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

Bakalářská práce

Realizace parametrizovaných úloh pro rozvoj jemné motoriky

Plzeň 2021 Vojtěch Váchal

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vojtěch VÁCHAL**
Osobní číslo:	**A18B0340P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Realizace parametrizovaných úloh pro rozvoj jemné motoriky**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s webovou aplikací BrainIn na pracovišti KIV.
2. Prostudujte problematiku neurorehabilitace pro rozvoj jemné motoriky a navrhněte vhodné vstupní a výstupní parametry.
3. Dle požadavků specialistů navrhněte 4 úlohy a použijte vstupní a výstupní parametry uvedené v bodě 2.
4. Dle návrhu v bodě 3 implementujte 4 úlohy a začleňte je do systému BrainIn.
5. Otestujte úlohy 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: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

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

V Plzni dne 26. října 2020

Prohlášení

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

V Plzni dne 4. května 2021

Vojtěch Váchal

Abstract

The aim of this bachelor's thesis is to develop four parameterized games that will help patients with cognitive disability develop their fine motor skills. The theoretical part focuses on becoming acquainted with the BrainIn system. Following that, the concepts of brain activity and cognitive functions, as well as the relationship between neurorehabilitation, are defined. Finally, the theoretical section explains the functionality and analysis of the games that were developed. Individual games will be implemented in the practical part using technologies from the current system. These games will primarily target patients with fine motor disabilities and will be used to restore them.

Abstrakt

Tato bakalářská práce si klade za cíl vytvořit 4 parametrizované hry, které budou sloužit pro zlepšení jemné motoriky u pacientů s poruchou kognitivních funkcí. Teoretická část je zaměřena na seznámení se systémem BrainIn. Následně jsou popsány principy mozkové aktivity a kognitivních funkcí a jejich spojení s neurorehabilitací. Nakonec teoretické části je popsána funkcionalita a analýza vytvářených her. V praktické části budou jednotlivé hry implementovány prostřednictvím technologií, které jsou v existujícím systému používány. Tyto hry budou převážně zaměřeny na pacienty s poruchou jemné motoriky a sloužit k jejich rehabilitaci.

Poděkování

Mockrát bych chtěl poděkovat vedoucímu bakalářské práce Ing. Petrovi Brůhovi za odbornou pomoc a za možnost realizovat tuto práci pod jeho dohledem. Samozřejmě velké díky patří také paní Šroglové, která mi byla nápomocná při vymýšlení her, vstupních a výstupních parametrech. Poděkování také patří všem lidem, kteří se podíleli na testování her a podporovali mě při studiu.

Obsah

1 Úvod

V dnešní době se mnohé obory snaží přejít do digitální formy. Výjimkou není ani neurorehabilitace, která se zabývá rehabilitací kognitivních funkcí člověka. V posledních pár letech vývoj neurorehabilitační her roste, protože nabízí možné zvýšení motivace pacienta. Nevýhodou je, že hry nemají vlastnosti dlouhodobé motivace. Jsou totiž méně poutavé než klasické počítačové hry, které ale postrádají neurorehabilitační účinek. Zároveň se většina těchto her nedá lehce upravovat podle potřeb jednotlivých pacientů a neposkytují zpětnou vazbu terapeutovi.

Neurorehabilitační hry by neměly být náročné, naopak by jejich obtížnost měla být velmi jednoduchá, aby pacienta dokázaly motivovat a zároveň trénovat poškozenou část mozku. Celkový princip hry by měl být jednoduchý, aby jej mohly pochopit i děti v předškolním věku.

Cílem bakalářské práce je vytvořit soubor čtyř parametrizovatelných her, které budou sloužit pro neurorehabilitaci a trénink kognitivních funkcí pacientů. Konkrétně se budou zaměřovat na zlepšení a rozvoj jemné motoriky u pacientů s poruchou těchto funkcí. Vytvořené hry budou zasazeny do systému BrainIn, který spravuje množinu her napomáhajících právě takovým lidem. Zároveň by tato práce měla usnadnit práci neurorehabilitačním sestrám a vlastně všem, kteří se snaží pomoci lidem s handicapem.

Bylo třeba dodržet náležitosti, které již byly zavedené v systému, ve kterém jsou tyto hry nahrány, ale i tak je potřeba, aby hra pacienta zaujala a chtěl jí proto hrát a zlepšovat se. Hraním si totiž pacient procvičuje své kognitivní funkce, a tudíž i motoriku. Další výhodou je, že se u toho zároveň baví.

Bakalářskou práci jsem rozčlenil na teoretickou a praktickou část. V teoretické části vytvořím teoretický základ, aby se dalo lépe porozumět dané problematice. Nejdříve se zaměřím na strukturu a vlastnosti systému BrainIn, ve kterém hry budou spravovány a testovány. Další kapitola se zabývá mozkovou aktivitou a neurorehabilitací. Nakonec teoretické části představím vybrané hry, popíši jejich základní principy, vstupní a výstupní parametry.

Praktická část se věnuje implementaci her. Obsahuje popis her více do detailů. Jsou zde vysvětleny základní kroky, které vedly ke správné implementaci. Další kapitola se věnuje testování her na reprezentativním vzorku experimentů. V závěru se snažím vyhodnotit výsledky z testování a jejich vliv na současnou podobu her.

2 Existující systémy zabývající se neurorehabilitací

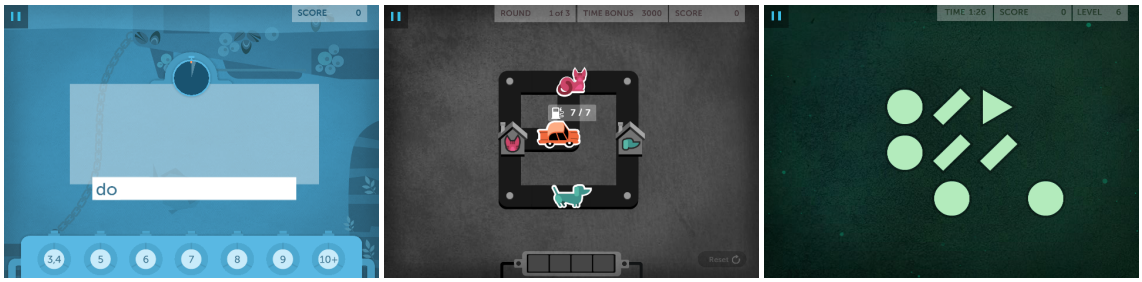
V této kapitole se zaměřím na aktuální průzkum již existujících aplikací, které se zaměřují na rehabilitaci a trénink kognitivních funkcí. Tyto systémy nabízejí možnost krátkodobé, ale i dlouhodobé rehabilitace. Jedná se o tři systémy, které se zabývají tímto problémem.

2.1 Lumosity

Lumosity je jedna z nejznámějších společností zabývající se touto problematikou. Jelikož už jsou na trhu přes několik let, mají velký počet uživatelů, který dosahuje na desítky milionů. Je to software, ke kterému je potřeba vlastnit placenou licenci. Sice lze trénovat zadarmo, ovšem v této verzi systém nabídne velice omezené možnosti oproti placené verzi programu. Jedna z licencí stojí okolo 60 dolarů na rok. [3]

Tato verze poskytuje okolo 60 her na trénink kognitivních funkcí (viz obr. 2.1), umožňuje sledování progresu uživatele a dokáže pomocí inteligentních algoritmů poskytovat personalizované cvičení podle předchozích zkušeností. [3]

Zároveň tento program podporuje určitou množinu světových jazyků, kam patří nejpoužívanější jazyky ve světě (angličtina, němčina, španělština, japonština a další). [3]

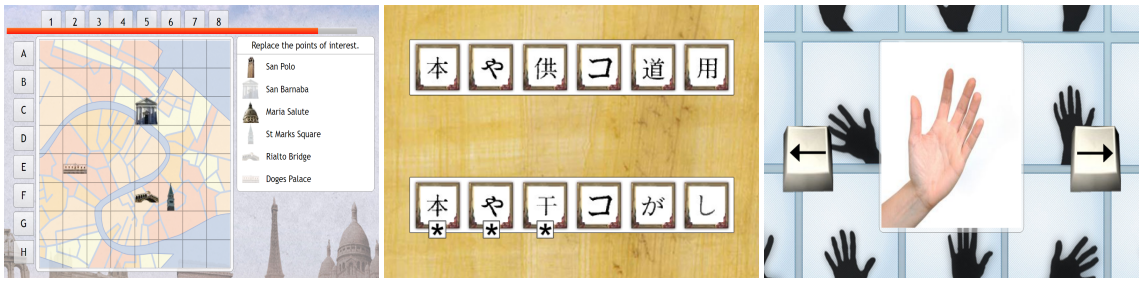


Obrázek 2.1: Ukázky úloh systému Lumosity

2.2 HappyNeuron

Tento systém se zabývá tréninkem pěti hlavních kognitivních funkcí, kam patří paměť, pozornost, vizuální a prostorové dovednosti, jazyk a výkonné funkce. Tento program je vyvíjen více než 10 let a nabízí velice kvalitní služby, které jsou bohužel zpoplatněny. Jelikož se jedná o placený software, tak má tato firma dostatek nových technologií. Protože společnost na trhu působí přes 10 let, má velkou množinu klientů, kteří její služby využívají. Ukázky her, které tento systém nabízí jsou zobrazeny na obr. 2.2. [2]

Tento systém nabízí také sledování výsledků za účelem motivovat pacienta v další léčbě a zlepšování. Tyto výsledky jsou zároveň kontrolovány terapeuty. [2]



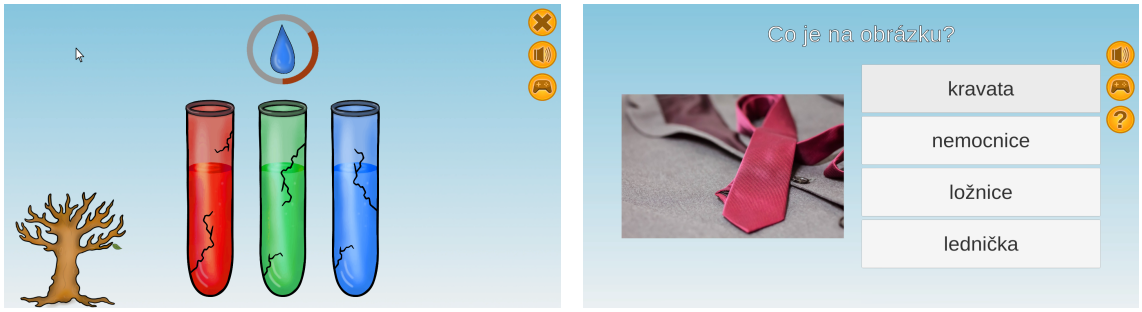
Obrázek 2.2: Ukázky úloh systému HappyNeuron

2.3 BrainIn

Další aplikace je BrainIn, která se začala vyvíjet v roce 2015. Při prvotním vývoji systému, ale nastaly problémy a vývoj skončil. Následně byl tento systém dovytvářen v rámci diplomové práce. [20]

Jedná se o webovou nadstavbu již existujícího řešení NP3[1]. Tento systém je od ostatních programů zdarma a poskytuje velkou dávku flexibility. Jelikož je systém rozdělen do několika sekcí, může terapeut velice jednoduše upravit jednu hru a dát jí hrát jinému pacientovi, ovšem s odlišnou vnitřní strukturou hry. Zároveň se systém snaží dávat pacientovi a terapeutovi zpětnou vazbu v podobě vytvořených informací. [20]

Tento systém aktuálně obsahuje několik her, které se ale stále rozvíjí a jejich počet se zvyšuje. Ukázka některých úloh je zobrazena na obrázku 2.3.



Obrázek 2.3: Ukázky úloh systému BrainIn

[1] Neurop 3

2.4 Zhodnocení existujících řešení

První dvě aplikace jsou samozřejmě mezi uživateli oblíbenější. Může za to hlavně fakt, že jsou na trhu již přes 10 let a jsou to celosvětové systémy, které přináší řešení této problematiky. Ovšem jejich nevýhodou je, že nemají dostatečný individuální přístup k pacientům a provádí vyhodnocování hodně autonomně. Zároveň se jedná o placený software.

Výhodou systému BrainIn je vysoká individualita k pacientům a také rozsáhlá flexibilita systému. Další výhodou je, že tento systém byl vytvořen z již existující desktopové aplikace Neurop3, která má za sebou několikaletou historii. Největší výhodou tohoto systému je rozhodně to, že je zcela zdarma a nabízí velikou rozmanitost, co se týče úrovní her.

Souhrn vlastností, ve kterých se systémy liší je zobrazen v tab. 2.1 na straně 12. Další kapitola se zaměřuje pouze na webovou aplikaci BrainIn, pro kterou budou vybrané hry realizovány a rozšíří tak aktuální soubor her.

Systém	Základní funkce	Výhody Premium účtu	Cena/rok
Lumosity	3 rychlé hry denně, ověření aktuálního stavu s použitím Fit Testu, data pro osobní zlepšení	Odemknutí všech ostatních her, detailních statistik a mnoho dalších odměn	60 $
HappyNeuron	7denní přístup, několik desítek her	Prodloužení platnosti licence	80 $
BrainIn	Neomezený počet hraní, zobrazování statistik (bez limitu)	Stejné jako základní funkce	Zdarma

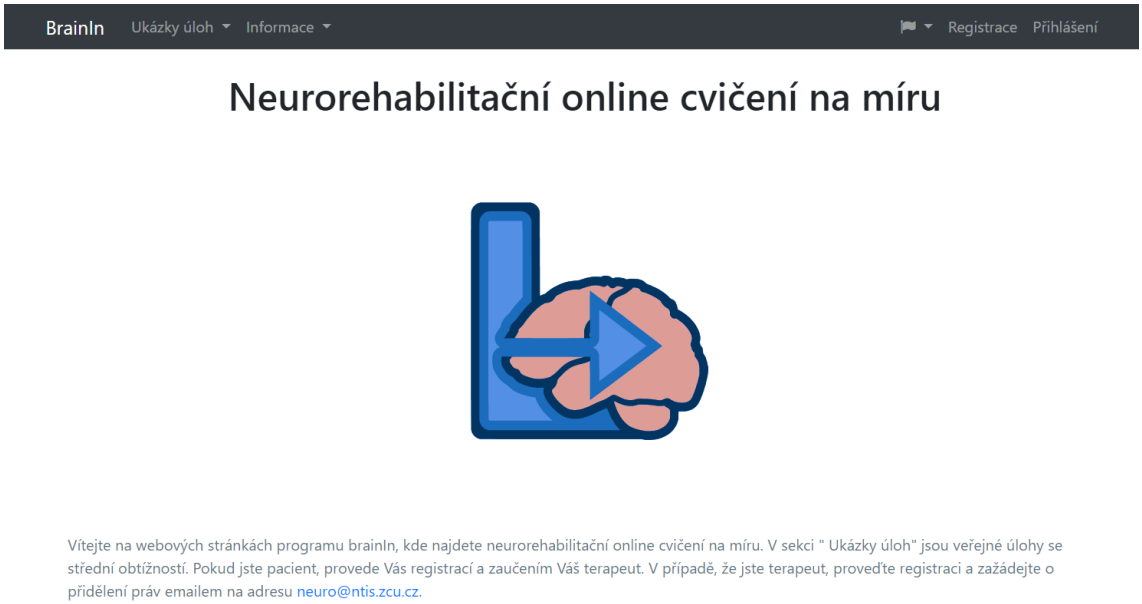
Tabulka 2.1: Porovnání systémů

3 Webová aplikace BrainIn

Jak jsem již zmínil v úvodu, nejdříve bylo nutné se seznámit se systémem BrainIn v neuroinformatické laboratoři na KIV[1]. Tento systém byl již dříve vytvořen studenty za účelem bakalářské či diplomové práce.

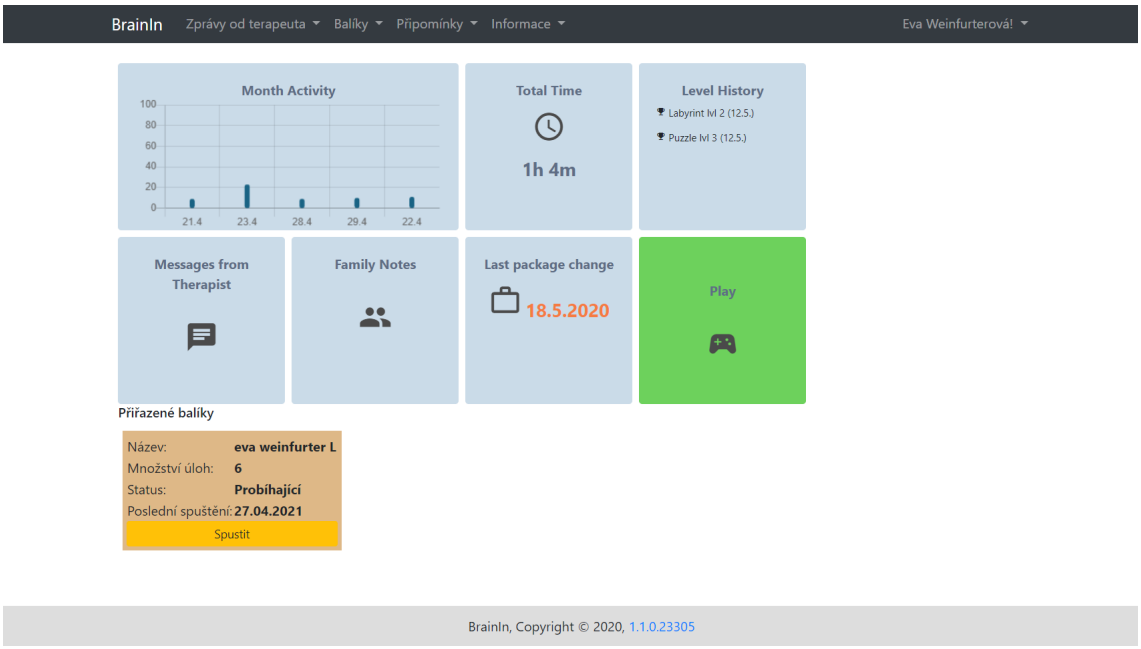
Systém BrainIn (viz obr. 3.1 a 3.2) slouží pro distribuci a správu různých her, které pomáhají rozvíjet a procvičovat převážně kognitivní funkce člověka. Tím, jak se stále vyvíjí svět počítačových technologií, se lidé často snaží o nahrazení her, které by se hrály např. na papíru anebo je k nim potřeba fyzického kontaktu. Systém BrainIn spravuje soubor her, které slouží právě k tomuto účelu. Díky novým technologiím jsou tyto hry převedeny do digitální formy a uloženy na jednom místě. [12]

Systém se zároveň snaží, aby hry byly parametrizovatelné. V zásadě to znamená, že se může vytvořit více instancí stejné hry, která bude mít stejný princip, ale obsah bude jiný. Umožní nám to upravit každou hru tak, aby se dala přizpůsobit pro každého pacienta zvlášť. Pokud bude mít pacient problémy s psaním levé ruky, vytvoříme mu instanci hry, kde bude psát levou rukou. Samozřejmě to samé můžeme udělat pro pacienta, který má naopak problémy s psaním pravé ruky. Tento efekt je dle mého názoru velice pozitivní, co se týče flexibility hry a umožňuje to více se věnovat konkrétnímu pacientovi a tím posílit účinnost rehabilitace.



