu nějaké proměnné, nebo vyvolat nějakou metodu. Tento problém je v Unity typicky řešen tak, že odkaz na komponentu A je přiřazen (přetáhnut myší v Unity) do atributu komponenty B. V ní už je možné s A pracovat v kódu jako s normální instancí třídy.

8.1.3 Problémy s tradičním přístupem

Výše uvedený přístup začíná být ve větších projektech problematický. Komponenty ve scéně mají velké množství atributů, které odkazují na jiné komponenty a ty mají zase své atributy. V tom je velmi těžké se orientovat, a proto typicky vzniká komponenta pojmenovaná `GameManager`, která má za úkol starat se o nejdůležitější herní objekty a jejich komponenty. Tím ale vzniká velká třída, která může být v mnoha různých stavech. Takové třídy jsou označovány jako návrhový antivzor God object. Přímé odkazy navíc komplikují opakované použití herních objektů v jiných částech hry nebo v jiných scénách.

Například mějme komponentu `SnowboardController`. Chceme, aby se po dojetí snowboardisty do cíle stalo několik událostí: zastavení běhu času, přepnutí vstupu z očí na myš, otevření menu, zobrazení výsledků a tak dále. Komponenta `SnowboardController` by tedy musela mít odkaz na několik herních objektů, které mají komponenty, které dané události vykonají. Pokud bychom chtěli přidat ještě například spuštění ohňostroje, museli bychom znovu otevřít třídu `SnowboardController` a přidat do ní odkaz na ohňostroj a zavolání příslušné funkce.

Ve skutečnosti ale není žádný dobrý důvod, proč by měl `SnowboardController` vědět o tom, co se stane po dojetí hráče do cíle. To je možné řešit

vytvořením dříve zmíněného `GameManageru`, který by měl odkazy na všechny potřebné komponenty a metodu `GameEnd()`, která by nad těmito komponentami spustila všechny potřebné metody. `SnowboardController` by ale neměl vědět ani o třídě `GameManager`. Ta může být pro různé scény různá.

V ideálním případě by mělo být možné vzít prefabrikát snowboardisty a vložit ho do prázdné scény, aniž by bylo nutné doplňovat nějaké závislosti. Na tomto příklad jsem se snažil ilustrovat, jak přímé odkazy mohou komplikovat vývoj hry.

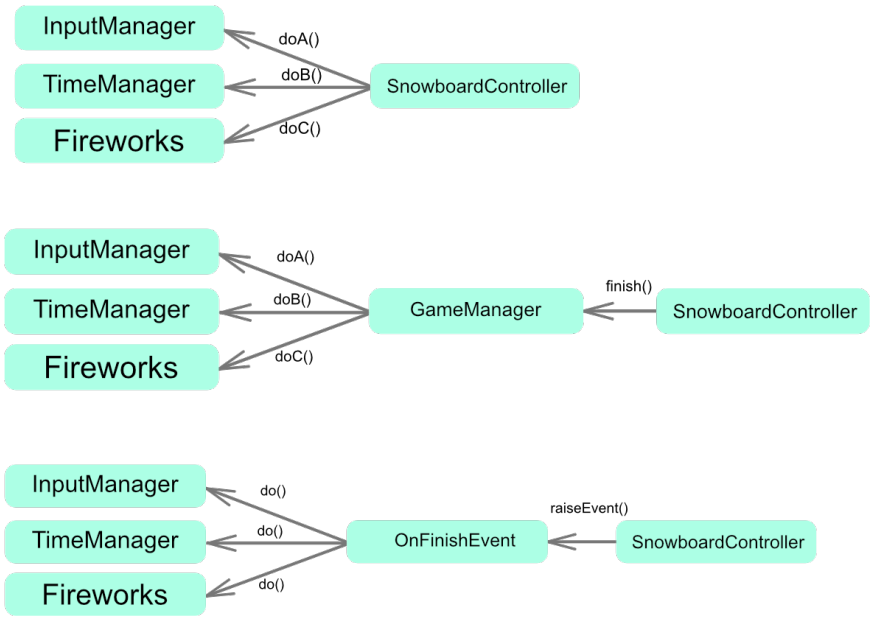
8.2 Událostmi řízená architektura

Událost je pomyslný most mezi dvěma objekty, které spolu potřebují komunikovat, ale nehodí se, aby o sobě přímo věděly. Stačí pouze, aby věděly o dané události.

Na obrázku 8.2 lze vidět přístupy řešení problému popsaném v sekci Problémy s tradičním přístupem. V horní části obrázku lze vidět situaci, ve které `SnowboardController` spouští vše potřebné sám. V prostřední části obrázku je vše spouštěno pomocí třídy `GameManager`. Ve spodní části obrázku spustí `SnowboardController` metodu `RaiseEvent()` události `OnFinish`. Této události naslouchají příslušné komponenty. Důležité je, že událost má pole referencí na metody komponent[1], které se mají vykonat. `RaiseEvent()` iteruje přes toto pole a postupně zavolá všechny příslušné metody.

Důležité je, že třída `SnowboardController` již neobsahuje pevné odkazy na jiné komponenty. `SnowboardController` pouze informuje o tom, že se stala nějaká událost. Herní objekt s komponentou `SnowboardController` je teď flexibilnější. Když se například rozhodneme zbavit se ohňostroje, stačí ho jednoduše smazat a nezpůsobíme tím žádnou výjimku null reference. Pokud chceme, aby se ohňostroj spustil jindy, stačí pouze změnit, které události naslouchá.

[1]Často řešeno pomocí lambda výrazů.



Obrázek 8.2: Ukázka přístupů k řešení problému popsaném v sekci Problémy s tradičním přístupem.

8.3 Událostmi řízená architektura v Unity

V Unity lze docílit událostmi řízené architektury pomocí SO po vzoru prezentace z události Unite 2017 Austin[38]. V mé práci jsem pro tento účel použil knihovnu Unity Atoms, která je dostupná na GitHub[39]. Tato knihovna vychází z uvedené prezentace.

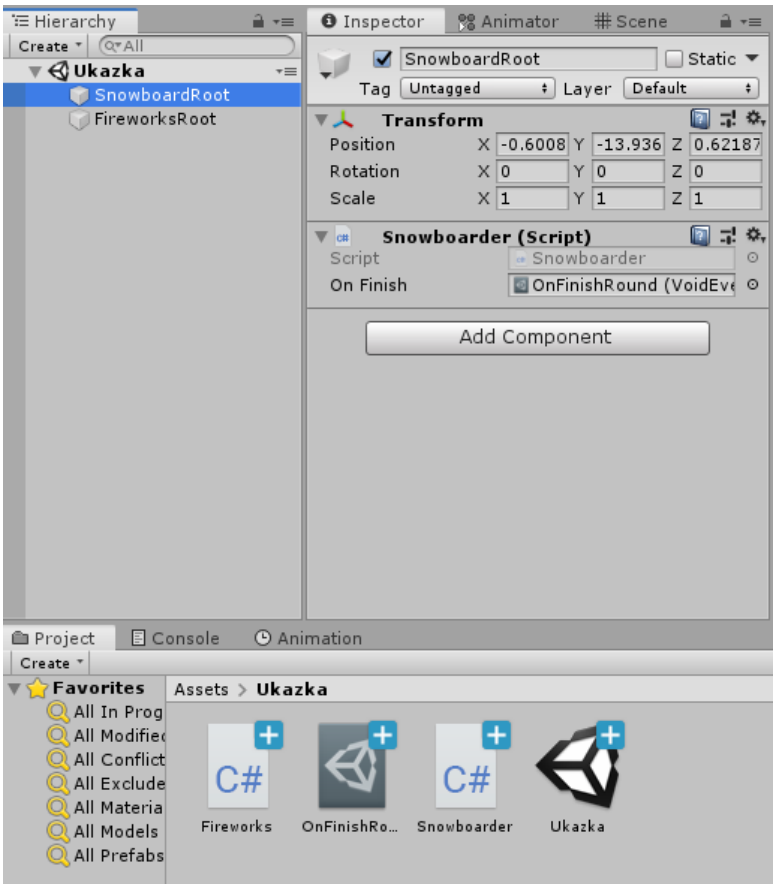
8.3.1 Unity Atoms

Unity Atoms je rozsáhlá knihovna, která pomáhá s modulárním návrhem projektů v Unity. K tomu využívá především SO. Unity Atoms poskytuje velké množství funkcionalit. Pro tuto práci je především důležitý systém pro událostmi řízenou architekturu. Kromě něj budou popsány i proměnné založené na SO, které jsou v této práci také využívány.

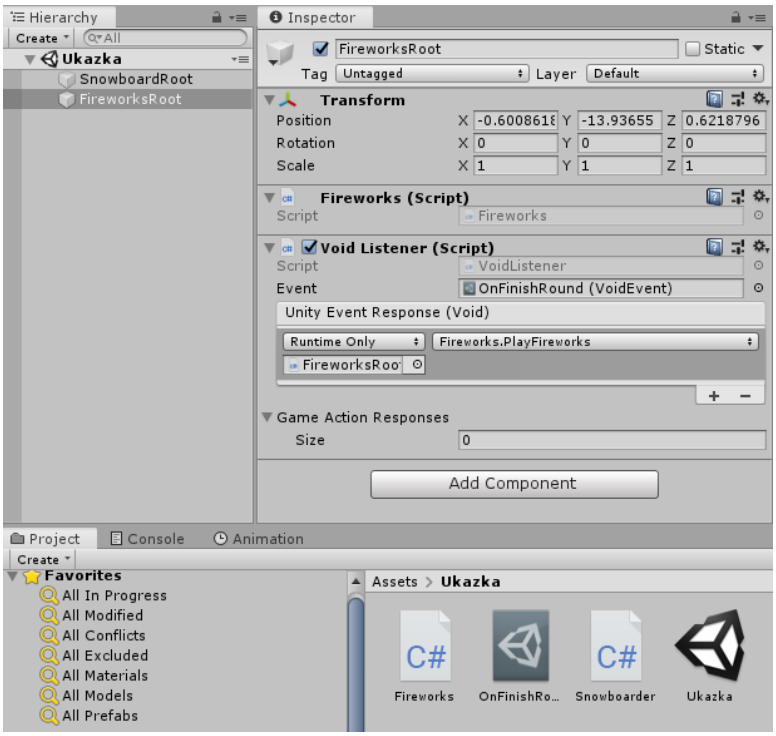
Události

Na obrázcích 8.3 a 8.4 lze vidět jednoduchou ukázku práce s touto knihovnou. Na obrázku 8.3 vlevo jsou ve scéně dva herní objekty, SnowboardRoot a FireworksRoot. SnowboardRoot je označen, a tudíž jsou jeho komponenty vidět napravo. SnowboardRoot má dvě komponenty, `Transform` a `Snowboarder`. `Snowboarder` má atribut `OnFinish`, který je typu `VoidEvent`. Tomuto

atributu je přiřazen SO `OnFinishRound` stejného typu. `OnFinishRound` lze vidět níže mezi assety. Typ `VoidEvent` je převzat z knihovny Unity Atoms. Na obrázku 8.4 lze napravo vidět komponenty herního objektu FireworksRoot. Komponenta `Void Listener` je také z knihovny Unity Atoms. Tam je specifikováno, na jakou událost má čekat, v tomto případě `OnFinishRound`, a jaká metoda má být následně spuštěna. V tomto případě se jedná o metodu `PlayFireworks` komponenty `Firework` herního objektu FireworkRoot.



Obrázek 8.3: Ukázka řešení problému popsaném v Problémy s tradičním přístupem s pomocí knihovny Unity Atoms.



