

Metodika podrobne opisuje postup tvorby aplikácie „Arachnofóbia“.

# METODIKA TVORBY VR APLIKÁCIE

Metodika podrobne opisuje postup tvorby aplikácie „Arachnofóbia“.

Táto kapitola opisuje jednotlivé časti vývojového procesu VR aplikácie od analýzy požiadaviek, cez výber vhodných hardvérových a softvérových nástrojov, prechádza k návrhu riešenia, podrobnému postupu tvorby, používateľskému testovaniu až po spätnú väzbu (od odborníka na KBT a potencionálnych pacientov). Úlohou týchto fáz je dekomponovať problém na menšie a jednoduchšie celky. Nami navrhnutá a vyvinutá VR aplikácia sa zameriava na liečbu jednej z najrozšírenejších špecifických fóbií, podľa ktorej dostala aj svoje meno **Arachnofóbia**, čo znamená chorobný strach z pavúkov. Účelom aplikácie je zefektívniť priebeh terapie a poskytnúť nástroje na prispôsobenie celého procesu podľa konkrétnych požiadaviek pacienta alebo psychoterapeuta.

Na začiatku projektu slúžiaceho na liečbu fóbie sa bolo treba zamyslieť nad tým, ktorú z veľkého množstva fóbií by bolo vhodné liečiť prostredníctvom VR a taktiež, ktorá by sa dala adekvátne implementovať pre priestorovú formu VR, akou je napríklad aj nami zvolená platforma HTC Vive. Práve arachnofóbia sa nám javila ako fóbia, pri ktorej sa najviac ukáže sila priestorovej VR – interakcia s virtuálnym prostredím pomocou ovládačov. V prvotných myšlienkach sme uvažovali nad akrofóbiou (strach z výšok), aviofóbiou (strach z lietania), klaustrofóbiou (strach z uzavretých priestorov) a sociálnou fóbiou (strach z medziľudského kontaktu), avšak pri žiadnej z týchto fóbií sme nevideli taký potenciál interakcie, ako pri špecifickej fóbii, akou je arachnofóbia.

## Špecifikácia požiadaviek

Pred samotným vývojom aplikácie sme vykonali rozsiahlu analýzu požiadaviek, ktorá nám dala jasnú predstavu o tom, aký produkt má byť vo finále vytvorený a čo všetko je potrebné doň implementovať. Špecifikácia požiadaviek na softvér Arachnofóbia vznikla na základe:

* konzultácií s odborníkom v oblasti psychoterapie a vedúcim diplomovej práce,
* zadania záverečnej práce,
* analýzy existujúcich softvérov s podobnou funkcionalitou,
* vlastného výskumu v oblasti VR a fóbií.

Pri zostavovaní katalógu špecifických požiadaviek sme sa riadili medzinárodnou metodikou pre vývoj softvéru RUP. Zosumarizované funkcionálne a nefunkcionálne požiadavky sa nachádzajú v nasledujúcich podkapitolách. Stĺpec s názvom „Priorita” označuje zaradenie požiadavky, respektíve jej dôležitosť, pričom menšie číslo predstavuje dôležitejšie prvky. Ak pri jednotlivých požiadavkách nebude explicitne uvedené inak, tak pod pojmom „používateľ“ budeme rozumieť pacienta alebo psychoterapeuta.

### Katalóg funkcionálnych požiadaviek

V nasledovnej tabuľke 1 uvádzame zoznam funkcionálnych požiadaviek, ktoré definujú výsledné správanie a funkcionalitu vyvíjanej aplikácie.

Tabuľka 1: Funkcionálne požiadavky

|  |  |  |
| --- | --- | --- |
| Funkcionálne požiadavky (funkcie, charakteristiky, schopnosti) | | |
| **ID** | **Špecifikácia** | **Priorita** |
| #F1 | Integrácia virtuálnej reality do systému. | 1 |
| #F2 | Systém bude schopný vykresľovať virtuálne prostredie. | 1 |
| #F3 | Systém zobrazí úvodné hlavné menu, v ktorom sa nachádzajú možnosti, ktoré umožnia navigáciu do ostatných častí aplikácie. | 2 |
| #F4 | Prvou možnosťou je spustenie simulovaného prostredia. | 2 |
| #F5 | Druhou možnosťou je prehľadný opis ovládacích prvkov aplikácie. | 2 |
| #F6 | Treťou možnosťou sú základné nastavenia aplikácie. Používateľ tu má možnosť zvoliť si pohlavie (muž alebo žena) a taktiež regulovať hlasitosť v aplikácii. | 2 |
| #F7 | Štvrtou možnosťou je zobrazenie informácií o aplikácii a jej autorovi. | 2 |
| #F8 | Poslednou možnosťou je ukončenie aplikácie. | 2 |
| #F9 | Systém umožňuje súčasné využitie aplikácie pacientom aj psychoterapeutom v procese liečby. | 1 |
| #F10 | Na monitor počítača sa premieta obraz z HMD, to znamená, že to isté čo vidí pacient, vidí aj psychoterapeut. | 1 |
| #F11 | Psychoterapeut má úplnú kontrolu nad tým, čo sa odohráva v simulovanom prostredí. | 1 |
| #F12 | Aplikáciu bude môcť využívať pacient aj samostatne, bez účasti psychoterapeuta. | 1 |
| #F13 | Systém používateľovi zobrazuje čas strávený vo virtuálnom prostredí. | 2 |
| #F14 | Pacient na ovládanie prostredia bude využívať laserové ukazovátko. | 1 |
| #F15 | Aplikácia zobrazí používateľovi obrázky a fotografie pavúkov so stupňujúcim sa stresorom. Používateľ dokáže medzi nimi listovať. | 2 |
| #F16 | Systém umožní nastaviť vzdialenosť stola od pacienta. | 1 |
| #F17 | Používateľ môže pridať 3D model pavúka, ktorého systém vykreslí na stole nachádzajúcom sa pred pacientom. Rovnako môže vybrať druh a sfarbenie pavúka. | 1 |
| #F18 | Maximálny počet pavúkov vyskytujúcich sa vo virtuálnom prostredí bude obmedzený na hodnotu 30. | 2 |
| #F19 | Modely pavúkov sa budú realisticky pohybovať po stole. | 1 |
| #F20 | Používateľ dokáže odobrať 3D model pavúka z virtuálneho prostredia. | 1 |
| #F21 | Používateľ musí byť schopný upravovať parametre 3D modelov pavúkov. Konkrétne regulovať ich rýchlosť pohybu a meniť ich veľkosť. | 1 |
| #F22 | Používateľ má možnosť zmazať všetky 3D modely pavúkov z virtuálneho prostredia naraz. | 1 |
| #F23 | Používateľ dokáže skryť a opätovne zobraziť 3D model priehľadnej nádoby. | 2 |
| #F24 | Systém umožní používateľovi návrat do hlavného menu. | 2 |
| #F25 | Používateľ môže kedykoľvek ukončiť aplikáciu. | 1 |
| #F26 | Pacient bude schopný manipulovať s vybranými objektmi pomocou 3D modelov rúk. Konkrétne pôjde o: dotknutie sa objektu, chytenie a pustenie objektu. | 1 |
| #F27 | Modely rúk budú vybavené vhodnými animáciami pre jednotlivé hmatové úkony. | 1 |
| #F28 | Pacientovi sa za každý dotyk a chytenie pavúka pripíšu body, ktoré zvýšia jeho motiváciu do ďalšieho sedenia. | 1 |
| #F29 | Systém reaguje na interakciu s objektmi pavúkov pomocou vibrácie ovládača. | 2 |
| #F30 | Aplikácia prehrá zvuk tlkotu srdca pri blízkom kontakte s pavúkom. | 2 |
| #F31 | Systém umožní pacientovi voľný pohyb po virtuálnom prostredí. | 1 |
| #F32 | Systém bude obsahovať ambientný zvuk hrajúci na pozadí, ktorý podporí vnorenie pacienta do simulovaného prostredia. | 1 |

### Katalóg nefunkcionálnych požiadaviek

Pre lepšiu prehľadnosť uvádzame zoznam nefunkcionálnych požiadaviek, ktoré definujú obmedzenia alebo vlastnosti vytváranej aplikácie v nasledujúcich tabuľkách 2, 3, 4, 5, 6.

Tabuľka 2: Nefunkcionálne požiadavky na použiteľnosť

|  |  |  |
| --- | --- | --- |
| Použiteľnosť | | |
| **ID** | **Špecifikácia** | **Priorita** |
| #NP1 | Predpokladá sa, že so systémom budú pracovať najmä psychoterapeuti a ich pacienti, od ktorých sa neočakáva veľká skúsenosť s technológiou VR. Použiteľnosť systému by mala byť čo najjednoduchšia. FOV priestorových VR headsetov je väčšinou len 110°, to znamená, že do tohto rozmedzia je potrebné umiestniť všetky hlavné ovládacie prvky GUI. | 1 |
| #NP2 | Noví používatelia sa dokážu naučiť používať systém za 10 až 15 minút. | 1 |

Tabuľka 3: Nefunkcionálne požiadavky na výkon

|  |  |  |
| --- | --- | --- |
| Výkon | | |
| **ID** | **Špecifikácia** | **Priorita** |
| #NV1 | Vizuálna simulácia musí prebiehať v reálnom čase bez oneskorenia, aby sa zachovala vysoká úroveň vnorenia. To znamená fungovanie na stabilnej obnovovacej frekvencii minimálne 90 snímok za sekundu. | 1 |
| #NV2 | Systém bude schopný stabilne fungovať pri scéne, ktorú tvoria zložité 3D modely (rádovo milióny polygónov). | 1 |
| #NV3 | Objekty, ktoré sa nenachádzajú v zornom poli používateľa sa nebudú vykresľovať. | 2 |
| #NV4 | Animácie objektov, ktoré sa nenachádzajú v zornom poli používateľa sa nebudú prehrávať. | 2 |

Tabuľka 4: Nefunkcionálne požiadavky na dizajnové obmedzenia

|  |  |  |
| --- | --- | --- |
| Dizajnové obmedzenia | | |
| **ID** | **Špecifikácia** | **Priorita** |
| #ND1 | Systém bude vyvíjaný pomocou herného enginu Unity 3D. | 1 |
| #ND2 | Systém bude implementovaný v jazyku C#. | 1 |
| #ND3 | Systém bude využívať knižnicu VIVE Input Utility (VIU). | 1 |
| #ND4 | K plnohodnotnému využitiu systému je potrebné použiť HMD s podporou priestorovej VR a dvojicu bezdrôtových ovládačov. | 1 |

Tabuľka 5: Nefunkcionálne požiadavky na rozhrania

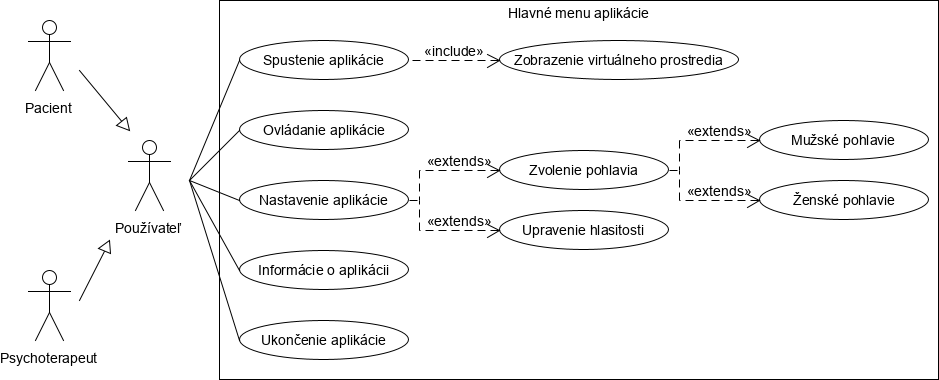
|  |  |  |
| --- | --- | --- |
| Rozhrania | | |
| **ID** | **Špecifikácia** | **Priorita** |
| #NR1 | Chod aplikácie má byť bezproblémový na výkonných počítačoch s dedikovanou grafickou kartou, respektíve na počítačoch s označením VR Ready. | 1 |
| #NR2 | Systém bude navrhnutý primárne pre zariadenie HTC Vive (vrátane verzie Pro), ale bude ľahko prenositeľný aj na iné priestorové typy VR headsetov. | 1 |
| #NR3 | Zariadenie HTC Vive na svoje fungovanie vyžaduje mať nainštalované potrebné ovládače pre operačný systém. Pri káblovom prepojení je nutný 1x USB a 1x HDMI port na počítači. V prípade bezdrôtového prepojenia pomocou adaptéra je nevyhnutný interný PCI-e slot. | 1 |
| #NR4 | Aplikácia bude spustiteľná na počítačoch s operačným systémom Microsoft Windows 7, 8, 8.1 a 10. | 1 |
| #NR5 | Systém bude pozostávať z dvoch používateľských rozhraní. Jedno bude slúžiť pre pacienta a druhé pre psychoterapeuta. | 1 |
| #NR6 | Používateľské rozhranie sa bude nachádzať v hlavnom menu a taktiež aj vo virtuálnom prostredí. | 2 |
| #NR7 | Aplikácia má mať jednoduché, prehľadné a používateľsky prijateľné rozhranie. | 2 |
| #NR8 | Text zobrazený v aplikácii musí byť pre používateľa viditeľný a ľahko čitateľný. | 1 |
| #NR9 | Aplikácia bude dostupná v slovenskom jazyku. | 2 |
| #NR10 | Interakcia a manipulácia so systémom musí byť intuitívna a ľahko ovládateľná. | 1 |
| #NR11 | Pacient bude schopný interagovať so systémom prostredníctvom pohybov tela, konkrétne hlavy a párom bezdrôtových ovládačov. | 1 |
| #NR12 | Psychoterapeut bude schopný interagovať so systémom prostredníctvom klasickej klávesnice a myši. | 1 |

Tabuľka 6: Nefunkcionálne požiadavky na bezpečnosť

|  |  |  |
| --- | --- | --- |
| Bezpečnosť | | |
| **ID** | **Špecifikácia** | **Priorita** |
| #NB1 | Pri používaní priestorovej VR musí byť v miestnosti voľný priestor o veľkosti najmenej 2 . V tejto oblasti by sa nemali nachádzať žiadne objekty, ktoré by mohli byť prekážkou pre pacienta. Tieto prekážky by mohli spôsobiť zranenie pacienta alebo poškodiť ovládače, prípadne HMD. | 1 |
| #NB2 | Dlhodobé používanie HMD môže u pacienta vyvolať závraty alebo nevoľnosť z pohybu, preto sa odporúča robiť prestávky po 50 minútach. | 2 |

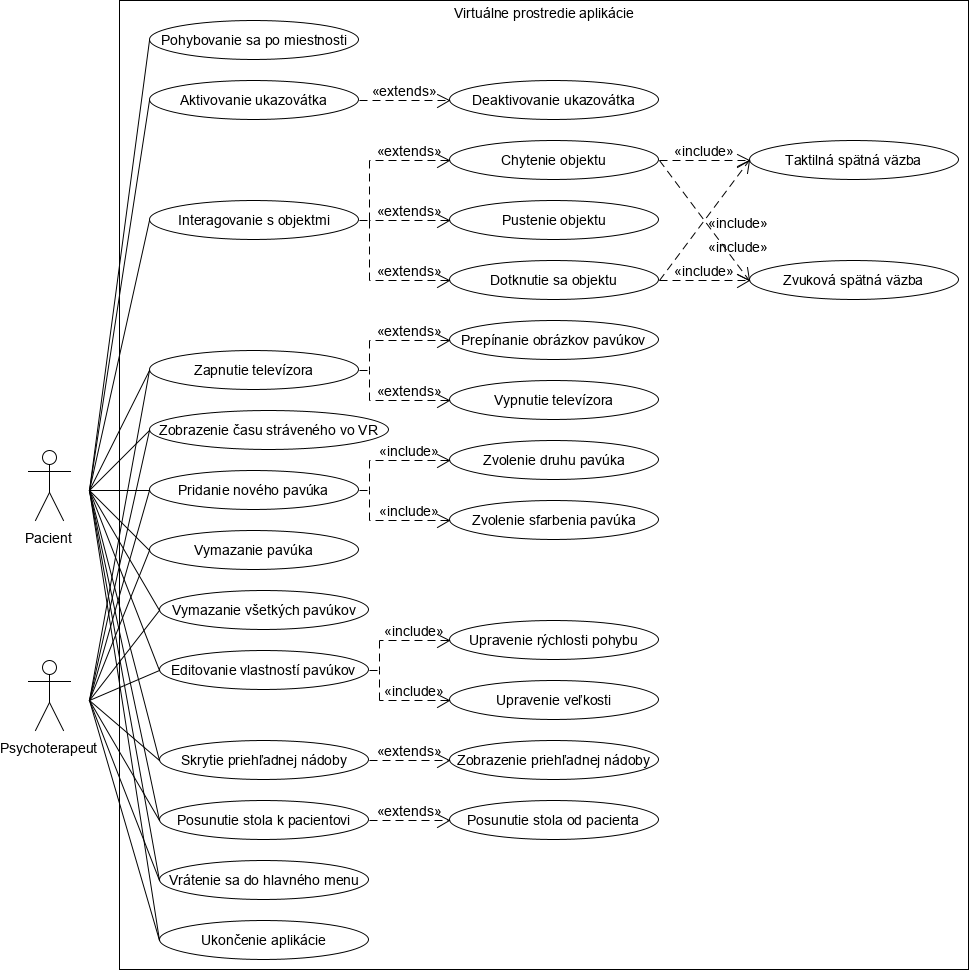
### Prípady použitia aplikácie

Na základe vyššie stanovených kritérií a požiadaviek sme pristúpili k modelovaniu systému prostredníctvom jazyka UML, výsledkom čoho sú diagramy prípadov použitia. Tieto diagramy opisujú správanie aplikácie a dostupné možnosti pre jednotlivých aktérov, ktorí v systéme vystupujú. V našom prípade rozlišujeme medzi dvoma aktérmi – pacientom a psychoterapeutom.