Obrázek 3.1: Úvodní stránka systému BrainIn

[1]Katedra informatiky a výpočetní techniky



Obrázek 3.2: Systém BrainIn po přihlášení pacienta

3.1 Architektura systému

Celý systém je vytvořen v **ASP.NET**. Ve stručnosti se jedná o framework, který obsahuje řadu knihoven, které umožňují tvorbu webových aplikací v programovacím jazyce **C#**. Tento framework obsahuje řešení základních problémů a funkcí, které ve webové aplikaci vznikají (autentizace, práce s databází, ...). [21]

Další použitou technologií je **MS SQL Server 2016**. Jedná se o verzi databázového systému pro správu databáze od společnosti Microsoft. Jelikož se jedná o databázový server, jeho funkcí je ukládání a načítání informací obsažených v tabulkách. Tyto data následně poskytuje dalším softwarovým produktům, které chtějí s těmito daty pracovat. Celý tento systém je založen na jazyce **SQL**, který je jedním z nejvyužívanějších a nejpopulárnějších jazyků pro správu databází. [19]

Pro přístup k samotným datům se využívá **Entity Framework**, který usnadňuje práci s objekty nad databází, kde jsou data uložena. Jednoduše řečeno, tento framework dokáže odstínit vývojáře od sloupců v tabulce, protože mu stačí pracovat s těmito daty již jako s jednotlivými objekty. Vývojáři pak dokáží vytvářet aplikace s méně kódem oproti tradičnímu řešení. [15]

3.2 Základní funkce systému

BrainIn funguje na bázi šablon, úloh a balíčků. Zároveň obsahuje několik uživatelských rolí, kam se řadí administrátor, terapeut nebo pacient. Každý z uživatelů má samozřejmě jinou množinu aktivit, které v systému může provádět. Schéma systému je popsáno na obrázku 3.3.

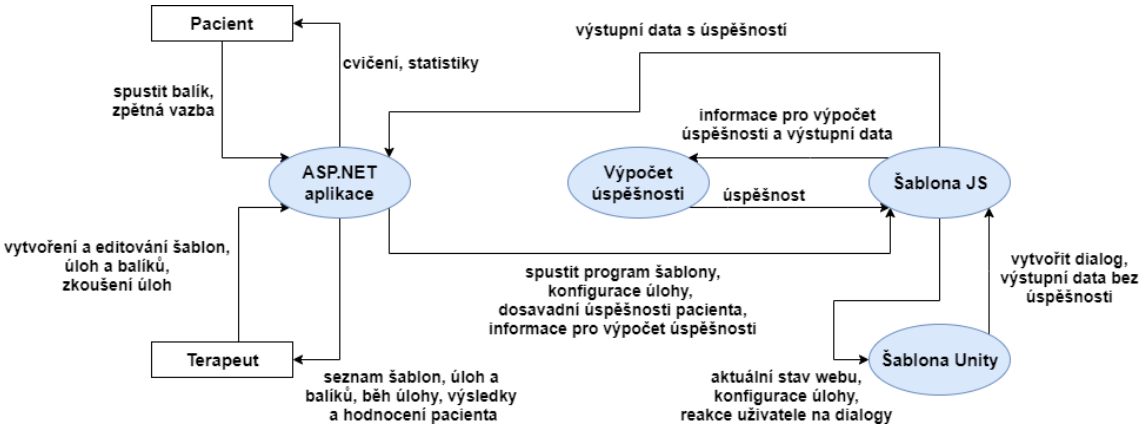
Šablony, úlohy a balíčky jsou datové typy systému. Každý z těchto typů má svůj jedinečný účel. Šablony se využívají při vytváření kompletně nové hry. To znamená,

že se zde nadefinují základní informace o šabloně, poté vstupní a výstupní parametry, a nakonec se do šablony nahrají soubory s již vytvořenou hrou.

Úlohy jsou instancí šablony. Udávají jednu konkrétní hru se zadanými parametry. Takto vytvořená instance hry se může jednoduše testovat, aby se předešlo případným chybám nebo nesrovnalostem ve vstupních parametrech. Nakonec je v systému vytvořen datový typ „balíček“. Balíček obsahuje různé množství úloh, které budou následně přiřazeny pacientovi, který bude úlohy hrát a tím procvičovat své kognitivní funkce.

Základní scénář je, že administrátor vytvoří hru, kterou následně přidá do systému a z této hry se stane šablona. Pak se z této šablony můžou vytvářet různé instance her s rozdílnými parametry. Z úloh se skládají balíčky, které jsou předány pacientům. Pacienti po přijmutí balíčku hrají a procvičují svoje kognitivní funkce. Hry v průběhu hraní sbírají definovanou množinu dat, které je možno vhodně využít k analýze výsledků. Informace, které se ze hry ukládají, může terapeut použít při plánování následné léčby a zároveň mu také předávají informaci, zdali se pacient v daném úkonu zlepšuje a rehabilitace je účinná.

Systém má tři jazykové verze. Proto je důležité překládat všechny texty, které se používají napříč celou aplikací. Webová aplikace má databázi překladů, které se mohou předávat i jako vstupní parametry. Zároveň se bere v potaz, že některé textové hodnoty mohou být i v každé instanci hry, kde se tento překlad musí také spravovat. Zvolený jazyk se předává spolu se vstupními parametry a instance hry se jím řídí a používá texty zvoleného jazyka.



Obrázek 3.3: Diagram systému

3.3 Vstupní a výstupní parametry

V této sekci popisuji a vysvětluji hlavní účel společných vstupních a výstupních parametrů. Vzhled parametrů je zobrazen na obr. 3.4, resp. obr. 3.5.

Parametrizace je stavebním kamenem celé aplikace, proto se na ni v této sekci zaměříme. Parametrizaci zde zaručují vstupní a výstupní parametry. Tyto hodnoty nám říkají, co všechno můžeme ve hře upravit, měnit a jaké výsledky budeme ve hře ukládat a následně analyzovat. Mezi tyto parametry patří například počet kol, který

může být pro každou hru jiný, nebo doba, jak dlouho bude zobrazena nápověda ve hře. Pro každou hru, ale mohou být tyto parametry různé a je třeba je rozdělit na společné a nespolečné. Vysvětlení konkrétních společných vstupních parametrů je uvedeno v sekci 3.3.1 a informace o výstupních parametrech jsou uvedeny v sekci 3.3.2 na straně 17. Parametry, které byly vytvořeny pro jednotlivé hry jsou popsány v kapitole 6.

Při vytváření šablony je potřeba nadefinovat vstupní a výstupní parametry. Obě tyto hodnoty mají podobnou strukturu - název, popis, datový typ. Výstupní parametry mají ještě navíc své označení. Nejdůležitější součástí parametrů je právě datový typ. Ten popisuje, jaké hodnoty budou předány z webové aplikace do hry. Z těch nejpoužívanější bych zde uvedl číslo, text nebo lokalizovaný text, ale existuje jich více.

Vstupní parametry v systému slouží pro parametrizaci různě nadefinovaných hodnot, které budou následně odeslány konkrétní hře, která je spuštěna. Instance hry se těmito hodnotami musí řídit, tj. musí upravit svoje chování tak, aby nebylo rozporuplné s parametry, které jsou označeny jako vstupní.

Výstupní parametry se dělí na celkové a dílčí. Rozdíl je v tom, že během hry se sbírají informace z jednotlivých kol a zároveň ze hry jako celku. Většina celkových výstupních parametrů se odesílá do systému v řetězci JSON[2], ale není to pravidlem.

Všechny hry mají společné vstupní a výstupní parametry. Jejich definice proto musí být uvedena u každé hry, aby se parametry mohly upravovat. Jedná se tedy o nedílnou součást každé šablony. Je více než vhodné, aby pořadí definovaných hodnot bylo pro každou hru stejné, což nám ulehčí následující implementaci a práci při získávání hodnot z webové aplikace.

3.3.1 Obecné vstupní parametry

Maximální doba

Hodnota udává, jak dlouho bude maximálně daná hra trvat. Pokud je pacient ještě ve hře, když uplyne tato doba, hra se ukončí a zbývající kola se přeskočí.

Počet kol

Parametr, který říká, z kolika kol se úloha skládá. Každá úloha musí mít alespoň jedno kolo.

Počet použití nápovědy

Parametr udává, kolikrát bude moci pacient použít nápovědu, pokud si nebude vědět rady, jak pokračovat. V některých úlohách může terapeut tuto možnost úplně zakázat.

[2] JavaScript Object Notation

Doba zobrazení řešení

Po dokončení či přeskočení kola se pacientovi ukáže správný výsledek. Může to být ten, ke kterému se dostal (úspěšné dokončení kola) nebo se jeho aktuální stav změní a zobrazí se správný výsledek (neúspěšné dokončení nebo přeskočení kola). Doba zobrazení řešení udává čas, jak dlouho bude výsledek viditelný před pokračováním na další kolo.

Doba zobrazení hodnocení

Časový úsek, který bude použit na konci celé úlohy, kde se pacientovi zobrazí jeho úspěšnost ve hře. Po tuto dobu bude informace o úspěšnosti viditelná a po uplynutí zmizí.

Trvání semaforu

Reálná hodnota, která řídí dobu trvání semaforu před každým kolem. Výchozí hodnota, po kterou běží semafor je přenásobena právě hodnotou vstupního parametru *Trvání semaforu* a můžeme tak docílit toho, že pacient nemusí dlouho čekat před každým kolem, což může působit rušivým dojmem.

3.3.2 Obecné výstupní parametry

Jelikož je výstupních parametrů více, uvedu zde pouze ty, které mi přišli důležité či často používané.

Celkový čas

Výstupní parametr, který nám říká, jak dlouho uživatel hru hrál.

Úspěšnost

Udává, jak moc úspěšný pacient byl. Jedná se o hodnotu, která je spočtena s použitím vzorce v definici šablony.

Čas spuštění

Hodnota informuje o čase, kdy byla daná úloha spuštěna. Hodí se, pokud chceme analyzovat, kdy je pro pacienta procvičování nejvhodnější.

Lokalizace

V jakém jazyce uživatel hru hrál.

Oblasti pro uživatelské rozhraní

Jedná se o více parametrů, které udávají oblast tlačítek pro ovládání nápovědy, přeskočení kola a ostatních komponent. Tyto hodnoty do webové aplikace přichází ve formátu JSON.

Počet kol

Počet kol, které uživatel dostal ve vstupních parametrech.

Čas kola

Společný dílčí parametr, který říká, jak dlouho pacientovi trvala jednotlivá kola.

Maximální doba	Maximální čas, který může pacient věnovat...	Číslo
Počet kol	Hodnota udává počet kol, která se měla o...	Číslo
Počet použití nápovědy	Počet použití nápovědy pro vyřešení dané ...	Číslo
Doba zobrazení nápovědy	Na jak dlouho bude zobrazena nápověda.	Číslo
Doba zobrazení odpovědi	Na jak dlouho bude zobrazena oprava a ře...	Číslo
Doba zobrazení hodnocení	Na kolik sekund bude zobrazeno celkové h...	Číslo
Trvání semaforu	Reálné číslo, kterým se násobí základní do...	Číslo
Body ke spojení	Udává body a zároveň cestu kudu musí pa...	Text
Počet po sobě následujících b...	Udává, kolik bude vidět najednou bodů. M...	Text
Doba zobrazení cesty	Na jak dlouho bude na začátku zobrazena ...	Číslo

Obrázek 3.4: Ukázka vstupních parametrů

Celkový čas	Celkový čas běhu úlohy v sekundách	totalTime	Číslo
Úspěšnost	Procentuálně vyjádřená úspěšnost úlohy	success	Číslo
Čas spuštění	Kdy byla úloha spuštěna v ISO formátu (nu...	startTime	Text
Lokalizace	V jakém jazyce uživatel úlohu řešil.	locale	Text
Oblast pro vzorovou matici	Informace o pozici a viditelnosti daného pr...	matrixArea	JSON
Oblast pro efekt kliknutí	Informace o pozici a viditelnosti daného pr...	hintArea	JSON
Tlačítko Vypnout/zapnout zvuky	Informace o pozici a viditelnosti daného pr...	soundsButton	JSON
Tlačítko Ovládání hry	Informace o pozici a viditelnosti daného pr...	controlsButton	JSON
Tlačítko Nápověda	Informace o pozici a viditelnosti daného pr...	helpButton	JSON
Tlačítko Přeskočit/Pokračovat	Informace o pozici a viditelnosti daného pr...	skipOrContinueButton	JSON
Počet kol	Neuvedeno	numOfRounds	Číslo

Definice dílčích výsledků z úlohy

Čas kola	Jak dlouho trvalo dané kolo v úloze.	roundTime	Číslo
Spuštění kola	Kdy bylo kolo spuštěno relativně k času sp...	roundStartTime	Číslo
Herní čas kola	Jak dlouho trvalo dané kolo v úloze se vší...	roundRealTime	Číslo

Obrázek 3.5: Ukázka výstupních parametrů

4 Mozek a jeho aktivita

Než se začneme zabývat neurorehabilitací mozku. Pozastavíme se nad tím, jak mozek funguje a co napomáhá šíření signálů, které do mozku nebo zpět putují.

Mozek je patrně ten nejsložitější orgán v každém těle. Tento orgán je součástí centrální nervové soustavy a řídí v podstatě celé tělo. Mozek je pro člověka něco jako pro počítač procesor. Aby byl mozek schopen posílat zprávy do různých částí těla, je potřeba využít nervového systému.

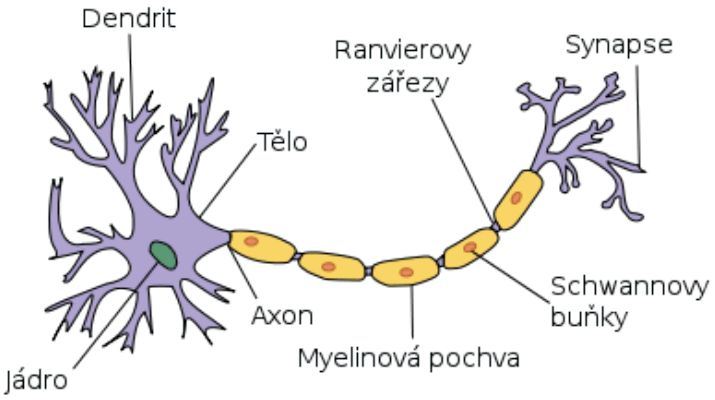
4.1 Neuron

Stavební složkou centrální nervové soustavy je buňka, která se nazývá neuron. Neurony jsou schopné přijímat, odesílat či zpracovávat speciální signály. Těmito signály řídí činnost člověka na základě vnitřních i vnějších podnětů. Všechny neurony jsou mezi sebou propojeny s využitím synapsí. Přes synapse spolu neurony dokáží komunikovat prostřednictvím elektrických signálů. [9]

4.1.1 Tělo neuronu

Každý neuron má tři hlavní části, které se od sebe dají snadno rozlišit (viz obr. 4.1):

- Buněčné tělo - někdy se mu také říká soma nebo perikaryon. Je jádrem všech neuronů v nervové soustavě. Tělo neuronu je uzavřeno membránou, která ho chrání a napomáhá mu komunikovat s okolím. [17]
- Axon - jedná se o výběžek, který předává vzruchy mezi jednotlivými neurony či neuronem a jinou buňkou v soustavě. K vedení elektrického signálu jim napomáhá myelin. [17]
- Dendrity - neboli vláknité kořeny, které vedou z buněčného těla. Jejich funkcí je přijímat a zpracovat signál, který přijde z axonů v okolí. Jednodušeji řečeno, dendrity fungují jako rádiová anténa. Dendrity mohou být v některých případech velice rozvinuté a vytvářet tzv. dendritický strom. Takový strom dokáže přijímat až tisíce signálů. [17]



Obrázek 4.1: Schéma neuronu [14]

4.1.2 Typy neuronů

Všechny neurony v těle se od sebe liší, ať už strukturou nebo funkcí. Je to podobné jako s otisky prstů, které má každý člověk jedinečné. Neurony můžeme rozdělit z hlediska jejich funkce do tří následujících skupin: [17]

- smyslové
- motorické
- interneurony

U prvních dvou již můžeme poznat, jakou funkci zajišťují, ovšem i v tomto případě si je ve stručnosti popíšeme.