Obrázek 8.4: Ukázka řešení problému popsaném v Problémy s tradičním přístupem s pomocí knihovny Unity Atoms.

Tato knihovna umožňuje i vytvoření událostí s parametrem určitého typu. Například události typu `IntEvent` navíc předají jako parametr jedno celé číslo. Pomocí genericity je velmi jednoduché přidat libovolné typy událostí.

Proměnné

Kromě jiného knihovna Unity Atoms umožňuje vytvořit proměnné jako instance tříd, které dědí od `ScriptableObject`[2]. Existuje například typ `IntVariable`. Tato proměnná je vytvořena mezi assety. Herní objekty s ní poté mohou pracovat. Tento přístup je již popsán a vysvětlen na příkladu s herním skóre v sekci 8.1.1. Proměnné a objekty návrhového vzoru přepravka, jako například již popsané `GameScore`, založené na SO, byly pro vývoj hry taktéž velmi důležité. Pomáhají oddělit data od logiky, která s nimi pracuje.

Ukázka vytvořených událostí a přepravek

Pro tuto hru bylo vytvořeno přes padesát různých událostí a přepravek založených na SO. Dále bude uvedeno několik událostí a přepravek jako ukázka,

[2]Instance jsou tedy vytvořeny mezi assety.

k čemu mohou sloužit.

- `Pause` (Událost) – Uživatel stiskl tlačítko, které pozastaví hru.
- `CountDownUp` (Událost) – Odpočítání na začátku hry skončilo a hra může začít.
- `ScoreChanged` (Událost) – Skóre se změnilo, jako parametr je vložena nová hodnota skóre.
- `GateCorrect` (Událost) – Hráč správně projel jednu branku.
- `MindwaveConnected` (Událost) – Mindwave bylo úspěšně připojeno.
- `GameScore` (Přepravka) – Obsahuje informace o tom, kolik branek bylo projeto správně, celkové skóre a podobně.
- `InputSettings` (Přepravka) – Obsahuje informace o tom, jak jsou nastavené vstupy.
- `SnowboarderSpeed` (Přepravka) – Obsahuje hodnotu rychlosti, kterou se snowboardista v danou chvíli pohybuje.

8.4 Využití prefabrikátů

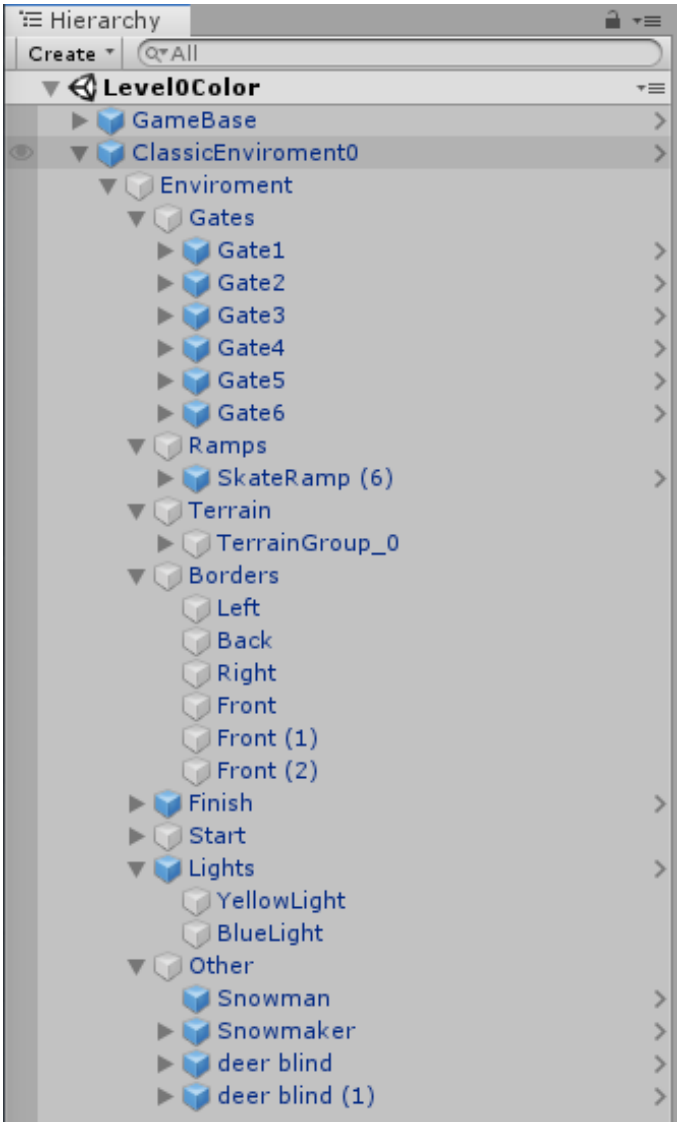
Hra obsahuje čtyři druhy her, ve kterých má hráč za úkol projet pouze ty branky, které splňují nějakou podmínku. Pátý druh hry je svou podstatou jiný a je ovládán pouze zařízením Mindwave (viz sekce Druhy her). Každý druh hry obsahuje devět kol. Při takovém množství kol je důležité správně využít prefabrikáty tak, aby nebylo nutné každou změnu provést manuálně v každém kole.

8.4.1 Prostředí

Pro první čtyři druhy her platí, že všechna kola číslo n se hrají na stejném svahu, s identicky rozestavěnými brankami. Z tohoto důvodu jsou vytvořeny prefabrikáty `ClassicEnviroment` s číslem kola na konci názvu. V nich se nachází několik herních objektů, které jsou zobrazeny na obrázku 8.5.

- `Gates` – Obsahuje prefabrikáty branek.
- `Ramps` – Obsahuje prefabrikáty skoků.
- `Terrain` – Krajina.

- `Borders` – Zábrany kolem svahu, které brání vyjetí z trati.
- `Finish` – Cíl.
- `Start` – Start.
- `Lights` – Obsahuje dvě světla, která osvětlují scénu.
- `Other` – Ostatní estetické prvky – sněhuláci, sněžná děla a podobné.



Obrázek 8.5: Ukázka prefabrikátu prostředí pro první kola.

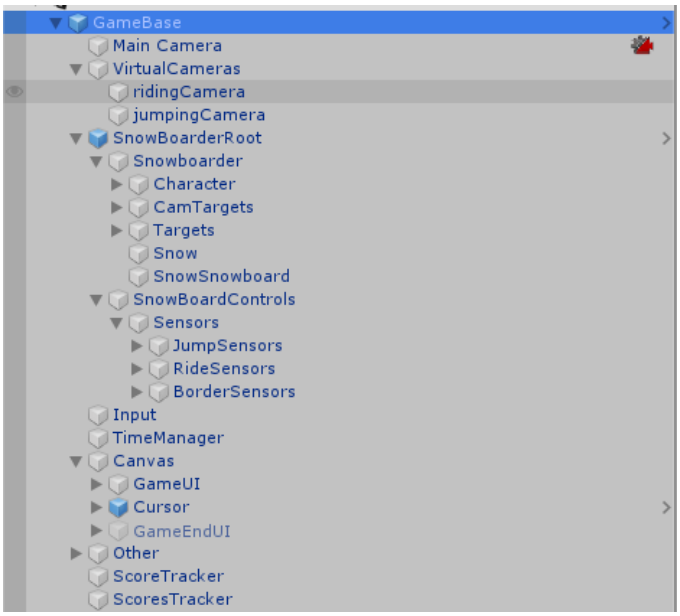
Tento návrh umožňuje bezproblémovou editaci například všech prvních kol najednou. Prefabrikáty jsou označeny modrou krychlí, oproti klasickým

herním objektům, které jsou označeny bílou krychlí. Za povšimnutí stojí, že branky `Gate1-6` jsou také prefabrikáty. Toto je příklad vnořených prefabrikátů. Ve všech prefabrikátech prostředí jsou využity tyto prefabrikáty branek. To znamená, že změna prefabrikátu branky se promítne do úplně všech kol najednou. Branky se ale v různých druzích her chovají trochu jinak. Toho je docíleno pomocí override viz Prefab.

8.4.2 Základ hry

Všechny druhy her s brankami mají jeden společný prefabrikát nazvaný `GameBase`, který je jejich společným základem. Druh her, které uživatel ovládá pomocí Mindwave, má jako základ jiný prefabrikát. Ten je nazvaný `BrainGameBase`, je ale velmi podobný `GameBase`. Na obrázku 8.6 lze vidět herní objekty, které tvoří prefabrikát `GameBase`.

- `Main Camera` – Stará se o vykreslování herního světa.
- `VirtualCameras` – Obsahuje dvě virtuální kamery (herní objekt s komponentou `VirtualCamera`, kterou poskytuje Unity), jednu pro jízdu a druhou pro skok. Ty slouží pro umístění skutečné kamery.
- `SnowboarderRoot` – Kořen snowboardisty, potomci obsahují herní objekty, které se starají o jeho pohyb, mesh, částicové efekty sněhu a další.
- `Input` – Stará se o vstupy.
- `TimeManager` – Slouží k zpomalení času při skoku.
- `Canvas` – Kořen všech elementů uživatelského rozhraní.
- `Other` – Obsahuje herní objekty s komponentami pro ladění.
- `ScoreTracker` – Stará se o počítání skóre.
- `ScoresTracker` – Ukládá dosažené skóre.



Obrázek 8.6: Prefabrikát společného základu.

8.4.3 Ostatní prefabrikáty

Zde budou krátce zmíněny některé další prefabrikáty, které byly vytvořeny a nebyla o nich zmínka v sekcích Prostředí a Základ hry.

- `MenuBase` – Společný základ menu, ve kterém si uživatel vybere, které kolo daného druhy hry chce hrát, viz obrázek Menu výběru úrovně.
- `PauseMenu` – Menu, které se zobrazí při pozastavení hry.
- `EyeButton` – Prefabrikát tlačítka, které se sepne při delším pohledu.
- `TrickPoint` – Vločka, na kterou se musí uživatel podívat při skoku. Tento prefabrikát je takzvaný Prefab Variant, což znamená, že je odvozený od jiného prefabrikátu[3]. V tomto případě je prefabrikát odvozen od `EyeButton`.
- `Cursor` – Částicový systém, který slouží k vizualizaci pozice, na kterou se uživatel dívá.
- `MenuBG` – Pozadí menu.

[3]Toto je podobné dědičnosti tříd.

9 Vstupy

9.1 Eye–tracking

Pro jakékoliv využití eye–trackeru Tobii 4C je nutné mít nainstalovaný software od výrobce, který je dostupný zde[36]. Pro čtení vstupu z eye-trackeru v Unity je nutné importovat oficiální balíček od Tobii, který je dostupný zde[30]. Práce s tímto balíčkem je jednoduchá. Příkladem může být následující řádka kódu, která vrací normalizovanou pozici, na kterou se uživatel dívá. Souřadnicový systém je nastaven tak, že $(0, 0)$ je v levém dolním rohu obrazovky.

```
Tobii.Gaming.TobiiAPI.GetGazePoint().Viewport
```

Při ovládání snowboardisty nebo menu pomocí eye–trackingu mohou způsobit problém náhlé sakády na jiné místo obrazovky. Například snowboardista by neměl začít okamžitě zatáčet hned, jak se uživatel podívá na levý okraj obrazovky. Hráč mohl totiž pouze rychle pohledem hledat další branku. Vstup z eye-trackeru je v některých případech dobré vyhladit.

Tento problém je řešen metodou `Vector2.SmoothDamp()`, kterou poskytuje Unity. Tato metoda je zavolána v každém snímku. Na následující řádce kódu lze vidět vyhlazování vstupu z eye–trackeru, v té chybí některé parametry metody `Vector2.SmoothDamp()`, které pro ilustraci nejsou podstatné.

```
smoothPoint = Vector2.SmoothDamp(smoothPoint, rawPoint,
smoothing);
```

Zde `smoothPoint` bude výsledný bod s hladkým průběhem, `rawPoint` je bod získaný z eye-trackeru a `smoothing` je čas, po kterém by byl `rawPoint` rovný `smoothPoint`.

