VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTUITIVNÍ OVLÁDÁNÍ STRATEGICKÝCH HER NA DOTYKOVÝCH ZAŘÍZENÍCH

INTUITIVE CONTROL OF STRATEGY GAMES ON TOUCH DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK HAAS**
AUTHOR

VEDOUCÍ PRÁCE **Ing. DANIEL BAMBUŠEK**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Haas Patrik**
Program: Informační technologie
Název: **Intuitivní ovládání strategických her na dotykových zařízeních**
Intuitive Control of Strategy Games on Touch Devices
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte současné přístupy uživatelské interakce u strategických her na dotykových zařízeních. Seznamte se s přenosným dotykovým projektorem Hachi Infinite M1 pro promítanou rozšířenou realitu.
2. Vyberte vhodné metody a nástroje a navrhněte strategickou hru využívající sadu uživatelsky přívětivých dotyků a gest na zařízení Hachi Infinite M1.
3. Navrženou aplikaci implementujte.
4. Proveďte uživatelské experimenty a vyhodnoťte vlastnosti výsledného řešení.
5. Vytvořte video prezentující klíčové vlastnosti výsledného řešení.

Literatura:

- SCHMALSTIEG Dieter, HÖLLERER Tobias. *Augmented Reality: Principles and Practice.* Addison-Wesley, 2016. ISBN 978-0321883575.
- HARTSON Rex. *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. 2012. ISBN 9780123852427.
- Dále dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bambušek Daniel, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 21. dubna 2022

Abstrakt

Cieľom tejto práce je poskytnúť odpoveď na otázku, ako vytvoriť prívetivo ovládateľnú stratégiu pre dotykové zariadenie a aké gestá by bolo vhodné použiť. Táto práca poskytuje odpoveď na danú otázku a to takým spôsobom, že sa tu nachádza ako naprogramovať takúto hru, a aké ovládanie je najlepšie zvoliť a prečo. Taktiež sa tu nachádza implementácia jednotlivých gest pre ovládanie.

Abstract

The aim of this work is to provide an answer to the question of how to create a user-friendly controllable strategy game on a touch device and what gestures should be used. This work provides an answer to the question in such a way that it is here how to program such a game, and what controls are best to choose and why. There is also an implementation of individual gestures for control.

Klíčová slova

rozšírená realita, premietaná rozšírená realita, strategická hra, dotyková plocha, ovládanie strategických hier, Hachi Infinite M1

Keywords

augmented reality, projected augmented reality, strategy game, touch surface, control of strategy games, Hachi Infinite M1

Citace

HAAS, Patrik. *Intuitivní ovládání strategických her na dotykových zařízeních.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Daniel Bambušek

Intuitivní ovládání strategických her na dotykových zařízeních

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Daniela Bambuška. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje z ktorých som čerpal.

.....
Patrik Haas
10. května 2022

Poděkování

Chcel by som poďakovať svojmu vedúcemu Ing. Danielovi Bambuškovi za výborné vedenie, rady a trpezlivosť pri tvorbe tejto práce. Taktiež chcem poďakovať svojej rodine za podporu, gramatickú korekciu a testovanie. Nakoniec chcem poďakovať všetkým účastníkom experimentu za ich ochotu.

Obsah

Kapitola 1

Úvod

V dnešnej dobe sa zvyšuje počet hier na mobily, tablety a teda na zariadenia s dotykovou plochou. Preto je potrebné, aby ovládanie na týchto zariadeniach bolo čo najvernejšou kópiou počítačovej verzie. Na počítači má hráč k dispozícii viacero možností ako ovládať hru, zatiaľ čo na dotykových plochách je k dispozícií len prst.

Keďže z množstva herných žánrov je mi najbližšia stratégia, cieľom tejto práce je pomocou experimentov nájsť čo najlepšiu možnú sadu ovládaní pre strategické hry na dotykových zariadeniach. To samozrejme vyžaduje vytvorenie funkčnej demo verzie hry, na ktorej sa môže experimentovať. Zvolené dotykové zariadenie je Hachi Infinite M1, pretože je revolučné a to tým spôsobom, že dokáže vytvárať dotykové plochy skoro na každej ploche.

V tejto práci sú najprv uvedené informácie o rozšírenej premietanej realite, následne jej použitiami a potom predstavením zástupcu a to Hachi Infinite M1. Ďalej je uvedený prieskum ovládania pri strategických hrách pre dotykové zariadenia. Ku koncu je uvedený návrh a implementácia tejto demo hry, a celá práca je zakončená vyhodnotením experimentom.

Kapitola 2

Teoretický úvod

V tejto kapitole je vysvetlený pojem rozšírená realita a virtuálna realita. Rozšírenej realite je venovaný väčší priestor. Pre ňu je vysvetlený spôsob jej zobrazenia, technológie ktoré používa a aj jej aktuálne využitie. Taktiež sú tu popísané zaujímavé časti histórie rozšírenej reality. Následne sa tu nachádza popis zariadenia Hachi Infinite M1, čo je zariadenie pre premietanie rozšírenej reality na ktoré je táto hra primárne určená. Na záver je uvedený prieskum súčasných spôsobov užívateľského ovládania strategických hier. Tento prieskum slúži na ukázanie rôznych spôsobov ovládania, kde niektoré budú použité v demo hre. Je to ukážka ako teraz vyzerajú rôzne strategické hry na dotykových zariadeniach.

2.1 Úvod do rozšírenej a virtuálnej reality

V roku 1977 vo filme Star Wars filmári ukázali 3D zobrazenie princeznej Leily, ako rozpráva nahovorenú správu. Toto bolo samozrejme vytvorené pomocou špeciálnych efektov. Avšak o 30 rokov bola v štúdiu CNN diskusia medzi dvoma moderátormi, kde jeden z nich sedel v štúdiu a druhý sa nachádzal tisíce míľ ďalej v inom meste. Tá druhá osoba bola zobrazená ako 3D obraz. V tomto prípade sa jednalo o skutočný obraz, ktorý bol zobrazený pomocou rozšírenej reality. Títo moderátori viedli diskusiu, akoby boli naozaj spolu v jednej miestnosti. Na obrázku 2.1 je možné pozorovať ich diskusiu. V konečnom dôsledku stačilo 30 rokov aby sa z Sci-fi stala realita [8].