Smyslové neurony se starají o základní funkci smyslů. Díky nim můžeme cítit chuť jídla, slyšet nebo vidět. Motorické neurony ovládají svaly v těle a pomáhají nám se hýbat. Jako pohyb nemusíme myslet pouze chůzi či běh, ale například dýchaní či mrkání. Interneurony předávají informace dál a stávají se tak prostředníky. Samozřejmě mozek je velice komplexní a složitý orgán v našem těle. Proto o něm, i v dnešní době, nevíme mnoho informací a stále se nové věci zkoumají.

Tento dokument se bude zaměřovat převážně na motorické neurony, které ovlivňují kognitivní funkce člověka.

4.2 Kognitivní funkce

Kognitivní funkce jsou základní funkce našeho těla a mozku. Díky nim dokážeme vnímat a reagovat na venkovní prostředí či myslet. Kognitivní funkce nám napomáhají žít plnohodnotný život a aktivně se zapojovat do dění všedního světa. Mezi tyto schopnosti se řadí paměť, pozornost, zrakově-prostorové schopnosti nebo myšlení. [11]

Tyto funkce mohou ztrácet své schopnosti z různých důvodů. Mezi ně patří nejčastěji stáří, vážný fyzický úraz nebo vrozená nemoc. S tím, jak se kognitivní funkce zhoršují, dochází k nejistotě a člověk může ztrácet sebevědomí a následně je denní fungování zkomplikované a v některých případech dokonce nemožné. [11]

4.2.1 Motorika

Motorika je jednou ze základních kognitivních funkcí, která nám pomáhá se hýbat a ovládat svalstvo. Nejúčinnější pohyb vyžaduje schopnost „cítit“, co jeho svaly dělají při provádění úkonu. Motorické poruchy nastávají, pokud člověk nedokáže úkol, který chce provést, vykonat. Tyto poruchy můžeme nejčastěji pozorovat u dětí ve školním či předškolním věku. Pokud se jedná o dítě, může to způsobit ztrátu sebevědomí či motivace. [4]

Mezi základní problémy způsobené motorikou patří:

- problémy s psaním nebo malováním
- neschopnost opisovat z tabule
- padání ze židle
- nepřirozené sezení nebo stání
- a další. [4]

Motorické poruchy lze řešit mnoha způsoby. Některé potíže lze vyřešit zralostí a vývojem člověka. Někdy se dítě odkáže na speciální vzdělávací služby. Ovšem v mnoha případech stačí, naučit se určitou aktivitu, která pomáhá eliminovat potíže s motorikou. [4]

Motorika se dále může rozdělit na jemnou a hrubou. Obě tyto skupiny se trénují společně, jelikož u většiny akcí, které vykonáváme, je důležitá koordinace jemné a hrubé motoriky.

Hrubá motorika

Hrubá motorika zahrnuje pohyby, které jsou velmi komplexní. Je k nim potřeba využívat větší množství svalových skupin a jsou zároveň více fyzicky náročné. Mezi tyto pohyby patří běhání, chůze, lezení či skákání. [13]

Stářím dítěte by se tyto funkce mělo tělo naučit a v každém věku je zlepšovat. Samozřejmě se nedá očekávat, že půlroční dítě bude umět běhat. Ovšem jsou věkové milníky, kdy by dítě mělo danou činnost ovládat. Základní příklady takovýchto milníků jsou: [13]

- 1 rok
  - chůze s držením ruky
  - chůze do schodů s pomocí
- 2 roky
  - skákání snožmo (z obou nohou najednou)
  - běhání bez padání
  - házení míčku

Jemná motorika

Jemná motorika se popisuje jako skupina pohybů, které jsou svým způsobem malé. Jedná se o pohyby rukou, zápěstí, prstů nebo i jazyka. [1]

Úkoly, které zajišťuje jemná motorikou jsou:

- psaní
- kreslení
- uchopování věcí
- a mnoho dalších. [1]

Jemná motorika by se měla procvičovat a rozvíjet již od raného dětství spolu s dalšími kognitivními funkcemi, které jsou důležité pro život člověka. Tyto vlastnosti by si měl každý osvojit, než nastoupí na základní školu. Interval od narození po nástup do základní školy je ta nejvhodnější doba, kdy trénovat nejen jemnou motoriku, ale všechny ostatní schopnosti člověka. [18]

Pokud jsou děti verbálně zralé, ale mají problémy s jemnou motorikou, může docházet k tomu, že budou mít potíže s předváděním svých znalostí na papíře (tj. psaním nebo kreslením). Zlepšování jejich schopností a vytrvalost v jemné motorice, zvyšuje školní připravenost a akademický výkon v kreslení nebo psaní. [18]

5 Poškození mozku a neurorehabilitace

V předešlé kapitole 4 na straně 19 jsem osvětlil funkčnost mozku a související pojmy, na které se v této kapitole snažím navázat. Zde uvedu základní pojmy, které souvisí s tématem bakalářské práce. Mezi ně patří poškození mozku, plasticita mozku a neurorehabilitace.

5.1 Poškození mozku

Poškození mozku vzniká v nejčastějších případech v důsledku mozkové mrtvice či úrazu. Tento úraz může být způsoben při sportovní činnosti nebo autonehodě. Další příčinou mohou být tzv. neuroinfekce. Ty jsou nejčastěji vyvolávány viry. Mezi neuroinfekce můžeme zařadit například klíšťovou encefalitidu, boreliózu nebo pneumokokovou meningitidu. Dalším typickým projevem je narušení emocionality, což způsobí kolísavé emoce, deprese či amnézie. [8]

Poškození mozku znamená velkou změnu v dosavadním životě člověka a má velký dopad především v oblasti kognitivních funkcí. Takové poškození nemá dopad jen na člověka, které ho takové zranění zasáhlo, ale ovlivňuje i jeho okolí (rodinu, přátele a mnoho dalších). [8]

5.2 Plasticita mozku

Plasticita mozku neboli neuroplasticita je pojem, který popisuje vlastnost mozku měnit a přizpůsobovat se na základě získaných zkušeností.

Až do roku 1960 se vědci domnívali, že se tento proces vyskytuje pouze v raném dětství a věřilo se, že v dospělosti je činnost mozku stálá a neměnná. Ovšem moderní výzkumy prokázali, že mozek i v dospělosti, stále vytváří nová nervová spojení a upravuje chování stávajících. Touto činností se přizpůsobuje nově získaným zkušenostem. [6]

Jak již bylo zmíněno, v některých případech je možnost, že se funkce mozku obnoví. Někdy k tomu dochází samovolně, v jiných případech je nahrazena již ztracená funkce mozku. Dokonce mohou být vytvářena i nová nervová spojení prostřednictvím plasticity mozku. Ačkoliv je proces obnovy spojení dlouhodobý (v řádech měsíců či let), dá se aktivním zapojením pacienta, většina spojení obnovit a navrátit jim jejich původní podobu a schopnost pracovat. [7]

5.2.1 Jak plasticita mozku funguje?

První roky života dítěte lze pozorovat rychlý nárůst mozku a jeho aktivity. Každý neuron má v tomto období zhruba 2 500 synapsí. Ve věku tří let toto číslo vzroste na 15 000 synapsí jednoho neuronu, ačkoliv má průměrný dospělý člověk poloviční počet synapsí. Jak člověk získává nové zkušenosti, některá nervová spojení jsou posílena, zatímco jiná jsou naopak vyloučena. Celý tento proces získávání a vylučování synapsí se nazývá synaptické prořezávání. [6]

5.3 Neurorehabilitace

Obor neurorehabilitace je velice komplexní, složitý a dlouhodobý proces. Jedná se o přístup k pacientům s neurologickým postižením mozku. Neurorehabilitace se zaměřuje na poruchy pozornosti, soustředění, zlepšování paměti nebo porozumění a řešení kognitivních potíží. [16]

5.3.1 Proces neurorehabilitace

Samotná rehabilitace začíná již v nemocnici, většinou na JIP[1]. S neurorehabilitací je nutné začít co nejdříve, jelikož obnova funkcí je nejrychlejší v prvních dnech po vzniku příčiny poškození mozku. Proces rehabilitace je veden rehabilitačním týmem, který tvoří psychologové, fyzioterapeuti nebo všeobecné sestry.

Po opuštění nemocnice dochází ke snížení frekvence rehabilitací, což většinou vede ke zpětnému nástupu poškození mozku. Možným řešením je právě v této fázi využít neurorehabilitační hry, které nevyžadují dohled rehabilitačního týmu a pacient může rehabilitovat z pohodlí domova. Ke správné efektivitě je potřeba, aby pacient měl motivaci a aktivní přístup, jinak hry nebudou účinné podle předpokladů.

I když hry vedou ke zvýšení motivace a přístupu pacienta k rehabilitaci, měly by hry splňovat určitá kritéria, aby se daly použít právě pro účely neurorehabilitace po úrazu mozku. Úlohy, které pacient bude plnit v rámci rehabilitace, by měly být zaměřeny převážně na schopnosti pacienta a aby soustředil svoji pozornost na úspěšné zvládnutí cíle hry, než aby vnímal, že provádí rehabilitační cvičení. Další z kritérií je zaměřit se na tyto hry i z pohledu terapeuta, který potřebuje dostatečně velkou skupinu parametrů, podle kterých bude moci vyhodnotit aktuální vývoj a zlepšování pacienta v dané úloze. [10]

[1] Jednotka intenzivní péče

6 Analýza a teoretická funkcionalita her

Zadáním bakalářské práce je vytvořit 4 parametrizované hry, které budou sloužit pro rozvoj jemné motoriky. Jako startovní bod jsem obdržel soubor her, které navrhla paní Šroglová z FN[1] Lochotín. Tyto hry se zaměřují na jemnou motoriku, kterou se rovněž zabývá. Pro každou hru bylo třeba navrhnout vstupní a výstupní parametry a také rozmístění prvků uživatelského rozhraní. Nakonec přišla na řadu implementace jednotlivých instancí her a jejich testování.

Po výběru 4 her, jsem se zaměřil na vstupní a výstupní parametry u každé hry. Bylo třeba promyslet celkovou funkcionalitu hry a rozmyslet, které hodnoty bude potřeba posílat do hry přes webovou aplikaci, aby se hra dala, co nejvíce parametrizovat. Přitom bylo nutné vybrat vhodné výstupní parametry, aby po odehrání hry bylo možné řádně analyzovat progres v rehabilitaci.

U každé hry uvedu i základní teoretický scénář, který by měl znázorňovat princip hry a nastínit posloupnost úkonů, které jsou nutné pro splnění celé úlohy.

6.1 Spojení bodů

První hrou, kterou jsem si z výše zmíněného souboru vybral, bylo Spojování bodů. Princip spočívá v tom, že se pacient pohybuje s použitím prstu či myši po obrazovce. Snaží se spojovat body, které se mu postupně zobrazují a to tak, že tah musí být co nejpodobnější rovné čáře. Prvotní myšlenka byla spojovat rohy obrazovky např. z levého dolního rohu do pravého horního. Při vývoji jsem se snažil, aby zadané cesty obsahovaly více bodů, což dodává možnost vytvořit obtížnější úrovně hry.

6.1.1 Základní herní scénář

Uživatel spustí úlohu. Zobrazí se mu klasický semafor, který odpočítá začátek prvního kola. Následně se pacientovi zobrazí počáteční bod, ze kterého zahájí kreslení pomocí pohybu prstu či myši po obrazovce. Poté co zahájí malování čáry, se mu zobrazí další bod v pořadí a bude se snažit malovat, co nejrovnější čáru dokáže, aby se dostal do následujícího bodu. Tento proces se opakuje, dokud nenarazí na koncový bod, kde ukončí kreslení. Pokud vyjede z koncového bodu a až poté přestane kreslit, bude se tento tah klasifikovat jako chybný a pacient bude muset začít od znovu.

[1] Fakultní nemocnice v Plzni

6.1.2 Vstupní parametry

Kromě společných vstupních parametrů bylo potřeba navrhnout takové parametry, které by dodaly dostatečnou flexibilitu a potřebnou funkcionalitu dané hře.

Body ke spojení

Jedná se o řetězec identifikátorů, které udávají jednotlivé body ve hře. Posloupnost, ve které jsou identifikátory zapsány, bude použita pro vyhodnocení sekvence po sobě jdoucích bodů. To znamená, že body se ve hře seřadí podle toho, jak jsou seřazeny ve vstupním parametru.

Doba zobrazení nápovědy

Tento parametr se použije u využití nápovědy. Udává, jak dlouho bude trvat zobrazení bodů, které bude muset pacient spojit.

6.1.3 Výstupní parametry

Zadaná cesta

Řetězec, který byl zadán jako vstupní parametr, aby mohl terapeut lépe zkontrolovat preciznost ostatních výstupních parametrů.

Výsledný obrázek

Jedná se o obrázek, který bude ukazovat, jak dobře pacient nakreslil dané spojení všech bodů.

Počet pokusů

Počet pokusů, které pacient potřeboval, než spojil všechny body v cestě.

Jednotlivé pokusy

Řetězec všech pokusů, které uživatel provedl.

Časové intervaly pokusů

Zde budou uvedeny časy mezi spojením jednotlivých bodů v každém pokusu, který uživatel provedl.

6.2 Domaluj polovinu

Druhá hra se věnuje malování. Princip hry je v tom, že se pacientovi zobrazí obrázek na jedné straně obrazovky. Pacient má za úkol na protější straně obrazovky domalovat druhou polovinu obrázku, aby byla jeho kresba symetrická s předaným obrázkem.

6.2.1 Základní herní scénář

Uživatel spustí hru a po odpočítání začátku kola, se zobrazí na určité straně obrazovky obrázek, který bude pacient dokreslovat. Po uplynutí počátečního zobrazení obrázku, začne pacient kreslit druhou polovinu obrázku na opačnou stranu obrazovky. Pokud si uživatel není jistý, jak obrázek vypadá, má možnost použít nápovědu, která mu zobrazí daný obrázek. Když uživatel dokončí kreslení a myslí si, že je s kreslením hotov, stiskne tlačítko pro dokončení kreslení a přesune se na další kolo.

6.2.2 Vstupní parametry

Obrázky

Prvním herním parametrem do hry je složka, kde budou uloženy obrázky, které budou použity v jednotlivých kolech. Pokud zde bude obrázků méně, použije se vždy ten abecedně poslední.

Strana obrázku

Řetězec udává, kde bude v daném kole umístěn obrázek, jestli na levé či na pravé straně obrazovky. Tímto docílíme větší flexibility při zadávání úlohy např. levákům.

Doba zobrazení obrázku

Časová hodnota určuje dobu zobrazení obrázku na začátku kola. Po uplynutí obrázek zmizí a pacient si ho může znovu zobrazit pomocí nápovědy. Pokud bude terapeut chtít, aby obrázek zůstal po celou dobu hraní viditelný, stačí nastavit hodnotu na 0. Tímto způsobem, ale zaniká možnost použití nápovědy.

6.2.3 Výstupní parametry

Výsledný obrázek

Hlavním výstupním parametrem bude obrázek, který bude ukazovat, jak dobře pacient domaloval obrázek, který mu byl zadán.

Doba kreslení

Zde budou jednotlivé časy, jak dlouho trvalo nakreslení druhé poloviny obrázku, než pacient stiskl tlačítko pro dokončení kreslení.

Počet přerušení

Pouze hodnota, která říká, kolikrát uživatel přerušil malování. V některých případech by tato informace nemusela být z obrázku viditelná.

6.3 Spojování dvojic

Základní funkcionalita hry je spojovat slovo s příslušným obrázkem. Například obrázek lesa se slovem les. V požadavcích na tuto hru bylo, aby uživatel spojoval slovo s obrázkem z horní, resp. dolní části obrazovky.

6.3.1 Základní herní scénář

Uživatel zapne úlohu, která mu zobrazí odpočítávání začátku prvního kola prostřednictvím semaforu. Po dokončení odpočítávání se ukáže určitý počet obrázků a adekvátní počet slov. Poté pacient začne postupně spojovat slova s obrázky nebo naopak. Poté co spojí všechna slova a obrázky, se kolo ukončí a započne odpočítávání dalšího kola. Pokud si uživatel není jistý, který obrázek patří k danému slovu, může použít nápovědu, která za něj doplní jeden pár nebo jej pouze zvýrazní.

6.3.2 Vstupní parametry

Obrázky

Jedním ze vstupních parametrů bude složka s obrázky, které budou používány pro všechna kola. Jako slova, která bude pacient spojovat, se použijí názvy obrázků.

6.3.3 Výstupní parametry

Výsledný obrázek

Obrázek, který ukazuje jednotlivé spojení obrázků a slov.

Jednotlivá spojení

Informace v jednotlivých kolech o tom, jestli je obrázek se slovem spojen správně či naopak.

Doba spojování

Doba, jak dlouho pacient spojoval jednotlivé obrázky se slovy.

6.4 Malování dvěma rukama

Hra funguje tak, že uživatel dostane slovo, které představuje obecně známou věc (např. strom). Pacient má za úkol tento objekt nakreslit oběma rukama zároveň na odlišných stranách obrazovky. Snaží se, aby objekty, které nakreslí, byly stejné nebo hodně podobné.