Pro ovládání snowboardisty a menu byla zvolena hodnota `smoothing` na čtvrt sekundy. Při delší době již začíná působit eye–tracking zpomaleně.

9.1.1 Ovládání snowboardisty

Pohyb snowboardisty je určen jedním dvourozměrným vektorem desetinných čísel, v dalším textu označen jako kontrolní vektor. První složka určuje míru zatočení, kde -1 znamená maximální zatáčení doleva a 1 maximální zatáčení doprava. Druhá složka, v rozsahu od -1 do 1, určuje akceleraci snowboardisty.

Výpočet

Jednoduchý výpočet kontrolního vektoru by mohl probíhat následovně. Levý kraj obrazovky znamená -1, pravý 1 a mezitím jsou hodnoty lineárně interpolovány. Stejně tak pro druhou složku. Problém ale nastává v tom, že hráč by při ostrém zatáčení neviděl, kam jede. Druhým problémem je, že hráč se přirozeně dívá do oblasti, kde se nachází snowboardista. Snowboardista však nemusí být vždy ve středu obrazovky.

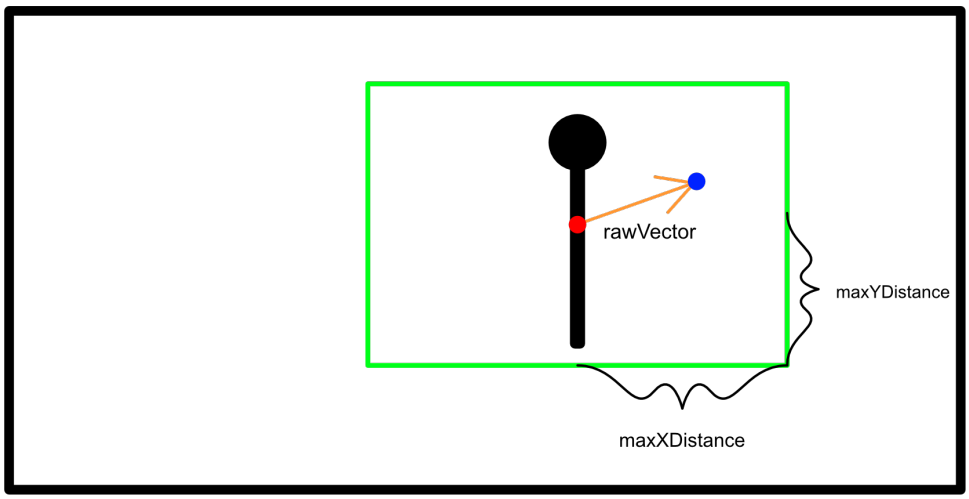
V této práci byl použit výpočet, s jehož vysvětlením pomůže obrázek 9.1. Černá oblast znázorňuje obrazovku, červená tečka normalizovanou pozici snowboardisty v souřadnicích obrazovky, modrá tečka normalizovanou pozici, na kterou se uživatel dívá. Zelená oblast určuje, kde již budou hodnoty kontrolního vektoru maximální.

Výpočet kontrolního vektoru ilustruje následující řádky kódu:

```
controlVector.x = rawVector.x / maxXDistance;
controlVector.y = rawVector.y / maxYDistance;

controlVector.x = clamp(controlVector.x, -1, 1);
controlVector.y = clamp(controlVector.y, -1, 1);
```

Tímto způsobem je zajištěn intuitivní pohyb, který závisí na tom, jakým směrem od snowboardisty se hráč dívá.



Obrázek 9.1: Pomocný obrázek k výpočtu vektoru, který určuje pohyb snowboardisty.

9.2 BCI

Pro jakékoliv použití Mindwave na OS Windows je nutné mít nainstalovaný software ThinkGear Connector, který je dostupný zde[35]. Ten slouží jako server, na který se lze v kódu připojit a přijímat z něj data. Protokol připojení klienta lze nalézt taktéž zde[35]. Důležité je, že data jsou ve formátu JSON[1]. Výrobce na svých stránkách poskytuje i jednoduchý projekt s ukázkou základního navázání připojení a čtení dat. Kód obsažený v tomto projektu ale neřeší některé okrajové situace. Z tohoto důvodu byl použit projekt z GitHub od uživatele DaCookie[27]. K použití tohoto projektu v této práci mám od autora svolení. Projekt obsahuje dokumentaci a okomentovaný kód, který zajistí připojení čtení dat.

9.2.1 Data

Mindwave poskytuje různá data o mozkové aktivitě. V kódu lze získat tato data.

- Hrubý EEG signál.
- Míru toho, jak moc jsou zastoupeny jednotlivé složky hrubého EEG, jako alpha, beta a podobně.
- Míru meditace a soustředění.

V této práci byla využita především uživatelova míra soustředění a meditace.

9.2.2 Implementace

Výše uvedený balíček obsahuje prefabrikát `MindWaveManager`. Ten má tři komponenty: `MindwaveManager`, `MindwaveController` a `MindwaveCalibrator`. Nejdůležitější komponenta je `MindwaveController`. Ta obsahuje veřejné metody `Connect()` a `Disconnect()` a několik událostí, kterým lze naslouchat. Pro tuto práci jsou důležité tyto události: `OnConnectMindwave`, `OnConnectionTimeout`, `OnDisconnectMindwave`, `OnUpdateMindwaveData`.

Využití v této hře probíhá následovně. Do scény, kde se má uživatel připojit, je vložen prefabrikát `MindwaveManager`. Na ten je přidána mnou

[1] JavaScript Object Notation, formát pro výměnu dat se syntaxem, který je odvozený z programovacího jazyka JavaScript.

vytvořená komponenta `MindWaveSetup`. Ta ve své metodě `Awake()`[2] zaregistruje událostem `OnConnectMindwave`, `OnConnectionTimeout` a `OnDisconnectMindwave` vyvolání události založené na SO z knihovny Unity Atoms. Tento krok je pouze pro udržení konzistence se zbytkem hry, protože v ní jsou události založené na SO široce využívány. Události `OnUpdateMindwaveData` je zaregistrována metoda `setMindwaveData(MindwaveDataModel mindwaveData)`, která se nachází uvnitř třídy `MindWaveSetup`. Na následujících řádkách kódu lze vidět zaregistrování událostí.

```
mindwaveController.OnConnectMindwave +=
connectedMindwave.Raise;

mindwaveController.OnConnectionTimeout +=
connectionTimeOut.Raise;

mindwaveController.OnDisconnectMindwave +=
disconnectedMindwave.Raise;

mindwaveController.OnUpdateMindwaveData +=
setMindwaveData;
```

V metodě `setMindwaveData(MindwaveDataModel mindwaveData)` poté mohou být přepravce `brainData` typu `BrainData`, který je založen na SO, přiřazeny přijatá data následovně.

```
brainData.Attention = mindwaveData.eSense.attention;
brainData.Meditation = mindwaveData.eSense.meditation;
```

Poté už stačí, aby v kterékoliv scéně měla jakákoliv komponenta referenci na SO `brainData`.

[2]Metoda, která je zavolána nad všemi komponentami ve scéně při jejím načtení.

10 Pohyb snowboardisty

Naprogramování pohybu snowboardisty po svahu je velmi zajímavým problémem. Z tohoto důvodu je tomuto problému věnována celá kapitola. Pohyb je implementován ve třídě `SnowboarderController`. Ta má jako atributy několik instancí tříd, které provádí části výpočtu pohybu. `SnowboarderController` to vše pouze slučuje do jednoho celku a přepíná mezi vykonávanými metodami podle toho, v jakém stavu se snowboardista nachází.

10.1 Informace o okolí

10.1.1 Třída Sensor

Pro zjištění informací o okolí snowboardisty je vytvořena třída `Sensor`. Nad její instancí je možné zavolat metodu `Recalculate()`. V té je pomocí metody `Physics.Raycast(Vector3 position, Vector3 direction)`, kterou poskytuje Unity, vrhnut pomyslný paprsek z pozice `position` ve směru `direction`. Při zaregistrování kolize s herním objektem scény o něm lze získat informace.

Pro tuto práci jsou důležité tři informace: normálový vektor herního objektu v místě kolize, vzdálenost od herního objektu a jeho `Tag`. `Tag` je string, který lze v Unity přiřadit skupinám herních objektů, které spolu nějak logicky souvisí. Například všechny skoky mají `Tag` "jump".

Na obrázku 10.1 lze vidět vizualizace použitých senzorů.

Červené senzory

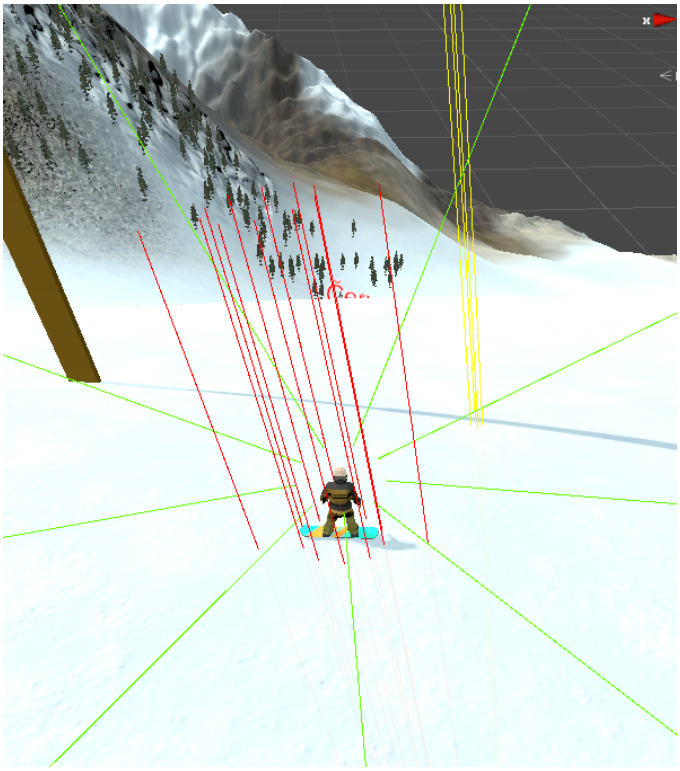
Senzory vyznačené červenou barvou slouží k získání informací o svahu. Senzor nestačí pouze jeden, protože povrch se může v některých místech prudce měnit, což by mohlo způsobit skokové přechody snowboardisty. Větší množství senzorů a jejich následné průměrování (viz sekce Třída SensorManager) umožňují hladší pohyb. Po vyzkoušení několika konfigurací je nakonec použito devět senzorů v kruhu kolem snowboardisty a dva o kus dále nalevo a napravo od snowboardisty.

Žluté senzory

Senzory vyznačené žlutě slouží k zjištění, zda snowboardista najel na skok.

Zelené senzory

Zelené senzory slouží k zajištění toho, aby snowboardista nevyjel z trati.



Obrázek 10.1: Vizualizace senzorů použitých ve hře.

10.1.2 Třída SensorManager

Na obrázku 10.1 lze vidět, že senzory jsou rozděleny do tří skupin. Třída `SensorManager` se stará vždy o jednu skupinu senzorů a poskytuje rozhraní pro získání informací o tom, co detekují. Z komponenty `SensorManager` lze získat tyto informace.