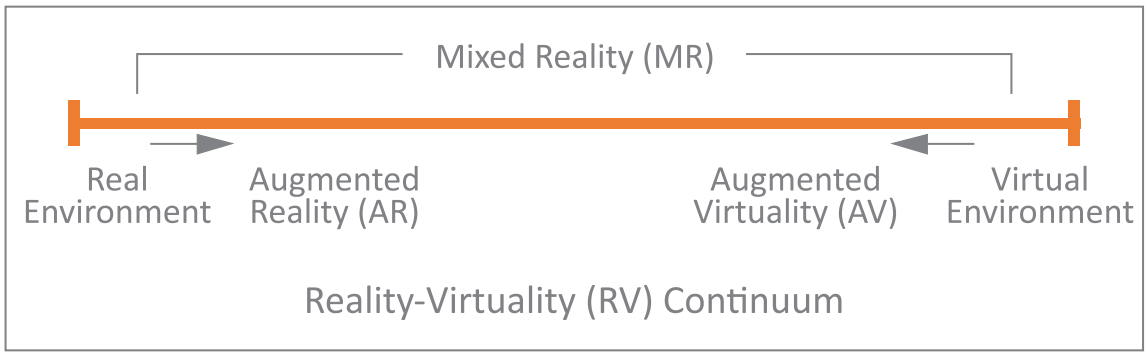
Rozšírená realita je variáciou virtuálnej reality. Na rozdiel od virtuálnej reality, ktorá užívateľa úplne ponorí do virtuálneho sveta a teda mu znemožní vidieť ten skutočný, je rozšírená realita menej reštriktívna. Neponára užívateľa úplne do virtuálneho sveta, len do skutočného sveta umiestňuje virtuálne objekty. Jej najlepší prípad je, keď užívateľovi pripadá, že virtuálne a reálne objekty sú spolu súčasťou reálneho sveta [3].

Pre virtuálnu realitu platí vytváranie vlastných fyzikálnych zákonov. Taktiež sa tu môžu nachádzať veci, ktoré patria do žánru sci-fi. Jednoducho povedané, vo virtuálnej realite je možné vidieť a vytvoriť ľubovoľné veci. Samozrejme to všetko v réžii virtuálneho sveta [12]. Napríklad môže existovať hra, v ktorej cieľom je užívateľovi umožnené lietať a všetky zvukové či vizuálne podnety ktoré tento svet poskytuje, stimulujú užívateľove zmysly aby mal pocit letu.

Rozdiel medzi rozšírenou a virtuálnou realitou je značný, na obrázku 2.2 je možné vidieť graf, ktorý popisuje tento rozdel. Tento obrazok bol prebraný od P. Milgrama a F. Kishino [13]. Z grafu je možné vyčítať, že rozšírená a virtuálna realita sa nachádzajú na dvoch opačných koncoch, teda sa jedná o dva extrémy. Keďže rozšírená realita len pridáva objekty do sku-

Obrázek 2.1: Obrázok konverzácie dvoch moderátorov [9].

točného sveta, je tak bližšie k reálnemu svetu. V tomto prípade užívateľ potrebuje nejaké zariadenie, cez ktoré bude schopný vidieť tieto nové objekty. Tieto zariadenia teda zobrazujú reálny svet ako aj virtuálny spolu. Na druhom konci je virtuálna realita a ako už bolo spomenuté vyššie, úplne ponára užívateľa do virtuálneho sveta. Na toto užívateľ potrebuje okuliare pre virtuálnu realitu, ktoré to ponorenie umožnia [1].



Obrázek 2.2: Milgramov graf rozdelenia rozšírenej a virtuálne reality [12].

Podľa Ronalda Azuma, je možné definovať rozšírenú realitu ako systém, ktorý má nasledujúce charakteristiky [3] :

1. Systém kombinuje skutočnú a virtuálnu realitu.
2. Systém je interaktívny v reálnom čase.
3. Je registrovaný v 3D.

Teraz je možné použiť túto klasifikáciu na prípad zo CNN. Prvý bod spĺňa, nakoľko bolo možné vidieť spolu v štúdiu moderátora a moderátorku ako 3D obraz. Taktiež spĺňa aj druhý bod, pretože ich konverzácia prebiehala v reálnom čase. A po tretie, v štúdiu bol zobrazený obraz, ktorý bol v 3D prevedení. Výsledkom podľa tejto klasifikácie je, že systém, ktorý vtedy CNN použilo, bola rozšírená realita [8].

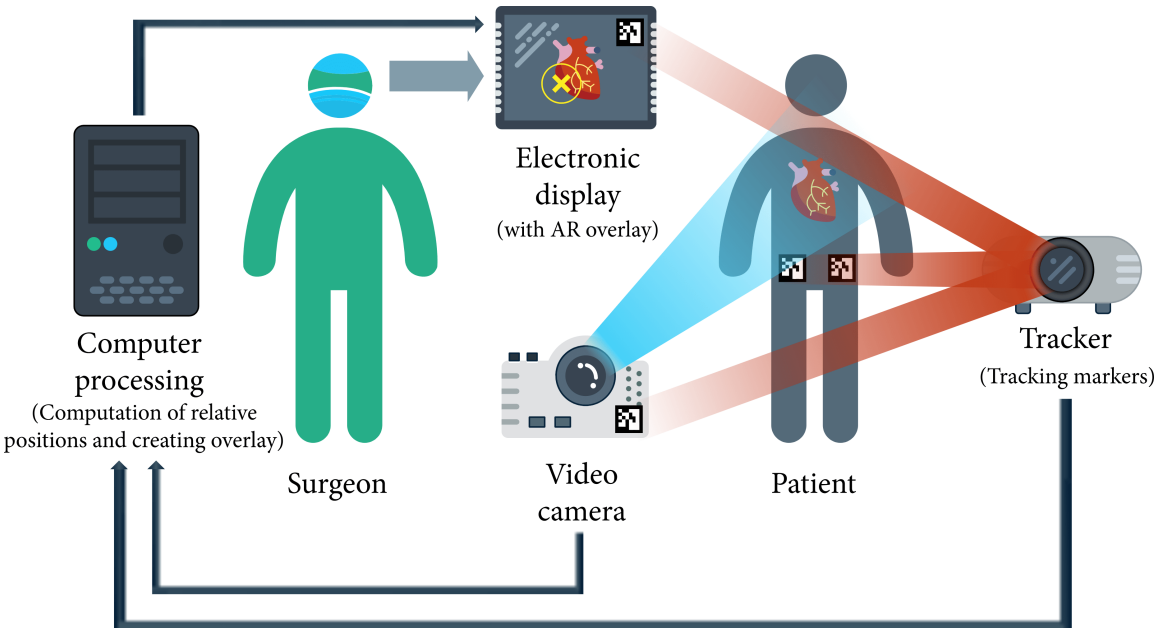
Tieto tri charakteristiky tiež definujú technické parametre na systém rozšírenej reality. A to tak, že tento systém má displej, ktorý kombinuje skutočné a virtuálne obrazy. Ďalej počítačový systém, ktorý dokáže generovať interaktívnu grafiku, ktorá vie odpovedať na užívateľské vstupy v reálnom čase a na koniec systém, ktorý vie umiestniť virtuálny obraz do reálneho sveta [8].