Ako môžeme vidieť z obrázku 3, úvodné hlavné menu bude poskytovať navigáciu vo forme tlačidiel, ktoré používateľa presmerujú do iných častí aplikácie. Používateľ bude mať na výber z piatich možností: spustenie aplikácie (načítanie scény), zobrazenie ovládacích prvkov, nastavenia (voľba pohlavia, zmena hlasitosti), informácie a prosté ukončenie aplikácie.

Obrázok 3: Diagram prípadov použitia hlavného menu aplikácie

Kľúčovú časť aplikácie tvorí virtuálne prostredie, ktorého funkcionality sú znázornené na obrázku 4. Komplexné prípady použitia s definovanými scenármi v tejto záverečnej práci neuvádzame, nakoľko je to pre nás irelevantné. Naším cieľom bolo skôr vytvoriť nadhľad nad vyvíjanou aplikáciou.



Obrázok 4: Diagram prípadov použitia virtuálneho prostredia aplikácie

## Použité technológie

Táto podkapitola slúži ako prehľad hardvérových a softvérových prostriedkov využitých pri vývoji výslednej VR aplikácie a zahŕňa taktiež motiváciu pre ich použitie. Výber jednotlivých technológií bol realizovaný na základe špecifických požiadaviek uvedených v kapitole 3.1.

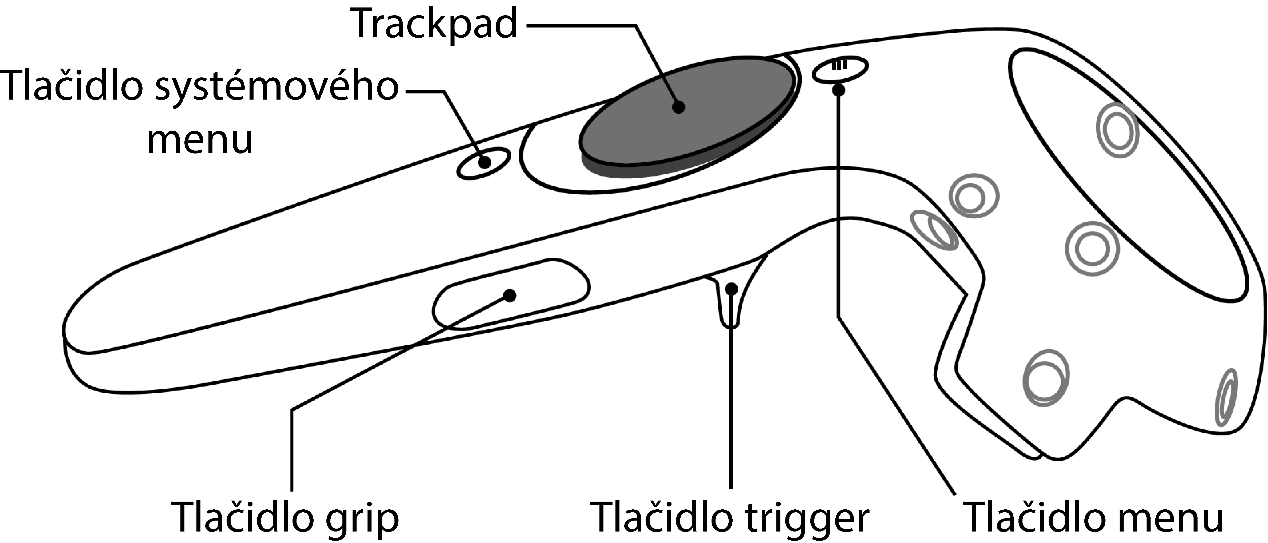
### HTC Vive

HTC Vive je platforma virtuálnej reality vyvinutá zo spolupráce taiwanskej technologickej spoločnosti HTC a hernej spoločnosti Valve, ktorá ponúka používateľom prístup k širokej škále VR aplikácií, videí a hier prostredníctvom programu **Steam** alebo predplatiteľskej služby **Viveport** **Infinity**. Táto platforma bola vydaná 5. apríla 2016. Headset obsahuje dvojicu OLED panelov s celkovým rozlíšením 2160 × 1200 px (teda 1080 × 1200 px na jedno oko), 90Hz obnovovacou frekvenciou a 110° zorným poľom. Okrem iného má taktiež zabudovaný mikrofón a prednú kameru, ktorá umožňuje pohľad na reálny svet bez potreby skladania dátovej prilby. Súčasťou sady HTC Vive je dvojica vysielačov s názvom **Lighthouse** (viď obrázok 5). Tieto vysielače sledujú polohu a orientáciu dátovej prilby v priestore. Montujú sa na steny vo výške približne 2 až 3 metre vo vzájomnej vzdialenosti maximálne 5 metrov. Vysielače vyžarujú infračervené lúče (60 pulzov za sekundu), ktoré sú zachytené jednotlivými senzormi osadenými na prednej strane headsetu a na prstencoch ovládačov. Presnosť snímania pozície je menšia než jeden milimeter a presnosť rotácie je vďaka kombinácii údajov z gyroskopu a akcelerometru asi jedna desatina stupňa.



Obrázok 5: Systém virtuálnej reality HTC Vive [23]

V sade nájdeme aj dva bezdrôtové ovládače so zabudovanými senzormi, ktoré sú rovnako sledované vysielačmi – ich pozícia a rotácia v priestore je prenášaná do počítača. Každý z ovládačov disponuje hmatovou spätnou väzbou vo forme vibrácií, štyrmi tlačidlami a jedným trackpadom (v ďalšom texte budeme používať ich anglické pomenovania, nakoľko sú bežne využívané v oblasti VR).



Obrázok 6: Opis tlačidiel a fotosenzorov ovládača zo sady HTC Vive

**Trackpad** predstavuje dotykovú oblasť v podobe kruhu, ktorá je umiestnená v hornej časti ovládača (pozri obrázok 6). Trackpad môžeme označiť ako 3DoF (z angl. Three Degrees of Freedom), pretože sníma pozíciu prsta v dvoch osiach a zároveň umožňuje jeho stlačenie (môžeme ho použiť ako tlačidlo). Tlačidlo **trigger** sa nachádza v spodnej časti ovládača a slúži ako spúšť. Má 1DoF (z angl. One Degree of Freedom), nakoľko zaznamenáva silu stlačenia. Zvyšné tlačidlá **menu**, **systémové** **menu** a **grip** nie sú ničím výnimočné – rozlišujú len dva stavy stlačené alebo uvoľnené. Vo všeobecnosti má ovládač 6DoF (z angl. Six Degrees of Freedom), čo poskytuje zaujímavé možnosti využitia.

Zariadenie HTC Vive bolo vybrané na základe zadania záverečnej práce a taktiež z dôvodu dostupnosti tejto platformy v priestoroch katedry informatiky FPV UMB, vďaka čomu sme si nemuseli zaobstarať vlastný VR headset.

### Unity 3D

Unity 3D je komplexný multiplatformový 2D a 3D herný engine vyvíjaný spoločnosťou Unity Technologies, ktorý sa používa nielen na vývoj hier, ale taktiež na tvorbu simulácií v reálnom čase a náučných aplikácií. Obrovské využitie má aj pri návrhu automobilov, architektonických vizualizácií či vo filmovom priemysle. Patrí medzi najpoužívanejšie herné enginy na svete. Jeho prvá verzia bola vydaná v roku 2005, avšak od tej doby sa toto softvérové prostredie rozšírilo o množstvo užitočných funkcií. Dnes sa môže pýšiť tým, že podporuje celkovo 29 platforiem vrátane VR, AR a MR (možnosť využitia rozličných SDK). Výraznou prednosťou v porovnaní s inými vývojovými prostrediami je jeho rýchlosť, jednoduchosť a prehľadnosť. Princíp Unity spočíva vo vytváraní aplikácií pre rôzne platformy prostredníctvom jediného projektu a teoreticky bez nutnosti prepisovania skriptov. Samozrejme isté úpravy sú potrebné, napríklad pre rôzne vstupné periférie je ovládanie nutné mapovať inak. Unity editor má aj vstavaný obchod nazývaný *Asset Store* s množstvom bezplatných, ale aj platených doplnkov (3D modelov, animácií, audio súborov, pluginov a podobne), ktoré je možné stiahnuť a okamžite použiť. Vďaka obrovskej komunite disponuje prepracovanou dokumentáciou, výukovými videami a diskusným fórom, kde používatelia zdieľajú svoje poznatky a inšpiráciu. Okrem fóra ponúka taktiež službu *Answers*, ktorá slúži na zodpovedanie otázok začínajúcich ako aj skúsených vývojárov ohľadom ich projektov. Unity je dostupné pre všetky desktopové operačné systémy a je možné ho zaobstarať vo verziách pre jednotlivca (Student, Personal) alebo pre firmy (Plus, Pro, Enterprise) priamo z ich oficiálnych stránok. Verzie pre jednotlivcov sú dostupné zdarma, pričom študenti sa musia preukázať pomocou *GitHub Student Developer Pack* – na jeho získanie je potrebný prístup na školský email. Verzia Personal obsahuje všetky hlavné funkcionality, to znamená, že je vhodná pre väčšinu používateľov, ktorých zisk z projektu je menší než 100 000 dolárov. Ostatné verzie Plus, Pro a Enterprise zahŕňajú niektoré prémiové služby a funkcie, ktoré sú však platené. Na vývoj našej VR aplikácie sme použili verziu Unity 2019.1.14f1 Personal (64-bit).

Okrem toho, že autor tejto práce mal predchádzajúce skúsenosti s vývojom aplikácií v Unity z bakalárskej práce [1] a z činnosti na vlastných projektoch [3] bol tento herný engine vybraný pre jeho podporu všetkých aktuálne dostupných VR zariadení a taktiež preto, lebo viac ako 60 % celkového VR/AR obsahu sa tvorí práve v ňom.

### Microsoft Visual Studio

V hernom engine Unity 3D je možnosť programovania logiky v jazykoch C# a JavaScript (prezývaný ako UnityScript – veľmi sa odlišuje od JavaScriptu, ktorý sa používa vo webovom vývoji). V minulosti bol taktiež podporovaný jazyk Boo. Pre riešenie nášho projektu sme zvolili programovací jazyk C#. Ide o jednoduchý, moderný, objektovo orientovaný a silne typovaný jazyk. To znamená, že jeho kód je dostatočne prehľadný. Syntax C# je podobná jazykom C, C++ a Java, nakoľko jeho korene pochádzajú z rodiny jazykov C. Dôvodom výberu tohto jazyka sú doterajšie skúsenosti autora s programovaním v tomto jazyku a taktiež fakt, že C# neexistuje len v rámci enginu Unity, teda jeho dokumentácia a podpora je oveľa robustnejšia než pre UnityScript. Ďalšou výhodou C# je to, že od roku 2017 obchod *Asset Store* dovoľuje publikovať doplnky len v tomto jazyku, takže z dlhodobého hľadiska na zaistenie kompatibility projektu je lepšie programovať práve v ňom. Samotný vývoj VR aplikácie sme realizovali na počítači s nainštalovaným operačným systémom Microsoft Windows 10, preto sme na kódovanie použili vývojové prostredie Microsoft Visual Studio 2019, ktoré podporuje jazyk C#. Inštaláciu IDE Microsoft Visual Studio 2019 sme vykonali v rámci inštalácie Unity 3D.

### Blender

Blender je grafický softvér s otvoreným zdrojovým kódom, ktorý ponúka profesionálne nástroje na tvorbu trojrozmerných počítačových modelov a animácií za nulovú cenu. Vyvíjaný je neziskovou organizáciou Blender Foundation v spolupráci s komunitou. V posledných rokoch je Blender vnímaný ako silná alternatíva k plateným softvérom slúžiacim na podobné účely, ako napríklad Autodesk Maya alebo Autodesk 3ds Max. Špecializuje sa na modelovanie, UV mapovanie, rigging postáv, animácie, fotorealistické renderovanie (s výpočtom na CPU aj GPU), vizuálne efekty, možnosť simulácie, kompozície, maskovania, ale zvládne taktiež sledovanie pohybu (kamery a objektov), či úpravu videa. Dokonca disponuje aj integrovaným herným enginom s podporou VR a programovacím jazykom Python, ktorý slúži na tvorbu vlastných skriptov a doplnkov. Tento herný engine sa v nových verziách Blenderu (od verzie 2.80) v jeho jadre nenachádza a jeho tvorcovia odporúčajú namiesto neho použiť konkurenčné voľne dostupné enginy. Dôvodom odstránenia bola jeho zastaranosť. Blender je vhodný pre všetkých – od jednotlivcov (umelcov, profesionálov, nadšencov, vedcov, študentov, modelárov) až po animačné štúdia. Nakoľko je dostupný zdarma, jeho komunita je veľmi široká a preto sa na webe nachádza množstvo tutoriálov, diskusných fór, či podrobná dokumentácia. Medzi jeho pozitívne vlastnosti patrí multiplatformovosť (funguje na operačných systémoch Windows, Linux aj macOS), možnosť exportu 3D modelov do viacerých formátov a prispôsobiteľné GUI. Na modelovanie dodatočných a úpravu pôvodných 3D modelov pre našu VR aplikáciu sme použili Blender vo verzii 2.80.

Zvolený bol hlavne kvôli predošlým poznatkom s prácou v tomto modelovacom softvéri počas štúdia – bakalárska práca zameraná na túto tému [1] a z vedeckej činnosti [2]. Ďalším faktorom pre voľbu tohto nástroja bola jeho licenčná politika.

### Adobe Photoshop

Photoshop je komerčný rastrový grafický editor slúžiaci primárne na tvorbu a úpravu rastrovej grafiky (obrázkov, fotografií) vyvinutý spoločnosťou Adobe Systems. Je k dispozícii pre operačné systémy Windows a macOS. V našom prípade sme tento softvér použili na tvorbu a úpravu textúr pre jednotlivé 3D modely vo verzii Adobe Photoshop CC 2019. Voľba tohto softvéru je opodstatnená ponúkanými nástrojmi a existenciou mnohých video tutoriálov.

### Adobe Illustrator

Illustrator je komerčný vektorový grafický editor slúžiaci primárne na tvorbu a úpravu vektorovej grafiky (digitálnej grafiky, ilustrácií, typografie) vyvinutý a publikovaný firmou Adobe Systems. Program je dostupný pre operačné systémy Windows a macOS. Pre náš projekt sme ho využili na tvorbu grafických prvkov, návrh loga aplikácie a rôznych tlačidiel vo verzii Adobe Illustrator CC 2019. K voľbe tohto programu sme pristúpili na základe vhodných nástrojov a množstva online návodov od profesionálov.

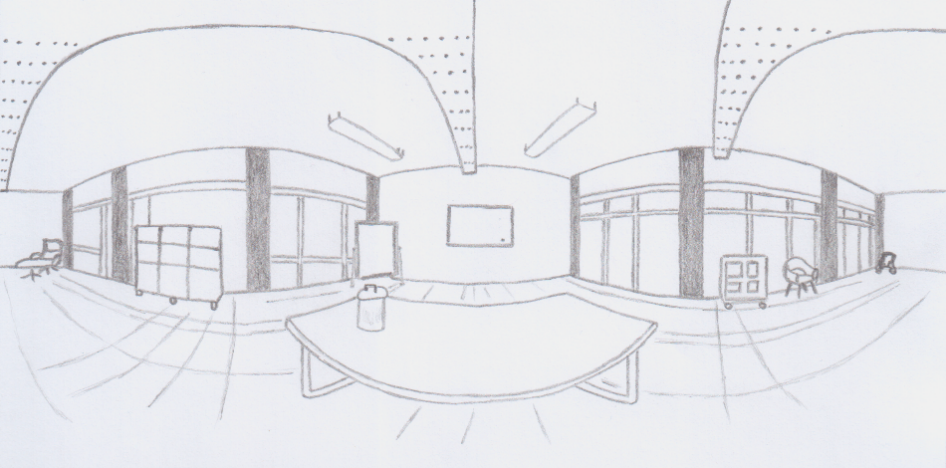
### Audacity

Audacity je bezplatný multiplatformový zvukový editor a rekordér s možnosťou viacstopového záznamu. Bol vytvorený v roku 1999 Dominikom Mazzonim a Rogerom Dannenbergom na univerzite Carnegie Mellon. Medzi jeho hlavné výhody patrí jeho jednoduchosť a dostupnosť na viacerých platformách ako Windows, GNU/Linux, macOS a ďalších. Audacity verzie 2.3.2 sme v našom VR projekte využívali na digitálnu úpravu a strih zvukových súborov. Dôvodom použitia tohto nástroja sú vyššie spomenuté vlastnosti a naša predchádzajúca skúsenosť.

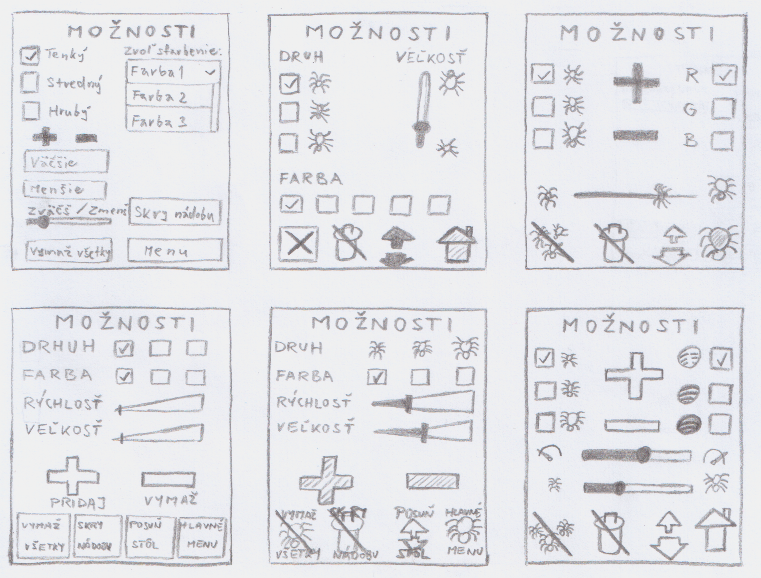
## Návrh riešenia

Po určení požiadaviek a výbere vhodných technológií na vývoj softvéru sme pristúpili k návrhu riešenia. Počas procesu navrhovania sme sa snažili splniť všetky stanovené požiadavky. Brali sme do úvahy potreby oboch aktérov – pacientov aj psychoterapeutov. Takýmto spôsobom sme navrhli intuitívne rozhranie a interakcie, ktorých úlohou je riešiť potreby cieľovej skupiny. Na dosiahnutie tohto cieľa sme naplno využívali výhody technológie VR. Taktiež samotný návrh v porovnaní s návrhom pre klasické desktopové alebo mobilné aplikácie prebiehal inak. Úlohou našej aplikácie je pomáhať ľuďom trpiacim neopodstatneným strachom z pavúkov. Pri návrhu vlastnej aplikácie sme sa inšpirovali existujúcimi riešeniami s podobným zameraním. Dôraz sme kládli na jednoduchosť a ľahkú ovládateľnosť, tak aby aplikácia bola prístupná a použiteľná pre všetkých. Táto vlastnosť bola dôležitá z toho dôvodu, aby nebolo potrebné zdĺhavé zaškoľovanie na ovládanie aplikácie.