6.4.1 Základní herní scénář

Pacient zahájí hraní úlohy. Na počátku hry se zobrazí semafor, který spustí odpočet prvního kola. Po dokončení odpočtu se ukáže slovo, které reprezentuje určitý objekt, který bude uživatel kreslit. Poté co si uživatel rozmyslí, jak bude objekt kreslit, dá dva prsty na obrazovku a oběma začne kreslit dva, co nejpodobnější, objekty. Když si je jistý, že jsou jeho objekty hotové, přesune se do dalšího kola stisknutím tlačítka a tím začne odpočítávání následujícího kola.

6.4.2 Vstupní parametry

Slova objektů

Řetězec slov, které bude uživatel kreslit. Jednotlivá slova pro konkrétní kola jsou rozdělena s použitím znaku středníku.

Rozdělení polovin

Logická hodnota, která se použije, pokud bude terapeut chtít vynutit malování na obou stranách obrazovky. V případě, že nebude nastaven, pacient bude moci kreslit po celé ploše bez omezení.

6.4.3 Výstupní parametry

Výsledný obrázek

Obrázek, který se vytvoří poté, co uživatel dokončí malování. Popisuje výsledek jeho práce.

Doba kreslení

Čas, který říká, jak dlouho jednotlivý objekt pacient maloval.

7 Využívané technologie

Tato kapitola se zaměřuje na technologie, které jsem při vývoji jednotlivých her použil a popisuji zde, jejich základní účel a použití.

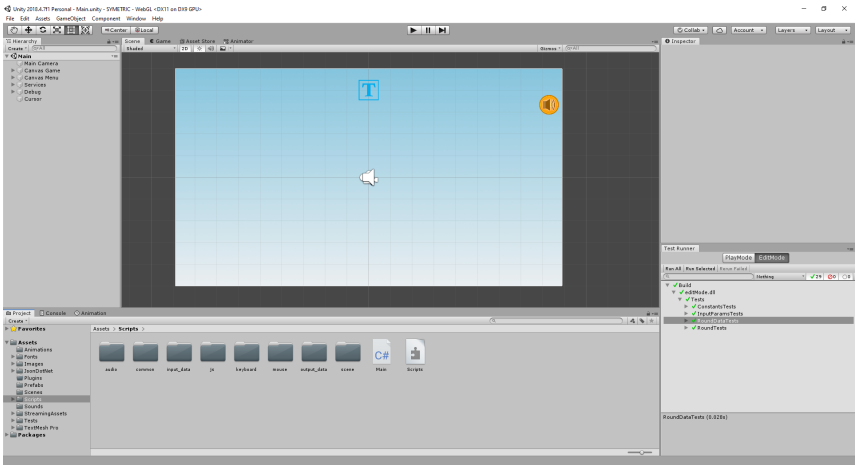
Jelikož systém je již v pokročilé fázi vývoje, už jsou zde zaběhnuté určité standardy a technologie, které se používají a zároveň je jejich využití opodstatněné. Ať už se jedná o herní engine nebo technologie používající se pro navázání spojení mezi instancí hry a webovou aplikací.

Pro vývoj her se používají primárně dvě technologie, pomocí kterých lze vytvořit jednotlivé úlohy, kontrolovat jejich průběh a zároveň zařídit, aby dokázaly komunikovat s webovou aplikací BrainIn.

7.1 Technologie pro vývoj her

Pro vývoj jednotlivých her je použit vývojový nástroj zvaný **Unity**, jehož grafické prostředí je ukázáno na obr. 7.1. V našem případě se bude jednatý konkrétně o **Unity 2018.4.7f1 (64bitová verze)**. Tato verze je použitá na základě doporučení vývojáře M. Horkého, který podobné hry vytváří. Tudíž se zamezí případným problémům, které by mohly vzniknout použitím rozdílných verzí.

Unity je herní engine, který slouží pro vývoj 2D, ale i 3D her. Usnadňuje vývoj her, jelikož dokáže ovládat fyziku herních objektů nebo jejich animace. Hry vytvořené v tomto enginu se dále dají využívat v různých zařízeních. Lze zde definovat prvky herní scény, tlačítka a další objekty, které mohou, ale i nemusí interagovat s uživatelem.



Obrázek 7.1: Grafické prostředí herního enginu Unity

7.1.1 Skripty

Běh herních objektů se zde řídí s využitím skriptů (viz algo. 7.1), které jsou psány v jazyce **C#**. Při vytváření těchto skriptů se definují tři základní funkce, které se mohou použít pro určení životního cyklu herního objektu. Tyto funkce jsou **Awake, Start** a **Update**.

```
1    void Awake() { }
2
3    void Start() { }
4
5    void Update() { }
```

Algoritmus 7.1: Funkce herních objektů

Výše popsané funkce definují akce, které se provedou v určité fázi hry. Metoda s názvem **Awake** se proveden v momentě, kdy je herní objekt načten společně s herní scénou nebo pokud je objekt převede z neaktivního stavu do aktivního.

Druhou důležitou funkcí je **Start**, která se zavolá až poté, co se daný skript povolí. Zde se vyskytuje většinou část kódu, která je závislá na načtení ostatních objektů, které nejdříve musejí zavolat funkci **Awake**. Tato funkce se zavolá pouze jednou v životním cyklu objektu, a to ve snímku předtím, než je vyvolána metoda **Update**.

Poslední funkce už je volána opakovaně, dokud se daný objekt nezničí nebo nedojde k ukončení hry. Slouží pro aktualizaci herního objektu v závislosti na čase. Nejčastěji se v praxi může jednat o změnu pozice figurky na základě stisknutých kláves a rychlosti pohybu hráčovi postavy.

7.2 Technologie pro komunikaci se systémem BrainIn

Pro komunikaci mezi hrou a serverem je použito objektově orientovaného programovacího jazyka **JavaScriptu**. Využití tohoto jazyka je ve skutečnosti logické, jelikož se v posledních několika letech hojně využívá a zájem o něj roste.

Pro komunikaci se tedy používají již vytvořené skripty, které mají různou funkci. Mezi tyto funkce patří například předávání vstupních a výstupních parametrů mezi systémem a instancí hry vytvořenou v **Unity** nebo ukládání výstupních obrázků, které jsou zachyceny v průběhu hry.

Veškerá logika je rozdělena do několika dynamický modulů, které se načítají za běhu programu. Implementace těchto skriptů se ve většině případů měnit nebude.

8 Obecný postup vývoje her pro systém BrainIn

Systém, pro který jsou tyto hry vyvíjeny, již obsahuje některé materiály a postupy, které jsou kostrou všem úlohám a ze kterých se každá hra vytváří, aby se dodržovalo sjednocení vytvářených úloh a usnadnila se tak práce ostatním, kteří by v případě nutnosti museli zasahovat do implementace her. Protože všechny hry mají stejnou strukturu, neměl by jiný vývojář mít velké problémy se v konkrétní implementaci hry vyznat.

Samozřejmě pro vytvoření jednotlivých úloh není třeba pouze úlohu naimplementovat v herním enginu, ale je potřeba i vytvořit patřičné šablony, překlady a úlohy ve webové aplikaci, aby se s danou instancí hry dalo správně pracovat.

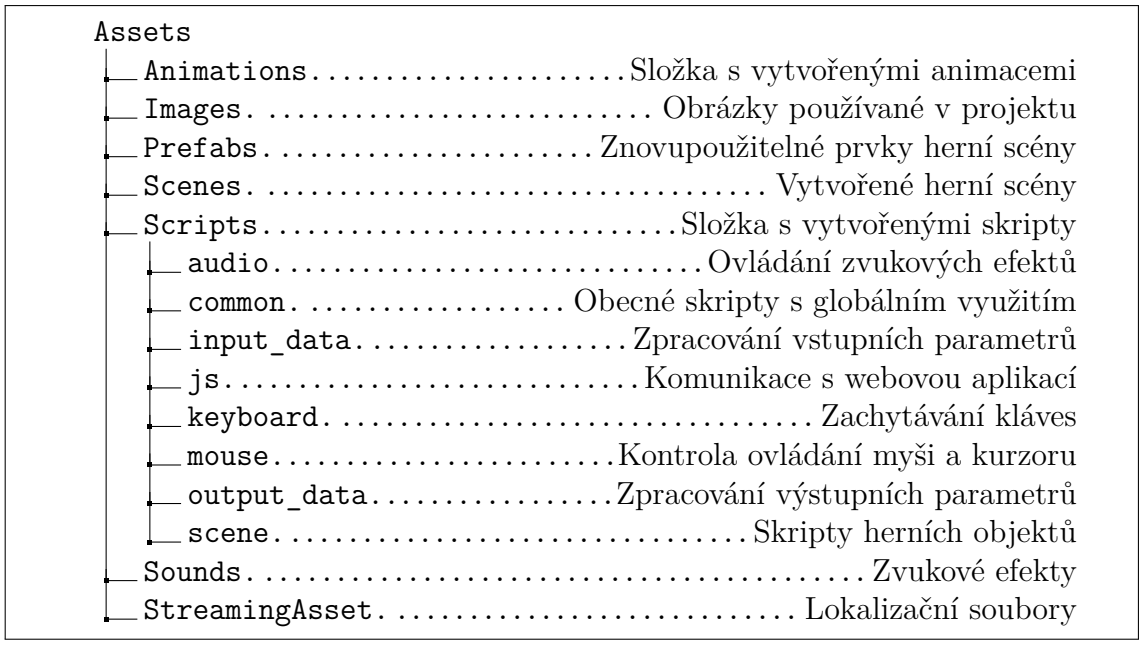
8.1 Poskytnuté materiály

Pro vývoj byla poskytnuta šablona, která je stejná pro všechny ostatní hry v systému. Slouží jako předloha pro vytváření úloh a zároveň usnadnění práce, jelikož se nemusí všechny stejné funkce pro sbírání informací a správu běhu programu stále opisovat.

Spolu s touto šablonou byly poskytnuté i skripty pro komunikaci s webovou aplikací. Do těchto skriptů bylo třeba přidat metodu, která bude zpracovávat ukládání výstupních obrázků.

8.1.1 Struktura projektu

Projekt je rozdělen do několika částí. Stromová struktura projektu je popsána níže na obrázku 8.1. Tato stromová struktura adresáře popisuje účel vytvořených adresářů a jejich obsah. Popis začíná ve složce **Assets**, kde se vyskytují všechny potřebné soubory, se kterými program pracuje a prostřednictvím kterých je definována herní scéna a logika jednotlivých objektů. Jedná se o zjednodušenou verzi adresáře, kde jsem si dovolil vynechat málo významné složky.



Obrázek 8.1: Struktura adresáře Assets

8.1.2 Význam a funkce důležitých skriptů

V této sekci se pokusím popsat funkci některých skriptů, se kterými se při vývoji nejvíce pracovalo.

Main.cs

Hlavní skript, který řídí spouštění jednotlivých kol, zobrazování výsledků a jejich ukládání. Zároveň se stará o zobrazování nápovědy a odchytávání požadavků na přeskakování kol.

Poté co se načtou vstupní parametry, začne spuštění hry, které se nachází v metodě **Update**. Tato metoda dbá na správné zahájení kol a jejich přepínání v průběhu hry.

Běh hry může být rozdělen do několika částí - zobrazení semaforu, spuštění kola, hraní hry, zobrazení řešení a vyhodnocení. Tento proces se opakuje tak dlouho, dokud uživatel neprojde všechna kola nebo nedojde k vyčerpání času na hraní úlohy.

Na počátku každého kola se pacientovi zobrazí odpočítávání kola v podobě semaforu. Následně se spustí kolo, ve kterém se uživatel snaží docílit nejlepšího možného výsledku. Když uživatel dokončí kolo, skript **Main** vyhodnotí výsledky, které uloží do výstupních parametrů a zahájí další kolo.

Dále se v tomto skriptu nachází reakce na stisk tlačítek zobrazených v herní scéně hry (nápovědy, přeskočení kola a další).

InputParams.cs

Tento skript se stará o rozdělení řetězce, který obsahuje hodnoty vstupních parametrů. Zároveň kontroluje, jestli je jejich obsah zadaný správně. Pokud všechno

proběhne v pořádku, vytvoří se objekty třídy **RoundData**, které obsahují informace pro dané kolo, díky kterým se vytváří obsah jednotlivých kol.

RoundData.cs

Tato třída slouží pro uchování a organizaci dat pro jedno kolo. Může se jednat o jakékoliv potřebné údaje. Některé informace se ovšem uchovávají i ve třídě **InputParams** ve statické podobě.

UnityToWeb.cs

Skript, který reprezentuje rozhraní mezi hrou a webovou aplikací. Obsahuje funkce, které se využívají pro odesílání dat do webové aplikace, která je poté dále zpracovává. Příkladem takové metody může být odeslání výstupního obrázku ve formátu **Base64**, což je ukázáno v algoritmu 8.1.

```
1    public static void MakeScreenShot(string base64string) {
2        Application.ExternalCall(Constants.WEB_FUNC_TAKE_SCREENSHOT
    , base64string);
3    }
```

Algoritmus 8.1: Odeslání obrázku ve formátu Base64

WebToUnity.cs

Soubor, který naopak přijímá data ze systému. Reaguje na příchozí zprávy a následně provádí adekvátní akce, které jsou třeba. Jako příklad lze uvést přijmutí zprávy s cestou k vytvořenému výstupnímu obrázku (viz algo. 8.2).

```
1    public void GetScreenShotPath(string path) {
2        Main.Results.SetImagePath(path);
3    }
```

Algoritmus 8.2: Přijmutí cesty k uloženému obrázku

DataCollector.cs

Objekt této třídy řídí ukládání potřebných informací o průběhu hraní úlohy. Uložené informace následně slouží pro terapeuty, kteří na jejich základě dokáží analyzovat úspěšnost a zlepšení nebo zhoršení v rehabilitaci a adekvátně k tomu upravit léčbu a způsob neurorehabilitace.

GameResults.cs

Skript obsahuje definici třídy, která ukládá a vytváří výstupní řetězec z celkových výstupních parametrů. Mezi tyto parametry patří čas spuštění úlohy nebo změny pozic a velikostí herních objektů v závislosti na čase.

Spolu s dílčími výstupními parametry tvoří výstupní řetězec, který se odesílá do webové aplikace **BrainIn**.

Round.cs

Třída uchovávající dílčí výstupní parametry kola. Při vytváření výstupních parametrů se prochází jednotlivé instance této třídy a vytváří se výstupní řetězec, který je následně odeslán do webové aplikace, kde se dále zpracuje.

Hodnoty proměnných v této třídě se upravují pomocí instance třídy **DataCollector**.

8.2 Lokalizace

Jelikož je systém vícejazyčný, je potřeba mít i tento způsob zařízený v instancích her. Aktuálně jsou vytvořeny tři verze jazyků (čeština, angličtina a němčina).

Pro správu překladů v hrách se využívají skripty, které obsahují překlady používané ve hře. Konkrétně se jedná o soubory **Inputs, Warnings** a **Errors**. Tyto soubory obsahují definice textů v různých jazycích. Podle obsahu těchto skriptů se vytvářejí serializovatelné soubory ve formátu *XML*[1]. *XML* soubory již využívá samotná webová aplikace pro úpravu obsahu dialogových oken v průběhu hraní hry. Příklad vytvoření jednoduchého překladu, který bude převeden do *XML* formátu je uveden v algoritmu 8.3.

```
1    [XmlElement("start")]
2    public StringItem Start = new StringItem { Cs = "Spustit", En =
     "Start", De = "starten" };
```

Algoritmus 8.3: Příklad vytváření lokalizovaných textů ve hře

Spolu s těmito texty a soubory je ještě potřeba vytvořit zvukovou nápovědu, která popisuje základní úlohu uživatele. Bude využita pokaždé, když uživatel klikne na nápovědu ovládání a zároveň bude mít zapnutý zvuk. Samozřejmě se z důvodů lokalizace vytvářejí tři soubory pro jednotlivé jazyky. Názvy jednotlivých zvukových souborů pro hru **Domaluj polovinu** jsou následující:

- SYMETRIC_cs.mp4
- SYMETRIC_en.mp4
- SYMETRIC_de.mp4

Pro vytváření těchto souborů byl využit nástroj **TextToSpeech**, který dokáže konvertovat text do mluveného slova v různých jazycích. Konkrétně se jednalo o nástroj, který se nachází na internetové adrese `https://wideo.co/text-to-speech/`.

[1] Extensible Markup Language - obecný značkovací jazyk pro serializaci

9 Implementace

V této kapitole se zaměřím na implementaci jednotlivých her a popíši zde jednotlivé funkce, ve kterých se hry liší. Zároveň popíši i prvky herní scény a případné problémy a řešení, na které jsem během postupného vývoje her narazil.

9.1 Implementace malování

Jelikož všechny hry souvisí s malováním, bylo potřeba vytvořit univerzální metody, které budou pro všechny úlohy podobné a budou se lišit pouze implementací rozhodování o úspěšnosti kreslení.

O malování se stará herní objekt **DrawManager**, který je spravován podle stejnojmenného skriptu. Obsah se může lišit od jednotlivých her, jelikož se v každé hře vyhodnocuje jiný výsledek kreslení.

9.1.1 Objekt čáry

Pro reprezentaci čáry byl vytvořen tzv. **Prefab**. Tento objekt se používá, pokud se daný herní prvek bude vytvářet opakovaně a je potřeba, aby vlastnosti tohoto prvku, byly pokaždé stejné.

Objekt, který provádí kreslení, obsahuje komponentu **Trail Renderer**, která vytváří cestu, po které procházel pohybující se objekt. Tato komponenta se většinou využívá, když je potřeba přidat do hry efekt pohybu. Ve výchozím nastavení je dáno, že cesta po chvilce zmizí. Naštěstí tento parametr je volitelný a po nastavení na hodnotu **Infinity** bylo docíleno toho, že cesta bude zobrazena po celou dobu běhu programu.