2.2 Využitie rozšírenej a premietanej reality

Rozšírená realita sa využíva v rôznych oblastiach. Jej využitie sa niekde už stáva bežnou praxou. Bude spomenuté využitie, respektíve možné využitie, rozšírenej reality v oblasti medicíny, vzdelávania a potom budú nasledovať konkrétne produkty premietanej rozšírenej reality.

Využitie v medicíne

Najnovšie objavy v medicíne sa zameriavajú na získavanie dát, ktoré by sa mohli zobrazovať v reálnom čase. Keďže dnešné spôsoby, ako röntgen apod., vyžadujú čas na ich spracovanie, tak systém rozšírenej reality, ktorý by mohol chirurgovi zobrazovať dáta v reálnom čase, je veľmi žiadúci. Tento systém, ktorého princíp je zobrazený na obrázku 2.3, by to dokázal. Základnou myšlienkou je pomocou neho vygenerovať počítačový snímok, ktorý by prekrýval pacienta a ukazoval pre chirurga najdôležitejšie informácie [16].



Obrázek 2.3: Schéma základného princípu rozšírenej reality v medicíne [16].

Tento systém je najlepšie používať pri operáciách, či rekonštrukčných operáciach, orgánov s malým pohybom ako je napríklad mozog a podobne. Pri pohybujúcich orgánoch, ako sú črevá, by bol potrebný veľmi výkonný systém, ktorý by aj tie najmenšie zmeny dokázal zobrazovať v reálnom čase. I keď tieto systémy napredujú míľovými krokmi, stále je veľa prekážok, ktoré treba prekonať. Napríklad pri rekonštrukčných operáciach je potrebné dopredu vytvoriť obrazy pomocou zložitých algoritmov, ktoré vyžadujú veľký výkon [16].

Využitie v oblasti vzdelávania

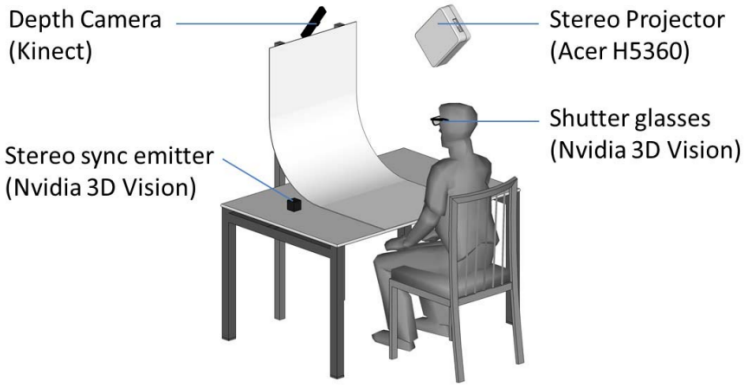
Pri výučbe sa v dnešnej dobe používajú dvojrozmerné média. Dôvodom je, že sú zaužívané a pohodlné, avšak neponúkajú dostatočnú dynamičnosť a teda študent si to tým pádom nemusí vedieť veľmi dobre predstaviť v praxi. Je možné využívať virtuálnu realitu, no jej nevýhodou je, že je zvyčajne veľmi drahá a neposkytuje dostatočný realizmus a študent je odtrhnutý od reálneho sveta. Preto rozšírená realita vychádza ako jeden z najlepších možných nástrojov pre výučbu. Tento systém zvyšuje atraktívnosť a efektívnosť vzdelávania a umožňuje študentom vidieť viac reálnejšie situácie, s ktorými sa môže stretnúť v praxi [10].

Ako príklad je možné uviesť študentov stavebného manažmentu. Títo študenti sa väčšinou nestretnú so stavebnými procesmi a procedúrami pri výučbe. Jednoducho povedané, nemajú takú možnosť si to „chytiť“ do rúk. I keď môžu mať exkurzie, ktoré by im mohli ponúknuť väčší náhľad do problematiky, väčšinou nedokážu poskytnúť dostatočné detaily, čo môže eventuálne viesť k nedostatku praktických vedomostí [4].

Ayer a spol. [2] uskutočnili štúdiu, kde sa skupinám študentov stavebníctva zadala úloha predizajnovať budovu. Niektoré skupiny použili obyčajný papier, pričom druhé použili hru v rozšírenej realite na dizajn. Výsledkom bolo, že skupiny, ktoré použili danú hru, vytvorili inovatívnejšie a kreatívnejšie návrhy, čím dosiahli celkovo lepší výkon ako študenti ktorí použili len papier [4].