Aplikáciu sme navrhli tak, aby ju dokázal ovládať nielen pacient, ale aj psychoterapeut. Takýmto spôsobom má psychoterapeut celý priebeh liečby pod kontrolou. V prípade pacienta je interaktivita zabezpečená dvojicou bezdrôtových ovládačov, ktorých pohyb sa do počítača prenáša prostredníctvom infračervených lúčov vychádzajúcich zo snímačov. Psychoterapeut dokáže do aplikácie zasahovať pomocou klávesnice a myši. Z tohto vyplynula potreba dvojakého rozhrania. **Dvojdimenzionálne rozhranie** (angl. non-diegetic UI) sa zobrazuje len na monitore počítača. Slúži psychoterapeutovi na ovládanie prvkov prostredia podľa vlastného uváženia a reakcií pacienta. **Trojdimenzionálne rozhranie** (angl. diegetic UI) majú možnosť vidieť obaja, avšak len pacient môže prostredníctvom neho modifikovať objekty prostredia.

Návrh aplikácie prebiehal vo viacerých fázach. Naše prvotné myšlienky sme načrtli na papier. Začali sme s návrhom prostredia miestnosti, ktorá pripomína psychoterapeutickú kanceláriu. Jej úlohou je navodiť pokojnú atmosféru a eliminovať stres. Vhodnou pomôckou pri skicovaní tohto drôtového modelu (angl. wireframe) bola ekvidištančná mriežka (angl. equirectangular grid), ktorá zachytáva celú 360 stupňovú panorámu miestnosti. Na obrázku 7 je zobrazená skica tejto miestnosti spolu s rozmiestnením hlavných elementov, ktoré kvôli obmedzenosti FOV dnešných VR headsetov museli byť situované v rozpätí 110°. Jednotlivé stavebné bloky (3D modely), ktoré spĺňali naše kritériá sme získali z obchodu *Asset Store* [24] a z voľne dostupného projektu na portáli GitHub [25].

Obrázok 7: Ekvidištančná skica prostredia kancelárie

Nasledoval návrh vzhľadu používateľského rozhrania, ktoré sa viackrát menilo, kým sme prišli k finálnej podobe rozloženia jednotlivých prvkov (pozri obrázok 8). Prvý dôvod na zmenu bolo nedodržanie nefunkcionálnej požiadavky na rozhranie s identifikátorom #NR7, ktorá hovorí o jeho jednoduchosti a prehľadnosti. Pôvodné textové tlačidlá boli nahradené výstižnými obrázkovými ikonami. Druhým dôvodom bolo pridanie nových funkcií.

Obrázok 8: Návrhy vzhľadu GUI pre pacienta

Grafické používateľské rozhranie pre pacienta je začlenené do virtuálneho prostredia tak, aby žiadnym spôsobom nenarúšalo jeho zážitok. Dosiahli sme to pomocou **3D modelu tabule**, do ktorej sme umiestnili jednotlivé komponenty GUI. Nachádzajú sa tu ikony, zaškrtávacie tlačidlá a posuvníky. Tieto prvky slúžia na pridávanie rôznych druhov pavúkov na stôl, výber ich sfarbenia, odoberanie, zväčšovanie, zmenšovanie, menenie rýchlosti pohybu, posúvanie stola k pacientovi a od pacienta, skrytie a následné zobrazenie priehľadnej nádoby, návrat do hlavného menu alebo v prípade impulzívnej reakcie pacienta, je tu možnosť odstránenia všetkých pavúkov naraz.

Ďalším prvkom GUI je **televízor**, nachádzajúci sa priamo oproti pacientovi, ktorý slúži na počiatočnú fázu terapie. Na obrazovke televízora sa pacientovi zobrazujú obrázky pavúkov, od malého kresleného pavúka až po skutočnú fotografiu tarantuly. Jednotlivé úrovne sa používajú preto, aby sme eliminovali nežiadúce, či prehnané reakcie pacienta. Vtipným začiatkom terapie povzbudíme pacienta, aby sa nebál pokračovať do ďalších úrovní. Posledný prvok GUI tvoria **digitálne hodinky** na ľavej ruke pacienta, ktoré zobrazujú uplynulý čas od začiatku expozície vo formáte HH:MM:SS. Samotný koncept aplikácie bol navrhnutý tak, aby sa pacient cítil pohodlne v oboch polohách – v sede či v stoji.

Každý model pavúka má priradenú náhodnú animáciu pomocou stavového automatu v systéme **Mecanim** a príslušného skriptu. Náhodným generovaním sme zabezpečili, aby sa každý pavúk správal v danom okamihu inak, čím sme dosiahli väčšiu hodnovernosť simulácie. Tieto modely pavúkov, ako aj priehľadná nádoba, sú pre pacienta ľahko uchopiteľné, čo mu umožňuje skúmať a pozorovať dané objekty z viacerých uhlov pohľadu.

Našu aplikáciu zatraktívňujú aj prvky **gamifikácie**, ktoré ju posúvajú na vyššiu úroveň. Herné prvky sme sa rozhodli zakomponovať, aby bol proces liečby zaujímavejší, interaktívnejší a pútavejší. Ďalším dôvodom je príťažlivejšia forma liečenia, podpora aktívnej účasti a zapojenia, čo umožňuje priamejšiu a okamžitejšiu spätnú väzbu. Informácia o tom, ako sa pacientovi počas terapie darilo (počet dotknutých a chytených pavúkov) ho motivuje k postupnému zlepšovaniu, čím dospeje k svojmu cieľu zbaviť sa neopodstatneného strachu.

Vizuálnu stránku aplikácie dopĺňajú **zvukové efekty** (hudba hrajúca v hlavnom menu, ruch mesta, zvuk pavúkov, tlkot srdca, zapnutie/vypnutie televízora a zvuky tlačidiel), ktoré prispievajú k lepšiemu vnoreniu pacienta do virtuálneho prostredia, čo následne pomáha zvýšiť účinnosť liečby. Zmienené zvuky sme zadovážili s bezplatnou licenciou z webových stránok [freesound.org](https://freesound.org/), [soundcloud.com](https://soundcloud.com/) a [soundbible.com](http://soundbible.com/).

### Návrh scenára

V tejto časti je opísaný štandardný liečebný (psychoterapeutický) postup pre použitie našej aplikácie, ktorý pozostáva z nasledovných deviatich fáz:

1. **Fáza terapie** – prezeranie obrázkov a fotografií pavúkov na televízore so stupňujúcim sa stresorom (od roztomilého kresleného pavúka až po reálnu fotografiu tarantuly).
2. **Fáza terapie** – pozorovanie pavúka umiestnenom na stole v priehľadnej nádobe vo vzdialenosti dva metre od pacienta.
3. **Fáza terapie** – pozorovanie pavúkov pohybujúcich sa po stole vo vzdialenosti dva metre od pacienta.
4. **Fáza terapie** – pozorovanie pavúkov pohybujúcich sa po stole vo vzdialenosti jeden meter od pacienta.
5. **Fáza terapie** – pozorovanie pavúkov pohybujúcich sa po stole vo vzdialenosti pol metra od pacienta.
6. **Fáza terapie** – zatvorenie pavúka do priehľadnej nádoby.
7. **Fáza terapie** – dotknutie sa pavúka jedným prstom.
8. **Fáza terapie** – chytenie pavúka do ruky.
9. **Fáza terapie** – chytenie dvoch pavúkov do rúk (jedného do jednej ruky, druhého do druhej).

Tieto kroky terapie však silne závisia od konkrétneho pacienta. Naša aplikácia predstavuje nástroj na odstupňovanú expozičnú terapiu virtuálnou realitou (VRET). Jednotlivé fázy sa následne opakujú najskôr s malým pavúkom, neskôr so stredným a nakoniec s veľkým pavúkom. Okrem veľkosti si používateľ vie nastavovať aj iné parametre simulácie ako napríklad druh pavúkov, ich sfarbenie a podobne.

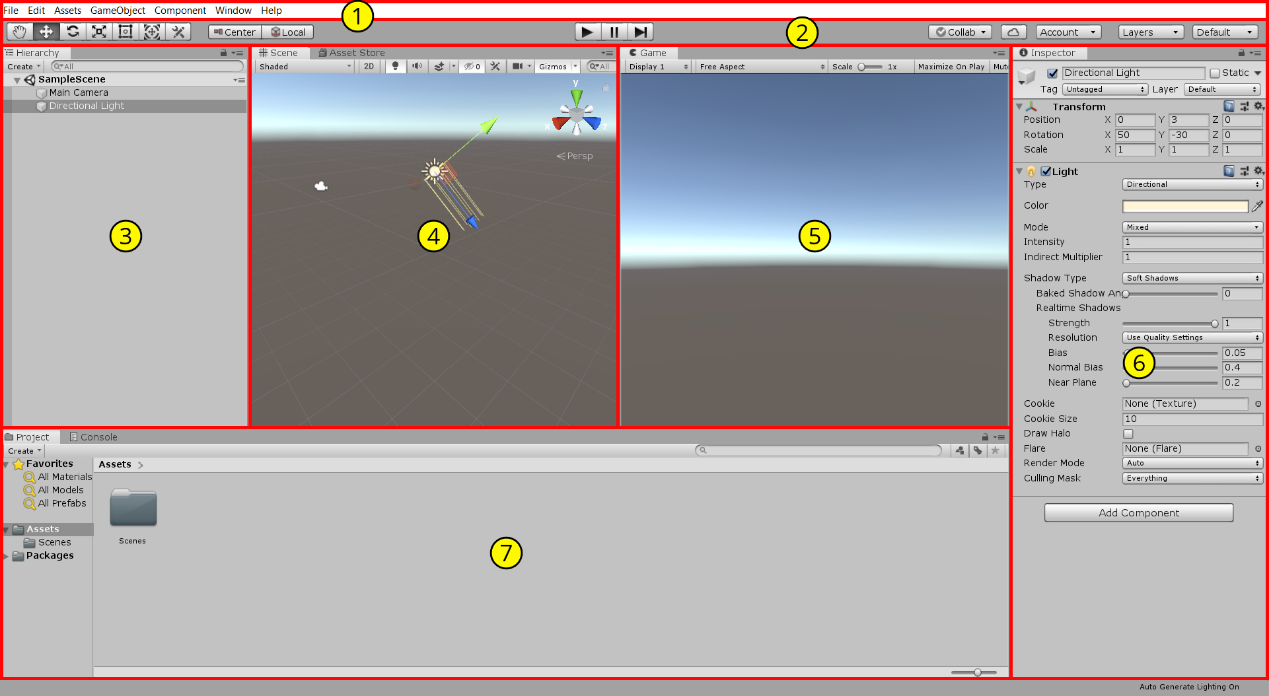
## Implementácia riešenia

Pri realizácii praktickej časti práce sme vychádzali z poznatkov opísaných v predchádzajúcich kapitolách, najmä z návrhu riešenia, ktorý zachytáva kapitola 3.3. Implementácia riešenia poskytuje podrobný pohľad na vývoj hlavných funkcionalít VR aplikácie Arachnofóbia, ktorej finálna verzia pozostáva z 3 scén. Prvá scéna tvorí hlavné menu aplikácie, druhá predstavuje pánsku a tretia dámsku verziu. Jednotlivé verzie sa od seba odlišujú len vizuálnou reprezentáciou rúk. Počas tvorby aplikácie sme využili viacero softvérových produktov, avšak jej jadro beží na hernom engine Unity 3D vo verzii 2019.1.14f1 (64-bit) s využitím knižnice VIVE Input Utility. Modelovanie doplnkových 3D modelov prebehlo v softvéri Blender 2.80, úprava textúr sa realizovala v Adobe Photoshop CC 2019, tvorba grafiky, tlačidiel v Adobe Illustrator CC 2019 a editácia, či strih zvuku v Audacity 2.3.2. Na programovanie aplikačnej logiky sme zvolili objektovo orientovaný jazyk C# a vývojové prostredie Microsoft Visual Studio 2019. Celkovú logiku projektu riadi 16 vzájomne prepojených skriptov, ktoré sú priradené určitým objektom v scéne. Ide o skripty na náhodný pohyb pavúka, vyhýbanie sa prekážkam, riadenie animácií, interakciu s objektmi, zobrazovanie času stráveného vo virtuálnom prostredí, ovládanie prvkov GUI a podobne. Pre niektoré z nich uvádzame aj ukážky kódov, ktoré boli pre jednoduchšiu čitateľnosť upravené. Výslednú VR aplikáciu sme zostavili do spustiteľného \*.exe súboru pre operačný systém Windows, ale rovnako je ju možné vybudovať aj pre platformu Windows Mixed Reality (Universal Windows Platform). Vytváranie a testovanie aplikácie prebiehalo na výkonnom počítači s pripojenou sadou HTC Vive.

### Zoznámenie sa s prostredím Unity 3D

Prostredníctvom nástroja na správu jednotlivých Unity projektov a inštalácií – **Unity Hub** vytvoríme nový projekt tlačidlom *New*, kde zvolíme 3D šablónu, názov projektu a umiestnenie projektu na disku.

Po kliknutí na *Create* sa zobrazí prostredie herného enginu Unity 3D, ktoré sa volá **Unity editor**. Tento editor slúži na vývoj aplikácií a hier pomocou grafických prvkov. Ako môžeme vidieť na obrázku 9, Unity editor pozostáva z niekoľkých hlavných častí, respektíve okien, pričom každé okno slúži na nejaký konkrétny účel.

Obrázok 9: Rozloženie prvkov GUI herného enginu Unity 3D

1. **Hlavné menu** – ponúka niekoľko položiek, ktoré slúžia ako rýchly prístup k základným operáciám vykonávaným v programe. Najčastejšie používané operácie však vieme ovládať aj pomocou klávesových skratiek.
2. **Panel nástrojov** – poskytuje nástroje na presúvanie, otáčanie a zmenu mierky objektov. V tejto časti sa nachádza aj ikona šípky *Play*, ktorou vieme spustiť aplikáciu alebo hru priamo v Unity editore.
3. **Okno hierarchia** – obsahuje zoznam všetkých objektov nachádzajúcich sa v scéne. Po kliknutí na ľubovoľný objekt sa daný objekt zvýrazní v pohľade scény.
4. **Pohľad scény** – je trojrozmerný priestor[[1]](#footnote-1), v ktorom sa nachádza grafická reprezentácia objektov z okna hierarchie (ich pozícia, rotácia a veľkosť). Scéna vždy obsahuje minimálne jednu kameru a osvetlenie. Ostatné objekty ako napríklad 3D modely, časticové systémy, plátna a podobne sú dobrovoľné. V scéne dokážeme taktiež vymodelovať jednoduché 3D modely alebo terén. Výhodou tohto pohľadu je jednoduchá orientácia v scéne a možnosť vnímania priestorových vzťahov objektov.
5. **Pohľad hry** – ide o záber, ktorý v reálnom čase zachytáva kamera. Kliknutím na ikonu *Play* v paneli nástrojov sa aplikácia spustí v tomto pohľade. Takýmto spôsobom môžeme testovať funkčnosť našej aplikácie ešte pred jej exportom na konkrétnu platformu.
6. **Okno inšpektor** – ak označíme niektorý z objektov (v hierarchii alebo v pohľade scény) zobrazí sa nám tu množstvo informácií – jeho názov, tag, vrstva, v ktorej sa nachádza a statickosť objektu. Nižšie je uvedený zoznam všetkých jeho komponentov a vlastností, pričom každý komponent slúži na niečo iné. Každý objekt musí mať minimálne jeden komponent a tým je Transform, ktorý definuje jeho pozíciu, rotáciu a veľkosť v priestore.
7. **Okno projekt** – zobrazuje adresárovú štruktúru celého projektu, v ktorej sa nachádzajú všetky potrebné súbory pre našu aplikáciu – takzvané assety. Slovo asset predstavuje súbory ako sú 3D modely, materiály, textúry, skripty, audio súbory, animácie a iné. V okne projektu môžeme kliknutím pravým tlačidlom myši vytvárať nové adresáre alebo assety. Taktiež máme možnosť premiestňovať sem súbory z disku metódou *drag and drop*.

### Prvotné nastavenie projektu

V počiatočnej fáze vývoja interaktívnej aplikácie pre platformu HTC Vive, ale podobne aj pri iných VR platformách, musíme vykonať niekoľko základných nastavení, ktoré sú špecifické práve pre virtuálno-realitné systémy. Presunieme sa preto do nastavení projektu cez položku hlavného menu *Edit* > *Project Settings*, kde zvolíme kategóriu *Player* a v paneli *XR Settings* aktivujeme podporu pre virtuálnu realitu. Následne OpenVR SDK prehodíme na začiatok zoznamu a *Stereo Rendering Mode* zmeníme na možnosť Single Pass, ktorá poskytuje súčasné vykresľovanie obrazu pre ľavé i pravé oko a tým pádom aj kvalitnejší výsledok.