- Jaká je největší vzdálenost, ve které jeden ze senzorů zaregistroval kolizi.
- Jaká je průměrná normála povrchu, se kterým dochází ke kolizi.
- `Tag` herního objektu, který skupina senzorů detekuje.

10.2 Druhy pohybů

V komponentě `SnowboardController` je v její metodě `FixedUpdate` volána jedna ze čtyř metod podle toho, v jakém stavu se snowboardista nachází. Jedná se o následující metody:

- `ridingPlayer()` – pokud snowboardista jede po svahu,
- `jumpingPlayer()` – pokud je snowboardista v letu,
- `ridingUpJump()` – pokud se snowboardista nachází na skokánku,
- `landingPlayer()` – pokud je snowboardista v letu, ale nachází se už blízko nad zemí.

10.3 Jízda

Jízda po svahu je obsluhována metodou `ridingPlayer()` komponenty `SnowboardController`. Zde je uveden kód její implementace.

```
void ridingPlayer()
{
    yRotationAdjuster.AdjustIfActive();
    normalDirectionAdjuster.AdjustIfActive();
    tiltAdjuster.AdjustIfActive();
    forwardMover.AdjustIfActive();
    borderAdjuster.AdjustIfActive();
    heightAdjuster.AdjustIfActive();
}
```

Všechny instance, nad kterými je ve výše uvedené ukázce zavolána metoda `AdjustIfActive`, jsou potomky abstraktní třídy `AAdjuster`. Dále bude popsán význam každé z těchto tříd a náhled na implementaci některých z nich.

10.3.1 YRotationAdjuster

Tato třída zajišťuje zatáčení. Na základě kontrolního vektoru (viz Ovládání snowboardisty) je snowboardista otočen kolem své osy Y. V Unity je osa Y směrem nahoru. Pro správné otáčení je nezbytné, aby byl snowboardista otáčen kolem osy Y, která je relativní k němu, ne k hernímu světu.

Ovládání snowboardisty eye–trackingem bylo při vyšších rychlostech nezvladatelné. Implementačním detailem, který výrazně zvyšuje ovladatelnost je, že se snowboardista otáčí tím pomaleji, čím rychleji se pohybuje.

10.3.2 NormalDirectionAdjuster

Třída `NormalDirectionAdjuster` otáčí snowboardistu tak, aby normála povrchu, po kterém se pohybuje, procházela jeho osou Y. Normálový vektor povrchu je získán z komponenty `SensorManager`. Senzory, které jsou pro tento účel použity, jsou na obrázku 10.1 vyznačeny červeně.

Pro jednoznačné určení požadované rotace nestačí pouze normálový vektor povrchu. Snowboardista by totiž mohl být libovolně otočený kolem své osy Y.

Unity poskytuje metodu `Quaternion.LookRotation(forward, upward)`. Ta dopočítá rotaci ze dvou vektorů, `forward` a `upward`. `forward` je vektor, který by vznikl po aplikování této rotace na vektor $(0, 0, 1)$. `upward` by vznikl po otočení vektoru $(0, 1, 0)$.

Pro výpočet rotace je `upward` normála povrchu. `forward` je směr, který má být pro snowboardistu po této rotaci směrem dopředu. Směrem dopředu je myšleno $(0, 0, 1)$ relativně k snowboardistovi. `forward` lze vypočítat promítnutím současného vektoru dopředu na rovinu určenou normálovým vektorem povrchu. Následuje ukázka kódu pro výpočet nové požadované rotace. Výsledná rotace je zde reprezentována jako kvaternion.

```
Vector3 newDesiredForward =
Vector3.ProjectOnPlane(target.forward, normal);
Quaternion desiredRotation =
Quaternion.LookRotation(newDesiredForward, normal);
```

Nově vypočítaná rotace `desiredRotation` ale není přímo aplikována na snowboardistu, aby náhlé změny povrchu nezpůsobily náhlé skoky v jeho orientaci. Namísto toho je použita interpolace `desiredRotation` a aktuální rotace snowboardisty. Interpolace dvou rotací je korektně definována nad kvaterniony. Unity poskytuje metodou `Quaternion.RotateTowards(from, to, maxDegreesDelta)`, která tuto interpolaci provede.

10.3.3 TiltAdjuster

Třída `TiltAdjuster` se stará o to, aby se snowboardista nakláněl při zatáčení. Snowboardista se při zatáčení mírně otočí kolem osy, která prochází špičkou a patou jeho snowboardu. Třída `TiltAdjuster` nemá žádný vliv na

jízdu snowboardisty. Pouze otáčí potomka herního objektu snowboardisty, na kterém je jeho mesh.

10.3.4 ForwardMover

Třída `ForwardMover` zajišťuje jízdu snowboardisty. Ten je v každém snímku posouván ve směru, který je dopředu relativně k němu. Následuje zjednodušená ukázka kódu, který snowboardistou posouvá.

```
Vector3 forwardMovement = target.forward * speed;
target.Translate(forwardMovement, Space.World);
```

V této třídě je nejdůležitější výpočet, jakou rychlostí má snowboardista jet, tedy atributu `speed`. Rychlost závisí na kontrolním vektoru (viz sekce Výpočet kontrolního vektoru snowboardisty) a na tom, jak moc z kopce snowboardista jede. Jak už bylo zmíněno v sekci Výpočet kontrolního vektoru snowboardisty, druhá složka kontrolního vektoru určuje snowboardistovu akceleraci.

Následuje algoritmus, který určuje, pomocí které metody se má zrovna přepočítávat rychlost.

```
if (!ridingUp())
{
    if (isVerticalInputPostive())
        acceleratingSpeed();
    else
        breakingSpeed();
}
else
    upRidingSpeed();
```

Metoda upRidingSpeed()

Pokud se snowboardista pohybuje směrem nahoru, měl by začít zpomalovat. Toto zpomalování by mělo být tím rychlejší, čím je kopec prudší. Následuje zjednodušená ukázka kódu, kterým je tohoto docíleno. `rotationOffset` je konstanta přičtená k rotaci snowboardisty, slouží k tomu, aby bylo možné jet do mírného kopce. Užití `rotationOffset` pomůže zamezit zaseknutí se na rovinkách.

```
speed += sin(target.eulerAngles.x + rotationOffset) *
upRidingMultiplier;
```

Metoda acceleratingSpeed()

Tato metoda je volána v okamžiku, kdy chce hráč zrychlovat. K tomu dochází v případě, že je druhá složka kontrolního vektoru kladná (tedy metoda `isVerticalInputPostive()` vrací hodnotu `true`). Při výpočtu je k rychlosti přičtena druhá složka kontrolního vektoru vynásobená konstantou. Tato konstanta (`accelerationMultiplier`) určuje velikost zrychlení. Následuje zkrácená ukázka kódu.

```
speed += accelerationMultiplier * controlVector.y;
```

Metoda breakingSpeed()

Pokud chce hráč zpomalit, proběhne stejný výpočet jako v `acceleratingSpeed()`. Zde je ale druhá složka kontrolního vektoru vynásobena jinou konstantou. Ta má nižší hodnotu než konstanta `accelerationMultiplier` v metodě `acceleratingSpeed()`, aby se zamezilo neúmyslnému prudkému zpomalení.

Zarovnání rychlosti

Kdyby byl algoritmus implementován pouze tak, jak je popsáno výše, tak by snowboardista neustále zrychloval, nebo se mohl dostat do situace, kdy jede pozadu[1]. Rychlost je tedy omezena následující zkrácenou řádkou kódu.

```
speed = max(0, min(speed, maxSpeed));
```

Pro ovládání snowboardisty je intuitivní, že hráč by měl dosahovat tím vyšších rychlostí, čím výše nad snowboardistu se dívá[2]. Z tohoto důvodu je maximální rychlost `maxSpeed` přímo úměrná druhé složce kontrolního vektoru.

10.3.5 BorderAdjuster

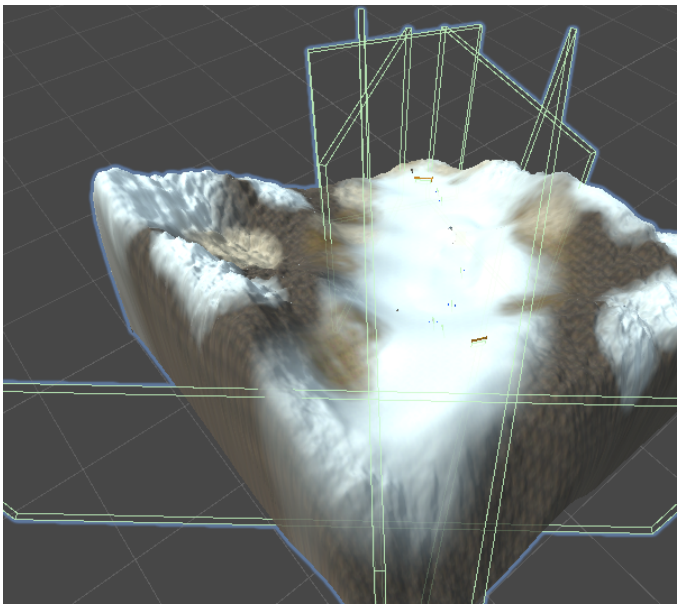
Třída `BorderAdjuster` zamezuje vyjetí snowboardisty z trati. Oblast, ve které se může snowboardista pohybovat, je pro každý svah vymezena několika herními objekty, které mají komponentu `BoxCollider`. Tyto objekty

[1] Například kdyby se hráč snažil vyjet svah. To sice odpovídá realitě, ale pro jednoduché ovládání je lepší, aby se snowboardista zastavil a hráč tím dostal příležitost se otočit.

[2] Druhá složka kontrolního vektoru.

jsou vidět na obrázku 10.2 a jsou zde vykresleny zeleně drátěným modelem. Díky této komponentě zaregistrují senzory s herním objektem kolizi.

Pomocí senzorů, které jsou na obrázku 10.1 vyznačeny zeleně, je zjištěno, která stěna je snowboardistovi nejblíže. Pokud se snowboardista přiblíží, stěna začne červenat tím více, čím je snowboardista blíže. Při dostatečném přiblížení jsou jako první pozastaveny všechny ostatní pohyby. Následně je snowboardista vrácen na pozici, na které se nacházel před jednou sekundou. Poté se snowboardista po dobu tří sekund nepohybuje dopředu, aby měl hráč čas otočit snowboardistu.



Obrázek 10.2: Ukázka vizualizace hranic svahu v prvním kole.

Obrázek 10.3: Ukázka toho, co vidí hráč při přiblížení k hranici.

10.3.6 HeightAdjuster

Třída `HeightAdjuster` udržuje snowboardistu ve správné výšce nad svahem. K tomu jsou znovu využity senzory, které jsou vyznačené na obrázku 10.1 červeně. V každém snímku je snowboardistova pozice změněna tak, aby byl v určené vzdálenosti nad povrchem ve směru normály daného povrchu.

10.4 Jízda na skokánku

Jízda na skokánku je řešena stejně jako jízda po svahu. Jediný rozdíl je, že pohyb dopředu zajišťuje jiná instance třídy `ForwardMover`. Ta má oproti normální jízdě nastavenu vyšší rychlost a dovoluje hráči jet do prudšího kopce. Díky tomu se nestane, že by se snowboardista zasekl uprostřed skoku, pokud se o to hráč přímo nesnaží.

10.5 Skok

Let snowboardisty po vyskočení na skokánku je obsluhován pouze třídou `JumpHandler`.