MirageTable

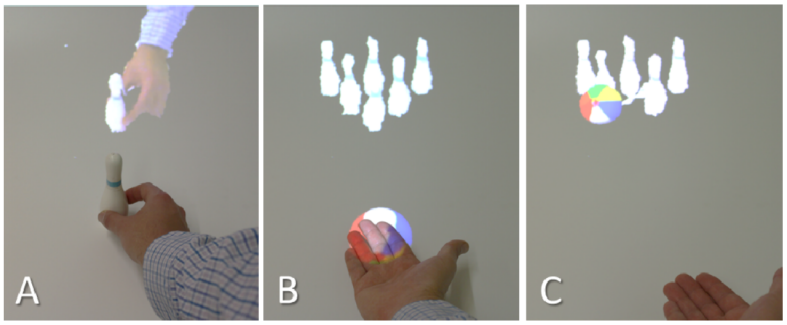
MirageTable je produkt zložený z jednej hĺbkovej kamery, projektoru a zakrivenej obrazovky. Je to interaktívny systém, ktorého úlohou je spojenie reálneho a virtuálneho sveta. Tento systém pomocou hĺbkovej kamery zachytáva oči užívateľa a taktiež zachytáva tvar a vzhľad ľubovoľného predmetu pred kamerou v reálnom čase. Toto umožňuje zobrazovať dané predmety v 3D prevedení pre jedného užívateľa korektne. Taktiež môže užívateľ bez akýchkoľvek rukavíc, či snímačov, interagovať s virtuálnymi predmetmi. Na obrázku 2.4 je možné pozorovať ako tento systém vyzerá [7].



Obrázek 2.4: Systém MirageTable [7].

Tento systém tiež umožňuje položiť reálne objekty na stôl, kde je systém MirageTable, a vytvoriť 3D virtuálnu kópiu daného objektu. Napríklad môže užívateľ prísť s bowlingovou figúrkou a zduplikovať si ju ako 3D virtuálne obrazy a zahrať si bowling. Toto je možné

pozorovať na obrázku 2.5 [7]. Tvorcovia daného produktu taktiež vytvorili video, ktoré znázorňuje všetky možné interakcie s MirageTable[1].



Obrázek 2.5: Bowling pomocou MirageTable [7].

Hracie zariadenia pre premietanú rozšírenú realitu

Premietaná rozšírená realita má veľké využitie pri zariadeniach, ktoré sú určené pre deti. Sú to zariadenia, ktoré umožňujú deťom sa hrať. Sú to napríklad interaktívna plocha, interaktívna strelecká galéria a interaktívne pieskovisko. Ďalej uvediem len interaktívnu streleckú galériu a interaktívne pieskovisko.

Ako prvý zo systémov, ktorý uvediem je Interaktívna strelecká galéria[2]. Tento systém ponúka interaktívnu stenu, ktorá vytvára transformovateľné herné prostredie. Toto zariadenie sa skladá z výkonného počítača, projektoru, pohybových senzorov, klávesnice s trackpadom, niekoľko hier a laserových zbraní. Na obrázku 2.6, ktorý je prebraný z webu[3] je možné pozorovať, ako tento systém vyzerá.

Ako ďalší systém pre hranie, ktorý používa premietanú rozšírenú realitu je interaktívne pieskovisko[4], po anglicky interactive sandbox. Toto zariadenie umožňuje meniť topografiu oblasti, ktorá je premietaná na pieskovisko. Ovládanie je jednoduché. Používatelia budú premiestňovať, hromadiť, alebo vykopávať piesok, pričom to budú interaktívne senzory sledovať a v reálnom čase prispôsobovať projekciu. Takto môžu vzniknúť hory, pláne a vodné plochy a ak sa tu premietajú aj nejaké zvieratá alebo predmety, tie sa budú prispôsobovať novej topografii. Toto môže užívateľov nepochybne naučiť rôzne či už nové, alebo celkové informácie o topografii. Na obrázku 2.6, ktorý je prebraný z webu[5], je možné vidieť ako tento systém vyzerá. Bod 1 reprezentuje projektor, ktorý je pripevnený na strop a ten premieta obsah na pieskovisko pod ním. Bod 2 je senzor, ktorý sníma zmeny urobené užívateľmi. A posledný bod číslo 3 je samotné pieskovisko.

[1] https://dl.acm.org/doi/abs/10.1145/2207676.2207704#video_stream_uuid%3A3b23a7ca-bfa8-46ca-8fde-05a097de3996

[2] https://kidsjumptech.com/interactive-shooting-gallery/

[3] https://kidsjumptech.com/wp-content/uploads/2022/02/Interactive_ShootingGallery.svg

[4] https://www.breezecreative.com/animated-sandbox

[5] https://static.wixstatic.com/media/ec028a_6c4002929e3540e1a8fb89db7b5eecc1~mv2.jpg/v1/fill/w_711,h_412,al_c,q_80,usm_0.66_1.00_0.01,enc_auto/sandbox.jpg

Obrázek 2.6: Vľavo interaktívne pieskovisko a vpravo Interaktívna strelecká galéria

2.3 Hachi Infinite M1

Hachi Infinite M1 je kompaktné zariadenie, ktoré obsahuje výkonný počítač a projektor. Dokáže vytvoriť interaktívnu dotykovú vrstvu na ľubovolnom plochom povrchu [6]. Toto zariadenie je zobrazené na obrázku 2.7.

Obrázek 2.7: Zariadenie Hachi Infinite M1

Spoločnosť Hachi, ktorá je dcérskou spoločnosťou Beijing Puppy Robotics Co., Ltd, prvýkrát ukázala Hachi infinite M1 roku 2020 na výstave spotrebnej elektroniky, skratka CES[6]. Na tejto výstave sa stretla s veľkým ohlasom. Dané zariadenie obsahuje technológiu hlbokého učenia (deep learning) spojenú s vysoko presnými senzormi, ktoré umožňujú rozpoznávanie rôznych obrazov a gest rúk. Hachi infinite M1 ponúka funkcie poháňané umelou inteligenciou a to v oblastiach vzdelávania, fitness, kuchyne a mnoho ďalších [6].

Niektoré technické parametre zariadenia sú:

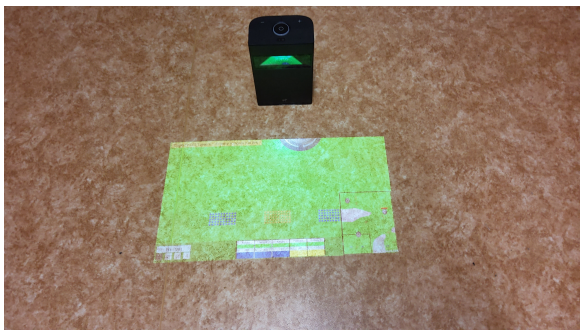
- CPU: Qualcomm Snapdragon SDA670
- RAM: 4GB

[6] https://www.ces.tech

- Úložný priestor: 64GB eMMC
- OS: Infinite OS
- Android systém: Android 9.0

Rozlíšenie displeja je 1280x720, kompatibilné rozlíšenia sú 1080p/2K/4K. Zariadenie sa ovláda presne tak ako na tablete alebo na mobile. Pomer projekcie je 0,39:1. Veľkosť premietaného obrazu je 58-255 cm, avšak odporúčaná veľkosť je 58-100 cm [14].