Integráciu zostavy HTC Vive do prostredia Unity nám nezabezpečuje SteamVR SDK, ale namiesto neho sme sa rozhodli použiť knižnicu **VIVE Input Utility**(ďalej len VIU), ktorú vytvorila spoločnosť ViveSoftware. Cieľom tejto knižnice je urýchliť a uľahčiť vývoj programátorom pomocou predpripravených nástrojov, ktoré stačí jednoducho importovať a použiť. Tým pádom nemusíme strácať čas písaním redundantného kódu, napríklad na správu vstupných zariadení, ale môžeme sa naplno venovať hlavným funkcionalitám aplikácie. VIU navyše poskytuje implementáciu ovládania kamery, pohybu v priestore, laserového ukazovátka, rozširuje systém používateľského rozhrania v Unity a mnoho iného. Hlavnou z výhod knižnice je podpora viacerých SDK (vrátane SteamVR, Oculus) alebo inak povedané podpora viacerých VR, ale aj MR platforiem ako napríklad:

* HTC Vive, Vive Pro, Vive Cosmos (zariadenia kompatibilné s OpenVR SDK),
* HTC Vive Focus, Focus Plus (zariadenia kompatibilné s WaveVR SDK),
* Oculus Rift, Oculus Touch, Oculus Rift S, Oculus Go, Oculus Quest,
* Daydream,
* MR zariadenia od spoločnosti Microsoft.

Obsahuje taktiež VR simulátor, s ktorým dokážeme v obmedzenej miere ovládať našu aplikáciu bez použitia VR headsetu a ovládačov, iba prostredníctvom klávesnice a myši. Vďaka týmto predpokladom zabezpečíme multiplatformovosť aplikácie, napríklad ak by sme sa v budúcnosti rozhodli zacieliť aj na iné platformy.

Knižnicu VIU môžeme získať dvoma spôsobmi. Prvý spôsob, ten jednoduchší je prostredníctvom vstavaného obchodu s assetmi, ktorý sa nachádza priamo v Unity editore – *Asset Store* [26]. Druhým spôsobom je stiahnutie najnovšej verzie pomocou repozitára z GitHub servera [27].

Po importovaní knižnice z adresára *HTC.UnityPlugin/ViveInputUtility/Prefabs* presunieme prefabrikát *ViveRig* do okna hierarchie. Keďže prefabrikát *ViveRig* bude slúžiť na ovládanie pohľadu kamery musíme pôvodný objekt *Main Camera* odstrániť, nakoľko v scéne sa nemôžu nachádzať viaceré aktívne komponenty kamery. Unity editor v takejto situácii nedokáže rozoznať, ktorý z obrazov má byť na výstupe.

Následne je potrebné vykonať pár zmien v nastaveniach prefabrikátu *ViveRig* pre naše účely. Začneme s jeho detským objektom *Camera* a jeho komponentom Camera, kde hodnotu Clipping Planes Near znížime na minimum, teda na 0.01. Taktiež tomuto objektu musíme pridať nový komponent Audio Listener, aby sme na výstupe počuli zvuk nachádzajúci sa v scéne. Keďže v našej aplikácii nebudeme používať žiadne dodatočné trackery[[2]](#footnote-2) môžeme objekt *ViveTrackers* kompletne odstrániť. Po rozkliknutí *ViveControllers* sa nám objavia objekty *Left* a *Right*, ktoré taktiež obsahujú vlastné podobjekty. Z týchto podobjektov ponecháme iba *LaserPointer* a ostatné odstránime, nakoľko v našom interiérovom virtuálnom prostredí nebudeme využívať modely ovládačov, predpripravený systém chytania objektov, ani možnosť pohybu prostredníctvom teleportovania, ktoré so sebou knižnica VIU prináša.

### 3D modely rúk

Pre väčší pocit vnorenia pacienta do umelo vytvoreného prostredia sme sa rozhodli namiesto pôvodných virtuálnych modelov ovládačov použiť realisticky vyzerajúce 3D modely rúk z *Asset Store* [28]. Tieto 3D modely zahŕňajú kompletný kostrový systém mužských a ženských horných končatín, rôzne druhy animácií, textúr, shaderov a podobne.

V okne hierarchie pod objektmi *Left* a *Right* vytvoríme nové objekty[[3]](#footnote-3) s názvom *LeftHand* a *RightHand*. Do príslušných prázdnych objektov vložíme prevzaté modely rúk *SteamVR\_male\_hand\_left* a *SteamVR\_male\_hand\_right* z adresára *VR Hands FP Arms/Prefabs*. Daným modelom odoberieme skripty slúžiace na ovládanie animácií, pretože po prvé budeme používať iné tlačidlá ovládačov na spúšťanie príslušných animácií a po druhé nebudeme využívať všetky animácie, ktoré tento balíček ponúka.

Predvolená veľkosť modelov rúk je príliš veľká, preto je potrebné ich veľkosť prispôsobiť približne podľa veľkosti vlastných rúk. Urobíme to tak, že sa pozrieme na svoje ruky a vzápätí na ruky virtuálne. Takýmto spôsobom prispôsobujeme parametre dovtedy, kým nie sme s výsledkom spokojní. Potrebné je aj mierne vycentrovanie pozície modelov rúk voči pozícii ovládačov.

### Rozpoznanie vstupných zariadení v scéne

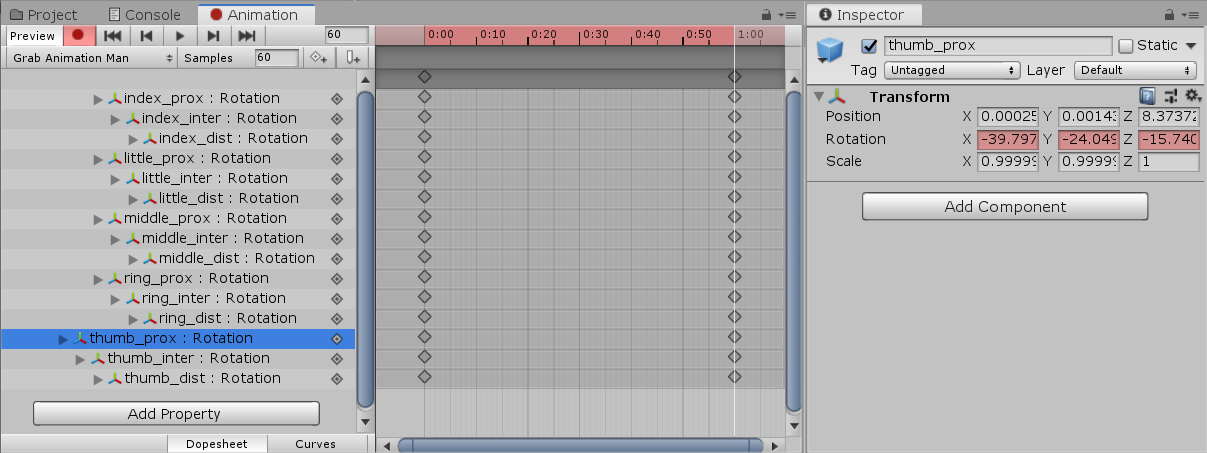
Aby trackovací respektíve sledovací systém pozostávajúci z dvoch senzorov (v našom prípade pri zariadení HTC Vive ide o systém s názvom **Lighthouse**) dokázal rozoznať ovládače, ktoré držíme v rukách je potrebné objektom *LeftHand* a *RightHand* priradiť skript *VivePoseTracker.cs*, ktorý je taktiež súčasťou knižnice VIU. Tento skript nám umožňuje zvoliť o aký typ vstupného zariadenia ide – ľavý ovládač alebo pravý ovládač. Prípadne by sme mohli mať k dispozícii aj viaceré ovládače alebo iné prídavné zariadenia ako napríklad trackery, ktoré je možné prostredníctvom systému sledovať a preniesť ich tak do virtuálneho sveta.

Ďalšou úlohou skriptu je zabránenie zobrazenia 3D modelu vypnutého ovládača v scéne. Slúži na to udalosť (event) onIsValidChanged a booleovská metóda GameObject.SetActive. Z tohto dôvodu vložíme do kolónky objekt *SteamVR\_male\_hand\_left* pre model ľavej ruky a objekt *SteamVR\_male\_hand\_right* pre model pravej ruky.

### Animácie modelov rúk

Ako už bolo spomenuté vyššie, balíček modelov rúk [28] obsahuje niekoľko rôznych druhov animácií, avšak len tri druhy z nich sú vhodné pre naše potreby. Konkrétne ide o animácie nečinnosti (angl. idle), dotyku (angl. touch) a zovretia ruky do päste (angl. fist).

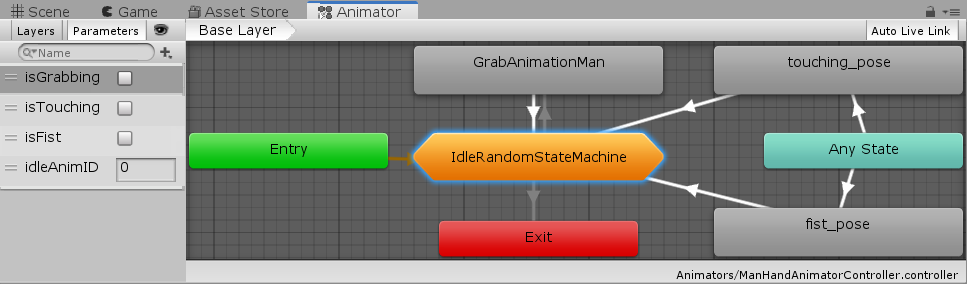
Nakoľko animácia chytenia respektíve uchopenia (angl. grab), ktorú poskytoval balíček nespĺňala nami požadované kritériá, rozhodli sme sa vytvoriť vlastnú takúto animáciu. Nemuseli sme na to využívať žiadny špeciálny animačný softvér tretej strany. Unity 3D totižto disponuje nástrojom na tvorbu pomerne kvalitných animácií priamo vo svojom prostredí, čo považujeme za obrovské plus.

V projekte teda vytvoríme nový adresár s názvom *Animations*, do ktorého budeme ukladať všetky nami vytvorené animácie. Cez ponuku hlavného menu *Window* > *Animation* zobrazíme časovú os, ktorá slúži na tvorbu animácií. Označením modelu ruky v hierarchii vytvoríme v okne *Animation* nový animačný klip pre úchop. Tlačidlom *Add Property* zvolíme ten podobjekt, v ktorom sú uložené všetky kosti modelu ruky a kĺby prstov. Pomocou ikony *+* pridáme jednotlivým podobjektom kľúčové snímky, ktoré nastavíme na jednu sekundu (snímka č. 60) a zapneme ich nahrávanie. Následne v inšpektore upravíme hodnoty rotácie na osi Z (pri prste palec) a deaktivujeme nahrávanie.

Obrázok 10: Nastavenie animácie chytenia pre model ruky

Takýmto spôsobom za nás Unity dopočíta všetky ostatné snímky medzi kľúčovými snímkami a vytvorí tak plynulú animáciu. Rovnaký postup aplikujeme pre všetky ostatné prsty a ich jednotlivé kĺby. Jediný rozdiel pri ostatných prstoch je v tom, že ich kĺby otáčame v smere osi X a nie v smere osi Z, ako to bolo pri palci. Ukážku nastavenia animácie chytenia môžeme vidieť na obrázku 10.

### Ovládanie animácií modelov rúk

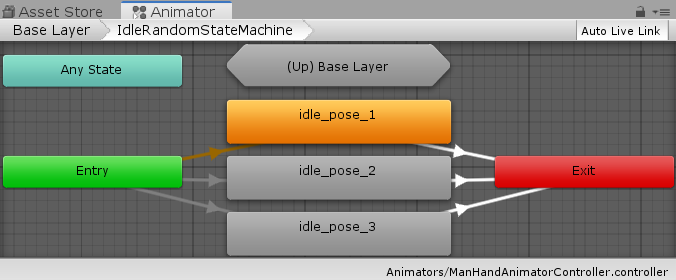
Na riadenie stavov animácií a prechodov medzi nimi musíme mať ako prvé k dispozícii *Animator Controller*. Vytvoríme ho v adresári *Animators* a pomenujeme ho ako *ManHandAnimatorController*. Dvojklikom naň sa otvorí okno *Animator*, do ktorého presunieme nami vytvorený animačný klip chytenia a päť animačných klipov z už spomínaného balíčka [28] – ide o trikrát nečinnosť, jedenkrát dotyk a jedenkrát zovretie ruky do päste. Animačnému klipu chytenia je potrebné zvýšiť jeho rýchlosť na hodnotu 10. Po dvojitom kliknutí mu v inšpektore taktiež deaktivujeme možnosť Loop Time, aby sa neopakoval v cykle donekonečna. Animačné klipy nečinnosti zoskupíme do jedného vnoreného stavu pomocou *Sub-State Machine*. Prechody medzi stavmi automatu a potrebné parametre dátového typu boolean a integer zhotovíme podľa obrázku 11.

Obrázok 11: Stavový diagram animácií modelov rúk s parametrami

Potrebné nastavenia prechodov v stavovom automate animačných klipov modelov rúk vykonáme v okne inšpektor podľa nasledovnej tabuľky 7:

Tabuľka 7: Nastavenia prechodov pre animácie modelov rúk

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prechod** | **Has Exit Time** | **Trvanie prechodu** (v sekundách) | **Podmienka** | **Hodnota** |
| Idle → Grab | − | − | isGrabbing | true |
| Grab → Idle | odškrtneme | 0.07 | isGrabbing | false |
| Any State → Touch | odškrtneme | 0.1 | isTouching | true |
| Touch → Idle | odškrtneme | 0.1 | isTouching | false |
| Any State → Fist | odškrtneme | 0.1 | isFist | true |
| Fist → Idle | odškrtneme | 0.1 | isFist | false |

Po vojdení do vnoreného stavu *IdleRandomStateMachine* spojíme stavy nečinnosti podľa obrázku 12. O ich náhodný výber sa stará skript *RandomAnimBehaviour.cs*, ktorý má indexy príslušných stavov uložené v poli.

Obrázok 12: Stavový diagram náhodných animácií nečinnosti modelov rúk

Konfiguráciu prechodov v stavovom automate náhodných animačných klipov nečinnosti modelov rúk vykonáme v okne inšpektor nasledovne (viď tabuľka 8):

Tabuľka 8: Nastavenia prechodov pre animácie nečinnosti modelov rúk

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prechod** | **Has Exit Time** | **Trvanie prechodu** (v sekundách) | **Podmienka** | **Hodnota** |
| idle\_pose\_1 → Exit | odškrtneme | 0.07 | isGrabbing | true |
| idle\_pose\_2 → Exit | odškrtneme | 0.07 | isGrabbing | true |
| idle\_pose\_3 → Exit | odškrtneme | 0.07 | isGrabbing | true |

Po úspešnom zhotovení *Controller*-a ho priradíme obidvom modelom rúk z okna hierarchie do komponentu Animator.

Aby sme tieto animácie dokázali spúšťať prostredníctvom určitého vstupného zariadenia (v našom prípade párom bezdrôtových HTC Vive ovládačov) je potrebné napísať skript v programovacom jazyku C#. Daný skript vytvoríme v adresári *Scripts* a nazveme ho *HandAnimation.cs*.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public class HandAnimation : MonoBehaviour  {  Animator anim;  public string axisGrab;  void Start() {  anim = GetComponentInChildren<Animator>();  }  void Update() {  if (Input.GetAxis(axisGrab) > 0) {  if (!anim.GetBool("isGrabbing")) {  anim.SetBool("isGrabbing", true);  }  }  else {  if (anim.GetBool("isGrabbing")) {  anim.SetBool("isGrabbing", false);  }  }  }  } |

Ukážka kódu 1: Trieda HandAnimation − riadenie stavov animácií podľa vstupu

V tejto sekvencii kódu sa nachádza funkčná implementácia spúšťania animácie chytania, ktorá sa vykoná po stlačení tlačidla, ktorého názov si uchováme v premennej axisGrab. Súčasťou skriptu je taktiež spúšťanie zvyšných dvoch animácií – dotyku a zovretia ruky do päste, ktoré v tejto ukážke neuvádzame, nakoľko ide o rovnaký kód (od riadku 11 po riadok 20) líšiaci sa len v názvoch premenných a parametroch.

Po naprogramovaní daného skriptu ho v Unity editore priradíme objektom *LeftHand* a *RightHand*. Do kolónky v inšpektore, ktorá prislúcha premennej axisGrab zadáme názov z *Edit* > *Project Settings* > *Input* > *Axes*, ktorý odpovedá tlačidlu grip na ľavom a pravom ovládači. Hodnoty osí joystickov zistíme z dokumentácie na oficiálnej stránke spoločnosti Unity Technologies [29]. Zisteným hodnotám v Unity editore odpovedá riadok *HTC\_VIU\_UnityAxis11* (pre ľavý ovládač) a riadok *HTC\_VIU\_UnityAxis12* (pre pravý ovládač). Analogickým spôsobom postupujeme aj pri animácii dotyku, ktorú namapujeme na stlačenie trackpadu a rovnako aj pri animácii zovretia ruky do päste, ktorej spúšťanie bude zabezpečovať tlačidlo trigger.

### Funkcionalita chytania a hádzania objektov

Ešte pred začatím programovania samotnej logiky držania a pustenia objektov prostredníctvom modelov rúk potrebujeme obidvom objektom *LeftHand* aj *RightHand* pridať požadované komponenty. V prvom rade ide o komponent Capsule Collider, ktorého veľkosť prispôsobíme rozmerom modelov rúk. Spravíme ho o čosi väčším, aby sme dokázali vziať aj objekty, ktoré nebudú v presnej pozícii modelov rúk. Collider po zaškrtnutí možnosti IsTrigger nebude navonok reagovať na kolízie s ostatnými objektmi (môže nimi prechádzať), ale namiesto toho bude pomocou metód OnTriggerStay a OnTriggerExit odosielať správy o tom, či prišiel do kontaktu s nejakým objektom alebo nie.