Na přechodu ze stavu jízdy do stavu skoku je uložen vektor dopředu relativně k snowboardistovi, vynásobený jeho aktuální rychlostí jako `moveVector`. Ten dále určuje směr a rychlost snowboardisty v letu. Efekt gravitace je poté simulován následovně.

```
moveVector += new Vector3(0, -gravity, 0);
```

Tento vektor je poté upraven tak, aby jeho délka nebyla větší než dané maximum. Tato úprava zajistí, aby snowboardista nepadal příliš rychle. Dále je snowboardista otočen tak, aby jeho směr dopředu byl směr totožný s `moveVector`. Nakonec je snowboardista posunut podle `moveVector`.

10.6 Dopad

Dopad je také kontrolován instancí třídy `JumpHandler`. Jediný rozdíl oproti letu je v tom, že má nastavenu nižší maximální rychlost než instance `JumpHandler`, která ovládá průběh skoku.

11 Herní prostředí

Tato kapitola se věnuje vytváření herního prostředí, ve kterém se snowboardista pohybuje.

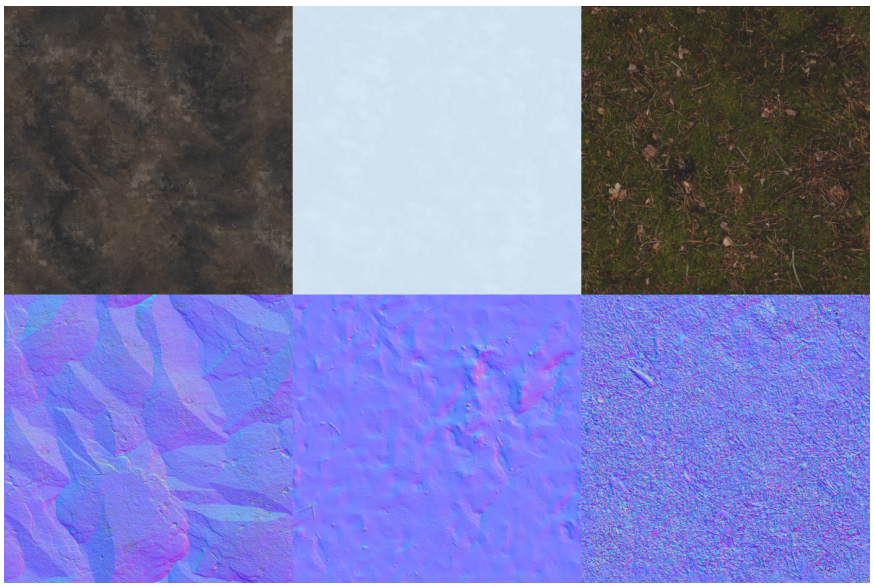
11.1 Převzaté grafické assety

Výsledná hra by se neobešla bez několika grafických assetů, které nebyly vytvořeny mnou. Ty budou v této sekci představeny a budou zde uvedeny jejich zdroje.

11.1.1 Textury

Při vytváření svahů byly použity tři textury, které jsou zdarma součástí Unity. Na obrázku 11.7 jsou tyto textury zobrazeny. Pod každou z textur je uvedena její normálová mapa.

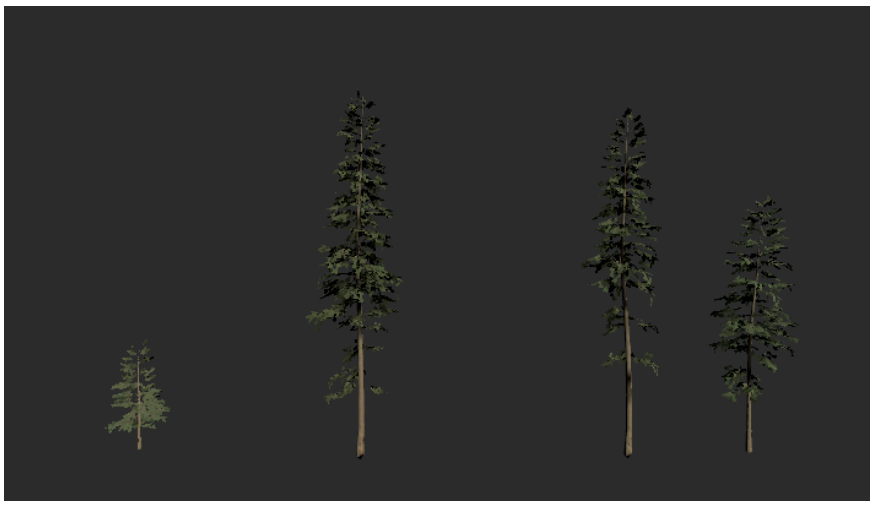
- Sníh – Na obrázku 11.7 uprostřed.
- Kámen – Na obrázku 11.7 vlevo.
- Mech – Na obrázku 11.7 vpravo.



Obrázek 11.1: Textury, které byly ve hře použity.

11.1.2 Stromy

Stromy jsou dostupné zdarma na Unity Asset Store[37]. Tento balíček obsahuje čtyři jehličnaté stromy různých velikostí. Stromy jsou vytvořeny s podporou pro tři úrovně detailu podle toho, jak blízko kamery se nacházejí. Úrovně detailu mezi sebou hladce přecházejí. Stromy navíc úplně zmizí, když je kamera dostatečně daleko.



Obrázek 11.2: Stromy, které byly ve hře použity.

11.1.3 Posed

Ve hře se na několika místech objevuje posed. Ten je taktéž dostupný zdarma na Unity Asset Store, odkaz zde[29].

Obrázek 11.3: Posed, který byl ve hře použit.

11.1.4 Nebe

Nebe je taktéž dostupné zdarma z Unity Asset Store. Odkaz je zde[34].

Obrázek 11.4: Nebe použité ve hře.

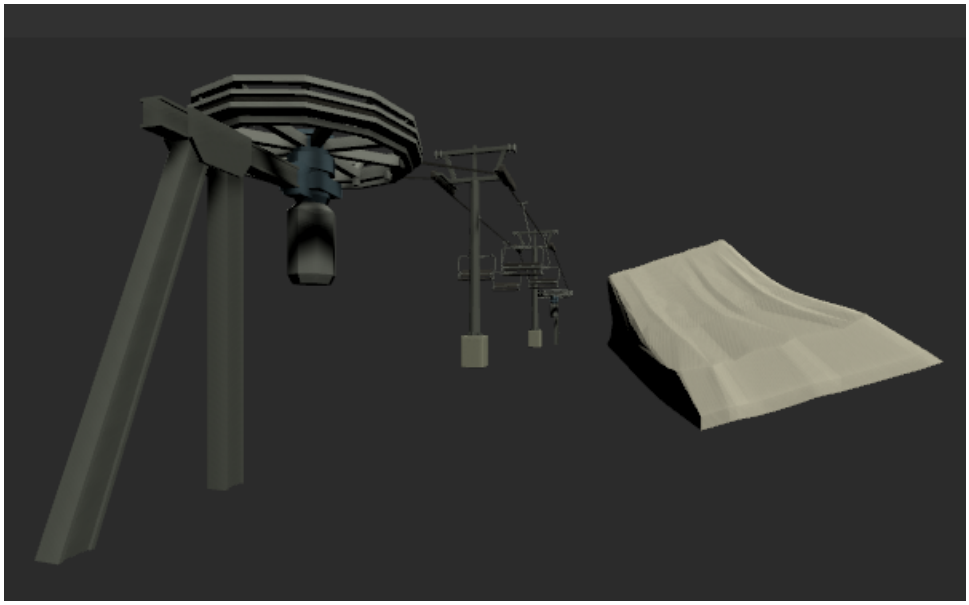
11.1.5 Balíček POLYGON Snow Kit

Zbytek grafických assetů je z balíčku POLYGON Snow Kit. Ten byl pro realizaci této bakalářské práce zakoupen na Unity Asset Store za 9 €. Odkaz

je zde[33]. Tento balíček obsahuje velké množství otexturovaných modelů, které souvisí se snowboardingem a lyžováním.

Ve hře byly z balíčku použity následující modely: snowboardista, snowboard, branka, sněžné dělo, skokánek, sněhulák a modely, ze kterých bylo možné poskládat lanovku.

Obrázek 11.5: Grafické assety z balíčku POLYGON Snow Kit: zleva branka, snowboardista, sněhulák, snowboard.



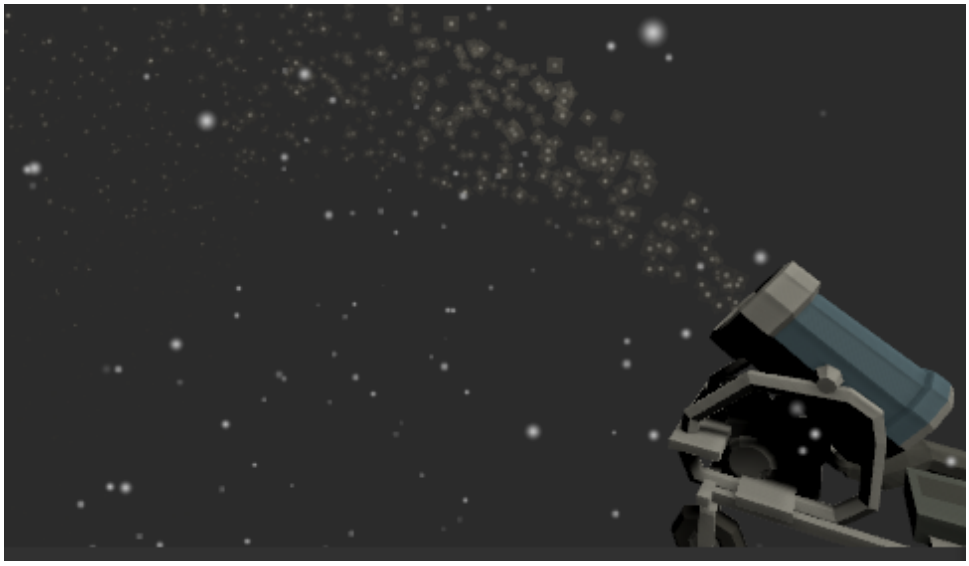
Obrázek 11.6: Grafické assety z balíčku POLYGON Snow Kit: lanovka, skokánek.

11.2 Nepřevzaté grafické assety

Některé grafické assety jsem pro tuto hru vytvořil sám. V této sekci budou tyto grafické assety představeny.

11.2.1 Částicové efekty

Pro hru jsem vytvořil tři částicové efekty: padající sníh, sníh, který se objevuje za snowboardem, a sníh, který vychází ze sněžného děla. Unity poskytuje komponentu, která umožňuje jednoduché vytváření částicových efektů. Tato komponenta se nazývá `ParticleSystem`. Částicový efekt padajícího sněhu není simulován v celém kole, ale pouze v oblasti, která se pohybuje se snowboardistou.



Obrázek 11.7: Částicový efekt padajícího sněhu a sněžného děla.

11.2.2 Animace snowboardisty

Animace snowboardisty byly vytvořeny v Unity pomocí systému pro tvorbu animací, který je založený na klíčových snímcích.

Snowboardista se může nacházet ve třech stavech, které určují jeho animace. Těmito stavy jsou jízda, skok a dopad.

Celkem bylo vytvořeno osm animací. Skok, salto, dopad, pomalá jízda dopředu, rychlá jízda dopředu, klid, jízda doleva, jízda doprava.

Dále bude vysvětlen princip, kterým je animována jízda snowboardisty. Na obrázku 11.8 lze vidět klíčové snímky pomalé jízdy dopředu. Poslední snímek je stejný jako první, aby se sekvence mohla plynule opakovat. Mezi

těmito třemi snímky je během animace interpolováno zvolenou křivkou. Na obrázku 11.9 lze vidět animace jízdy doleva. Ta je tvořena pouze jedním snímkem. Tyto animace jsou mezi sebou za jízdy prolínány[1]. Prolínání je určeno dvěma parametry. Rychlostí, kterou snowboardista jede, a první složkou kontrolního vektoru, která určuje, jak moc chce hráč zatočit.