Dané zariadenie je poháňané patentovanou technológiou AnyTouch, ktorá má funkčnosť na všetkých povrchoch, ktoré Hachi vytvorí a umožňuje odozvu ako kapacitné displeje. Zariadenie je možné využiť v dvoch režimoch a to horizontálnom 2.8 a vertikálnom móde 2.9. Takto vytvorené dotykové pole, ktoré je podobné napríklad tabletom, dokáže s použitím počítačového videnia, ktoré sleduje a číta každý užívateľský dotyk, okamžite a bez oneskorenia rozpoznať každé klepnutie na dotykovej vrstve [6].



Obrázek 2.8: Zariadenie Hachi Infinite M1 v horizontálnej polohe

Obrázek 2.9: Zariadenie Hachi Infinite M1 vo vertikálnej polohe

2.4 Súčasné ovládanie strategických hier na dotykovej ploche

Táto sekcia slúži ako úvod do spôsobov ovládania strategických hier na dotykových zariadeniach.

Pri hľadaní existujúcich riešení som narazil na veľa strategických hier pre zariadenia s dotykovou plochou, avšak drvivá väčšina nemala žiadne ovládanie. Respektíve boli to typy hier, ktoré sa dajú označovať ako „klikačky“, kde sa všetko ovláda klikaním. Výstavba alebo boje majú trvanie v reálnom čase od pár minút, až po niekoľko hodín. Medzi takéto hry napríklad patrí Abyss of Empires[7], ktorá má ikonu nápadne podobnú veľmi slávnej hre Age of Empires II[8].

Tieto uvedené hry majú podľa môjho názoru ovládanie odpovedajúce strategickej hre. Sú to už typické hry, kde hráč buduje svoju základňu, získava suroviny a cvičí vojsko. Patrí k nim aj hra Art of War 3[9]. Táto hra ponúka zaujímavé označovanie jednotiek a to pomocou dvojkliku. Zaujali ma ešte aj tlačidlá, ktoré slúžia pre zapnutie označovania buď pomocou obdĺžnika alebo kreslenia kruhu. Ďalšie ovládanie ktoré má, je už typické klikanie pre označenie alebo stavbu budovy, či posun vojska. Na obrázku 2.10 môžeme vidieť screenshot z hry.

Obrázek 2.10: Snímok z hrania hry Art of War 3.

Hra Expansion[10] je ďalšia typická stratégia. Medzi zaujímavé ovládacie prvky patrí, že má tlačidlá pre označenie všetkých vojakov daného typu, napríklad označenie všetkých vojakov pechoty. To je podľa môjho názoru veľkou výhodou, pretože jednotky sú celkom malé a ich označovanie je teda o niečo náročnejšie. Inak obsahuje typické ovládanie kliknutím ako presun, označenie či stavba budovy. Na obrázku 2.11 môžete vidieť obrázok z hrania tejto hry.

[7] https://play.google.com/store/apps/details?id=com.flamingogames.gok&hl=sk&gl=US
[8] https://store.steampowered.com/app/221380/Age_of_Empires_II_2013/
[9] https://play.google.com/store/apps/details?id=com.geargames.aow&hl=cs&gl=US
[10] https://play.google.com/store/apps/details?id=air.air.Expansion&hl=cs&gl=US

Obrázek 2.11: Snímok z hrania hry Expansion.

Ako poslednou a zároveň aj najväčšou inšpiráciou mi bola hra ROME: Total War - Alexander[11]. Systém tejto hry podstatne rozdielnejší než u vyššie spomenutých. V tejto hre sa nenachádzajú samostatní vojaci, len ich skupiny. Je to hra postavená na využití bojovej stratégie, pretože je potrebné zarátať, že každá jednotka má svoje klady a zápory a taktiež že útok do tyla nepriateľa je veľmi účinný a spôsobuje väčšie poškodenie jednotkám ako útok spredu. Táto hra obsahuje ovládanie presunu vojska pomocou nakreslenia čiary, čo je podľa mňa veľmi dobré gesto. Umožní to lepšie ovládanie jednotiek. Avšak pár vecí by som vytkol a to odznačenie vojakov, zmenu formácie a zobrazenie menu. Majú len jeden spôsob a to znovu kliknúť na jednotku, čo je pri väčšom počte označených jednotiek zdĺhavé. Taktiež majú veľký počet spôsobov ovládania a asi preto ich zmena formácie vyžaduje dva prsty, čo je podľa môjho názoru neprívetivé. Pre menu je treba urobiť swipe troma prstami, čo je síce jednoduché, ale obyčajné tlačidlo by bolo lepšie. Na obrázku 2.12 môžeme vidieť obrázok z hrania tejto hry.

Obrázek 2.12: Snímok z hrania hry ROME: Total War - Alexander.

[11] https://play.google.com/store/apps/details?id=com.feralinteractive.rometwalex&hl=cs&gl=US

Kapitola 3

Návrh

V nasledujúcej kapitole sú uvedené myšlienky a postupy, ktorých som sa držal pri tvorbe tejto demo hry. V tomto návrhu je možné dočítať sa o čom je táto demo hra. Ďalej jej grafické spracovanie, návrhy jednotlivých sád ovládania a ďalších aspektov.