Druhým komponentom je Rigidbody. Nastavíme ho na Is Kinematic, čo znamená, že modely rúk nebudú reagovať na žiadne vonkajšie fyzikálne sily, ale ich pohyb (polohu a rotáciu) budeme riadiť my pomocou dvojice ovládačov. Táto možnosť nám taktiež umožní prístup k vektoru rýchlosti ovládača, prostredníctvom ktorého dokážeme verne simulovať hodenie predmetu vo VR tak, ako keby to bolo v skutočnosti.

Prejdeme na vytvorenie skriptu, ktorý pomenujeme *GrabController.cs*. Tento skript rovnako ako predchádzajúce komponenty priradíme objektom, ktoré reprezentujú ľavú a pravú ruku v hierarchii. Logika skriptu pozostáva z nasledovných krokov:

1. Chytenie objektu – zabezpečuje tlačidlo grip,
2. Držanie objektu – rodičom objektu sa stane ovládač,
3. Po uvoľnení tlačidla grip sa rodičovský vzťah preruší,
4. Vektor rýchlosti ovládača sa prenesie na objekt.

Na to, aby sme mohli kontrolovať, či môžeme niečo chytiť alebo nie použijeme dvojicu metód OnTriggerStay a OnTriggerExit.

|  |  |
| --- | --- |
| 1  2  3  4  5 | void OnTriggerStay(Collider other) {  if (other.CompareTag("HandItem")) {  handItemReady = other.gameObject;  }  } |

Ukážka kódu 2: Metóda OnTriggerStay

Vo vnútri tela metódy OnTriggerStay sa nachádzame vtedy, keď sa model ruky (ovládač) prekrýva s určitým objektom, ktorý taktiež obsahuje komponent Collider. Naopak metóda OnTriggerExit sa volá vtedy, keď sa model ruky (ovládač) prestane s daným objektom prekrývať, to znamená, že už nie sme schopní ho vziať do rúk. V skutočnosti, ak už niečo držíme, teda premenná isCarrying bude nastavená na hodnotu true, tak nás metóda OnTriggerExit vôbec nezaujíma (viď riadok dva v ukážke kódu 3).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void OnTriggerExit(Collider other) {  if (isCarrying) return;  if (other.CompareTag("HandItem")) {  handItemReady = null;  }  } |

Ukážka kódu 3: Metóda OnTriggerExit

Samozrejme chytanie objektov musíme nejakým spôsobom obmedziť. Nechceme dosiahnuť to, aby sme vedeli interagovať so všetkými objektmi nachádzajúcimi sa v scéne, ale iba s určitými, ktoré budú označené špecifickým tagom (v našom prípade HandItem). Toto nám zabezpečí podmienka v ukážke kódu 2 na riadku dva a v ukážke kódu 3 na riadku štyri. Premenná typu GameObject s názvom handItemReady nám slúži na uchovanie aktuálneho objektu, s ktorým sa s modelmi rúk prekrývame. V momente, keď sa s ním prestaneme prekrývať, nastavíme túto premennú na prázdnu hodnotu.

V metóde Update, ktorá sa volá pri každom vykreslení snímky ako prvé kontrolujeme vstup od používateľa. Obdobne ako v predchádzajúcej podkapitole, aj tu zvolíme pre úchop najprirodzenejšie tlačidlo pre tento úkon a tým je tlačidlo grip. V Unity editore preto priradíme do premennej axis reťazec *HTC\_VIU\_UnityAxis11* pre ľavú ruku a *HTC\_VIU\_UnityAxis12* pre pravú ruku.

Podmienka na riadku tri v ukážke kódu 4 zisťuje, či sa prekrývame s uchopiteľným objektom, ktorý je označený správnym tagom a zároveň sleduje, či náhodou už niečo v rukách nedržíme (booleovská premenná isCarrying je prednastavená na hodnotu false, teda po zapnutí aplikácie máme ruky voľné). V prípade, ak máme ruky prázdne zavoláme metódu CarryItem, ktorá to zmení.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | void Update() {  if (Input.GetAxis(axis) > 0) {  if (handItemReady != null && !isCarrying) {  CarryItem();  }  }  else if (isCarrying) {  handItemVelocity = (transform.position - previousPos)  / Time.deltaTime;  ThrowItem();  }  previousPos = transform.position;  } |

Ukážka kódu 4: Metóda Update – držanie alebo hodenie objektu podľa vstupu

Metóda CarryItem nastaví ovládač ako rodiča pre uchopený objekt (handItemReady), čo znamená, že môžeme s ovládačom ľubovoľne pohybovať a pri stlačenom tlačidle grip, ho bude daný objekt všade nasledovať (viď riadok dva v ukážke kódu 5). Na ďalšom riadku pristúpime k jeho komponentu Rigidbody, pomocou ktorého ho spravíme kinematickým, takže po chytení už nebude reagovať na pôsobenie gravitácie, či iné vonkajšie sily. Jediné na čo bude reagovať, je zmena transformácie prostredníctvom ovládačov. Poslednú vec, ktorú metóda vykoná je, že aktualizuje príznak toho, že niečo v rukách držíme.

|  |  |
| --- | --- |
| 1  2  3  4  5 | void CarryItem() {  handItemReady.transform.parent = transform;  handItemReady.GetComponent<Rigidbody>().isKinematic = true;  isCarrying = true;  } |

Ukážka kódu 5: Metóda CarryItem − držanie objektu

V tomto momente máme funkčnú jednu časť skriptu – chytanie objektov. Prejdime na druhú časť, v ktorej implementujeme logiku hádzania objektov. Hodenie respektíve pustenie objektu z modelu ruky nastáva, keď používateľ uvoľní tlačidlo grip. Samozrejme, že to platí, len v prípade, ak už nejaký objekt v rukách držal (pozri riadok sedem v ukážke kódu 4). Táto vetva podmienky bude vykonávať metódu hádzania objektu ThrowItem.

Pred tým však ešte potrebujeme vypočítať predchádzajúcu pozíciu ovládača, ktorú budeme neustále ukladať do premennej previousPos na konci metódy Update (viď riadok trinásť v ukážke kódu 4). Teda aktuálna pozícia sa bude ukladať ako nová „predchádzajúca“ pozícia. Keď máme k dispozícii hodnotu predchádzajúcej pozície, vieme vypočítať rýchlosť ovládača handItemVelocity, ktorá je dátového typu Vector3. Túto rýchlosť získame pomocou fyzikálneho vzorca pre výpočet rýchlosti rovnomerného pohybu telesa:

kde  je rýchlosť a  je dráha prejdená za čas . V našom prípade dráha respektíve vzdialenosť odpovedá rozdielu aktuálnej pozície ovládača s jeho predchádzajúcou pozíciou. Výsledný rozdiel predelíme časom (viď riadok osem v ukážke kódu 4).

Pri hádzaní určitého objektu sa vykoná niekoľko akcií. Ako prvé použijeme komponent Rigidbody objektu, ktorý držíme a prehlásime ho za nekinematický, čo spôsobí to, že daný objekt začne reagovať na fyzikálne sily (viď riadok tri v ukážke kódu 6). V riadku štyri prenesieme vypočítaný vektor rýchlosti ovládača na držaný objekt. Následne sa zbavíme rodičovského vzťahu tým, že mu priradíme prázdnu hodnotu a nakoniec aktualizujeme príznak isCarrying na hodnotu false, aby sme vedeli, že už nič v rukách nedržíme.

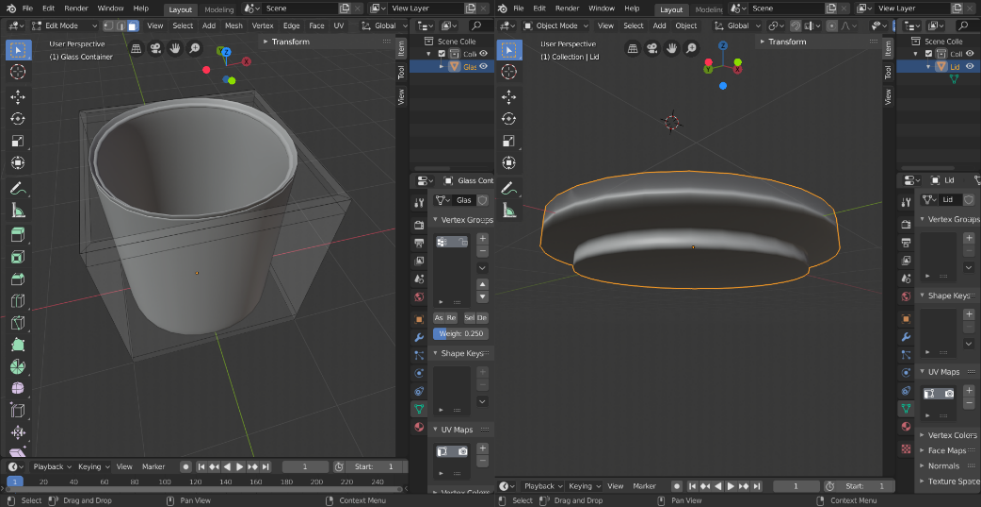
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void ThrowItem() {  Rigidbody rb = handItemReady.GetComponent<Rigidbody>();  rb.isKinematic = false;  rb.velocity = handItemVelocity;  handItemReady.transform.parent = null;  isCarrying = false;  } |

Ukážka kódu 6: Metóda ThrowItem − hodenie objektu

### Modelovanie prostredia

V tejto fáze tvorby aplikácie prichádza na rad modelovanie jednoduchého prototypu virtuálneho prostredia. Začneme s vytvorením podlahy, uprostred ktorej bude sedieť alebo stáť pacient. Podlahu v Unity editore najjednoduchšie zhotovíme použitím základného telesa *Plane*, ktoré nájdeme po kliknutí pravým tlačidlom do okna hierarchie v ponuke *3D Object*. Dané teleso pomenujeme ako *Floor* a umiestnime ho do počiatku súradnicovej sústavy, čomu odpovedá pozícia X: 0, Y: 0, Z: 0.

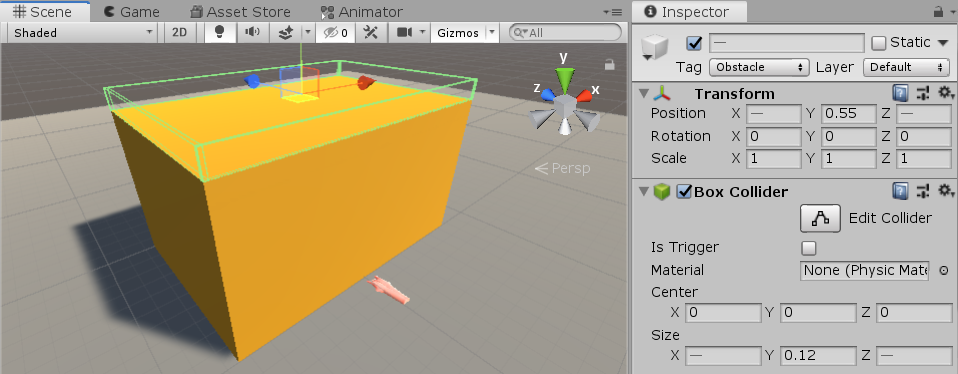
Druhým neodmysliteľným prvkom aplikácie je stôl, po ktorom sa budú neskôr pohybovať 3D modely pavúkov. Vytvoríme ho analogickým spôsobom, akým sme vytvárali podlahu, to znamená, že použijeme ďalšie zo základných telies, ktoré nám Unity editor ponúka. Tento krát to však bude teleso *Cube*, ktoré nazveme *Table* a jeho veľkosť prispôsobíme rozmerom reálneho stolíka. Keďže 1 Unity jednotka predstavuje v skutočnosti 1 m, tak pre šírku stola sme sa rozhodli zvoliť hodnotu 1.5, pre výšku hodnotu 0.8 a pre hĺbku hodnotu 1. Jeho polohu v scéne sme upravili podľa pozície prefabrikátu *ViveRig*, ktorý reprezentuje pohľad používateľa.

Na modelovanie dodatočných objektov ako napríklad priehľadná nádoba s uzatvárateľným vekom, ktorú môžeme vidieť na obrázku 13, sme využili voľne šíriteľný 3D modelovací softvér **Blender**.

Obrázok 13: Vymodelovaná nádoba s vekom v programe Blender

Zostalo nám už len farebne odlíšiť tieto vytvorené telesá. Bežne sa na to používajú textúry alebo materiály, ktorým môžeme nastavovať rôzne vlastnosti, ako sú napríklad povrchové úpravy, metalíza, hladkosť a podobne. V adresárovej štruktúre projektu si vytvoríme adresár, do ktorého budeme ukladať všetky materiály používané v našej aplikácii. Nový materiál vytvoríme v okne projektu kliknutím RMB > *Create*, kde vyberieme možnosť *Material*. Vhodne si ho pomenujeme, najlepšie podľa toho, na ktorý objekt ho chceme naniesť. Keďže ho chceme naniesť na podlahu zvolíme názov *FloorMaterial*. V inšpektore daného materiálu v časti Albedo vyberieme farbu podlahy napríklad sivú a pretiahneme tento materiál na objekt *Floor*. Rovnakým spôsobom postupujeme aj pri aplikovaní materiálu pre objekt *Table*. Jediný rozdiel bude len pri voľbe farby, pretože stolík chceme mať hnedý.

#### Hranice okolo stola

Pod objektom *Table* v okne hierarchie vytvoríme prázdny objekt[[4]](#footnote-4), ktorý pomenujeme *Borders* − bude slúžiť ako kontajner pre objekty hraníc okolo stola. V tomto kontajneri vytvoríme štyri objekty kocky, ktorých názvy zvolíme podľa strán stola: *BorderLeft*, *BorderRight*, *BorderBack* a *BorderFront*. Aby jednotlivé kocky neboli viditeľné používateľom, ale zároveň aby plnili svoj účel, musíme im odstrániť všetky komponenty z okna inšpektor okrem Box Collider-ov. Postupne po jednom im upravujeme dĺžku Box Collider-ov na hodnoty podľa veľkosti stola. Ich šírka môže byť malá, nakoľko to bude slúžiť len ako zábrana okolo stola. Pre ich výšku sme zvolili hodnotu 0.12, ktorá odpovedá 12 cm v skutočnosti. Taktiež zmeníme aj pozíciu objektov hraníc v scéne a prispôsobíme ich okrajom stola. Zostáva už len pomocou inšpektora vytvoriť nový tag s názvom Obstacle, ktorý budeme neskôr využívať v zdrojovom kóde pre pohyb pavúka. Daným tagom označíme všetky štyri objekty, ktoré tvoria hranice stola.

Obrázok 14: Hranice okolo stola vytvorené z komponentov Box Collider

Úlohou hraníc nachádzajúcich sa na obrázku 14 bude obmedzenie pohybu pavúka, aby sa dokázal hýbať len vo vyhradenej časti územia, ktorým je v našom prípade povrch stola.

### 3D model pavúka

Jednoduché virtuálne prostredie máme vymodelované, avšak chýba nám v ňom najdôležitejší prvok našej aplikácie. Týmto prvkom je hlavný spúšťač jednej z najznámejších špecifických fóbií (arachnofóbie) – pavúk. Trojrozmerný model pavúka sme sa nakoniec rozhodli nemodelovať a to hlavne z dôvodu časovej náročnosti tohto procesu. Treba si uvedomiť, že by nešlo iba o samotné modelovanie, ale aj o textúrovanie, tvorbu kostry, rôznych druhov animácií a podobne. Už len jedna animácia pohybu článkov nôh a tela pavúka by bola náročná na realizáciu, nehovoriac o ďalších. Z tohto dôvodu sme sa priklonili k efektívnejšej variante, ktorou je import 3D modelov pavúkov z obchodu *Asset Store* [30]. Obsahom balíčka sú tri druhy 3D modelov pavúkov, rozličné animácie a materiály.

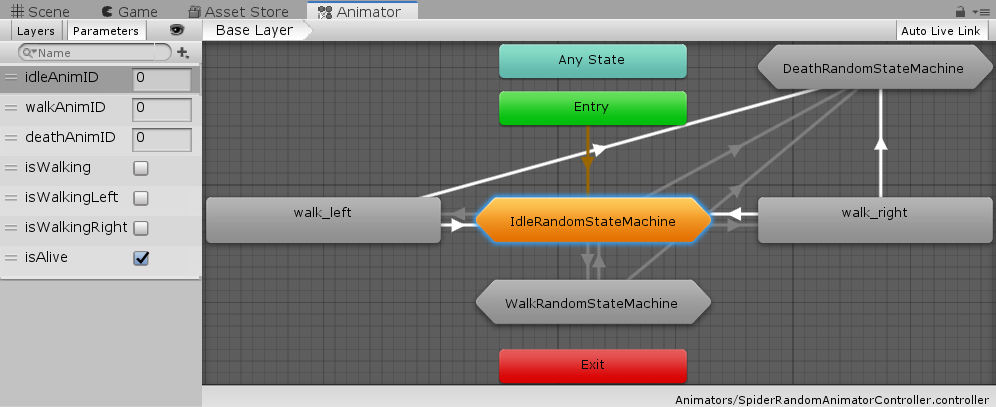
V okne hierarchie vytvoríme nový prázdny objekt[[5]](#footnote-5), ktorého úlohou bude niesť prvý druh modelu pavúka, preto ho pomenujeme ako *Spider1*. Z naimportovaného balíčka vyberieme model *spider\_01* a presunieme ho do predpripraveného prázdneho objektu *Spider1*. Predvolená rotácia modelu pavúka je smerom od nás, preto je nutné otočiť jeho rodičovský objekt o 180 stupňov podľa osi Y. Veľkosť modelu pavúka[[6]](#footnote-6) a jeho polohu v scéne prispôsobíme podľa rozmerov stola. Snažíme sa ho umiestniť tak, aby stál nohami na stole. Pre jednoduchšie pristupovanie k materiálu modelu pavúka prostredníctvom skriptu zmeníme názvy jeho detských objektov *spider\_01* na *SpiderModel* a objekt *spider* premenujeme na *SpiderRender*.