Dále zde byly nastaveny další parametry, které udávají podobu čáry. Jedná se hlavně o barvu a šířku.

9.1.2 Skript DrawManager

Tato třída je vytvořena z důvodu ovládání kreslení s použitím myší č dotyků prstů na obrazovce. Toto ovládání se dělí na tři části, mezi kterými se rozhoduje v metodě **Update**. Tyto části jsou začátek kreslení, aktualizace čáry a ukončení malování. Proto byly vytvořeny tři metody **StartDraw, UpdateDraw** a **EndDraw**, které se o tyto tři fáze starají.

Ovládání prostřednictvím dotykové obrazovky se řídí pomocí třídy **Input**, která poskytuje rozhraní pro získávání informací o pozici a různých fázích průběhu stisknutí. Tato třída rozeznává celkově 5 fází, které mohou nastat při pohybu prstu. U jednotlivých metod, které jsou popsány níže v této sekci, jsem upravil jejich obsah, aby obsahovaly pouze potřebné části pro pochopení toho, jak funguje kreslení.

Implementaci metody **Update** lze vyčíst z algoritmu 9.1.

```
1    void Update() {
2        if (Input.touchCount == 1 && Input.GetTouch(0).phase ==
   TouchPhase.Began) {
3                StartDraw(Input.GetTouch(0).position);
4            }
5            else if (Input.GetMouseButtonDown(0)) {
6                StartDraw(Input.mousePosition);
7            }
8            else if (Input.touchCount == 1 && Input.GetTouch(0).
   phase == TouchPhase.Moved && isPainting) {
9                UpdateDraw(Input.GetTouch(0).position);
10           }
11           else if (Input.GetMouseButton(0) && isPainting) {
12               UpdateDraw(Input.mousePosition);
13           }
14           else if (Input.touchCount == 1 && Input.GetTouch(0).
   phase == TouchPhase.Ended || Input.GetMouseButtonUp(0)) {
15               EndDraw();
16           }
17   }
```

Algoritmus 9.1: Metoda Update ve skriptu DrawManager

StartDraw

Tato funkce se stará o vytvoření nové instance čáry a umístění na správnou pozici, tj. na aktuální pozici kurzoru nebo prstu. Metoda nejdříve posune pozici objektu, na pozici kurzoru, na které se následně s využitím metody **Instantiate** vytvoří nový herní prvek čáry. Tato metoda se používá, pokud je třeba vytvořit nový objekt z předpřipraveného **prefabu** (viz algo. 9.2).

```
1    private void StartDraw(Vector3 pointerPosition)
2    {
3        transform.position = Camera.main.ScreenToWorldPoint(
   pointerPosition);
4        Trail = Instantiate(LinePrefab, transform.position,
   Quaternion.identity);
5    }
```

Algoritmus 9.2: Metoda StartDraw ve skriptu DrawManager

UpdateDraw

Metoda reaguje na pohyb kurzoru, poté co je již vytvořena nová instance čáry. Na základě aktuální pozice ukazatele posune bod, který řídí dráhu čáry a **TrailRenderer** se následně postará o vykreslení cesty, jak se změnila oproti předešlé pozici. Tímto způsobem se ve všech hrách vykreslují čáry, což je vidět na algoritmu 9.3.

```
1     private void UpdateDraw(Vector3 pointerPosition)
2     {
3         Ray drawRay = Camera.main.ScreenPointToRay(pointerPosition)
      ;
4         if (plane.Raycast(drawRay, out float dist) && Trail != null
      )
5         {
6             Trail.transform.position = drawRay.GetPoint(dist);
7         }
8     }
```

Algoritmus 9.3: Metoda UpdateDraw ve skriptu DrawManager

EndDraw

Tato funkce už pouze přeruší kreslení, pokud dojde k ukončení ze strany uživatele. Následně provede vyhodnocení a zároveň se zde kontroluje, jestli při kreslení nebyla doba kreslení příliš krátká. V tom případě, by se totiž mohlo jednat pouze o stisknutí tlačítka a vytvoření „zbytečné“ tečky, která by mohla pacienta mást.

9.2 Načítání obrázků

U dvou úloh jsem musel implementovat načítání obrázků ze souborového systému serveru. Kvůli tomu byla vytvořena třída **ImageLoader**, která se může v některých částech lišit pro obě hry, ale načítání obrázků má stejnou funkcionalitu.

Metoda postupně načítá obrázky s využitím jednotlivých cest k obrázkům. Jelikož ovšem webová aplikace neposílá absolutní cestu k obrázku, ale pouze cestu k souboru ze složky dané úlohy, bylo potřeba tuto cestu ještě zkompletovat. Důvodem toho je používání třídy **UnityWebRequest**, která se stará o komunikaci s webovým serverem. Použita je zde hlavně z důvodu, že je to jeden ze způsobů, jak můžeme do hry, která je vytvořená pro technologii **WebGL**, načíst obrázky.

Algoritmus funguje tak, že se nejdříve pošle dotaz na webový server systému, pomocí kterého se následně stáhnou binární data obrázku. Prostřednictvím těchto dat se následně vytvoří textura, která se bude zobrazovat na obrázcích.

Jelikož při vývoji není potřeba webové aplikace, bylo třeba vzít v potaz i případ načítání obrázků z počítače, k čemuž slouží větvení programu. Následně se pouze načtou binární data, ovšem jiným způsobem než v případě použití webového serveru.

Celý tento proces je popsán v algoritmu 9.4 na straně 39.

```
1     private IEnumerator LoadImagesFolder2Texture() {
2         foreach (string url in InputParams.ImagesPaths)
3         {
4             byte[] imgData = null;
5             if (url.Contains("://") || url.Contains(":///"))
6             {
7                 UnityWebRequest www = UnityWebRequest.Get(url);
8                 yield return www.SendWebRequest();
9                 imgData = www.downloadHandler.data;
10            }
11            else
12            {
13                imgData = File.ReadAllBytes(url);
14            }
15
16            Texture2D tex = new Texture2D(0, 0);
17            tex.LoadImage(imgData);
18            tex.wrapMode = TextureWrapMode.Clamp;
19
20            textureImages.Add(tex);
21        }
22    }
```

Algoritmus 9.4: Metoda LoadImagesFolder2Texture ve třídě ImageLoader

9.3 Úprava vstupních a výstupních parametrů

Postupem vývoje jsem narazil na problémy spojené s prvotní analýzou her. Tento problém byl hlavně spojen s tím, že práce s obrázky byla původně zamýšlena s využitím datového typu **Složka**, do kterého by terapeut jako vstupní parametr s obrázky zadal právě složku ze souborového systému serveru a obrázky, by se načetly z této složky. Ovšem tato představa byla ze začátku těžko realizovatelná a bylo nutné změnit typ vstupního parametru na **Skupinu otázek**. Tento typ totiž přímo dovoluje k dané otázce nahrát obrázek a tím odeslat do hry přímo existující cestu k obrázku. Zároveň se díky tomu zpřehlednilo zadávání dalších parametrů, což podle mě i zlepšilo uživatelskou přívětivost, protože je vidět, jaké parametry se budou nastavovat jednotlivým obrázkům. Tato změna se týká her **Domaluj polovinu** a **Spojování dvojic**.

9.3.1 Změna u Domaluj polovinu

Původně bylo v plánu nahrávání obrázků přes složku, ve které by byly jednotlivé obrázky uloženy. Jak jsem již ale zmiňoval, technické možnosti mi neumožňovali tuto možnost implementovat, a tak jsem využil možnosti **Skupiny otázek**. Zde se obrázky nahrávají jednotlivě a definují se odpovědi. U této hry jsem těchto odpovědí využil tak, že jsem ostatní parametry (strana obrázku a doba zobrazení obrázku), které byly popsány v sekci 6.2, přiřazoval jako jednotlivé odpovědi otázky. Jako první odpověď se zadává strana, na které bude uživatel kreslit. Zde se pro identifikaci stran

používají lokalizované texty *0* a *1*. Druhá odpověď reprezentuje dobu zobrazení obrázku při zahájení kola. Z důvodu větší parametrizace se dá nastavit, aby byl obrázek vidět po celou dobu hraní. Tohoto výsledku lze docílit zadáním textu s hodnotou *0*.

Nevýhodou této možnosti je zadávání odpovědí pouze v podobě lokalizovaných textů a nepovoluje zadávání jednotlivých čísel samostatně.

9.3.2 Změna u Spojování dvojic

Z analýzy úloh je patrné, že se dvojice měly tvořit také s použitím datového typu **Složka**, odkud by se nahrály všechny možné obrázky a jednotlivá slova by byla vybrána ze jmen obrázků. Na základě předchozí zkušenosti u hry **Domaluj polovinu**, byly upraveny vstupní parametry i této hry.

Nahrazení bylo provedeno s použitím typu **Skupina otázek**, která opět usnadnila přehlednost zadávání úloh. Pro každou otázku se definuje obrázek, zadání otázky a její odpovědi. V tomto případě zadání otázky slouží pro zadání slova, se kterým se bude obrázek spojovat. Odpovědi poté reprezentují kola, ve kterých se dvojice zobrazí.

Pokud by nastala situace, že by některému kolu nebyla přiřazena ani jedna dvojice, program vygeneruje počet dvojic, které se v daném kole využijí a postupně toto kolo naplní dvojicemi z ostatních kol. K tomuto účelu byla vytvořena funkce **FillRound** (viz algo. 9.5), která kolo naplní podle pořadí zadaných dvojic ve vstupních parametrech. K tomu přispívá pomocná proměnná **randomIndexHelper**, která značí ukazatel do seznamu všech dvojic a v případě, že by hodnota přesáhla velikost tohoto seznamu, přeskočí zpět na počáteční hodnotu. Zároveň je potřeba, aby v jednom kole bylo zadáno maximálně pět dvojic.

```
1     private static void FillRound(RoundData roundData)
2     {
3         int max = Pairs.Count > 5 ? 6 : Pairs.Count + 1;
4         int count = Tools.GetRandomValue(max, 1);
5
6         for (int i = 0; i < count; i++)
7         {
8             if (randomIndexHelper >= Pairs.Count)
9                 randomIndexHelper = 0;
10
11            roundData.AddPair(Pairs[randomIndexHelper]);
12            randomIndexHelper++;
13        }
14    }
```

Algoritmus 9.5: Metoda FillRound ve hře Spojování dvojic

9.4 Ukládání výstupních obrázků

Jednou z komplikací, která se při vývoji her vyskytla, byla ukládání obrázků, které se v průběhu her získávají kvůli hodnocení pro terapeuta. Jelikož v systému tato funkcionalita naprosto chyběla, bylo třeba kontaktovat tvůrce webové aplikace, aby toto rozšíření provedl a obrázky, se mohly ukládat na úložiště webového serveru. Zároveň bylo třeba vymyslet, jakým způsobem se bude obrázek posílat nebo ukládat prostřednictvím souborů v **JavaScriptu**.

9.4.1 Zachycení obrázku

Pro zachycení obrázku ve hře byla vytvořena třída s názvem **ScreenShotManager**, která je stejná pro všechny čtyři hry. Tato třída čeká na zavolání funkce **TakeNewScreenShot**, která povolí, aby se v dalším snímku překreslování zachytilo aktuální dění na obrazovce.

Následující snímek byl zvolen proto, aby se v některých případech mohly zobrazit dodatečné herní prvky, které je třeba při terapii vidět. Například u hry **Domaluj polovinu** se může stát, že se pacientovi obrázek schová a doba zobrazení řešení bude nastavená na hodnotu *0*. Kdyby se v tomto případě výsledný obrázek rovnou zachytil, terapeut by neviděl obrázek, který pacient dokresloval a musel by si jej dohledat z jiného zdroje.

Pokaždé když dojde k dokreslení aktuálního snímku, herní engine zavolá metodu **OnPostRender** (viz algo. 9.6), která se vykonává vždy po vykreslení a aktualizaci scény. Konkrétní případ použití lze pozorovat na následujícím příkladu, který se stará o převedení vzhledu scény do podoby obrázku.

```
1   private void OnPostRender() {
2       if (takeScreenShotOnNextFrame) {
3           takeScreenShotOnNextFrame = false;
4           RenderTexture renderTexture = myCamera.targetTexture;
5
6           Texture2D renderResult = new Texture2D(renderTexture.
    width, renderTexture.height, TextureFormat.ARGB32, false);
7           Rect rect = new Rect(0, 0, renderTexture.width,
    renderTexture.height);
8
9           renderResult.ReadPixels(rect, 0, 0);
10
11          byte[] byteArray = renderResult.EncodeToJPG();
12          string base64string = ConvertToBase64(byteArray);
13
14          UnityToWeb.MakeScreenShot(base64string);
15
16          RenderTexture.ReleaseTemporary(renderTexture);
17          myCamera.targetTexture = null;
18      }
19  }
```

Algoritmus 9.6: Metoda OnPostRender pro ukládání obrázků

Pokud dostane metoda povolení zpracovat snímek, udělá to prostřednictvím aktuální textury, kterou zobrazuje kamera. Následně se díky tomuto obrázku vytvoří kopie, která se funkcí **EncodeToJPG** převede do formátu **JPG**, který byl vybrán na základě velikosti výsledných obrázků. Jelikož se velmi často bude jednat o velice jednoduché obrázky, nebude nám vadit ztráta kvality, protože daleko zásadnější je vliv na datové úložiště serveru, které by se v jiném případě zaplňovalo rychleji. Právě proto byl využit tento kompresní formát.

9.4.2 Zpracování v aplikaci

Pro odeslání obrázku na web se využívá metoda **UnityToWeb.MakeScreenShot**, která odešle obsah obrázku s využitím skriptu do webové aplikace, kde se následně provedou další kroky, ke zpracování obrázku. Původně bylo zamýšleno odesílat do webové aplikace celé pole jednotlivých bytů. Ovšem zde se naskytl problém, že skript nezvládl zpracovat tolik argumentů. Proto vzniklo stávající řešení, čímž je převedení obrázku do textového řetězce ve formátu **Base64**, což je způsob kódování, které převádí binární data do tisknutelných znaků. Tento formát se používá velice často při práci s obrázky v počítačových programech či aplikacích.

Protože byl použit formát **Base64** bylo nutné jej ještě před uložením převést zpět do pole bytů, které se následně převede do objektu s názvem **BLOB**. Tento objekt představuje velký objekt binárních dat. Většinou se jedná o videa, audio nebo právě obrázky. K převedení řetězce ve formátu do objektu **BLOB** se využívá následující funkce, která je popsána v algoritmu 9.7.

```
function base64ToBlob(base64string, contentType , sliceSize = 512){
    const byteCharacters = atob(base64string);
    const byteArrays = [];

    for (let offset = 0; offset < byteCharacters.length; offset += sliceSize) {
      const slice = byteCharacters.slice(offset, offset + sliceSize);

      const byteNumbers = new Array(slice.length);
      for (let i = 0; i < slice.length; i++) {
        byteNumbers[i] = slice.charCodeAt(i);
      }

      const byteArray = new Uint8Array(byteNumbers);
      byteArrays.push(byteArray);
    }

    return new Blob(byteArrays, {type: contentType });
}
```

Algoritmus 9.7: Ukázka převodu řetězce ve formátu Base64 do objektu BLOB

Jak je vidět z výše uvedené části kódu, lze při konverzi řetězce definovat výsledný formát souboru. V našem případě se bude jednat o formát **JPG**, jak jsem již vysvětlil v sekci 9.4.1.

Poté co se provede konverze z řetězce na **BLOB**, je možnost již výsledný objekt uložit na úložiště serveru. Tuto funkcionalitu vytvářel vývojář P. Šnejdar, který se podílí na implementaci webové aplikace. Zároveň vytvořil nový výstupní formát ve webové aplikaci, a to datový typ **Obrázek**, který se stará o zobrazování výsledného obrázku. Všechny obrázky se ukládají s názvem podle časové značky v době pořízení obrázku, a to hlavně z důvodu zamezení duplikování jmen.

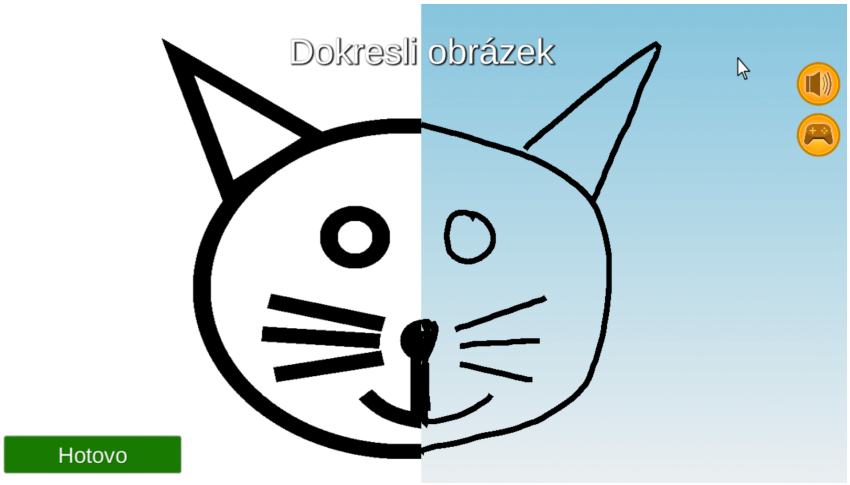
Nakonec se cesta k souboru odešle zpět do instance hry a tato cesta se uloží jako řetězec do výstupních parametrů.

V současné době ještě neexistuje rozšířené řešení zobrazovaní obrázků, tudíž je při analýze vidět pouze výsledná cesta v souborovém systému serveru. Ovšem na zobrazování obrázků, přímo prostřednictvím webové aplikace, současně pracuje tým vývojářů, který se tímto problémem zabývá.