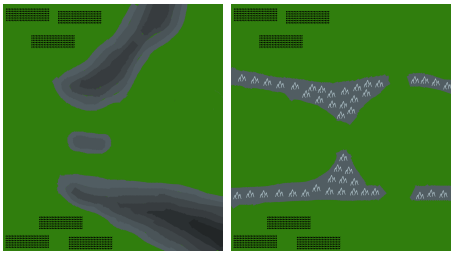
3.1 Popis hry

Táto demo verzia hry slúži na zistenie najideálnejšieho ovládania strategických hier na dotykových zariadeniach. V demo verzii je k dispozícii jedna misia, v ktorej má hráč za úlohu obsadiť všetky veže a poraziť všetky nepriateľské jednotky. Pre experiment som vytvoril scenár ako má používateľ postupovať. Viac sa o ňom je možné dočítať ďalej v kapitole 5. V prípade plnej hry by bola k dispozícii kampaň s príbehom, ktorú by si užívateľ mohol zahrať. K dispozícii by bolo aj viac typov jednotiek, spôsobov výhry a podobné vylepšenia, aby sa z toho mohla stať relatívne plnohodnotná strategická hra. Táto však bola vyvíjaná ako 2D, pohľad zhora.

3.2 Grafické spracovanie

Tvorba hernej mapy

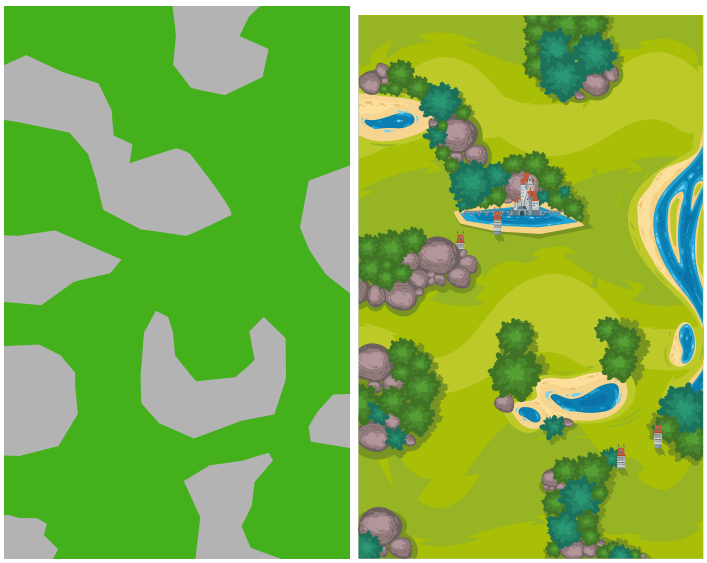
Medzi prvé komponenty pri tvorbe grafického spracovania jednotlivých aspektov demo hry, patrilo vytvorenie hernej mapy. Ako prvú vec na tvorbe hernej mapy som zisťoval, ako najlepšie zobraziť prekážky, na obrázku 3.1 môžete vidieť dva spôsoby ako zobraziť kopce. Prvý z nich zobrazuje kopce pomocou vrstevníc a druhý pomocou malého symbolu kopca. Pri konzultácii s vedúcim a kamarátmi som došiel k záveru, že prvý spôsob sa im páči viac a teda som sa rozhodol tento použiť, avšak neskôr som vypustil zobrazenie vrstevníc.



Obrázek 3.1: Výber typu zobrazenia kopcov.

Obrázek 3.2: Prvotný návrh vľavo, druhotný v strede a jej vylepšenie vpravo.

Prvotný návrh zobrazený na obrázku 3.2 vľavo bol podľa mňa príliš malý a väčšina hernej plochy bola obsadená kopcami, čo sa mi nepáčilo. Preto som prešiel na druhý návrh, ktorý je v strede a jej vylepšenie je pridaním vrstevníc vpravo. Aj v tomto prípade som bol stále nespokojný s veľkosťou, a preto som ju znovu prerobil. S ďalším návrhom som bol už spokojný, pretože bol dostatočne veľký, mal dobre rozložené umiestenie kopcov a je zobrazený na obrázku 3.3. Taktiež je tam zobrazené vylepšenie tejto hernej mapy, na ktorom som pracoval s otcom Mariánom Haasom. Toto je moja predstava, ako by tá hra mohla vyzerala v plnej verzii. Ako už bolo spomenuté, toto zatiaľ nie je hra, je to len demo, ktoré slúži na zistenie aké ovládanie pre dotykové plochy je najlepšie pri tvorbe strategických hier. Teraz je používaný typ hernej plochy, ktorý bol zmenšený a zjednodušený natoľko, aby experimenty netrvali zbytočne dlho a bol teda uspôsobený na demo verziu hry. Je zobrazený na obrázku 3.4.

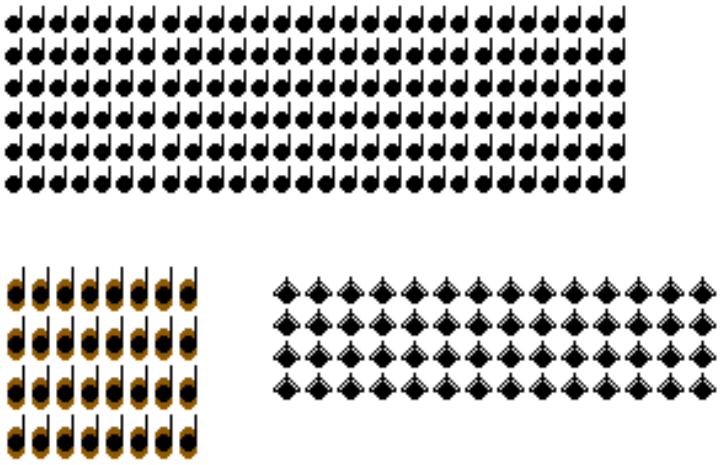


Obrázek 3.3: Návrh mapy pre plnú hru. Druhý obrázok zobrazuje vylepšenú grafiku. Dizajn: www.Freepik.com.

Obrázek 3.4: Aktuálna herná mapa.