Teraz vytvoríme adresár *Prefabs*, do ktorého presunieme objekt nesúci model pavúka *Spider1*. Takýmto spôsobom sa z obyčajného objektu stal prefabrikát. To znamená, že ak do scény umiestnime ďalší prefabrikát tohto typu, bude mať totožné hodnoty a vlastnosti (komponenty) ako jeho predchodca. Pri zmene niektorej z hodnôt sa táto zmena prejaví na všetkých prefabrikátoch tohto typu, čo považujeme za nesmiernu výhodu. Prefabrikáty sú v okne hierarchie odlíšené od bežných objektov modrou farbou. Identický postup opísaný vyššie aplikujeme aj pre zvyšné dva druhy modelov pavúkov z balíčka [30].

Momentálne máme zhotovené statické modely pavúkov, s ktorými zatiaľ vo virtuálnom prostredí nedokážeme nijako manipulovať. Aby sme to napravili musíme jednotlivým prefabrikátom pavúkov pridať dva komponenty. Najskôr im pridáme kolízne teleso obdĺžnikového tvaru Box Collider, ktorého veľkosť modifikujeme podľa rozmerov konkrétneho modelu pavúka. Ak však chceme s objektami pavúkov aj hýbať musíme im taktiež priradiť komponent Rigidbody, ktorý simuluje fyzikálne zákony reálneho sveta. Posledným krokom je vytvorenie nového tagu HandItem, ktorým označíme všetky tri prefabrikáty pavúkov. V danej chvíli je možná interakcia s modelmi pavúkov prostredníctvom virtuálnych rúk.

### Ovládanie animácií modelu pavúka

Počiatočným krokom pri vytváraní dynamického modelu pavúka je mať k dispozícii *Animator Controller*, ktorý zhotovíme v adresári *Animators* a nazveme ho *SpiderRandomAnimatorController*. Už z názvu vyplýva, že pôjde o náhodnú voľbu určitých typov animácií. Túto náhodnosť bude mať na starosti zdrojový kód *RandomAnimBehaviour.cs*, ktorý z vnorených stavov bude voliť vždy jednu animáciu podľa indexov uchovaných v poli. Otvoríme okno *Animator*, do ktorého popresúvame animačné klipy nečinnosti, chôdze vpred, chôdze do strán a taktiež animácie smrti. Celkovo ide o desať animačných klipov zo zmieňovaného balíčka [30]. Týmto animačným klipom (okrem animácií smrti) aktivujeme v inšpektore možnosť Loop Time, ktorá zabezpečí, aby sa animácia opätovne spustila po dosiahnutí jej konca. Nakoľko disponujeme viacerými animačnými klipmi nečinnosti, pohybu vpred a animáciami smrti, zhromaždíme ich do vnorených stavov takzvaných *Sub-State Machine*. Prechody medzi jednotlivými stavmi automatu a potrebné parametre dátového typu integer a boolean zhotovíme podľa obrázku 15.



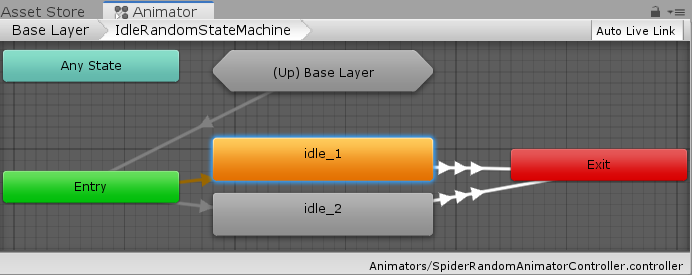
Obrázok 15: Stavový diagram animácií modelu pavúka s parametrami

Definovanie prechodov v automate animačných klipov modelu pavúka urobíme v okne inšpektor nasledovne (viď tabuľka 9):

Tabuľka 9: Nastavenia prechodov pre animácie modelu pavúka

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prechod** | **Has Exit Time** | **Trvanie prechodu** (v sekundách) | **Podmienka** | **Hodnota** |
| Idle → Walk | − | − | isWalking | true |
| Walk → Idle | − | − | isWalking | false |
| Idle → walk\_left | − | − | isWalkingL | true |
| walkLeft → Idle | odškrtneme | 0.25 | isWalkingL | false |
| Idle → walk\_right | − | − | isWalkingR | true |
| walk\_right → Idle | odškrtneme | 0.25 | isWalkingR | false |
| Idle → Death | − | − | isAlive | false |
| Walk → Death | − | − | isAlive | false |
| walk\_left → Death | odškrtneme | 0.25 | isAlive | false |
| walk\_right → Death | odškrtneme | 0.25 | isAlive | false |

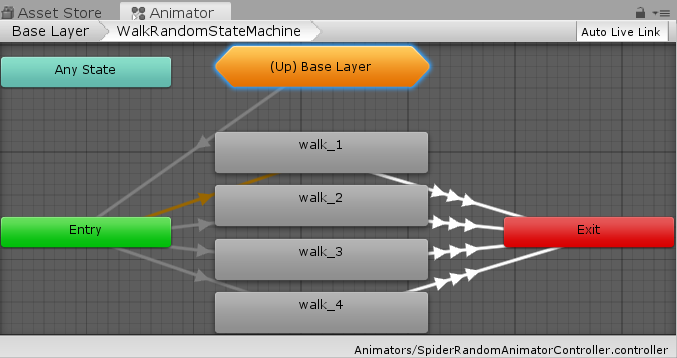
Prejdením do vnoreného stavu *IdleRandomStateMachine* skonštruujeme prechody medzi stavmi nečinnosti podľa obrázku 16.

Obrázok 16: Stavový diagram náhodných animácií nečinnosti modelu pavúka

Konfiguráciu prechodov v tomto automate náhodných animačných klipov nečinnosti modelu pavúka uskutočníme v okne inšpektor podľa nasledovnej tabuľky 10:

Tabuľka 10: Nastavenia prechodov pre animácie nečinnosti modelu pavúka

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prechod** | **Has Exit Time** | **Trvanie prechodu** (v sekundách) | **Podmienka** | **Hodnota** |
| idle\_1 → Exit | odškrtneme | 0.25 | isWalking | true |
| idle\_1 → Exit | odškrtneme | 0.25 | isWalkingL | true |
| idle\_1 → Exit | odškrtneme | 0.25 | isWalkingR | true |
| idle\_1 → Exit | odškrtneme | 0.25 | isAlive | false |
| idle\_2 → Exit | odškrtneme | 0.25 | isWalking | true |
| idle\_2 → Exit | odškrtneme | 0.25 | isWalkingL | true |
| idle\_2 → Exit | odškrtneme | 0.25 | isWalkingR | true |
| idle\_2 → Exit | odškrtneme | 0.25 | isAlive | false |

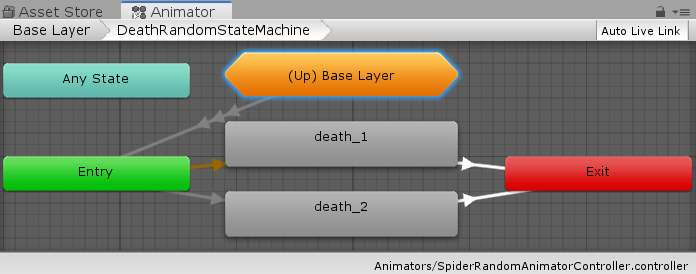
Po presunutí sa do vnoreného stavu *WalkRandomStateMachine* poprepájame stavy chôdze vpred podľa obrázku 17.

Obrázok 17: Stavový diagram náhodných animácií chôdze modelu pavúka

Potrebné nastavenia prechodov v stavovom automate náhodných animačných klipov pohybu vpred modelu pavúka vykonáme v okne inšpektor nasledovne (viď tabuľka 11):

Tabuľka 11: Nastavenia prechodov pre animácie chôdze modelu pavúka

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prechod** | **Has Exit Time** | **Trvanie prechodu** (v sekundách) | **Podmienka** | **Hodnota** |
| walk\_1 → Exit | odškrtneme | 0.25 | isWalking | false |
| walk\_1 → Exit | odškrtneme | 0.25 | isAlive | false |
| walk\_2 → Exit | odškrtneme | 0.25 | isWalking | false |
| walk\_2 → Exit | odškrtneme | 0.25 | isAlive | false |
| walk\_3 → Exit | odškrtneme | 0.25 | isWalking | false |
| walk\_3 → Exit | odškrtneme | 0.25 | isAlive | false |
| walk\_4 → Exit | odškrtneme | 0.25 | isWalking | false |
| walk\_4 → Exit | odškrtneme | 0.25 | isAlive | false |

Posledným vnoreným stavom je *DeathRandomStateMachine*, v ktorom dáme dohromady stavy smrti podľa obrázku 18.

Obrázok 18: Stavový diagram náhodných animácií smrti modelu pavúka

Ostáva už len zadefinovanie prechodov v automate náhodných animačných klipov smrti modelu pavúka. Túto činnosť realizujeme v okne inšpektor podľa nasledovnej tabuľky 12:

Tabuľka 12: Nastavenia prechodov pre animácie smrti modelu pavúka

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prechod** | **Has Exit Time** | **Trvanie prechodu** (v sekundách) | **Podmienka** | **Hodnota** |
| death\_1 → Exit | odškrtneme | 0.25 | − | − |
| death\_2 → Exit | odškrtneme | 0.25 | − | − |

Po finálnom dokončení *Controller*-a ho pripneme všetkým trom prefabrikátom pavúkov do poľa *Controller* v komponente Animator.

### Pohyb modelu pavúka

Logiku, ktorá sa skrýva za pohybom pavúka nám zabezpečí skript *SpiderRandomMovement.cs*, ktorý vytvoríme v adresári *Scripts*. Ako už z názvu môžeme tušiť, pôjde o náhodný pohyb tohto drobného živočícha s ôsmimi nohami. Tento skript priradíme všetkým trom prefabrikátom pavúkov *Spider1*, *Spider2* a *Spider3*, ktoré sú umiestnené v adresári *Prefabs*.

Samotný pohyb pavúka a jeho otáčanie do strán vykonáva trojica metód MoveForward, TurnLeft a TurnRight. Okrem toho je ich úlohou spúšťanie príslušných animácií na základe smeru pohybu pavúka (viď riadok dva v ukážkach kódov 7, 8 a 9).

|  |  |
| --- | --- |
| 1  2  3  4  5 | void MoveForward() {  anim.SetBool("isWalking", true);  transform.position += transform.forward \* Time.deltaTime  \* moveSpeed;  } |

Ukážka kódu 7: Metóda MoveForward – pohyb modelu pavúka vpred

|  |  |
| --- | --- |
| 1  2  3  4 | void TurnLeft(float rSpeed) {  anim.SetBool("isWalkingLeft", true);  transform.Rotate(transform.up \* Time.deltaTime \* -rSpeed);  } |

Ukážka kódu 8: Metóda TurnLeft – otočenie modelu pavúka vľavo

|  |  |
| --- | --- |
| 1  2  3  4 | void TurnRight(float rSpeed) {  anim.SetBool("isWalkingRight", true);  transform.Rotate(transform.up \* Time.deltaTime \* rSpeed);  } |

Ukážka kódu 9: Metóda TurnRight – otočenie modelu pavúka vpravo

Na implementovanie náhodnosti pohybu pavúka nám vhodne poslúži koprogram (angl. Coroutine), ktorý využíva špeciálne jazykové konštrukcie programovacieho jazyka C# – iterátory. Iterátor je metóda, ktorá vracia výsledok postupne, podľa potreby. To znamená, že pomocou nej dokážeme zámerne pozastaviť a následne opätovne obnoviť výpočet. Daný výpočet pokračuje v zdrojovom kóde tam, kde skončil, avšak už na nasledujúcej snímke (pozri príkaz yield s volaním WaitForSeconds v ukážke kódu 10).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | IEnumerator Wander() {  int rotateTime = Random.Range(1, 3);  float rotateWait = Random.Range(0.5f, 1);  int rotateLeftOrRight = Random.Range(1, 2);  float walkTime = Random.Range(0.5f, 1);  int walkWait = Random.Range(1, 4);  isWandering = true;  yield return new WaitForSeconds(walkWait);  isWalking = true;  yield return new WaitForSeconds(walkTime);  isWalking = false;  yield return new WaitForSeconds(rotateWait);  if (rotateLeftOrRight == 1) {  isRotatingRight = true;  yield return new WaitForSeconds(rotateTime);  isRotatingRight = false;  }  if (rotateLeftOrRight == 2) {  isRotatingLeft = true;  yield return new WaitForSeconds(rotateTime);  isRotatingLeft = false;  }  isWandering = false;  } |

Ukážka kódu 10: Iterátor Wander – náhodný pohyb pavúka (úprava [31])

Na začiatku iterátora si postupne vygenerujeme náhodné hodnoty zo zadaného rozsahu, ktoré použijeme ako argumenty metódy WaitForSeconds. Práve tieto hodnoty zvyšujú autentickosť pohybu pavúka. Na spustenie koprogramu zavoláme v metóde Update funkciu StartCoroutine s argumentom iterátora Wander. V tele metódy Update taktiež voláme metódy z ukážok kódov 7, 8 a 9 na základe hodnôt booleovských premenných z iterátora (isWalking, isRotatingLeft, isRotatingRight).

Okrem náhodného pohybu pavúka tento skript obsahuje taktiež prvky umelej inteligencie, ktoré slúžia na vyhýbanie sa prekážkam s určitým označením, v našom prípade objektom s tagom Obstacle.

### Používateľské rozhranie aplikácie

Používateľské rozhranie aplikácie rozdelíme na 2D a 3D rozhranie. Najskôr vytvoríme **trojdimenzionálne rozhranie** (angl. diegetic UI), pomocou ktorého bude môcť pacient modifikovať objekty v prostredí. Neskôr sa budeme venovať tvorbe **dvojdimenzionálneho rozhrania** (angl. non-diegetic UI) pre psychoterapeuta.

#### Používateľské rozhranie pacienta

Plocha, ktorá bude zoskupovať všetky ovládacie prvky používateľského rozhrania sa nazýva **plátno**. Pre jej vytvorenie klikneme RMB do okna hierarchie, kde zvolíme možnosť *UI* > *Canvas*. V inšpektore objektu *Canvas* zmeníme spôsob vykresľovania na World Space, čím zabezpečíme jeho zobrazenie v 3D priestore a priradíme mu ako nový komponent skript *CanvasRaycastTarget.cs*. Ide o skript rozširujúci základný systém používateľského rozhrania v Unity pre potreby VR, ktorý pochádza z knižnice VIU. Dané plátno osadíme na vhodné miesto, podľa ktorého mu prispôsobíme jeho rozmery. Ďalej vo vnútri objektu *Canvas* vytvoríme ďalší objekt z možnosti *UI*, tentokrát to však bude *Panel*, na ktorý môžeme aplikovať ľubovoľnú farbu alebo textúru. Keďže na *Panel*-i nechceme mať nanesenú textúru, ale len farbu, v komponente Image zvolíme zdroj žiadny a farbu zvolíme z ponuky Color.

Všetky nasledovné ovládacie prvky GUI budeme vytvárať v hierarchii pod objektom *Panel*. Prvým prvkom je nadpis respektíve samotný **text**, ktorý zhotovíme voľbou *UI* > *Text*. Veľkosť písma, jeho zarovnanie na stred a farbu kontrastnú s pozadím definujeme v inšpektore.

Druhým prvkom sú **zaškrtávacie tlačidlá**, ktoré získame pomocou *UI* > *Toggle*. Rozmery boxov a veľkosť písma pozmeníme podľa vlastnej potreby. Ak máme objekt *Toggle*, s ktorým sme spokojní, dvakrát ho duplikujeme. Každému z týchto troch objektov vhodne pomenujeme ich *Label* v komponente Text. V našom prípade to bude: „Tenký“, „Stredný“, „Hrubý“. Následne objekty *Toggle* zoskupíme do prázdneho objektu s názvom *AddSpiderToggleGroup*, ktorému priradíme komponent Toggle Group (na tomto mieste povolíme zaškrtávanie). Po označení všetkých objektov *Toggle* im presunieme do poľa Group ich rodičovský objekt *AddSpiderToggleGroup*. Prvé zaškrtávacie tlačidlo s názvom „Tenký“ nastavíme, aby bolo predvolene aktívne pomocou premennej Is On v inšpektore.

Tretím prvkom je **tlačidlo**, ktoré nájdeme pod *UI* > *Button*. Nastaveniami v inšpektore docielime správnu veľkosť tlačidla. Čo sa týka vlastností jeho textu tie nájdeme v podobjekte *Text*. Názov tlačidla zmeníme v komponente Text na „Pridaj pavúka“. Analogicky postupujeme aj pri tvorbe zvyšných tlačidiel, odhliadnuc od toho, že každé tlačidlo bude mať iný názov a bude slúžiť na iný účel.

Štvrtým prvkom je vodorovný **posuvník**, ktorý sa ukrýva pod možnosťou *UI* > *Slider*. Pri posuvníku je potrebné modifikovať rozmery samotného posuvníka, podobjektu *Fill* (tvorí časť, ktorá sa vypĺňa) a taktiež podobjektu *Handle*.

Posledným použitým ovládacím prvkom GUI je **rozbaľovací zoznam**, ktorý sa nachádza pod *UI* > *Dropdown*. Rozbaľovací zoznam sa skladá z troch hlavných detských objektov: *Label*, *Arrow* a *Template*. Posledný menovaný má ešte niekoľko ďalších, z ktorých sú pre nás zaujímavé: *Content*, *Item* a *Item Checkmark*. Všetkým spomenutým objektom musíme upraviť ich rozmery. Detský objekt *Label* zobrazuje prvú možnosť zo zoznamu, preto prepíšeme jeho komponent Text na „Farba 1“. Ostatné možnosti zo zoznamu zahŕňa podobjekt *Template*, ktorý sa aktivuje respektíve „rozbalí“ len po kliknutí na šípku. Aby sme mohli s rozbaľovacím zoznamom interagovať vo VR, musíme priradiť objektu *Template* skript *CanvasRaycastTarget.cs* z knižnice VIU. Zostáva už len rodičovský objekt *Dropdown* naplniť správnymi hodnotami, pričom v našom prípade ide o reťazce: „Farba 1“ až „Farba 5“.