9.5 Domaluj polovinu

Tato hra byla jako první z těch, které jsem vyvíjel. Úloha byla vybrána jako první z důvodu, že se zde objevily všechny potřebné funkce, které se následně daly využít v dalších úlohách. Jedná se o načítání obrázků, kreslení a samozřejmě samotné ukládání obrázků, které jsem již vysvětlil v sekci 9.4 na straně 41.

Pro vývoj této hry byl zvolen pracovní název **SYMETRIC**. Tento název je použit i jako název šablony ve webové aplikaci. Snímek z rozehrané hry je zobrazen na obrázku 9.1.



Obrázek 9.1: Ukázka ze hry Domaluj polovinu

9.5.1 Definice herních objektů

Kromě běžných herních objektů byly vytvořeny i další dva objekty. **LeftImage** a **RightImage** jsou objekty, které v této úloze zobrazují načtené obrázky. Pro vytvoření těchto prvků byl stvořen **Prefab** s názvem **RawImage**.

Oba prvky jsou rozmístěny tak, aby vyplňovaly konkrétní část obrazovky a vymezovaly pacientovi, na které straně má kreslit. Do těchto objektů se v průběhu celé úlohy načítají textury, které byly staženy z webové aplikace.

9.5.2 Zpracování vstupních parametrů

Při zpracování vstupních parametrů je potřeba vstupní řetězec rozložit na jednotlivé části a ty poté podrobit zkoušce, jestli neobsahují nepovolená data, která by mohla způsobovat pády aplikace. Může se jedna o záporný počet kol nebo nezadanou cestu k obrázku.

V této úloze je hlavní rozdíl ve zpracování v posledním vstupním parametru, který popisuje obrázky a jejich přídavné vlastnosti. Nejdříve se řetězec s otázkami rozdělí na samostatné otázky, které se kontrolují každá zvlášť. Kontroluje se zadání správných číselných hodnot, rozhoduje se o straně, kde bude pacient kreslit a zároveň se ověřuje, jestli je cesta k souboru, která byla zadána, správná.

Pro rozdělení vstupů, byl využit předpoklad, že správná cesta k souboru obsahuje přesně 5 částí. Ovšem, pokud by cesta k souboru zadána nebyla, tj. ve webové aplikaci by nebyl vybrán příslušný obrázek, odešle se cesta, která má pouze 3 části.

Pokud dojde k tomu, že cesta byla zadána správně, musí se vytvořit **URL** adresa k danému obrázku, aby se mohl v dalším kroku stáhnout jeho obsah pomocí internetového požadavku.

Na základě těchto parametrů se vytvoří objekty, které hodnoty vstupních parametrů budou uchovávat po celou dobu běhu programu. K tomuto účelu jsou vytvořeny instance třídy **RoundData**.

RoundData

V každém kole je potřeba si uchovat patřičné informace, které jsou důležité k jeho ovládání a přípravě. V tomto případě se jedná o cestu k souboru, stranu, kde bude uživatel kreslit a dobu zobrazení obrázku. Zobrazení základních parametrů je možné najít v konstruktoru třídy **RoundData** (viz algo. 9.8).

```
1     public RoundData(bool side, int showTime, string path)
2     {
3         ImagePath = path;
4         ImageSide = side;
5         ImageShowTime = showTime;
6     }
```

Algoritmus 9.8: Konstruktor třídy RoundData ve hře Domaluj polovinu

9.5.3 Průběh kola

Po zpracování vstupních parametrů, implementaci načítání obrázků a malování s použitím myši či prstu, bylo potřeba nadefinovat zobrazování obrázků na počátku úlohy a zároveň kontrolovat, aby uživatel nemohl malovat přes zobrazený obrázek, což by mohlo způsobovat negativní efekt.

Počáteční zobrazení úkolu

Po spuštění kola se nastaví herní objekt, kde má být zobrazen obrázek a následně se načte stažená textura jako obsah daného obrázku, který se následně zobrazí. Poté se nastaví objektu třídy **DrawManager** strana, na které uživatel bude provádět kreslení, aby věděl, jaké hranice má kontrolovat v průběhu pacientova malování. Nakonec se pouze počká zadanou dobu ze vstupních parametrů, aby měl pacient možnost si zapamatovat předaný obrázek a případně se zobrazený obrázek schová.

Implementace omezení

Potřebným omezením úlohy je, aby měl uživatel vyhrazený prostor, ve kterém může bez problémů kreslit a plnit cíl úlohy. Proto třída **DrawManager** kontroluje při každé aktualizaci nebo vytváření čáry, jestli uživatel toto omezení neporušuje.

Protože třída zná, na které straně uživatel kreslí, stačí zjistit, jestli je nová pozice na stejné straně. K tomuto účelu byla vytvořena metoda **TestPointerPosition**, která kontroluje pozici ukazatele a případně upraví její pozici, aby byla v polovině obrazovky. Tudíž se nemůže stát, že by uživatel kreslil přes obrázek.

Dalším možným řešením bylo, aby uživatel kreslil pod obrázek, ale jelikož dochází i ke schovávání obrázku, mohlo by to pacienta zbytečně mást a způsobovat zbytečné problémy.

Zobrazení nápovědy

Jelikož jsem se při vývoji her snažil dodržet standardy systému, bylo třeba zde vymyslet i nějakou formu nápovědy. U této úlohy se pouze uživateli ukáže schovaný obrázek, ovšem pokud je obrázek zobrazen po celou dobu hraní, pacient již nemá možnost nápovědu využít.

Samozřejmě možnost nápovědy se může lišit kolo od kola, jelikož doba, po kterou bude zobrazen obrázek, je jiná pro každé kolo.

9.5.4 Zpracování výstupních parametrů

Kromě ukládání základních výstupních parametrů bylo potřeba ukládat i další data. Pro ukládání se používá metoda **SetInstanceOutputData** (viz algo. 9.9), která uloží všechna data, která jsou speciální pro danou hru. Spolu s těmito informacemi se ukládá i výstupní obrázek.

Po skončení úlohy se z těchto informací vytvoří výstupní řetězec, který se odešle do systému, kde se data uloží pro pozdější analýzu.

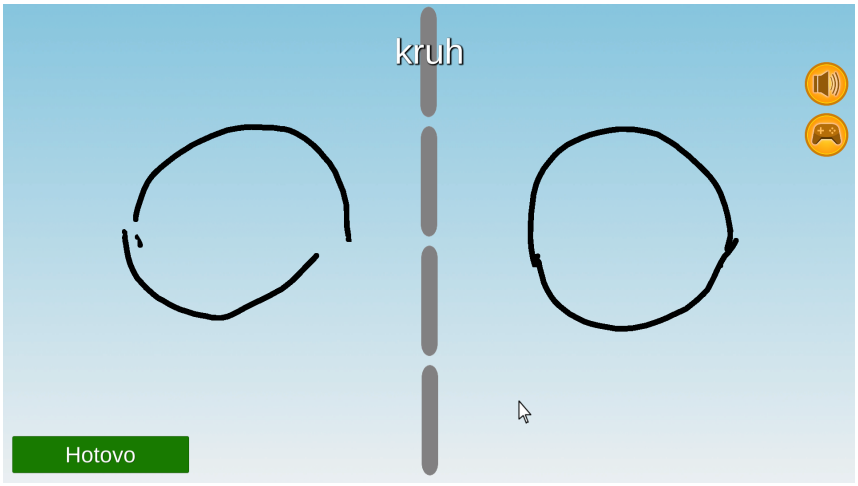
```
1    public void SetInstanceOutputData(float time, int count, string
     path) {
2        currentRound.InputImagePath = path;
3        currentRound.DrawingTime = time;
4        currentRound.LinesCount = count;
5    }
```

Algoritmus 9.9: Metoda SetInstanceOutputData ve hře Spojování dvojic

Tyto informace se shromažďují v průběhu hraní ve třídě **DrawManager**, a to na základě všech pokusů a namalovaných čar. Pokud ovšem nedojde ani k jednomu pokusu o malování a uživatel stiskne tlačítko pro dokončení kola, tak jsou výstupem pouze výchozí hodnoty a kolo je bráno jako neúspěšné.

9.6 Malování dvěma rukama

Název šablony je **DOUBLE_TROUBLE**. V pořadí vývoje tato hra obsadila druhé místo, kvůli pokročilejší podobě malování dvěma prsty. Díky tomu se zamezilo případným problémům, které by mohly vzniknout, pokud by hra byla vytvářena jako poslední. Grafické rozhraní hry je zobrazeno na obrázku 9.2.



Obrázek 9.2: Ukázka ze hry Malování oběma rukama

9.6.1 Definice herních objektů

V této hře byl vytvořen jeden dodatečný herní objekt, který představuje rozdělení obrazovky v půlce. Toto rozdělení nutí uživatele kreslit na rozdílných stranách obrazovky. Tento přidaný objekt se nazývá **Blocker** a pouze zobrazuje obrázek s rozdělením.

9.6.2 Zpracování vstupních parametrů

Vstupní parametry se zde zpracovávají podobně jako u předešlé úlohy, akorát se rozdělují pouze slova, která bude pacient kreslit. Z těchto slov se následně vytvoří instance třídy **RoundData** (viz algo. 9.10), která obsahuje pouze ukazatel na konkrétní slovo.

```
1     public RoundData(string Word) {
2         this.Word = Word;
3     }
```

Algoritmus 9.10: Třída RoundData ve hře Maluj oběma rukama

9.6.3 Ovládání průběhu kola

Rozšířené malování

Protože se zde používá malování oběma rukama, bylo nutné upravit stávající podobu třídy **DrawManager** dle potřeb. Třída kontroluje, kolik dotyků je na obrazovce a na základě těchto informací, se snaží aktualizovat podobu scény a vytvářet jednotlivé čáry.

Pohyb jednoho prstu po obrazovce se dá rozdělit do 5 případů: **zahájený, pohyblivý, nepohyblivý, zrušený** nebo **ukončený**. Pro zjištění aktuálního stavu dotyku se využívá rozhraní **Input** a konstant uložených ve výčtovém typu **TouchPhase**. Na základě těchto informací se **DrawManager** rozhoduje, jaká je fáze kreslení, a tudíž i o tom, co má v danou chvíli dělat.

Zahájení kreslení může nastat ve dvou případech. Prvním z nich je, že oba dotyky vzniknou ve stejném snímku zároveň. Další možností je, že uživatel nejdříve umístí jeden prst na obrazovku, se kterým může pohybovat nebo ne. Následně umístí i druhý prst a tím se vytvoří dvě čáry, které se aktualizují dle pohybu pacienta.

K ukončení dochází, pokud alespoň jeden prst bude z obrazovky odstraněn nebo bude zrušen operačním systémem zařízení.

Zde je největší rozdíl v tom, že k zahájení kreslení jsou potřeba dva dotyky na obrazovce, což ovšem prostřednictvím samotné myši nelze docílit. Proto zde vniká omezení hraní úlohy, a to pouze na dotykových zařízeních, které podporují **WebGL**. V aktuální době **WebGL** plně nepodporuje mobilní internetové prohlížeče a tím pádem vzniká při spouštění problém, který se dá obejít, pokud se režim prohlížeče v telefonu přepne do režimu pro počítače.

Implementace omezení

Protože se v této úloze používá rozdělení obrazovky na poloviny, bylo nutné implementovat i omezení, které bylo zpracováno v úloze **Domaluj polovinu**. Kontroly se musí zúčastnit oba dotyky a zároveň je také potřeba, aby se identifikovalo, jaké identifikační číslo má dotyk na pravé, resp. levé polovině obrazovky. Tato identifikace je potřebná z důvodu aktualizace čar, která by se nemusela provádět správně. Proto byly vytvořeny dvě pomocné proměnné **positiveTouchId** a **negativeTouchId**, které značí dotyky na levé a pravé straně.

Pokud nastane situace, že prostřední oddělovač nebude nastaven, jsou proměnné vybrány v pořadí, v jakém byly vytvořeny jednotlivé dotyky na obrazovce.

Zobrazení nápovědy

Po diskuzi s paní Šroglovou z FN Lochotín, bylo rozhodnuto, že u této úlohy nápověda nebude moci být využita. Důvodem je, že jsme společně nenalezli žádnou vhodnou podobu, která by pacientovi dokázala poradit. Nejvíce se nabízí, uživateli zobrazit obrázek předmětu, který má kreslit. Nakonec bylo ovšem konstatováno, že by mohlo docházet k demotivaci, protože by pacient již mohl mít část obrázku

nakreslenou jinak, což by se mu nemuselo líbit. Z tohoto důvodu zde nebyla nápověda vytvořena.

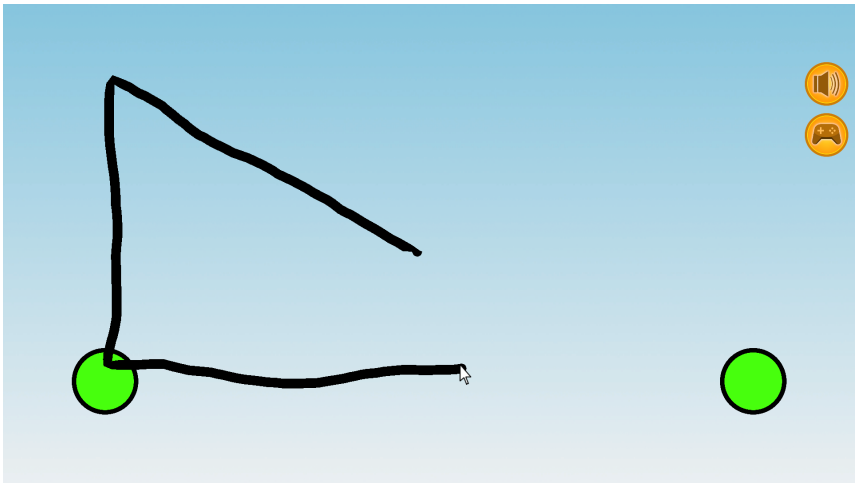
9.6.4 Zpracování výstupních parametrů

Stejně jako u každé jiné hry ze systému, i zde se ukládají výstupní parametry. Kromě základních parametrů a cesty k výstupnímu obrázku, se zde ukládají i další dvě informace, které informují o době, jak dlouho uživatel strávil samotným kreslením a o počtu čar, které vytvořil, protože by to z výstupního obrázku nemuselo být v některých případech patrné. Tyto informace sbírá třída **DrawManager**, která je aktualizuje v průběhu kreslení čar.

Všechny ostatní důležité informace jsou již obsahem výstupního obrázku, tudíž nemá smysl je ukládat tímto způsobem.

9.7 Spojení bodů

Další vyvíjenou úlohou bylo **Spojení bodů**, pro kterou byl vybrán název **VERTEX_JOIN**. Při implementaci bylo potřeba vytvořit několik obrázků a nové skripty, které budou reagovat na vstup do bodu a jeho opuštění. Na obrázku 9.3 je ukázka ze hry, která byla vytvořena.



Obrázek 9.3: Ukázka ze hry Spojení bodů

9.7.1 Definice herních objektů

Vytvořené herní objekty jsou body, které má pacient spojovat a následně šipky, které se zobrazují v případě použití nápovědy.

Body

Všech pět bodů, které se mohou spojovat jsou uskupeny do herního objektu **PointArea**. Díky tomu můžeme přistupovat přímo k bodům bez zbytečného hledání

(pouze přistupujeme k objektům rodičovské komponenty).

Pro body byl vytvořen herní objekt **Prefab** s názvem **Point**, který obsahuje obecné vlastnosti. Vytvořené instance se liší pouze v pozici na obrazovce. Bod je reprezentován obrázkem kruhu, který má různou barvu v závislosti na tom, jestli je bod počáteční, koncový nebo je pouze prostředníkem spojení mezi začátkem a koncem. Pro přepínání mezi těmito stavy se využívá komponenta zvaná **Animator**, která dokáže na základě různých podmínek či podnětů spouštět animace, které mohou měnit vzhled herního objektu, ke kterému je **Animator** přidružen.

Šipky

Nápověda postupně zobrazuje, jaké body má pacient spojovat. Proto, aby nápověda byla přehledná a lépe srozumitelná, byly vytvořeny tři šipky, které mají různou velikost v závislosti na tom, mezi kterými body budou použity. Například mezi horními body předpokládáme šipku větší než mezi prostředním a levým dolním bodem. Obrázek 9.4 zobrazuje jednu z možných šipek, která se zobrazuje při použití nápovědy.

Obrázek 9.4: Ukázka šipky

Všechny tyto objekty slouží pouze k zobrazení obrázků se šipkami, které při použití nápovědy mění svoji původní pozici na požadované místo, zobrazí se a následně se jejich pozice nastaví na původní hodnotu, tj. do středu obrazovky a skryjí se. Šipky jsou uspořádány do objektu **HelpPointers**, a to hlavně kvůli přehlednosti v hierarchii objektů herní scény.

Popis, jak se provádí změny pozice je uveden v sekci 9.7.3 na straně 50.

9.7.2 Zpracování vstupních parametrů

V analýze hry v sekci 6.1 na straně 25 byly zmíněny přidané vstupní parametry hry. Pro identifikaci bodů bylo tedy třeba využít identifikátory, které by se dokázaly jednoznačně rozlišit při zpracování a zároveň nebylo jejich zadávání příliš obtížné přes webové rozhraní aplikace.

V aktuální podobě úlohy je zvolena identifikace podle toho, kde se daný bod nachází. Například pokud definujeme úlohu, kde se bude spojovat levý horní bod s pravým horním a zároveň toto spojení půjde přes prostřední bod, byl by zadaný řetězec zadán následovně: *LH,S,PD*. Tímto dokážeme jednoduše rozlišit bod nejen v programu, ale i při zadávání úlohy.