Obrázek 11.8: Klíčové snímky animace pomalé jízdy snowboardisty.

Obrázek 11.9: Klíčový snímek animace jízdy doleva.

11.2.3 Terén

Svah je vytvořen pomocí komponenty `Terrain`, kterou poskytuje Unity. Obrázek 11.10 ilustruje práci s komponentou `Terrain`. Napravo v nabídce označené číslem dva si uživatel vybere operaci, v tomto případě `Raise or Lower`

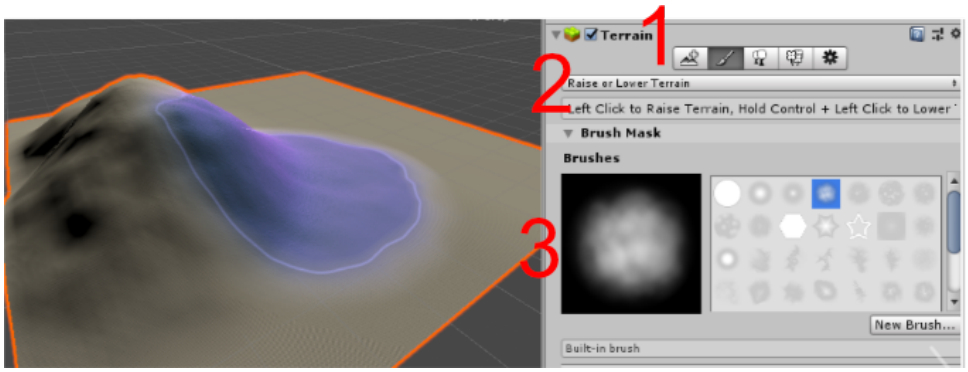
[1] Anglicky animation blending.

`Terrain` (zvednout nebo snížit terén) a pod tím si v nabídce označené číslem tři vybere štětec. Štětec určuje, jak velký bude mít operace vliv. Bílá znamená nejsilnější, černá žádný. Nalevo lze vidět vytvářený terén, modrofialová oblast je vizualizace štětce. Jeho pozici určuje myš.

Kromě modelování svahu pomocí `Raise or Lower Terrain` jsem použil ještě operaci `Smooth Height`. Ta vyhladí vyznačenou oblast. Tím jsem docílil hladkého povrchu svahu.

Dále jsem využil operace `Paint Texture`. Pomocí ní je možné aplikovat určenou texturu na vytvářený terén.

Kromě vytváření a texturování modelu terénu umožňuje komponenta `Terrain` i plošné pokládání stromů. Tuto volbu uživatel vybere kliknutím na prostřední možnost v nabídce označené číslem jedna.



Obrázek 11.10: Ukázka práce s komponentou `Terrain`.

12 Persistence

Některá data je třeba uchovat i po vypnutí hry. Jedná se o dosažené výsledky pro každé kolo, která kola již uživatel splnil, které vstupy chce uživatel používat.

Unity poskytuje jednoduchý systém pro persistentní ukládání dat pomocí třídy `PlayerPrefs`. Tento systém ale umožňuje pouze uložení primitivních datových typů, ty jsou identifikovány řetězci. Další možností by bylo ukládání do databáze. Toto řešení by, dle mého názoru, bylo pro uložení těchto dat zbytečně komplexní. Pro tuto hru byl zvolen systém založený na `ScriptableObject`. Implementace takového systému je jednoduchá a lze ho snadno používat bez přidávání dalšího kódu.

12.1 Implementace

Potřebná data jsou za běhu hry uložena do tříd návrhového vzoru přepravka, které dědí od `ScriptableObject`. Data uložená tímto způsobem zůstávají uložena napříč scénami. Unity poskytuje třídu `JsonUtility`, která umožňuje serializovat a deserializovat instance `ScriptableObject` do formátu JSON (Javascript Object Notation). Toto zajišťuje třída `Serializer`.

Komponenta `Serializer` je na prázdném herním objektu v hlavním menu. Do jejího pole typu `ScriptableObject` jsou v Unity přidány všechny assety dat, které je potřeba uložit. Ve třídě `Serializer` je zavolána metoda `DontDestroyOnLoad()`. To zapříčiní, že se herní objekt, na kterém komponenta je, nesmaže při načtení jiné scény. Při zapnutí hry jsou všechna data podle názvu jejich assetu načtena do příslušných instancí `ScriptableObject`. Při vypnutí hry jsou data serializována a uložena do složky `NeuroGames/Snowboard/data` ve skryté složce `%appdata%` v OS Windows. Následuje ukázka kódu uložení jedné instance `ScriptableObject`, zde pojmenovaná obj. Načtení dat probíhá obdobně.

```
BinaryFormatter binaryFormatter = new BinaryFormatter();
FileStream file = File.Create(basePath + "/" + obj.name);
var json = JsonUtility.ToJson(obj);
binaryFormatter.Serialize(file, json);
file.Close();
```

13 Testování

První testování částečně rozpracované hry probíhalo při Dnu otevřených dveří FAV. Při tomto testování byly získány první připomínky, které byly následně zapracovány při vývoji hry. Další testování jsem si domluvil na Gymnáziu Ostrov. Dále jsem měl v rámci zadání bakalářské práce hru otestovat při příležitosti projektových dnů FAV. K žádnému z těchto testování nedošlo z důvodu vyhlášení nouzového stavu v důsledku pandemie nemoci Covid–19. S vedoucím bakalářské práce jsem se proto dohodl, že hru otestují moji rodinní příslušníci a kamarádi. K testování se mi podařilo sehnat i absolventku studijního oboru ergoterapie. Hru jsem otestoval se sedmi lidmi ve věkové kategorii dvacet až dvacet pět let. Dále si hru vyzkoušely dvě osoby ve věku padesát až padesát pět let a jedna osoba ve věku sedmdesát let. Testování probíhalo tak, že jsem hru vždy s někým vyzkoušel a poté jsem ji před dalším testováním upravil.

Pro otestování atributů třídy `SnowBoardController`, které zodpovídají za pohyb snowboardisty, jsem napsal dvanáct jednotkových testů. Tyto testy určitě nestačí pro důkladné pokrytí třídy `SnowBoardController` a už vůbec ne celé hry. K vytvoření mock objektů pro účely testování byla využita knihovna `NSubstitute`[32].

V průběhu testování bylo odhaleno několik nekritických chyb, které byly následně opraveny. Z testování jsem získal především různé připomínky, které jsem následně zapracoval do finální verze hry. Výsledky testování jsem rozdělil do několika kategorií.

13.1 Chyby

Při testování byly objeveny pouze tři chyby. Jedna spočívala v tom, že po kliknutí na některá tlačítka se hráč dostal do jiné scény, než měl. Důležité bylo povšimnutí si absence zábran, které jsou kolem trati. Zábrany nebyly v sestavené verzi, protože byly v Unity označeny jako `EditorOnly`. Poslední chybou bylo to, že se text „Výborně, postupuješ do dalšího kola.“ vypisoval dvakrát.

13.2 Využití eye–trackingu

Využití eye–trackingu pro ovládání hry se téměř všem testovaným líbilo. Pro dobrou zkušenost s eye-trackingem je ale důležité uživatele nejdříve s touto technologií seznámit a eye–tracker zkalibrovat. Obojí lze udělat s pomocí Tobii Eye Tracking Core Software[36]. Ten nabízí kalibraci, která zabere asi jednu minutu, a hru, která zabere asi čtyři minuty. Ze začátku je také dobré zapnout funkcionalitu, která uživateli ukazuje, kam se dívá. Ta pomůže uživateli ujistit se, že eye–tracking funguje dostatečně přesně, nebo může odhalit to, že snímání nefunguje dobře. S tím jsem se v průběhu testování setkal u dvou lidí, kteří nosí brýle.

Hru jsem vyzkoušel na čtyřech uživatelích s brýlemi. Dva z nich neměli s ovládáním žádné problémy. Dvěma eye–tracker přestával chvilkami snímat oči a nebyl dostatečně přesný. Největší problém měl uživatel, který používá multifokální brýle. Uživatel se domníval, že je to způsobené tím, že je naučen dívat se na obrazovku počítače přes jednu konkrétní zónu skla. Při sledování obrazovky tedy nepohybuje tolik očima, ale hlavou.

Celkově byly zkušenosti s touto technologií pozitivní. Uživatelé si většinou do patnácti minut zvykli ovládat hru i menu bez větších potíží.

13.3 Využití BCI

BCI zprostředkované zařízením Mindwave se celkově líbilo méně než eye–tracking. Uživatelům se nelíbilo, že neměli úplnou kontrolu jako při eye–trackingu. Někteří uživatelé měli pocit, že zařízení nefunguje dostatečně dobře, tedy nepozná, kdy se uživatel soustředí a kdy ne. Většina uživatelů i přesto ocenila zkušenost s touto technologií. Asi polovina se snažila se zlepšit, zato druhou polovinu odradil pocit, že nic neovlivní.

13.4 Vizuální stránka

Velká část připomínek se týkala vizuální stránky. Toto jsou některé připomínky, které jsem v rámci testování řešil.

- Málo viditelný text instrukcí.
- Vločky, na které se má uživatel dívat pro úspěšné splnění skoku, splývají barvou s nebem.
- Skoky by měly být zvýrazněny, protože nejsou ve sněhu dostatečně viditelné.

- Černé branky v herním druhu Barvy jsou příliš podobné modrým.
- Světlo v určitém kole je otočeno tak, že velká část trati je ve stínu.

13.5 Hratelnost

Díky testování jsem mohl vidět, jak ostatní projíždějí všechna kola této hry. To bylo velmi důležité pro vyladění hry tak, aby se co nejlíp hrála.

Během testování jsem posouval branky na jiná místa tak, aby bylo možné hru projet s minimálním zpomalováním. Zpomalování se totiž většina hráčů nenaučila. Některé branky byly následně i odebrány, aby nebyla první kola příliš náročná. Také jsem ve většině kol po testování nastavil nižší maximální rychlost snowboardisty a menší parametr `maxXDistance` (viz sekce Výpočet kontrolního vektoru snowboardisty). To má za důsledek, že se uživatel nemusí podívat až tak daleko od snowboardisty, aby prudce zatočil.

13.6 Kognitivní náročnost

Z testování bylo jasné, že nejnáročnější jsou kola, ve kterých si uživatel musí zapamatovat čísla branek, které má projet. Zajímavé je, že hráč si většinou pamatuje čísla branek do té doby, než vyskočí na skoku. Ve druhu hry, ve kterém má hráč vždy projet branku s číslem o jedna větší než předchozí, uživatelé taktéž zapomínali, kterou branku již projeli až po vyskočení. Ostatní druhy her sice vyžadovaly vysokou míru soustředění, ale nedělaly nikomu vážnější problémy.

13.7 Ostatní připomínky

- Instrukce by neměly zmizet po určitém čase hráč by měl mít možnost potvrdit, že si instrukce již dočetl.
- Udělat jedno kolo, ve kterém bude vysvětleno ovládání a princip hry.

13.8 Shrnutí

Velká část z těchto připomínek byla získána od vystudované ergoterapeutky, která má s využitím eye–trackingu pro kognitivní trénink praktické zkušenosti. Hra se jí jinak celkově líbila. I přes malé množství uživatelů, kteří si hru vyzkoušeli, si myslím, že byla hra důkladně uživatelsky otestována.