Tvorba herných postáv a objektov

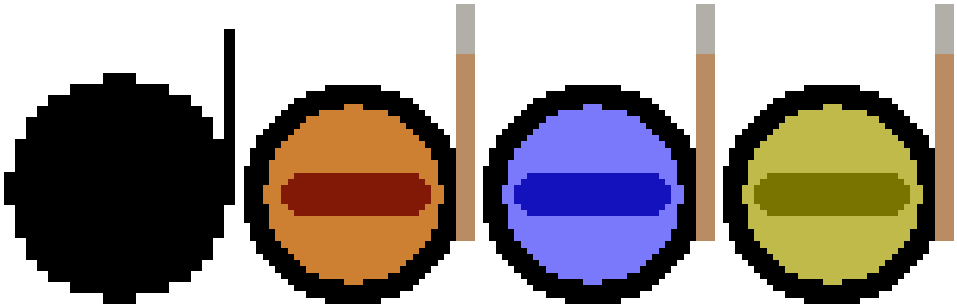
Pri tvorbe herných postáv som začal najprv s jednoduchým zobrazením vojakov ako čierne bodky s paličkami, čo boli kopije. V pláne bolo pridať niekoľko typov vojakov, ako je napríklad jazda a lukostrelci, čo je vidieť na tomto obrázku 3.5. Lenže v tejto demo verzii hry na nich nezostal priestor, no v plnej verzii by boli zastúpení. Neskôr som im pridal farbu, pričom stále zostali v jednoduchom zobrazení ako bodky, keďže som sa inšpiroval spôsobom zobrazení a ovládaní vojakov z hier Total War[1]. Tam sa ovládajú len celé pluky vojakov. Nie je možné ovládať len jedného vojaka. Na obrázku môžete vidieť vývoj tvorby jednoduchého zobrazovania vojakov zľava od prvého typu, potom zobrazenie nepriateľských vojsk a ďalej sú zobrazení vojaci samotného hráča 3.6. Na týchto obrázkoch 3.7 je zobrazené vylepšenie na ktorom som pracoval s otcom.



Obrázek 3.5: Návrh typov vojakov. Kde hore je pechota, vľavo je jazda a na pravo sú lukostrelci.

Ďalšími objektami v hre sú dedinky a veže. Dedinky slúžia pre vyliečenie ranených vojakov a tiež ako obnovenie stratených vojakov v boji. Pri demo verzii boli odstránené, nakoľko už neboli potrebné z dôvodu zjednodušenia celej hry na demo verziu. Na obrázku môžeme

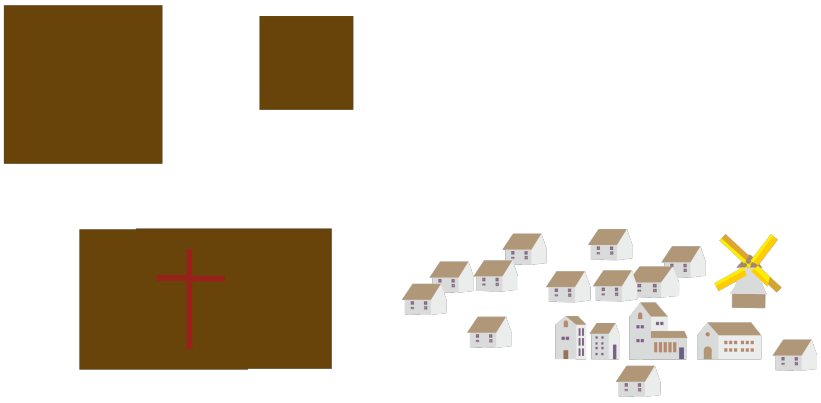
[1] https://www.totalwar.com/



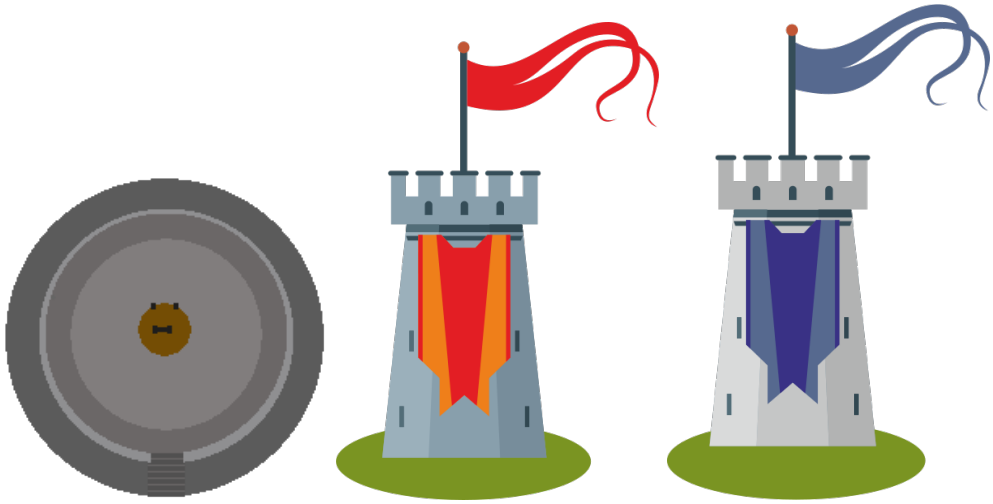
Obrázek 3.6: Návrhy pechoty.

Obrázek 3.7: Vylepšený návrh vojakov. Dizajn: www.Freepik.com.

pozorovať vľavo návrh dedinky a vpravo jej vylepšenú verziu 3.8. Pre väčšiu interakciu v hre som pridal veže. Veže slúžia ako body pre obsadenie, respektíve ako úloha pre hráča. V terajšej podobe slúžia len na obsadzovanie pomocou presunutia vojska k ich blízkosti. V plnej hre by slúžili ako ochrana pre lukostrelcov, ktorý môžu do nej vojsť, strieľať po nepriateľoch a ich dostrel by sa vo veži zvýšil. Na obrázku môžeme vidieť vľavo ako prvé práve používaný návrh veže, uprostred a na pravo vylepšené verzie veží, kde v strede je veža obsadená nepriateľmi a na pravo veža obsadená hráčom 3.9.



Obrázek 3.8: Prvotný návrh dedinky a jej vylepšenie. Dizajn: www.Freepik.com..



Obrázek 3.9: Súčasná grafika veže a jej vylepšenie. Dizajn: www.Freepik.com.

3.3 Mechanizmy

Ovládanie

Keďže cieľom experimentu je odpovedať na otázku, aké ovládanie pre strategické hry na zariadenia s dotykovou plochou je najlepšie, tu je spísaný návrh jednotlivých spôsobov ovládania. Od označovania jednotiek cez ich presun až po zmenu formácie.

Univerzálne ovládanie

Univerzálnym ovládaním sa myslí pohyb kamery a odznačenie vojakov.