Podle těchto identifikátorů se nakonec provede mapování na identifikační čísla, která se používají v programu. Při zpracování výstupních parametrů se opět provede toto mapování, ovšem opačně.

9.7.3 Ovládání průběhu kola

Reakce na vstup do bodu a jeho opuštění

Pro identifikaci bodů a jejich reakci na vstup do bodu byl vytvořen skript **PointButton**. Ještě před samotnou reakcí na vstup do bodu, je potřeba je identifikovat, aby ostatní instance věděly, do kterého bodu bylo vstoupeno. Proto tato třída obsahuje několik proměnných, které bod jednoznačně identifikují a definují, jestli se jedná o počáteční, koncový nebo prostřední bod.

Vytvořený skript dědí vlastnosti od třídy **CursorHoverEffect**, která při najetí myši nad objekt uživatelského rozhraní změní vzhled ukazatele myši. Této funkcionality bylo využito a metody, které se o tuto funkci starají, byly ve skriptu **PointButton** přepsány, aby zároveň informovaly třídu **DrawManager**. Přepsané metody jsou **OnPointerEnter** a **OnPointerExit**. Pokud nastane situace, že uživatel vstoupí do počátečního bodu, informuje o tom třídu **DrawManager** a zároveň zobrazí následující bod v cestě, aby pacient věděl, který bod má dále spojit. Potom se uživatel snaží spojit následující bod a poté, co se mu to podaří, se předešlý bod schová a ukáže se následující v pořadí. Tento proces se opakuje, dokud pacient nespojí všechny body v zadané cestě.

Zobrazení nápovědy

Jak již bylo zmíněno, pro zobrazování nápovědy se používají šipky, které postupně zobrazují cestu po jednotlivých spojích. Pro přesunutí šipky na místo mezi body byla použita metoda **ShowArrow**, která nejdříve zjistí pozice bodů. Díky tomu se vybere šipka, která bude pro nápovědu použita a spočte se její výsledná pozice. Tyto informace se musí držet v paměti do doby, než dojde k zobrazení nové šipky. Je to z důvodu, aby se mohla pozice zobrazované šipky obnovit do původní pozice.

Díky tomu, že se pozice umístění vypočítávají při každém zobrazení šipky, dosáhneme toho, že ukazatelé budou vždy ukazovat na body přesně, jak chceme. Kdybychom je definovali přímo v herní scéně, mohlo by se při změně velikosti obrazovky stávat, že by šipka nemířila přímo k bodu, což by mohlo působit nepříjemně.

9.7.4 Zpracování výstupních parametrů

Kromě výstupního obrázku a dalších parametrů, které již byly mnohokrát zmiňovány se ukládají i informace o jednotlivých pokusech, které uživatel v celém kole provedl. Nakonec zde byla provedena menší úprava výstupních parametrů oproti předchozí analýze. Pokusy a jejich časové intervaly byly spojeny do jednoho objektu, který se ukládá ve formátu **JSON**, což zvyšuje přehlednost daného pokusu.

Zároveň byly přidány další dvě informace o pokusu.

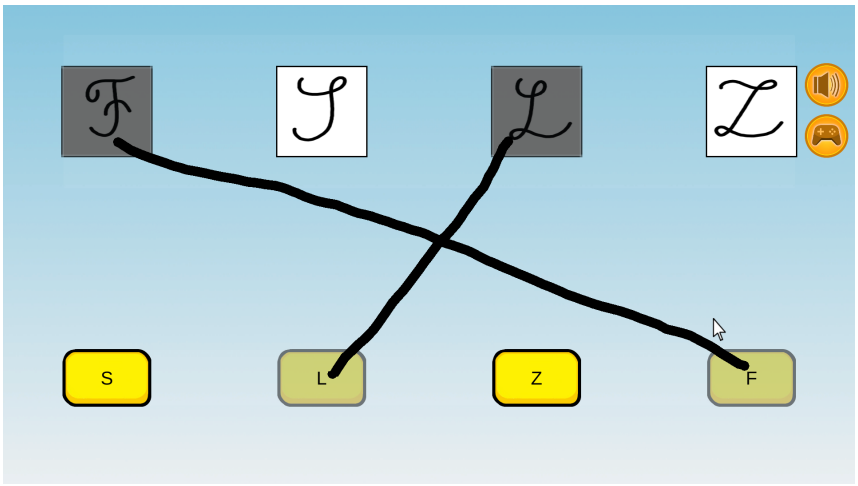
Informace o jednom pokusu se ukládají do instancí třídy **RoundInstanceResult** (viz algo. 9.11), která tyto údaje udržuje a aktualizuje na základě podnětů ze třídy **DrawManager**. Jeden pokus je v aktuální podobě hry popsán čtyřmi parametry. Mezi ně patří cesta, kterou pacient vykonal, počáteční čas, doba kreslení a nakonec identifikační číslo stisknutí myši, aby se dalo lépe vyznat v tom, v jakém pořadí byly pokusy provedeny.

```
1    public RoundInstanceResults(string journey, float startTime,
     float drawTime, int actionId)
2    {
3        Journey = journey.Equals("") ? Constants.NULL : journey;
4        StartTime = startTime;
5        DrawTime = drawTime;
6        ActionId = actionId.ToString(Constants.ACTION_ID_FORMAT);
7    }
```

Algoritmus 9.11: Třída RoundInstanceResults ve hře Spojení bodů

9.8 Spojování dvojic

Poslední vyvíjenou hrou bylo spojování dvojic (viz obr. 9.5). Vývoj této hry byl značně urychlen díky tomu, že se zde využívala vytvořená funkcionalita z předešlých úloh. Jako pracovní název byl zvolen **525**.



Obrázek 9.5: Ukázka ze hry Spojování dvojic

9.8.1 Definice herních objektů

Pro vytváření jednotlivých slov nebo obrázků byly vytvořeny příslušné předpřipravené herní prvky, které se dynamicky vytváří během celého běhu programu.

Pro organizaci těchto prvků slouží herní objekty **Images**, resp. **Words**, které obsahují komponentu **GridLayoutGroup**, která dokáže dynamicky upravit pozici prvků, které jsou do ní vloženy a tím docílit, aby tyto prvky byly uloženy v mřížce.

Teoreticky to znamená, že tlačítka a obrázky mezi sebou budou mít stejně velkou velikost a zároveň se budou soustředit uprostřed.

9.8.2 Zpracování vstupních parametrů

Jak již bylo zmíněno v sekci 9.3.2 na straně 40, vstupní parametry se oproti původní analýze trochu změnily, a to hlavně v datovém typu **Složka**.

Pro rozdělení byl proto použit velmi podobný způsob jako ve hře **Domaluj polovinu**.

9.8.3 Ovládání průběhu kola

Rozšíření mezer

Na základě podnětu, který vznesla paní Šroglová, byla přidána funkčnost, aby mezery mezi dvojicemi byly co největší.

Proto byl vytvořen skript s názvem **DynamicGridLayoutGroup**, který v každém novém snímku spočte maximální možnou mezeru mezi jednotlivými body a zároveň upraví jejich velikost, pokud by to bylo potřeba. Metoda prochází některé možné velikosti obrázků a hledá největší vytvořenou mezeru, která vznikla. Nalezená hodnota se použije pro aktuální snímek. Výhodou tohoto přístupu je, že při změně velikosti okna se mezery přepočítají a případně upraví. Spolu s obrázky se upravují i tlačítka se slovy. Díky tomu se pro kreslení využije celá plocha obrazovky a dochází k efektivnějšímu procvičování motorických funkcí.

Zobrazení nápovědy

Pro zobrazení nápovědy byly vytvořeny dva režimy, které se volí s využitím vstupních parametrů.

První možností je pouze výběr náhodné dvojice, která se na chvilku zvýrazní. Tato možnost byla vybrána zároveň jako upřednostňovaná, jelikož v neurorehabilitaci je potřeba, aby si uživatel procvičil spojování.

Druhý typ nápovědy je, že se náhodná dvojice skryje a uživatel ji nemusí spojovat. Je zde přidaná hlavně kvůli rozmanitosti, která může vzniknout.

9.8.4 Zpracování výstupních parametrů

Stejně jako u **Spojení bodů** byly trochu upraveny výstupní parametry. Pro popis jednotlivých pokusů se vytváří instance třídy **SingleTryInfo**, která má podobné parametry jako předešlá hra. Ovšem jsou zde i informace o tom, v jakém objektu pacient zahájil spojování. Následně dokáže terapeut poznat, který objekt začal pacient spojovat jako první a ve kterém bodu malování ukončil.

10 Budoucí možná vylepšení

Při vývoji her pro širokou škálu pacientů či uživatelů se naskýtá velká škála zlepšení a inovací. Mezi takové patří implementace nových her, které budou mít jiný princip než ty, které byly vyvíjené v rámci bakalářské práce, ale budou procvičovat podobné funkce (tj. jemnou motoriku). Další vylepšení by mohla přinést úprava stávajících her.

10.1 Vylepšení stávajících úloh

Samozřejmě je možné, že časem se v implementovaných hrách může objevit jakýsi problém nebo je bude potřeba upravit kvůli nedostatečnému procvičování kognitivních funkcí. Ovšem zde se pokusím popsat některá vylepšení, která by se v budoucnu dala do her přidat a vylepšit tak jejich funkci v praxi.

Společným vylepšením pro všechny hry, by mohl být jakýsi časovač, který by uživatele limitoval dobou, jakou bude moci kreslit. Například v každém kole by mohl strávit kreslením 5 vteřin. Tudíž by ho to nutilo, aby maloval rychleji a stejně přesně.

Dále by se určitě daly přidat možnosti na úpravu velikosti nebo barvy čáry, která se vykresluje na herním plátně. To by pacientovi pomohlo v motivaci hru hrát, jelikož by to pro něj bylo lákavější.

10.1.1 Malování oběma rukama

Výrazným zlepšením v této hře by bylo, kdyby se výsledek pacientova malování hodnotil prostřednictvím umělé inteligence, která by porovnávala podobnost dvou obrázků. Usnadnilo by to nejen hodnocení terapeutům, ale i zpřesnilo hodnotu úspěšnosti na konci úlohy. Velké úskalí vidím v tom, že vývoj této umělé inteligence by byl náročný, jelikož by se v některých případech musela vypořádat s tím, že by pacient nemaloval přesně na opačných stranách, ale třeba pouze na jedné. Tím pádem by musela provádět různé posuny a odhadovat, jak tyto tvary rozdělit, aby se daly porovnat.

10.1.2 Spojení bodů

U této hry by se dalo upravit například to, aby se uživateli nezobrazoval pouze následující bod, ale několik bodů, což by bylo definována přes vstupní parametry. Samozřejmě, že v některých případech by tento způsob byl velice nepřehledný, protože pokud bychom měli více bodů, které by se střídaly a najednou by se uživateli zobrazil bod, který bude následovat až po dalším spojení, uživatele by to mohlo zmást. Proto by asi byla potřeba využít nějakého způsobu, který by mu pomohl zjistit, jaký bod má aktuálně spojovat.

Zároveň si myslím, že na vyhodnocování preciznosti splnění úlohy, by se také dal implementovat program s umělou inteligencí, který by kontroloval, jestli jsou čáry rovné, případně by dopočítal, jak moc se liší a na základě této odlišnosti upravil adekvátně hodnocení úspěšnosti úlohy.

10.1.3 Domaluj polovinu

Stejně jako u předešlých dvou úloh, by úloze prospělo vyhodnocování umělou inteligencí, které by porovnávalo obrázek, který byl předán ze vstupních parametrů a jeho druhou polovinu, která byla nakreslena pacientem.

Dále by se program mohl rozšířit tak, že by uživatel nemaloval pouze na druhé polovině obrazovky, ale herní scéna by se mohla rozdělit do několika částí (kvadranty, ...), kde by jednu část zabíral kousek obrázku a pacient by se ho snažil symetricky domalovat. Například pokud bychom vzali obrázek čtyřlístku a jednu část obrázku (jeden lístek) zobrazili v některém z kvadrantů, uživatel by následně do ostatních třech nakreslil zbylé lístky čtyřlístku. Ovšem už by se asi jednalo o velmi složitou úlohu a zároveň by se dalo uvažovat, jestli by už takto upravená hra neměla odlišný princip než domalovávání symetrické poloviny.

10.1.4 Spojování dvojic

Možným vylepšení by mohlo být, aby se dalo generovat stejné rozvržení několikrát za sebou. Jelikož se pořadí dvojic náhodně zamíchá, lze definovat hodnotu, která by udávala tzv. semínko. Jedná se o hodnotu, která se vkládá do generátoru náhodných čísel a díky tomu lze generovat stejná čísla při opakovaném spouštění. Tato hodnota by mohla být zadána pomocí vstupních parametrů a mohl by ji tedy terapeut průběžně upravovat a měnit.

10.2 Nápady na nové hry

10.2.1 Spojování čísel

Jednou z možných nových her by mohlo být spojování čísel do určitého tvaru nebo obrázku. Uživatel by měl za úkol spojit body podle daného kritéria a spojením těchto bodů by vznikl výsledný tvar nebo právě obrázek. Kritérii by mohla být vzestupná posloupnost čísel nebo pořadí písmen abecedy. Velkou výhodou by zde byla variabilita kritérií.

Základní herní scénář

Uživatel spustí hru, která mu zobrazí semafor a tím odstartuje první kolo hry. Poté se zobrazí všechny body, které bude muset pacient spojit dle nějakého kritéria. Toto kritérium se uživateli zobrazí v horním popisku hry. Uživatel pak začne spojovat jednotlivé body. Když by dokončil malování ve špatném bodě nebo by pořadí

spojených bodů nebylo správné, tak by pacient musel provést spojení znovu, aby kolo úspěšně absolvoval. Na konci hry by se mu zobrazil spojený obrázek.

Nápověda

Jako nápověda by se dal použít textový řetězec, který by uživateli zobrazil pořadí, ve kterém mají být jednotlivé body spojeny nebo by se mohlo jednat o postupné zobrazení celé posloupnost, resp. její části.

Obtížnost

Vyšší obtížnost by mohla být docílena s využitím složitějších obrazců či kritérií na pořadí. Například by se mohlo jednat o vzestupné řazení záporných či reálných čísel nebo řadit prvky střídavě (větší, menší, ...). Pro jednodušší úlohy by mohly být použity geometrické tvary a vzestupné řazení čísel či písmen abecedy.

10.2.2 Obkreslování obrázku

Další hrou, která by mohla být implementována je obkreslování obrázku. Zde by měl uživatel zobrazen nějaký obrázek nebo tvar, který by se snažil obkreslovat bez toho, aniž by z něho vybočil. Musel by se tedy snažit, aby jeho kreslený tvar byl stejný jako zobrazovaný a trénoval tak preciznost jeho svalů. V této hře by se zároveň dala využít umělá inteligence na rozeznávání, jak moc byl pacient v obkreslování přesný a dokázala by to ohodnotit. Tím by se značně ulehčila práce terapeutům, kterým by výsledná úspěšnost dávala větší informaci o přesnosti obkreslování pacienta.

Základní herní scénář

Pacient by zapnul hru. Po odstartování kola prostřednictvím semaforu by se uživateli zobrazil obrázek, který by měl za úkol obkreslit. Na obkreslení by měl pouze jeden pokus, tudíž je potřeba se při každém kole soustředit, aby výsledné obkreslení bylo co nejpodobnější originálu. Po dokončení malování by uživatel stiskl tlačítko, které ukončí aktuální kolo a přesune jej na další. K obkreslování by využíval pouze jeden prst, myš nebo například stylus.

Nápověda

Možný rozšířením by mohlo být, že by zobrazovaný obrázek po chvilce zmizel. Tudíž by si ho uživatel musel zapamatovat a obkreslit z paměti. Pokud by ale v tomto případě zapomněl, jak přesně obrázek vypadal, mohl by použít nápovědu, která by mu jej ukázala na specifikovanou dobu, po které by opět zmizel.

Obtížnost

V této úloze by se obtížnost dala zvyšovat velice jednoduše. Pouze by se uživateli nahrál jiný obrázek, který by byl komplikovanější nebo by se pro pacienta mohlo připravit více kol, aby svaly v prstech byly déle zaměstnány a tím více trénovaly.

11 Testování

Testování je nedílnou součástí vývoje softwaru, které se provádí, aby se ověřila předpokládaná funkčnost a chování programu. K nalezení chyb je třeba zvolit rozumně velkou množinu testů, protože v určitém časovém úseku dojde k tomu, že se již v programu nebude nacházet tolik chyb. Pokud by jsme chtěli otestovat každý program tak, aby neobsahoval žádné chyby, museli bychom vynaložit velké úselí a obětovat hodně času, protože psaní testovacích scénářů pro každý možný případ je velice zdlouhavé. Testování softwaru lze rozdělit do dvou částí - automatické a manuální. V následujících sekcích popíši, jak jsem tyto způsoby testování využil v rámci této práce.

11.1 Manuální testování

Nejdříve jsem se testování ujal já sám a tím docílil toho, že byly nalezené drobné chyby, které byly hned opraveny. Testování probíhalo vždy v průběhu implementace ve vývojovém prostředí. Nakonec bylo potřeba otestovat správnost výstupních parametrů a chování ve webové aplikaci BrainIn. Toto testování probíhalo vždy na třech odlišných úlohách dané šablony, kde jsem se snažil nasimulovat možný postup pacienta při rehabilitaci. Při testování jsem kontroloval správnost generování kol ze vstupních parametrů, funkčnost malování a nakonec jsem analyzoval, jestli jsou výstupní parametry ve správném formátu a obsahují hodnoty adekvátní k odehrané úloze.