14 Zhodnocení dosažených výsledků

V rámci této bakalářské práce byla vytvořena kompletní a funkční hra pro terapii pacientů s kognitivními poruchami. Tuto hru včetně menu lze ovládat pouze pohledem. Uživatel nebo terapeut mají možnost měnit, čím bude ovládáno menu, menu ve hře a samotná hra. Menu mohou být ovládána pohledem, hra může být navíc ovládána pomocí šipek. Hra navíc podporuje připojení zařízení Mindwave, které ve většině kol pouze zobrazuje míru soustředění nebo meditace podle toho, co si uživatel zvolí. Ve hře je ale i devět kol, která lze ovládat pouhým soustředěním nebo mírou meditace.

Vlastní hra se dělí na pět druhů her, které se liší kognitivním úkonem. Čtyři druhy her spočívají v tom, že hráč očima ovládá snowboardistu, který se pohybuje po svahu. Snowboardista má vždy projet pouze některé branky podle nějakého pravidla. Které branky má snowboardista projet, je generováno náhodně. Na svahu jsou i skokánky. Když snowboardista vyskočí, zpomalí se čas a hráč se musí rychle podívat na pět náhodných bodů na obrazovce. Pokud tak učiní, snowboardista udělá salto a hráč získá body. Kola ovládaná pomocí Mindwave spočívají v tom, že snowboardista jede na sérii skokánků, a když se hráč dostatečně soustředí, snowboardista udělá salto a hráč získá body.

Celkově hra obsahuje čtyřicet pět různých kol, obsahuje prvky pro kognitivní trénink a je navržena tak, aby v ní byly uplatněny poznatky ze sekce Společné charakteristiky. Důležité je především to, že hra je ze začátku jednoduchá a obtížnost se pomalu zvyšuje. Terapeut může obtížnost hry i měnit individuálně, pro každého pacienta. Obtížnost hry lze změnit tím, že je změněn počet branek, které je nutné správně projet pro postoupení do dalšího kola. Další důležitou vlastností hry je to, že před každým kolem jsou uvedeny instrukce. Byla také vytvořena scéna s detailním návodem, ve které se hráč naučí ovládat snowboardistu a pochopí princip hry. Pro seznámení se s menu jsem vytvořil uživatelskou příručku (viz Příloha A: Uživatelská příručka).

Myslím si, že hra je po grafické stránce povedená a je zábavné ji hrát. Na druhou stranu nejsem zcela spokojený se vzhledem menu. Všechny elementy menu jsou velmi velké a obarvené výraznými barvami. Takto uspořádané menu ale vzniklo z důvodu, aby jej bylo možné jednoduše ovládat pomocí eye–trackingu. Jedním z možných rozšíření by bylo vizuálně vylepšit celé

menu.

Jako další rozšíření by bylo dobré vytvořit systém, ve kterém by mohl terapeut jednoduše vytvářet nová kola a ty přidat mezi ostatní. Dále by bylo možné přidat další herní druhy a implementovat jiný úkon, který má hráč provést v letu. Dále by bylo dobré dodělat systém pro vytvoření uživatelských profilů. Z grafické stránky by bylo ještě dobré přidat více různých animací snowboardisty. To, jestli budou tato rozšíření implementována, záleží především na tom, zda budou nemocnice mít o hru zájem.

Hru by bylo také dobré nejdříve otestovat v nemocnicích s cílovou skupinou. Určitě by bylo vhodné napsat větší množství jednotkových a integračních testů, pro zajištění kvality.

Celkově jsem spokojen s konečnou podobou hry. Myslím si, že ovládání této hry pomocí eye–trackingu je intuitivní a hru se lze rychle naučit. Celkově jsem s prací s eye-trackerem Tobii 4C velmi spokojen. Za celou dobu jsem s ním nenarazil na žádné problémy a pro tyto účely byl dostatečně přesný.

Na druhou stranu to stejné neplatí pro Mindwave. Se softwarem TGC[1] jsem měl několik problémů a připojení Mindwave k mému PC jako uživateli mi zabralo několik hodin. Dále nejsem zcela spokojen s kvalitou odhadu míry soustředění. Celkově to už nepovažuji za dobrý způsob ovládání počítačové hry.

[1]Software, který je nutné nainstalovat pro připojení Mindwave na OS Windows.

15 Závěr

V rámci této bakalářské práce jsem se nejdříve dozvěděl velké množství informací o tom, na jakém principu fungují technologie EEG a eye–tracking. Především jak je možné vytvořit BCI. Dále jsem zjistil, jak je možné tyto technologie využít pro ovládání her. Následně jsem prozkoumal různá dostupná zařízení, která jsou na trhu. Z nich jsem s vedoucím bakalářské práce vybral dvě, která se nejvíce hodí pro implementaci hry ovládané s využitím těchto technologií. Vybraná zařízení jsem prozkoumal. Zjistil jsem jejich technické parametry, v jakých aplikacích už byly využity a jak je možné využít jejich API.

Dále jsem prozkoumal, jaké hry se využívají pro kognitivní trénink. Na základě těchto informací jsem určil, jaké vlastnosti by měla taková hra mít. Tyto vlastnosti jsem využil při návrhu hry, která může sloužit pro kognitivní trénink a využívá obě technologie.

Následně byla vytvořena hra, která využívá eye–trackingu a EEG signálu pro své ovládání. Tato hra může být použita pro terapii pacientů s kognitivními poruchami. Pro vyžití této hry pro kognitivní trénink není nutné vlastnit žádné zařízení, protože ji lze ovládat i myší nebo klávesnicí. Hra je ale jinak uzpůsobena tak, aby ji bylo možné celou ovládat pouze pomocí eye–trackeru.

Z důvodu pandemie viru Covid-19 jsem nemohl kompletní hru otestovat na projektových dnech FAV, její první verzi jsem ale otestoval na Dnu otevřených dveří FAV v lednu a poté s rodinou a přáteli. Z testování jsem i přesto dostal mnoho užitečných připomínek, které hru vylepšily. Ergoterapeutka, která hru testovala, mi potvrdila, že by hra mohla být využívána v nemocnicích.

Hra je tvořena padesáti šesti scénami, více než devadesáti zdrojovými soubory a velkým množstvím ostatních souborů. Jedná se o celkem rozsáhlý projekt.

Doufám, že hra najde své využití v praxi, kde by mohla zábavnou formou pomáhat lidem.

Literatura

[1] *EEG signal processing and its applications* [online]. Z youtubového kanálu Neuroinformatics KIV ZČU. Dostupné z: https://www.youtube.com/watch?v=eInr7NCBT7o.

[2] *EEG signal processing and its applications* [online]. Z youtubového kanálu Neuroinformatics KIV ZČU. Dostupné z: https://www.youtube.com/watch?v=GnAPfsyovfE.

[3] *Training Mental Commands* [online]. 2020. Dostupné z: https://www.emotiv.com/knowledge-base/category/detection-suites/mental-commands/.

[4] *Fabien Lotte about his goal to improve BCI usability* [online]. 2017. Dostupné z: http://blog.gtec.at/interview-fabien-lotte/?fbclid=IwAR2J6uLX2eiLmSL3rmLBxJfuOt1cBxWm8RX3Rtywi47Bc90sFOCwV5QdLNA.

[5] *EEG Biosensor Entertainment Apps* [online]. 2019. Dostupné z: https://store.neurosky.com/collections/entertainment.

[6] *Puzlebox Orbit Mobile Edition* [online]. 2019. Dostupné z: https://www.amazon.com/gp/product/B00BTK2SRS?ie=UTF8&tag=neur02-20&camp=1789&linkCode=xm2&creativeASIN=B00BTK2SRS.

[7] *Introducing Muse 2* [online]. 2019. Dostupné z: https://choosemuse.com/muse-2/.

[8] *EMOTIV EPOC+ 14 Channel Mobile EEG* [online]. Dostupné z: https://www.emotiv.com/product/emotiv-epoc-14-channel-mobile-eeg/#tab-description.

[9] *Brainwave Sensing Headset* [online]. 2015. Dostupné z: https://store.neurosky.com/pages/mindwave.

[10] *EMG and EOG Artifacts – Friend, Foe or Both?* [online]. 2015. Dostupné z: http://neurosky.com/2015/05/emg-and-eog-artifacts-friend-foe-or-both/.

[11] *How Does Eye Tracking Work? Gamescom 2017* [online]. 2017. Dostupné z: https://www.youtube.com/watch?v=9uePzwLGVTY.

[12] *How do Tobii Eye Trackers work?* [online]. Dostupné z: https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/.

[13] *No. 1 in Eye Tracking* [online]. 2020. Dostupné z: https://www.tobii.com/group/about/.

[14] *Assistive technology for communication* [online]. 2020. Dostupné z: https://www.tobiidynavox.com/.

[15] *Eye tracking for research* [online]. 2020. Dostupné z: https://www.tobiipro.com/.

[16] *Welcome to Tobii Tech* [online]. 2020. Dostupné z: https://tech.tobii.com/.

[17] *Pupil Labs Shop* [online]. 2020. Dostupné z: https://pupil-labs.com/products/.

[18] *Shop* [online]. 2020. Dostupné z: https://www.gazept.com/shop/.

[19] *our products* [online]. 2020. Dostupné z: https://eyetechds.com/eye-tracking-products/.

[20] *The most crucial part of video-game development explained* [online]. 2018. Dostupné z: https://www.gamesradar.com/what-is-a-game-engine-and-what-does-it-do/.

[21] *Electrophysiological Research* [online]. 2020. Dostupné z: https://psychology.unt.edu/unt-neurocognitive-laboratory/current-research/electrophysiological.

[22] *Brain-Computer Interfaces* [online]. 2020. Dostupné z: http://www.moregrasp.eu/technologies/Brain-Computer-Interfaces.

[23] *UNREAL ENGINE 4 - Photorealistic Graphic* [online]. 2019. Dostupné z: https://www.youtube.com/watch?v=zKu1Y-LlfNQ.

[24] *Developer toolkit* [online]. 2019. Dostupné z: http://developer.neurosky.com/.

[25] *PC Gaming* [online]. 2019. Dostupné z: https://developer.tobii.com/pc-gaming/.

[26] *UNITY* [online]. 2019. Dostupné z: http://developer.neurosky.com/docs/doku.php?id=using_thinkgear_with_unity.

[27] *Mindwave Unity* [online]. 2019. Dostupné z: https://github.com/DaCookie/mindwave-unity.

[28] *Mindwave Unity* [online]. 2019. Dostupné z: https://store.neurosky.com/products/eeg-tgam.

[29] *Hunter'S Deer Blind* [online]. 2020. Dostupné z: https://assetstore.unity.com/packages/3d/props/exterior/hunter-s-deerblind-31958.

[30] *Tobii Eye Tracking SDK* [online]. 2020. Dostupné z: https://assetstore.unity.com/packages/tools/input-management/tobii-eye-tracking-sdk-90604.

[31] *Specifications for the Tobii Eye Tracker 4C* [online]. 2019. Dostupné z: https://help.tobii.com/hc/en-us/articles/213414285-Specifications-for-the-Tobii-Eye-Tracker-4C.

[32] *NSubstitute* [online]. 2020. Dostupné z: https://nsubstitute.github.io/.

[33] *POLYGON - Snow Kit* [online]. 2020. Dostupné z: https://assetstore.unity.com/packages/3d/characters/polygon-snow-kit-134501.

[34] *Skybox Series Free* [online]. 2020. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/sky/skybox-series-free-103633.

[35] *ThinkGear Connector* [online]. 2020. Dostupné z: http://developer.neurosky.com/docs/doku.php?id=thinkgear_connector_tgc.

[36] *Eye Tracking Software* [online]. 2020. Dostupné z: https://gaming.tobii.com/getstarted/.

[37] *Conifers* [online]. 2020. Dostupné z: https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076.