#### Používateľské rozhranie psychoterapeuta

Nakoľko používateľské rozhranie slúžiace pre psychoterapeuta bude veľmi podobné tomu pre pacienta, nebudeme sa mu venovať tak dopodrobna. V podstate ide o identickú činnosť, až na pár detailov. Keďže toto rozhranie bude dvojdimenzionálne, nemusíme vykresľovanie plátna nastavovať na World Space. Rovnako nie je potrebné žiadnemu objektu priraďovať skript *CanvasRaycastTarget.cs*, pretože psychoterapeut bude aplikáciu ovládať pomocou klasickej klávesnice a myši. Posledným rozdielom oproti vyššie opísanému postupu je, že v tomto prípade nie je nevyhnutné meniť pôvodnú veľkosť súčastí GUI, pretože sa zobrazujú korektne.

### Tvorba aplikačnej logiky

Prvá vec, ktorú potrebujeme spraviť je vytvorenie troch prázdnych objektov v scéne, ktoré budú niesť pozíciu miesta vzniku, respektíve pridania nových pavúkov. Tieto prázdne objekty si pomenujeme ako *Spawn1*, *Spawn2*, *Spawn3* a umiestnime ich do troch rôznych častí na povrch stola (napríklad vľavo, do stredu a vpravo).

Keď to máme, zhotovíme ďalší prázdny objekt s názvom *3DAppUIManager* – bude slúžiť ako kontajner pre nový skript *ThreeAppUiManager.cs*, ktorý mu priradíme ako komponent. V skripte sme naprogramovali hlavnú časť logiky aplikácie, ktorá je obsiahnutá v nasledujúcich trinástich metódach.

Metóda PushSpider sa volá po zvolení druhu a sfarbenia pavúka, ktoré sú implementované ako zaškrtávacie tlačidlá a následnom stlačení tlačidla „Pridaj pavúka“. Týmto úkonom sa do dátovej štruktúry zásobník (typu LIFO) pridá inštancia jedného z troch modelov pavúkov („Tenký“, „Stredný“ alebo „Hrubý“) a zároveň sa pavúk objaví na stole vo virtuálnom prostredí. Jeho počiatočná pozícia závisí od umiestnenia spomínaných objektov *Spawn1*, *Spawn2* alebo *Spawn3*.

Metóda PopSpider sa volá po stlačení tlačidla „Vymaž pavúka“. Spôsobí odobranie inštancie pavúka z vrchu zásobníka a taktiež jeho odstránenie zo scény.

Metóda DeleteAllSpiders sa volá po stlačení tlačidla „Vymaž všetky pavúky“ a odstráni zo zásobníka všetky doposiaľ vytvorené inštancie pavúkov.

Metóda ScaleItUp sa volá po stlačení tlačidla „Väčšie pavúky“. Jej úlohou je zväčšovať všetky inštancie pavúkov uložených v zásobníku o konštantnú hodnotu. Maximálna veľkosť pavúka je obmedzená hornou hranicou.

Metóda ScaleItDown sa volá po stlačení tlačidla „Menšie pavúky“. Robí presný opak toho, čo robí predchádzajúca metóda. To znamená, že zmenšuje inštancie pavúkov umiestnených v zásobníku konštantnou hodnotou s tým, že je určená minimálna veľkosť inštancie pavúka.

Metóda SpeedChanged sa volá pri posúvaní vodorovného posuvníka, pomocou ktorého môžeme regulovať rýchlosť inštancií pavúkov uložených v zásobníku.

Metóda SizeChanged sa volá pri posúvaní vodorovného posuvníka, pomocou ktorého dokážeme zväčšovať alebo zmenšovať inštancie pavúkov v zásobníku.

Metóda ChangeColoration sa volá pri výbere sfarbenia pavúka z rozbaľovacieho zoznamu. Sfarbenie sa mení na základe voľby z troch materiálov, ktoré sú uložené v dátovej štruktúre pole.

Metóda MoveTableCloser sa volá po stlačení tlačidla „Posuň stôl bližšie“. Táto metóda slúži na priblíženie 3D modelu stola spolu s priehľadnou nádobou o jeden meter bližšie k pacientovi.

Metóda MoveTableAway sa volá po stlačení tlačidla „Posuň stôl ďalej“ a robí to isté, čo predchádzajúca metóda, ibaže opačne – oddiali 3D model stola spoločne s priehľadnou nádobou o jeden meter od pacienta.

Metóda HideShowTransparentContainer sa volá po stlačení tlačidla „Skry/zobraz nádobu“, čím z virtuálneho prostredia zmizne 3D model priehľadnej nádoby spolu s uzatvárateľným vekom. Opätovným stlačením tohto tlačidla sa nádoba s vekom objavia na tom istom mieste ako pred zmiznutím.

Metóda MainMenuScene sa volá po stlačení tlačidla „Hlavné menu“ a načíta scénu hlavného menu aplikácie. Teda presunie používateľa z virtuálnej miestnosti do hlavného menu.

Metóda QuitApplication sa volá po stlačení tlačidla „Ukonč aplikáciu“ a vynúti ukončenie aplikácie.

## Používateľské testovanie aplikácie

Testovanie aplikácie sa uskutočňovalo v laboratóriu virtuálnej reality a skúmania používateľského zážitku (miestnosť 140A) v priestoroch FPV UMB na katedre informatiky v Banskej Bystrici. Táto výskumná činnosť prebiehala niekoľko dní v rámci rôznych aktivít alebo podujatí. Rovnako ako v procese návrhu, tak aj vo fáze testovania mal ako prvý možnosť aplikáciu zhodnotiť psychoterapeut z praxe pán Mgr. Ján Záskalan, ktorý je odborníkom na KBT a má bohaté skúsenosti s liečbou úzkostných porúch vrátane špecifických fóbií. Nasledovalo testovanie študentmi aplikovanej informatiky druhého ročníka magisterského štúdia, ktoré sa konalo počas úvodných vyučovacích hodín povinne voliteľného predmetu Virtuálna realita. Vhodnou udalosťou na otestovanie projektu stredoškolskými študentmi bol taktiež deň otvorených dverí UMB, ktorý sa konal dňa 13.11.2019. Ďalší priestor na získanie spätnej väzby nám poskytol pán prodekan PaedDr. Mgr. Vladimír Siládi, PhD., ktorý nás pozval na jednu z vyučovacích hodín so študentmi druhého ročníka magisterského štúdia v študijnom programe Forenzná a kriminalistická chémia. Okrem spomínaných aktivít sa testovaniu aplikácie podrobili aj vysokoškolskí učitelia, technik a najmladšou respondentkou bola 12-ročná dcéra psychoterapeuta. Dôležité je poznamenať, že pri všetkých testoch bol prítomný autor tejto diplomovej práce, ktorý riešil prípadné technické problémy alebo pomáhal účastníkom so správnym nasadením, či nakalibrovaním dátovej prilby.

Obrázok 19: Testovanie aplikácie prvým používateľom počas DOD UMB

Hardvérová zostava, na ktorej sme vykonávali tento experiment pozostávala z procesora Intel Core i9-9900K, 32 GB operačnej pamäte RAM a grafickej karty NVIDIA GeForce GTX 1060 6GB. Na zostave bol nainštalovaný 64-bitový operačný systém Windows 10 Education. Počítačová zostava prenášala výstupný obraz aplikácie do headsetu HTC Vive prostredníctvom bezdrôtového adaptéra. Súčasťou HTC Vive bola taktiež dvojica ovládačov. Pre voľný pohyb sme mali k dispozícii priestor s rozmermi 2,5 m x 2,5 m (6,25 ).

Obrázok 20: Testovanie aplikácie druhým používateľom počas DOD UMB

Každého respondenta sme pred začiatkom testu požiadali, aby v priebehu vykonávania jednotlivých krokov v aplikácii nahlas rozmýšľal a vyjadroval svoje pocity a komentáre k tomu, čo sa vo virtuálnom prostredí momentálne deje. Celkovo sa testovania zúčastnilo 39 respondentov, žiadny z nich neukončil testovanie predčasne, avšak šesť z nich nevyplnilo alebo neodoslalo nami vytvorený elektronický dotazník. Návratnosť dotazníka bola 84,62 %. Ako už bolo spomenuté vyššie, vzorku tvorilo široké spektrum ľudí. Z týchto 33 osôb bolo 24 mužov a 9 žien vo veku od 12 do 65 rokov, pričom najpočetnejšie vekové zastúpenie bolo 21 - 25 rokov. Nadpolovičná väčšina (66,67 %) mala ukončené vysokoškolské vzdelanie I. stupňa. Išlo o pomerne dobre vyváženú vzorku subjektov trpiacich arachnofóbiou (21,21 %), klaustrofóbiou (9,09 %), sociálnou fóbiou (3,03 %), bližšie nešpecifikovanou fóbiou (6,06 %) a úplne zdravých jedincov (60,61 %). Skoro polovica z nich (45,45 %) mala skúsenosť s určitým typom technológie VR. Väčšinou však išlo o pasívnu VR s nízkou mierou interakcie, ako je napríklad mobilná VR alebo 360 stupňové video. Všetci túto predchádzajúcu skúsenosť s VR hodnotili pozitívne. Dĺžka testovania aplikácie nebola vopred určená, ale v priemere každý subjekt strávil vo virtuálnom prostredí približne 16 minút (v rozsahu od 8 do 80 minút).

## Spätná väzba

Po skončení testovania boli uchádzači vyzvaní na vyplnenie anonymného dotazníka prostredníctvom služby Google formuláre, ktorý pozostával z nasledovných sekcií: základné informácie o účastníkoch, fyziologické hľadisko, psychologické hľadisko, ovládateľnosť aplikácie, vizuálne hľadisko a iné. Dotazník obsahoval celkovo 36 otázok, z toho 4 boli otvoreného charakteru. Otázky slúžili na zistenie subjektívnych názorov a pocitov vyplývajúcich z používania našej aplikácie. Celé znenie dotazníka spolu s výsledkami testovania sa nachádzajú v prílohe C.

Vyhodnotením získaných údajov z dotazníka sme dospeli k niektorým zaujímavým záverom. V nasledujúcom texte zhrnieme tie najpodstatnejšie. Majoritná časť používateľov sa úplne (57,58 %) alebo čiastočne (39,39 %) zhodla na tom, že GUI bolo pre nich prehľadné a intuitívne. Vo všeobecnosti sa im interakcia a manipulácia s virtuálnym prostredím javila ako prirodzená. Pozitívne hodnotili aj možnosti a funkcie, ktorými aplikácia disponuje (viď graf 1).

23. Možnosti a funkcie, ktoré poskytovala aplikácia sa mi zdali byť zaujímavé.

Graf 1: Vyhodnotenie dotazníka – otázka č. 23

Menší problém sa vyskytol iba s bezdrôtovými ovládačmi HTC Vive. Viac než polovici používateľov (57,58 %) chvíľu trvalo, kým sa s nimi oboznámili. Pripisovali to najmä tomu, že s nasadenou prilbou nemali možnosť vidieť vlastné ruky ani ovládače a preto občas nevedeli nahmatať tlačidlo. Vhodnou pomôckou preto bolo zoznámenie sa s ovládacími prvkami v hlavnom menu aplikácie, po ktorom boli schopní ju naplno využívať.

Pocit prítomnosti dosiahol pomerne vysokú úroveň (48,48 % subjektov absolútne súhlasilo a 45,45 % čiastočne súhlasilo s tým, že sa cítili byť pohltení virtuálnym prostredím, ktoré ich úplne zapojilo do procesu simulácie). Túto domnienku potvrdzuje aj otázka týkajúca sa vizuálneho aspektu virtuálneho interiérového prostredia, ktoré 51,52 % opýtaných považuje za realistické a 48,48 % za priemerné. Viacerí uvádzali občasné neostré videnie, ktoré mohlo byť spôsobené buď ešte stále nedostatočným rozlíšením displejov alebo niektorou z očných chýb, pretože až 42,42 % respondentov trpí určitou očnou vadou (najčastejšie krátkozrakosťou 30,3 %, ďalekozrakosťou 9,09 % a inou 3,03 %). Avšak aj napriek zisteným nedostatkom bol text nachádzajúci sa v aplikácii (vďaka svojej veľkosti) dostatočne viditeľný a čitateľný.

Nevoľnosť z VR sa meria obťažne, nakoľko jej účinky nepôsobia na každého rovnako a nedá sa zmerať pomocou jedného parametra. Štandardným nástrojom na meranie nevoľnosti je dotazník o nevoľnosti zo simulátora (SSQ) [32]. Ide o pomerne jednoduchú subjektívnu metódu, ktorá má za sebou dlhú históriu používania. V niektorých prípadoch došlo k určitým nepriaznivým účinkom z dôvodu dlhého pobytu vo virtuálnom priestore (v rozsahu od 8 do 80 minút). Namerané výsledky sa nachádzajú v tabuľke 13 a grafe 2. Z grafu 2 môžeme vyčítať, že najčastejšie problémy boli spojené so zrakom (problém so zaostrením 48,48 %, rozmazané videnie 48,48 %, námaha očí 39,39 %) a celkovým pocitom nepohodlia 33,33 %. Zároveň je potrebné poznamenať, že tieto príznaky neboli závažné a zmizli niekoľko minút po testovaní. Zaujímali sme sa taktiež o to, či existuje prepojenie medzi nevoľnosťou vzniknutou z VR a kinetózou pri jazde v automobile. Zistili sme, že 69,7 % respondentov pri jazde v dopravnom prostriedku kinetózou netrpí. Jedna z otázok bola zameraná aj na používateľský komfort s dôrazom na jeho polohu. 36,36 % subjektom bolo počas testovania príjemnejšie stáť, ďalším 33,33 % vyhovovali obe polohy rovnako a 30,3 % subjektov dalo prednosť sedeniu. Používateľský komfort zameraný na polohu subjektu je totiž dôležitý pre správne rozmiestnenie prvkov v scéne.

8. Prejavil sa pri testovaní aplikácie niektorý zo symptómov nevoľnosti? [32]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Symptóm** | **Počet uchádzačov, ktorí vybrali danú možnosť** | | | |
| **Žiadny/a** | **Slabý/á** | **Priemerný/á** | **Závažný/á** |
| 1. Celkový pocit nepohodlia | 22 | 10 | 1 | 0 |
| 1. Malátnosť | 26 | 4 | 1 | 2 |
| 1. Bolesť hlavy | 28 | 4 | 1 | 0 |
| 1. Námaha očí | 20 | 11 | 2 | 0 |
| 1. Problém so zaostrením | 17 | 10 | 6 | 0 |
| 1. Zvýšené slinenie | 30 | 1 | 2 | 0 |
| 1. Potenie | 25 | 5 | 3 | 0 |
| 1. Nevoľnosť | 28 | 4 | 0 | 1 |
| 1. Porucha koncentrácie | 31 | 1 | 1 | 0 |
| 1. Plnosť hlavy | 28 | 3 | 2 | 0 |
| 1. Rozmazané videnie | 17 | 10 | 5 | 1 |
| 1. Závrat (s otvor. očami) | 29 | 3 | 1 | 0 |
| 1. Závrat (so zatvor. očami) | 30 | 3 | 0 | 0 |
| 1. Vertigo | 28 | 2 | 2 | 1 |
| 1. Krátky pocit nevoľnosti | 26 | 5 | 1 | 1 |
| 1. Grganie | 32 | 0 | 1 | 0 |

Tabuľka 13: Vyhodnotenie dotazníka − otázka č. 8

Graf 2: Vyhodnotenie dotazníka – otázka č. 8

Vystavenie sa obávaným objektom, v našom prípade pavúkom vyvolalo u jednotlivých účastníkov rôzne reakcie. Zdravým jedincom tieto objekty nespôsobovali žiadny stres ani napätie, práve naopak evokovali v nich skôr pocity veselosti, príjemnosti, prípadne žiadne emócie. Pre ľudí trpiacich arachnofóbiou používanie aplikácie znamenalo čeliť svojmu strachu. Zo zistených údajov môžeme skonštatovať, že 3D modely pavúkov pohybujúcich sa po stole spôsobovali väčší strach (36,36 % kladných odpovedí), než 2D obrázky pavúkov zobrazené na televíznej obrazovke (24,24 % kladných odpovedí). Stretli sme sa taktiež s respondentmi, ktorí úplne odmietli akúkoľvek interakciu s pavúkmi (dotyk alebo chytenie pavúka do rúk). Úroveň strachu pri vystavení sa obávanému objektu sme merali na stupnici od 1 do 10, kde 1 predstavuje žiadny strach a 10 obrovský strach. Najčastejšie vyskytujúce sa skóre strachu bolo 2, 3 a 7. Jeden z účastníkov sa vyjadril nasledovne: *„Najviac ma vystrašili nepredvídateľné pohyby pavúkov“*. Ukázalo sa teda, že náš systém môže byť užitočný na vyvolanie úzkosti u účastníkov. Rovnako graf 3 naznačuje, že dlhší pobyt vo virtuálnom prostredí má priaznivé účinky na zmiernenie strachu a úzkosti. Výsledky získané z tohto výskumu poukazujú na obrovský potenciál VR v liečbe špecifických fóbií, konkrétne arachnofóbie (viď graf 4). Až 84,85 % respondentov by bolo ochotných podstúpiť liečbu fóbie takouto formou. Analyzovali sme taktiež to, či by túto aplikáciu odporučili psychoterapeutom a ich pacientom. Väčšina respondentov (90,91 %) by ju odporučila.