Další etapou bylo otestování jednotlivých her přímo na pacientech, kteří se systémem budou v budoucnu pracovat. Kvůli tomuto testování bylo potřeba nejdříve zaškolit paní Šroglovou, která v několika dnech až týdnech otestovala úlohu celkem na 2 pacientech. Počet těchto osob je velmi ovlivněn onemocněním COVID-19, které nepříznivě ovlivňuje běh v celé zemi a zároveň znepříjemňuje docházení většího množství pacientů na neurorehabilitace. Spolu s tím, bylo třeba sledovat jednotlivé pacienty, jak se systémem pracují a co je potřeba, aby úlohu splnili.

11.2 Automatické testování

Poslední fází bylo vytvoření jednotkových testů. Pro každou úlohu bylo v průměru vytvořeno 30 jednotkových testů, které testují pouze základní funkcionalitu a konzistenci využívaných tříd.

Vytvořené testy určitě nestačí k otestování všech možných případů a výjimek. Výhodou těchto testů je, že pozitivně přispívají k testování funkcionality programu bez přítomnosti programátora. To znamená, že vývojář pouze spustí testování a všechno ostatní provedou jednotkové testy. Na obrázku 11.1 lze vidět, jaký je rozdíl mezi manuálním a automatickým testování. Ve zkratce lze říct, že automatické

testování je v závislosti na čase lepší a to hlavně z důvodu, že prostřednictvím automatického testování lze ušetřit čas a náklady, které se spotřebují na manuální testování. Zároveň nemůže dojít k chybám způsobené člověkem, které by negativně ovlivnily testování.

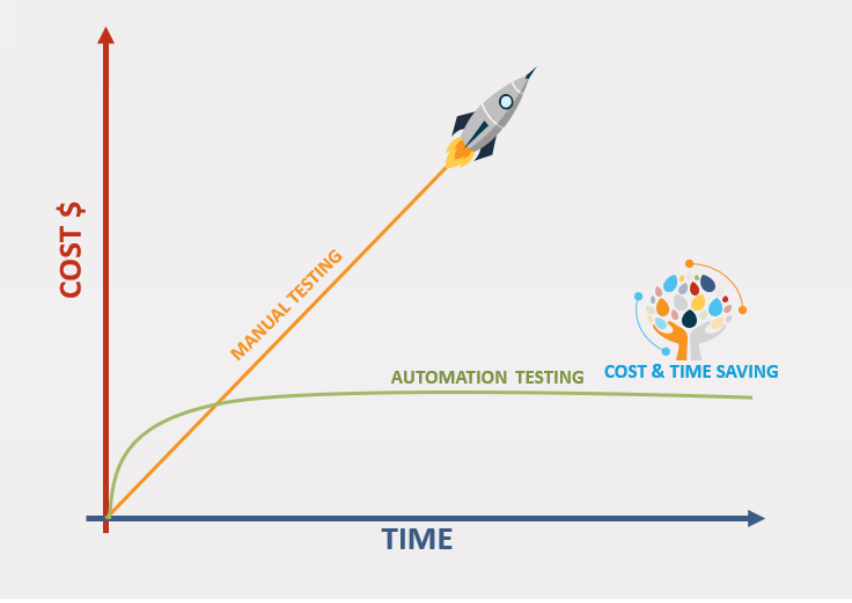
K vytvoření těchto testů byl využit nástroj **Unity Test Runner**, který dokáže vytvářet testy ve vývojovém prostředí **Unity**. Tyto testy se dají rozdělit do dvou režimů. Názvy jednotlivých režimů jsou **PlayMode** a **EditMode**.

EditMode

V tomto módu se testují třídy, které nepotřebují k použití žádné herní objekty ani herní scénu. Díky tomuto způsobu lze testovat konzistenci tříd, které v průběhu úlohy vznikají a zanikají. Většina testů všech her byla tvořena právě v tomto režimu. Jedná se například o rozdělení vstupních parametrů nebo správné vytváření tříd pro ukládání dat pro jednotlivá kola.

PlayMode

Tento režim testuje funkčnost herních objektů, které jsou rozmístěné v herní scéně. Pro každý herní objekt se vytvoří příslušný testovací soubor, kde se již definují akce, které se mají s objektem provést a následně se zkontroluje, jestli je výsledek stejný s očekáváním. V tomto režimu byly vytvářeny testy pro ověření načítání obrázků z webové aplikace.



Obrázek 11.1: Rozdíl mezi automatickými a manuálními testy [5]

11.3 Zhodnocení testování

V rámci celého vývoje i testování se ke mně dostávalo několik připomínek z různých zdrojů. Tyto připomínky byly vytvářeny převážně od paní Šroglové, která hry před použitím testovala a zkoušela si jejich chod, aby se mohla dostatečně připravit na rehabilitace. Po opravení všech nalezených chyb a nedodělků, se práce dostala do finální verze, která je aktuálně nasazená v systému BrainIn a probíhá na ní další testování.

Testování všech her proběhlo v pořádku. Kontrolou byly nalezeny chyby, které byly v průběhu vývoje odstraněny. Na základě zpětné vazby od paní Šroglové bylo zjištěno, že je spokojena a hry má v plánu využívat i nadále. Sice se testování neúčastnilo velké množství lidí, ale dá se očekávat, že se to časem změní, až se zlepší epidemiologická situace spojená s nemocí COVID-19. Pokud se v budoucnu narazí na nějaké problémy, bude mě paní Šroglová kontaktovat a dojde k opravení nalezených chyb.

12 Zhodnocení dosažených výsledků

V rámci bakalářské práce byly vytvořeny 4 hry, které jsou kompletní, funkční a pomáhají pacientům s rozvojem jemné motoriky. Ke hraní her lze použít nejen myš, ale i dotyky na obrazovce, pokud to dovolují technické možnosti zařízení. Pro jednotlivé hry byly vytvořeny vstupní a výstupní parametry, které dostatečně popisují potřebné vlastnosti her a jejich obsah. Zároveň hry rozšířily aktuální systém BrainIn a přidaly do něj novou funkcionalitu v podobě výstupních obrázků.

Pro všechny hry byla vytvořena množina úloh, které se dají využít k přiřazování do balíků. Hry splňují prvotní zadání, které bylo obdrženo od paní Šroglové, která je s hrami taktéž spokojena. Samozřejmě se v budoucnu budu snažit s ní stále komunikovat a řešit případné problémy ve hrách, pokud by na některé narazila při testování s pacienty.

Velkou výhodu vidím v tom, že se hry už používají v praxi. Paní Šroglová již otestovala některé úlohy na 2 pacientech, se kterými v poslední době přišla do styku. Bohužel nedošlo k velkému počtu testování, což je z velké části způsobené pandemií koronaviru. Zároveň jsem měl tu možnost, se s jednou z pacientek dvakrát potkat na osobní schůzce, která se konala za dohledu paní Šroglové. Bylo vidět, že pacientka hry doma procvičuje, kde jí napomáhá rodina a blízcí. Pouze z těchto schůzek se dalo poznat, že se její stav zlepšil už po prvním týdnu rehabilitace, což vidím jako velký úspěch těchto her.

Hry obsahují některé automatické testy, které ověřují základní funkčnost. Ovšem nejsem spokojen s jejich počtem. Určitě by se hodilo mít testů více, aby se daly otestovat všechny případy, které by mohly při hraní vzniknout.

13 Závěr

V rámci této bakalářské práce jsem se nejdříve zaměřil na existující systémy, které se již neurorehabilitací zabývají a vyhledání některých důležitých informací. Zároveň tyto systémy byly srovnány se systémem BrainIn, do kterého byly hry integrovány.

Poté jsem se seznámil se základní funkčností systému BrainIn a získávání všech potřebných dat, které by mohly být potřeba k tomu, aby bylo možné hry do systému integrovat.

Následně se moje pozornost zaměřila na získání obecných informací o funkci mozku, kognitivních funkcí a rehabilitacích, které jsou na ně zaměřené. Výhodou tohoto přístupu bylo, že mi to dalo vhled do problematiky.

Dále jsem provedl analýzu jednotlivých her, které byly vyvíjeny. Bylo třeba navrhnout vhodné vstupní a výstupní parametry každé hry a zároveň připravit základní herní scénář, který udává posloupnost kroků, které byly potřeba implementovat. Analýza vstupních a výstupních parametrů probíhala za dohledu paní Šroglové, která se mi snažila popsat, která data potřebuje sbírat, aby dokázala odehranou hru vhodně a lehce analyzovat.

Před vývojem jednotlivých instancí her, bylo potřeba se seznámit s poskytnutým projektem, ve které se hry vytvářely. Spolu s tímto seznámením přišlo i porozumění používaných technologií, které se v celém systému používají.

Když jsem měl dostatek informací, zahájil jsem implementaci her ve zvoleném pořadí. Tyto hry mohou být použité pro rehabilitaci pacientů, kteří mají potíže s jemnou motorikou. Výhodou těchto her je, že se dají používat na dotykových zařízeních, které mají přístup k systému BrainIn.

Testování bylo velmi ovlivněno pandemií viru COVID-19, která stěžovala docházení pacientů na neurorehabilitace. Nakonec bylo testování provedeno celkem na 2 pacientech. Spolu s tím bylo vytvořeno několik desítek jednotkových testů, které k testování přispěly.

Velmi si přeji, aby hry našly velké využití v praxi a jsem rád, že už jsou tyto hry využívány alespoň paní Šroglovou z FN Lochotín, což pozitivně přispěje k rozšiřování systému a testování jednotlivých her.

Literatura

[1] *Fine motor skills* [online]. Encyclopedia of Children's Health. [cit. 8. 11. 2020]. Dostupné z: `http://www.healthofchildren.com/E-F/Fine-Motor-Skills.html`.

[2] *HAPPYneuron Brain Fitness Program* [online]. [cit. 26. 11. 2020]. Dostupné z: `http://www.happy-neuron.com/`.

[3] *Lumosity Brain Training: Challenge & Improve Your Mind* [online]. [cit. 26. 11. 2020]. Dostupné z: `https://www.lumosity.com/en/`.

[4] *What are Motor Skills?* [online]. EduCLIME, 2020. [cit. 7. 11. 2020]. Dostupné z: `https://www.educlime.com/wharemosk.html`.

[5] *Myths of Test Automation* [online]. QodeStack, 2019. [cit. 30. 4. 2021]. Dostupné z: `https://qodestack.com/myths-of-test-automation/`.

[6] Cherry, K. *How Our Brain Neurons Can Change Over Time From Life's Experience* [online]. Verywell Mind, 2020. [cit. 8. 11. 2020]. Dostupné z: `https://www.verywellmind.com/what-is-brain-plasticity-2794886`.

[7] CogniFit. *Neuroplasticity* [online]. Brain and Neuroplasticity, Neurogenesis, and Cognition exercises with CogniFit, 2015. [cit. 8. 11. 2020]. Dostupné z: `https://www.cognifit.com/brain-plasticity-and-cognition`.

[8] Hasalíková, M. *Mechanismy a příčiny poranění a poškození mozku* [online]. Návraty. [cit. 8. 11. 2020]. Dostupné z: `http://www.navraty.info/verejnost/mechanismy-priciny-poraneni-poskozeni-mozku`.

[9] Jakubcová, T. Mozek a nové teorie učení. Diplomová práce, Masarykova univerzita, Filozofická fakulta, Brno, 2010.

[10] Jaroš, D. Aplikace pro neurorehabilitace. Bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2016.

[11] Klucká, J. – Volfová, P. *Kognitivní trénink v praxi.* Grada, 2009. ISBN 978-80-247-2608-3.

[12] Kraft, V. Systém pro distribuci a správu kognitivních her. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2018.

[13] Mauro, T. *Fine and Gross Motor Skills in Children* [online]. Verywell Family, 2019. [cit. 8. 11. 2020]. Dostupné z: `https://www.verywellfamily.com/what-are-motor-skills-3107058`.

[14] Medalová, K. *Neuron a jeho stavba* [online]. Memtem.cz, 2015. [cit. 24. 4. 2021]. Dostupné z: `https://www.mentem.cz/blog/neuron/`.

[15] Musso, E. *Entity Framework using C#* [online]. C# Corner, 2016. [cit. 25. 12. 2020]. Dostupné z: https://www.c-sharpcorner.com/article/entity-framework-introduction-using-c-sharp-part-one/.

[16] Neubauerová, L. – Javorská, M. – Neubauer, K. *Ucelena rehabilitace osob s postizenim centralni nervove soustavy.* Gaudeamus, 2011. ISBN 978-80-7435-109-9.

[17] Rehabilitace.info. *Co je to neuron a jakou roli hraje v nervovém systému?* [online]. 2019. [cit. 5. 11. 2020]. Dostupné z: https://www.rehabilitace.info/lidske-telo/co-je-to-neuron-a-jakou-roli-hraje-v-nervovem-systemu/.

[18] Saparahayuningsih, S. – Badeni, B. Improving Children's Fine Motor Skills through Pencil Skills. *Proceedings of the International Conference on Educational Sciences and Teacher Profession (ICETeP 2018)*. 2019.

[19] Sirkin, J. *What is Microsoft SQL Server 2016?* [online]. SearchSQLServer, 2016. [cit. 25. 12. 2020]. Dostupné z: https://searchsqlserver.techtarget.com/definition/Microsoft-SQL-Server-2016.

[20] Skala, P. Webová aplikace pro správu neurorehabilitačních programů. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2019.

[21] Čápka, D. *Lekce 1 - Úvod do ASP.NET* [online]. ITNetwork.cz, 2012. [cit. 25. 12. 2020]. Dostupné z: https://www.itnetwork.cz/csharp/asp-net-core/zaklady/tutorial-uvod-do-asp-dot-net.

Seznam obrázků

Seznam algoritmů

Uživatelská dokumentace

Sestavení hry

Tato sekce obsahuje návod na sestavení hry pro **WebGL**.

1. Stáhněte si aplikaci **Unity Hub**, která je dostupná na stránkách `https://unity3d.com/get-unity/download`.
2. Po stažení si aplikaci nainstalujte a spusťte.
3. Až se aplikace spustí zvolte možnost **Installs**.
4. Následně stiskněte tlačítko **ADD** a vyberte verzi **Unity 2018.4.7f1**.
5. Ve vytvořeném menu nezapomeňte zaškrtnout políčko **WebGL Build Support**.
6. Klikněte na tlačítko **DONE** a počkejte, až se vybraná verze nainstaluje.
7. Po dokončení instalace přejděte do záložky **Projects** a klikněte na tlačítko **ADD**.
8. Najděte cestu k projektu s vytvořenou hrou. Projekt s příslušným názvem by se měl zobrazit mezi ostatními projekty.
9. Spusťte přidaný projekt.
10. Až se spustí aplikace **Unity** přejděte do **File/Build Settings**.
11. Jako **Platform** vyberte **WebGL** a následně klikněte na tlačítko **Build**.
12. Vyberte cílový adresář a vyčkejte na vytvoření.
13. Sestavenou verzi hry následně naleznete ve vybraném adresáři. Tato hra se nakonec nahraje do webové aplikace.

Nahrání do webové aplikace

Sekce popisuje, jak nahrát sestavenou hru do webového systému BrainIn.

1. Po sestavení hry je potřeba navštívit stránky `https://brainin.kiv.zcu.cz`, kde je potřeba se přihlásit, případně registrovat.
2. Po přihlášení je nutné si zažádat o práva, která Vám dovolí hru nahrát do systému.

3. Až obdržíte potřebná práva, přejděte do záložky **Šablony/Správa šablon**, kde vyberete šablonu, do které chcete vytvořenou hru nahrát. Zde je nutné vybrat takovou šablonu, pro kterou byla hra vytvářena, jinak by mohlo dojít k problémům.

4. Poté, co se zobrazí detail šablony přejděte do **Správa souborů**, kde kliknete na tlačítko **Správa souborů**.

5. Zobrazí se dialogové okno, kde s použitím tlačítka **Nahrát nové soubory** nahrajete vybrané soubor z Vašeho souborového systému na server.

6. Je potřeba, aby nahrané soubory byly ve stejné struktuře, jako ve vytvořeném adresáři. Proto je třeba vytvořit složky **Build** a **TemplateData** v kořenovém adresáři šablony.

7. Zároveň je nutné nahrát i skripty, které slouží pro komunikaci se serverem. Tyto skripty se nahrávají do složky, kde se nachází složka **Build** nebo **TemplateData**.

Spuštění a otestování hry

V této sekci je popsán návod, jak vytvořenou hru spustit a otestovat její běh a funkčnost.

1. Když je vytvořena šablona, ve které jsou nahrány všechny potřebné soubory, je možnost ji spustit.

2. Pro spuštění je třeba vytvořit novou úlohu v záložce **Úlohy/Vytvořit úlohu**.

3. Zde zadejte Vámi vybraný název úlohy a šablonu, která bude testována a pak klikněte na modré tlačítko **Vytvořit**.

4. V detailu úlohy zvolte záložku **Obsah**, kam zadáte hodnoty vstupních parametrů, které budou předány do hry.

5. Po zadání všech potřebných parametrů stiskněte tlačítko **Uložit** na spodu obrazovky.

6. Pro otestování správnosti vstupních parametrů zvolte tlačítko **Otestovat úlohu** v pravém horním rohu obrazovky.

7. Následně se hra spustí a je možnost ji odehrát.

Obsah CD

Přiložené CD obsahuje jeden textový soubor a dva další adresáře. Textový soubor **README.txt** obsahuje popis adresářové struktury, která je uložena na CD. První adresář s názvem **doc** obsahuje text bakalářské práce spolu s adresářem **Latex**, který obsahuje zdrojové LaTeX soubory a obrázky použité v textu bakalářské práce. Druhý adresář (**/src**) obsahuje všechny projekty a skripty, které se vytvářely v rámci této práce.