[38] *Unite Austin 2017 - Game Architecture with Scriptable Objects* [online]. 2017. Dostupné z: https://www.youtube.com/watch?v=raQ3iHhE_Kk&t=2793s.

[39] *Unity Atoms* [online]. 2020. Dostupné z: https://github.com/AdamRamberg/unity-atoms.

[40] *Foveated Rendering* [online]. 2020. Dostupné z: https://vr.tobii.com/foveated-rendering/.

[41] Bonnet, Laurent, L. L. Two Brains, One Game: Design and Evaluation of a Multi-User BCI Video Game Based on Motor Imagery. *IEEE Transactions on Computational Intelligence and AI in games*. 2013, 5, s. 185–192. Dostupné z: https://hal.inria.fr/hal-00784886/document?fbclid=IwAR3kmxUjCIpO3U4GFvKjIWqXOaaZwLRJ4DlVbMygYnXwhmNhlO96PPdzCbM.

[42] Farnsworth, B. *EEG Headset Prices An Overview of 15+ EEG Devices* [online]. 2019. Dostupné z: https://imotions.com/blog/eeg-headset-prices/.

[43] Farnsworth, B. *What is Eye Tracking and How Does it Work?* [online]. 2019. Dostupné z: https://imotions.com/blog/eye-tracking-work/.

[44] Fingelkurts, Alexander, F. *Composition of the EEG signal from different frequencies* [online]. 2010. Dostupné z: https://www.researchgate.net/figure/Composition-of-the-EEG-\signal-from-different-frequencies-oscillatory-activities-Five_fig3_43341609.

[45] Kraft, V. Realizace jednoduchého BCI na bázi snímače Mindwave Mobile. 2015. Dostupné z: https://otik.zcu.cz/bitstream/11025/23794/1/Bakalarska_prace_A12B0096P.pdf.

[46] Maršálek, P. *Vyšetření nervového systému* [online]. Dostupné z: http://patf-biokyb.lf1.cuni.cz/wiki/_media/vyuka/vysnervsyst.cz.ppt.

[47] Mautner, P. *Neuroinformatika metoda evokovaných potenciálů* [online]. 2007. Dostupné z: https://www.kiv.zcu.cz/studies/predmety/uir/predn/P6/Neuroinformatika.pps.

[48] Pomplun, Marc, k. *Introduction to eye tracking technology and research* [online]. 2017. Dostupné z: https://www.youtube.com/watch?v=msLPy0Jrj9I&t=3877s.

[49] Popelka, S. *EYE-TRACKING (NEJEN) V KOGNITIVNÍ KARTOGRAFII.* Univerzita Palackého v Olomouci, 2018. Dostupné z: https://theses.cz/id/ve7g7h/Popelka.pdf. ISBN 978-80-244-5313-2.

[50] Shah, K. Aashit, M. Invasive electroencephalography monitoring: Indications and presurgical planning. *Annals of Indian Academy of Neurology.* 2014. doi: 10.4103/0972-2327.128668. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4001224/.

[51] Sivarajah, Yathunanthan, T. P. T. H. *ERP signal obtained for a simple target detection task by averaging across 100 target trials* [online]. 2014. Dostupné z: https://www.sciencedirect.com/science/article/pii/S107158191300178X.

[52] Sundstedt, V. – Chalmers, A. Evaluation of Perceptually-Based Selective Rendering Techniques using Eye-Movements Analysis. *Proceedings - SCCG 2006: 22nd Spring Conference on Computer Graphics.* 04 2006. doi: 10.1145/2602161.2602179. Dostupné z: https://www.researchgate.

net/publication/228954795_Evaluation_of_Perceptually-Based_Selective_Rendering_Techniques_using_Eye-Movements_Analysis.

Seznam zkratek

API Application Programming Interface

BCI Brain Computer Interface

EEG Elektroencefalografie

EMG Elektromyografie

EOG Elektrookulografie

FAV Fakulty aplikovaných věd

GPU Graphics processing unit

JSON JavaScript Object Notation

KIV Katedra informatiky a výpočetní techniky

SDK Software developement kit

TGC ThinkGear Connector

USB Universal Serial Bus

Seznam obrázků

Seznam tabulek

16 Přílohy

16.1 Příloha A: Uživatelská příručka

První obrazovkou, kterou uživatel uvidí při spuštění hry, je hlavní menu. Hlavní menu je zobrazeno na obrázku 16.1. V celém menu je důležité barevné rozlišení tlačítek. Modrá tlačítka jsou normální tlačítka, která reagují na kliknutí. Zelená tlačítka fungují jinak. Při prvním spuštění hry jsou aktivována tím, že na něm uživatel nechá kurzor myši po dobu jedné sekundy. V nastavení je možné toto chování změnit, aby se zelená tlačítka aktivovala poté, co se na něj uživatel jednu sekundu dívá. K tomu je nezbytné mít připojený eye–tracker Tobii 4C.

Funkce tlačítek v hlavním menu (obrázek 16.1) je následující:

- Tlačítko 1 – Vypne hru.
- Tlačítko 2 – Spustí návod pro hraní hry.
- Tlačítko 3 – Otevře obrazovky s dosaženými výsledky.
- Tlačítko 4 – Otevře obrazovku menu nastavení – zobrazena na obrázku 16.2.
- Tlačítko 5 – Přesune uživatele do výběru kola, obrázek 16.5.

Obrázek 16.1: Obrazovka hlavního menu.

Funkce tlačítek menu nastavení na obrázku 16.1 je následující:

- Tlačítko 1 – Přesune uživatele do nastavení ovládání, obrázek 16.3.
- Tlačítko 2 – Přesune uživatele do nastavení obtížnosti, obrázek 16.4.
- Tlačítko 3 – Vrátí uživatele do hlavního menu, obrázek 16.1.

Obrázek 16.2: Menu, kde si uživatel vybere, co chce nastavit.

Nastavení ovládání hry je zobrazeno na obrázku 16.3. Jsou zde označena tlačítka a rozbalovací menu, která mají následující funkcionalitu.

- Rozbalovací menu 1 – Zde si lze zvolit, jak má být ovládána hra. Možnosti jsou myš, klávesnice a oči.
- Rozbalovací menu 2 – Zde si lze zvolit, jak má být ovládána většina menu. Možnosti jsou myš a oči.
- Rozbalovací menu 3 – Zde si lze zvolit, jak má být ovládáno menu, které se objeví při pozastavení hry. Možnosti jsou myš a oči.
- Rozbalovací menu 4 – Zde si lze zvolit, co má snímat čelenka Mindwave. Možnosti jsou soustředění a meditace.
- Tlačítko 5 – Pokusí se připojit zařízení Mindwave.

Obrázek 16.3: Menu, ve kterém uživatel může nastavit ovládání.

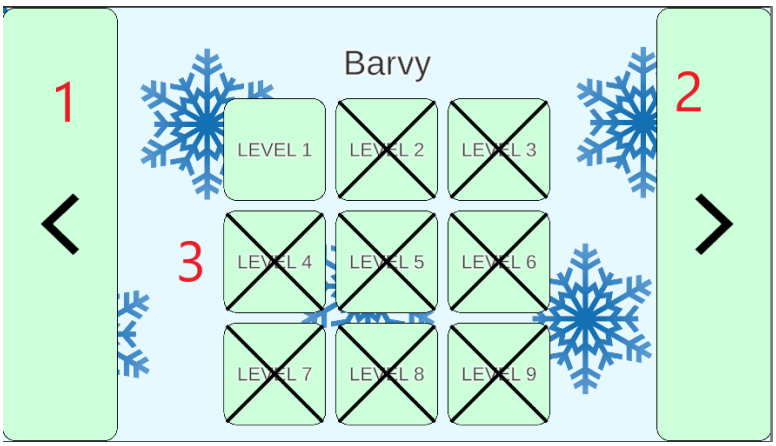
Na obrázku 16.4 je zobrazeno uživatelské rozhraní pro nastavení obtížnosti hry. Funkcionalita tohoto menu je vysvětlena dále.

- Rozbalovací menu 1 a 2 – Pomocí těchto dvou rozbalovacích menu si uživatel vybere, kterému kolu chce změnit obtížnost.
- Rozbalovací menu 3 – Zde lze změnit, kolik branek musí hráč správně projet pro postup do dalšího kola.
- Rozbalovací menu 4 – Zde lze změnit, na kolika skocích musí hráč správně vyskočit pro postup do dalšího kola.
- Element 5 – Pro kola, která jsou ovládána čelenkou Mindwave, lze nastavit hodnoty soustředění, kterých má uživatel dosáhnout pro úspěšný skok. Hodnotami musí být celá čísla v rozmezí 0–100. Pro potvrzení je nutné stisknout tlačítko se zeleným obrázkem odškrtnutí.

Obrázek 16.4: Menu, ve kterém uživatel může nastavit obtížnosti kol.

Obrázek 16.5 zobrazuje menu, ve kterém si uživatel vybírá, jaké kolo chce hrát. Pro každý druh hry je obdobné menu.

- Tlačítko 1 – Předchozí menu s výběrem kola.
- Tlačítko 2 – Následující menu s výběrem kola.
- Tlačítko 3 – Tlačítka, která spustí dané kolo. Přeškrtnutá tlačítka značí, že hráč musí nejdříve úspěšně splnit předchozí kola.



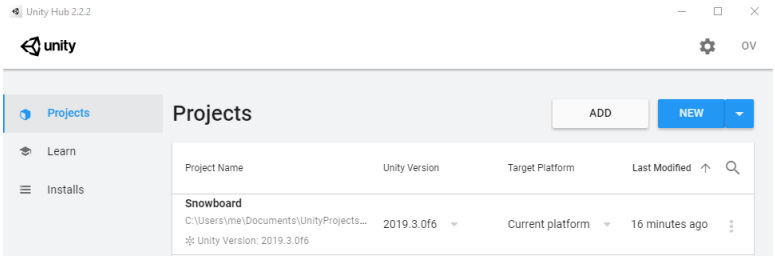
Obrázek 16.5: Menu, ve kterém si uživatel vybere, jaké kolo chce hrát.

16.2 Příloha B: Návod k sestavení

Toto je návod k sestavení hry pro OS Windows.

- Stáhněte a nainstalujte aplikaci **Unity Hub** ze stránek Unity.
- V aplikaci **Unity Hub** vyberte záložku **Installs**.
- Napravo klikněte na modré tlačítko **ADD**.
- Vyberte verzi **Unity 2013.3**.
- V následujícím menu se ujistěte, že políčko **Window Build Support (IL2CPP)** je zaškrtnuto.
- Klikněte na tlačítko **Done** a počkejte na dokončení instalace.
- V aplikaci **Unity Hub** v záložce **Projects** klikněte na bílé tlačítko **ADD**.

- Zvolte cestu ke kořenové složce projektu.
- Projekt pojmenovaný **Snowboard** by se měl přidat mezi projekty.
- Na obrázku 16.6 lze vidět výsledný stav.
- Pokud verze Unity není u projektu specifikována, nastavte ji na **2019.3.xxx** pod **Unity Version**.
- Klikněte na projekt Snowboard.
- V Unity pokračujte do **Edit–>Project Settings–>Player–>Other Settings–>Configuration**.
- Pro **Api Compatibility Level** vyberte **.NET 4.x**.
- Dále pokračujte do **File–>Build Settings**.
- Ujistěte se, že **Target Platform** je **Windows**.
- Klikněte na tlačítko **Build**.
- Vyberte cílovou složku.
- Hru lze poté spustit z kořenu cílové složky souborem **Snowboard.exe**.



Obrázek 16.6: Obrázek stavu po nainstalování Unity a přidání projektu Snowboard mezi projekty.