18. Cítil/a som, že dlhším pobytom vo virtuálnom prostredí sa znižovala úroveň strachu a stresu (negatívne pocity).

Graf 3: Vyhodnotenie dotazníka – otázka č. 18

10. Myslíte si, že aplikácia môže byť nápomocná pri liečbe arachnofóbie?

Graf 4: Vyhodnotenie dotazníka – otázka č. 10

Celkové hodnotenie a spokojnosť používateľov s aplikáciou je vyjadrená v grafe 5. Len jeden zo subjektov mal z aplikácie zmiešané pocity. Prijatá spätná väzba bola veľmi pozitívna a zároveň nápomocná. Aj vďaka nej sa nám podarilo odhaliť niekoľko drobných nedostatkov, ktoré boli odstránené pred začiatkom nasledovného testovania. Okrem iného priniesla aj nový pohľad na vec a cenné návrhy na vylepšenie aplikácie. Väčšina používateľov si pochvaľovala interakciu s objektmi a možnosť chytania pavúkov do rúk. Iní zasa vyzdvihli kreatívnosť nápadu a samotné spracovanie projektu. Boli nadšení z toho, ako ich simulované prostredie dokázalo vnoriť do iného sveta. Ako uviedol jeden z respondentov: *„Bolo to veľmi reálne. Páčila sa mi možnosť vytvorenia rôznych druhov pavúkov. V podstate si dokážete nasimulovať pavúka, ktorého sa najviac bojíte a následne sa tento strach snažíte prekonať“.*

32. Celkovo sa mi aplikácia páčila.

Graf 5: Vyhodnotenie dotazníka – otázka č. 32

Samozrejme boli sme otvorení aj konštruktívnej kritike. Účastníci poukazovali hlavne na slabé rozlíšenie headsetu a obmedzenosť pohybu po virtuálnej miestnosti. Akýkoľvek spôsob pohybu, akým je napríklad teleportácia sme však nezahrnuli schválne, aby sme obmedzili vznik pohybovej choroby na minimum. Podľa účastníkov by priestor na zlepšenie malo dostať grafické prevedenie menu s možnosťami (tabuľa s GUI). To sa nám podarilo vo finálnej verzii upraviť na prehľadnejšie a intuitívnejšie. Čo sa týka odporúčaní do budúcej verzie aplikácie, najčastejšie sa opakovala žiadosť o pridanie viacerých prostredí, kde by si používateľ mohol vybrať jedno z nich (napríklad obývacia izba, kancelária, ordinácia, záhrada a podobne). Taktiež by podľa nich bolo vhodné zahrnúť aj ďalšie fóbie zo zvierat. Spomínali tiež pridanie viacerých možností interakcie so systémom alebo zakomponovanie určitých úrovní (rôznych vzdialeností pozorovateľa od pavúka). Posledný námet prišiel od samotného odborníka v tejto oblasti – psychoterapeuta, ktorý poznamenal: *„V budúcnosti by bolo vhodné vykonať testovanie aplikácie s použitím časovača s možnosťou dychového metronómu“.*

Z týchto odporúčaní sme vybrali tie najužitočnejšie a implementovali ich do výslednej aplikácie. Tie, ktoré sa v aplikácii nenachádzajú ďalej opisujeme v závere tejto práce ako možné rozšírenie.

ZÁVER

Predložená diplomová práca sa zaoberá návrhom a následnou tvorbou interaktívnej aplikácie virtuálnej reality, ktorá už začala slúžiť na liečbu vybranej fóbie, konkrétne arachnofóbie. Preukázalo sa, že náš softvér dokáže vyvolať úzkostné stavy a môže byť účinný pri liečbe tohto chorobného strachu z pavúkov. Psychoterapeut, ktorý s nami spolupracoval od návrhu aplikácie až po jej testovanie zahájil liečbu so skutočnými pacientmi a zatiaľ sa mu darí viac než dobre. Ide však o veľmi zložitý a dlhodobý proces, preto výsledky nemôžeme očakávať hneď po prvých sedeniach.

V teoretickej časti diplomovej práce sme sa oboznámili s problematikou súčasného stavu digitálnych realít, kde sme definovali virtuálnu, rozšírenú a zmiešanú realitu. Následne sme vysvetlili vznik týchto termínov a poukázali na ich zásadné rozdiely. V druhej kapitole sme opísali jednu z oblastí úzkostných porúch, akými sú fóbie. Zamerali sme sa na kategóriu špecifických fóbií, do ktorých spadá aj spomínaná arachnofóbia. V poslednej časti teórie sme rozobrali najpoužívanejšie metódy liečby špecifických fóbií, pričom dôraz sme kládli na liečbu fóbií pomocou virtuálnej reality.

Praktická časť práce obsahuje opis všetkých etáp vývojového procesu aplikácie od zberu požiadaviek až po spätnú väzbu od odborníka na KBT a eventuálnych pacientov. Významná časť je venovaná podrobnému postupu tvorby riešenia, ktorá môže slúžiť pre vzdelávacie účely, ako návod na vývoj virtuálno-realitných aplikácií. Súčasťou práce je celistvý projekt multimediálnej aplikácie s názvom Arachnofóbia a taktiež výsledky výskumu, ktorý prebiehal pomocou dotazníkovej metódy.

Prínos našej práce spočíva v tom, že prostredníctvom nami vyvinutej aplikácie sa začalo pomáhať tým, ktorí to skutočne potrebujú. Okrem iného sa aplikácia dá využiť pri rôznych školských akciách ako sú prezentácie, popularizačné účely, workshopy, dni otvorených dverí a podobne. Aplikácia spĺňa všetky stanovené požiadavky, z čoho usudzujeme, že cieľ práce sme naplnili a veríme, že práca bude pre tieto účely prospešná aj naďalej. Práca má do budúcnosti potenciál pre vylepšenia a rozšírenia. Napríklad pridanie ďalších fóbií v podobe nezávislých modulov, hlas virtuálneho terapeuta, databáza pacientov s užitočnými informáciami, lokalizácia do cudzích jazykov, podpora pre iné platformy, pozitívne myšlienky, biofeedback a mnohé ďalšie.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

[1] MITTER, M. *Úlohy v kurze Virtuálna realita v prostredí LMS Moodle*. Banská Bystrica, 2018. Bakalárska práca. Univerzita Mateja Bela v Banskej Bystrici, Fakulta prírodných vied, Katedra informatiky. Vedúci práce Ing. Dana Horváthová, PhD.

[2] HORVÁTHOVÁ, D., VOŠTINÁR, P. a MITTER, M. Using Blender 3D for learning virtual and mixed reality, In: *ICERI2019 Proceedings*, 2019, s. 9309-9314, ISBN 978-84-09-14755-7.

[3] VOŠTINÁR, P., HORVÁTHOVÁ, D., MITTER, M. a BAKO, M. Possibilities of human interaction in the virtual environment, In: *INFORMATICS 2019*, 2019, s. 456-460. ISBN 978-1-7281-3178-8.

[4] SCRIBANI, J. What is Extended Reality (XR)? *Visual Capitalist* [online]. Vancouver, 2019 [cit. 2020-04-01]. Dostupné z: <https://www.visualcapitalist.com/extended-reality-xr/>

[5] MILGRAM, P., TAKEMURA, H., UTSUMI, A. a KISHINO, F. Augmented reality: A class of displays on the reality-virtuality continuum, In: *Proceedings of SPIE*, 1994, s. 282-292

[6] AUKSTAKALNIS, S. a BLATNER, D. *Silicon Mirage: The Art and Science of Virtual Reality*. Berkeley, CA: Peachpit Press, c1992. ISBN 0-938151-82-7.

[7] BOTELLA, C., PÉREZ-ARA, M. Á., BRETÓN-LÓPEZ, J., QUERO, S., GARCÍA-PALACIOS, A. a BAÑOS, R. M. In Vivo versus Augmented Reality Exposure in the Treatment of Small Animal Phobia: A Randomized Controlled Trial. *PLOS ONE* [online]. 2016, 11(2) [cit. 2020-03-20]. DOI: 10.1371/journal.pone.0148237. ISSN 1932-6203. Dostupné z: <https://dx.plos.org/10.1371/journal.pone.0148237>

[8] MEALY, P. *Virtual & Augmented reality for dummies*. Indianapolis, In: John Wiley and Sons., 2018. ISBN 978-1-119-48134-8.

[9] AZUMA, R. T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments* [online]. 1997, 6(4), 355-385 [cit. 2020-03-30]. DOI: 10.1162/pres.1997.6.4.355. ISSN 1054-7460. Dostupné z: <http://www.mitpressjournals.org/doi/10.1162/pres.1997.6.4.355>

[10] RAHANI, V. K., VARD, A. a NAJAFI, M. Claustrophobia Game: Design and Development of a New Virtual Reality Game for Treatment of Claustrophobia. *J Med Signals Sens* [online]. 2018 Oct-Dec, 8(4), 231-237 [cit. 2020-03-20]. Dostupné z: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6293643/>

[11] STĂNICĂ, I.-C., DASCĂLU, M.-I., MOLDOVEANU, A. a MOLDOVEANU, F. An Innovative Solution Based on Virtual Reality to Treat Phobia. *International Journal of Interactive worlds* [online]. 2017, 1-13 [cit. 2020-03-20]. DOI: 10.5171/2017.155350. ISSN 21659508. Dostupné z: <https://ibimapublishing.com/articles/IJIW/2017/155350/>

[12] KORNARAKIS, I. *A virtual reality serious game platform to support treatment of phobia patients*. Crete, 2016. Master's theses. ΤΕΙ of Crete, School of Applied Technology, Department of Applied Informatics and Multimedia.

[13] PATEL, V., SAXENA, S., LUND, C. et al. The Lancet Commission on global mental health and sustainable development. *The Lancet* [online]. 2018, 392(10157), 1553-1598 [cit. 2020-03-20]. DOI: 10.1016/S0140-6736(18)31612-X. ISSN 01406736. Dostupné z: <https://linkinghub.elsevier.com/retrieve/pii/S014067361831612X>

[14] INSEL, T. *The Global Cost of Mental Illness*. National Institute of Mental Health [online]. 2011 [cit. 2020-03-20]. Dostupné z: <https://www.nimh.nih.gov/about/directors/thomas-insel/blog/2011/the-global-cost-of-mental-illness.shtml>

[15] *Diagnostic and statistical manual of mental disorders DSM-5tm*. 5th ed. Washington, DC: American Psychiatric Publishing, c2013. ISBN 978-0-89042-555-8.

[16] CULBERTSON, F. *The Phobia List* [online]. 2016 [cit. 2020-03-20]. Dostupné z: <http://www.phobialist.com/>

[17] MILOFF, A., LINDNER, P., HAMILTON, W., REUTERSKIÖLD, L., ANDERSSON, G. a CARLBRING, P. Single-session gamified virtual reality exposure therapy for spider phobia vs. traditional exposure therapy: study protocol for a randomized controlled non-inferiority trial. *Trials* [online]. 2016, 17(1) [cit. 2020-03-20]. DOI: 10.1186/s13063-016-1171-1. ISSN 1745-6215. Dostupné z: <http://www.trialsjournal.com/content/17/1/60>

[18] WITTCHEN, H.-U., LECRUBIER, Y., BEESDO, K. a NOCON., A. Relationships Among Anxiety Disorders: Patterns and Implications. NUTT, D.J. a J.C. BALLENGER, ed. *Anxiety Disorders* [online]. Oxford, UK: Blackwell Science, 2002, s. 23-37 [cit. 2020-03-21]. DOI: 10.1002/9780470986844.ch2. ISBN 9780470986844. Dostupné z: <http://doi.wiley.com/10.1002/9780470986844.ch2>

[19] NIECHWIADOWICZ, K. *Virtual Reality and Game Mechanics in Generalized Social Phobia Treatment*. Netherlands, 2017. Master's theses. University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

[20] BLACK, R. Arachnophobia: Fear of Spiders and How to Overcome It. *Psycom* [online]. 2019 [cit. 2020-03-23]. Dostupné z: <https://www.psycom.net/arachnophobia-fear-of-spiders>

[21] HOFFMAN, H. G. Virtual-Reality Therapy. *Scientific American* [online]. 2004, 291(2), 58-65 [cit. 2020-03-24]. ISSN 0036-8733. Dostupné z: <http://www.nature.com/doifinder/10.1038/scientificamerican0804-58>

[22] GARCIA-PALACIOS, A., HOFFMAN, H. G., KWONG SEE, S., TSAI, A. a BOTELLA, C. Redefining Therapeutic Success with Virtual Reality Exposure Therapy. *CyberPsychology & Behavior* [online]. 2001, 4(3), 341-348 [cit. 2020-03-24]. DOI: 10.1089/109493101300210231. ISSN 1094-9313. Dostupné z: <http://www.liebertpub.com/doi/10.1089/109493101300210231>

[23] Unveiling the Vive Consumer Edition and Pre-order Information. In: *Vive* [online]. Xindian District, 2016 [cit. 2020-01-11]. Dostupné z: <https://blog.vive.com/us/2016/02/21/unveiling-the-vive-consumer-edition-and-pre-order-information/>

[24] VERTIGO GAMES. Design Furniture Pack. In: *Unity: Asset Store* [online]. 2018 [cit. 2020-03-02]. Dostupné z: <https://assetstore.unity.com/packages/3d/props/interior/design-furniture-pack-20147>

[25] KKMPROJECTS. Arachnophobia. In: *GitHub* [online]. 2016 [cit. 2020-03-02]. Dostupné z: <https://github.com/kkmProjects/arachnophobia>

[26] VIVESOFTWARE. VIVE Input Utility. In: *Unity: Asset Store* [online]. 2019 [cit. 2019-10-13]. Dostupné z: <https://assetstore.unity.com/packages/tools/integration/vive-input-utility-64219>

[27] VIVESOFTWARE. ViveInputUtility-Unity. In: *GitHub* [online]. 2019 [cit. 2019-10-13]. Dostupné z: <https://github.com/ViveSoftware/ViveInputUtility-Unity>

[28] NATUREMANUFACTURE. VR Hands and FP Arms Pack. In: *Unity: Asset Store* [online]. 2016 [cit. 2019-10-16]. Dostupné z: <https://assetstore.unity.com/packages/3d/characters/humanoids/vr-hands-and-fp-arms-pack-77815>

[29] UNITY, Technologies. Input for OpenVR controllers: Unity input system mappings. *Unity3d: Documentation* [online]. 2019 [cit. 2019-10-17]. Dostupné z: <https://docs.unity3d.com/2019.1/Documentation/Manual/OpenVRControllers.html>

[30] 3DRT.COM. Spiders Pack. In: *Unity: Asset Store* [online]. 2013 [cit. 2019-11-04]. Dostupné z: <https://assetstore.unity.com/packages/3d/characters/animals/spiders-pack-16546>

[31] Blob Run Code Samples: *IE Numerator for Wander AI* [online]. [cit. 2019-11-28]. Dostupné z: <https://blobrun829143539.wordpress.com/code-samples/>

[32] KENNEDY, R. S., LANE, N. E., BERBAUM, K. S., a LILIENTHAL, M. G. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology* [online]. 1993, **3**(3), 203-220 [cit. 2019-10-19]. DOI: 10.1207/s15327108ijap0303\_3. ISSN 1050-8414. Dostupné z: <http://www.tandfonline.com/doi/abs/10.1207/s15327108ijap0303_3>

Obsah

[1 METODIKA TVORBY VR APLIKÁCIE 1](#_Toc53749929)

[1.1 Špecifikácia požiadaviek 1](#_Toc53749930)

[1.1.1 Katalóg funkcionálnych požiadaviek 2](#_Toc53749931)

[1.1.2 Katalóg nefunkcionálnych požiadaviek 3](#_Toc53749932)

[1.1.3 Prípady použitia aplikácie 6](#_Toc53749933)

[1.2 Použité technológie 7](#_Toc53749934)

[1.2.1 HTC Vive 7](#_Toc53749935)

[1.2.2 Unity 3D 9](#_Toc53749936)

[1.2.3 Microsoft Visual Studio 10](#_Toc53749937)

[1.2.4 Blender 11](#_Toc53749938)

[1.2.5 Adobe Photoshop 12](#_Toc53749939)

[1.2.6 Adobe Illustrator 12](#_Toc53749940)

[1.2.7 Audacity 12](#_Toc53749941)

[1.3 Návrh riešenia 13](#_Toc53749942)

[1.3.1 Návrh scenára 16](#_Toc53749943)

[1.4 Implementácia riešenia 17](#_Toc53749944)

[1.4.1 Zoznámenie sa s prostredím Unity 3D 17](#_Toc53749945)

[1.4.2 Prvotné nastavenie projektu 19](#_Toc53749946)

[1.4.3 3D modely rúk 21](#_Toc53749947)

[1.4.4 Rozpoznanie vstupných zariadení v scéne 22](#_Toc53749948)

[1.4.5 Animácie modelov rúk 22](#_Toc53749949)

[1.4.6 Ovládanie animácií modelov rúk 24](#_Toc53749950)

[1.4.7 Funkcionalita chytania a hádzania objektov 26](#_Toc53749951)

[1.4.8 Modelovanie prostredia 30](#_Toc53749952)

[1.4.9 3D model pavúka 33](#_Toc53749953)

[1.4.10 Ovládanie animácií modelu pavúka 34](#_Toc53749954)

[1.4.11 Pohyb modelu pavúka 38](#_Toc53749955)

[1.4.12 Používateľské rozhranie aplikácie 40](#_Toc53749956)

[1.4.13 Tvorba aplikačnej logiky 42](#_Toc53749957)

[1.5 Používateľské testovanie aplikácie 44](#_Toc53749958)

[1.6 Spätná väzba 46](#_Toc53749959)

[ZÁVER 52](#_Toc53749960)

[ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV 53](#_Toc53749961)

1. Tento priestor môže byť aj dvojrozmerný po prepnutí tlačidlom *2D*. [↑](#footnote-ref-1)
2. Tracker – doplnkový bezdrôtový senzor. [↑](#footnote-ref-2)
3. RMB > *Create Empty*. [↑](#footnote-ref-3)
4. RMB > *Create Empty*. [↑](#footnote-ref-4)
5. RMB > *Create Empty*. [↑](#footnote-ref-5)
6. Pre naše potreby sme veľkosť pavúka prednastavili na hodnotu 0.03 v každom smere (X, Y, Z). [↑](#footnote-ref-6)