1. Pohyb kamery funguje na princípe potiahnutia prstu po mape.
2. Oddialenie\priblíženie funguje na princípe gesta pinch in\out[2].
3. Odznačenie jednotiek funguje pomocou dvojkliku prstom.

Označovanie

Navrhnuté ovládanie označovania jednotiek je uvedené následovne:

1. Kliknutím na jednotku.
2. Potiahnutím prsta cez jednotku.
3. Označenie pomocou obdĺžnika.

Označenie pomocou kliknutia

Ako prvé ovládanie ktoré som navrhol, bolo označovanie vojakov kliknutím. Je to veľmi jednoduché, zistím si pozíciu kde užívateľ klikol, následne skontrolujem všetky užívateľove jednotky a zistím či sa toto kliknutie nachádzalo v nejakej jednotke. Ak áno, tak ju označím.

[2]To je gesto, keď dvoma prstami sa približujem k sebe(in) alebo od seba(out)

Na tablete toto označovanie funguje skoro bezchybne, avšak na zariadení Hacchi Infinite M1 dochádza k problémom. Neviem síce prečo, ale je potrebné rázne a rýchlo kliknúť na jednotku aby sa označila. Predpokladám, že problémom môže byť, že zariadenie zisťuje dotyk inak ako napríklad tablet.

Označenie pomocou prejdenia prstom

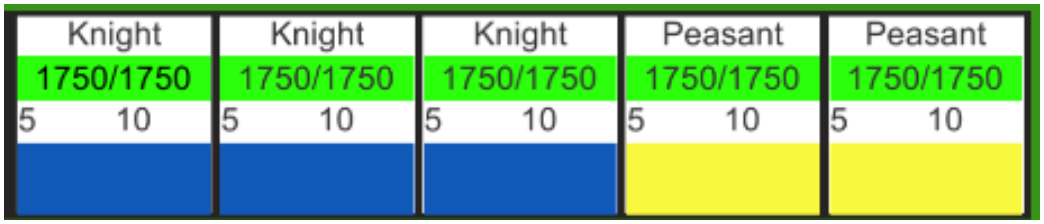
Druhý spôsob označovania jednotiek mi na obhajobách navrhol Ing. Vítězslav Beran Ph.D.. Tento spôsob funguje tak, že sa prejde cez vojakov prstom. Tento pohyb vytvára čiaru a všetky užívateľove jednotky cez ktoré prejde označí. Avšak treba začať túto čiaru kresliť v užívateľovej jednotke.

Označenie pomocou Obdĺžnika

Ako posledné je označenie pomocou označovacieho obdĺžnika. Po podržaní prsta sa zobrazí fialová farba indikujúca, že môže začať označovať. Začne sa teda zobrazovať obdĺžnik, kde jeho ľavý horný roh je v mieste začatia a jeho dolný pravý roh sleduje užívateľov prst. Jednotky, ktoré sa nachádzajú vo vnútri a sú užívateľove, budú označené.

Ďalšie možné spôsoby označovania

Medzi ďalšie spôsoby označovania bol navrhnutý dvojklik prstom, pre označenie všetkých užívateľových jednotiek, ktoré sa práve nachádzajú na hlavnej kamere, avšak neskôr som toto gesto použil pre odznačenie jednotiek. Pôvodné gesto pre odznačenie jednotiek bol klik dvoma prstami, ktoré sa mi zdalo pomalšie ako gesto, ktoré sa práve používa a preto som to zmenil. V plnej verzii hry by boli tie najlepšie vybrané gestá, ktoré tento experiment odhalí použité súčasne. Taktiež som zvažoval označovanie pomocou kariet[3], ktoré môžete vidieť na obrázku 3.10. Ďalej hľadám najklasickejšie označovanie, a to dvojklik na danú jednotku, čo vyberie všetky jednotky rovnakého druhu, ktoré sú viditeľné na kamere.



Obrázek 3.10: Pohyb jednotiek.

Ovládanie presunu vojska

Navrhol som dva spôsoby presunu vojska:

1. Kliknutím na miesto.
2. Nakresleným čiary, po ktorej sa vojsko vydá.

[3]Karta zobrazuje jednotku, jej meno, body života, hodnotu obrany a útoku.

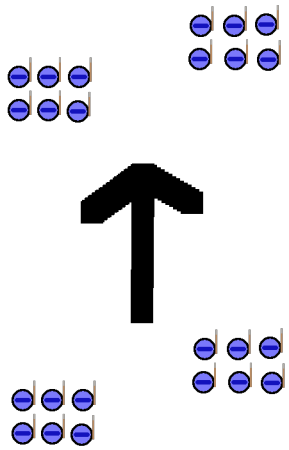
Ako ďalšie uvediem ovládanie presunu vojska. Avšak najprv zdôrazním, že pokiaľ má užívateľ označenú viac ako jednu jednotku, tak ich nové pozície po presune budú v rovnakom rozpoložení, čo môžete pozorovať na obrázku 3.11. Dôvodom tohto návrhu je, že z hry z ktorej som sa inšpiroval, je toto prednastavené pohybovanie. Ak chce užívateľ mať jednotky pri sebe v rade, použije ovládanie zmeny formácie. V týchto hrách sa veľmi zakladá na taktike a je nežiadúce aby jednotky menili svoje rozpoloženie, pretože pred zaútočením si svoje jednotky užívateľ rozpoloží do požadovanej formácie a nechce aby sa to pri presune menilo.

Presun vojska kliknutím

Ako prvé som navrhol presun kliknutím na miesto. Zistil som miesto kde užívateľ klikol a jednotka sa tam presunula. Ďalej v sekcii 3.3 Pathfinding a pohyb jednotiek, bude popísaný spôsob tohto presunu.

Presun vojska nakreslením cesty

Nasledujúci typ pohybu bol o niečo komplikovanejší. Užívateľ začne kresliť cestu prstom, ktorú jednotky budú nasledovať. Cesta je jemne znázornená pokiaľ jednotky nedorazia do cieľa. Treba začať kresliť v jednotke, ktorá je označená. V niektorých prípadoch môže cesta zostať nakreslená a to preto, lebo keď sa jednotky presunuli, tak minimálne jedna jednotka nemá dostatok miesta pre umiestnenie, toto je dôvodom prečo čiara zostane svietiť.



Obrázek 3.11: Pohyb jednotiek.

Ovládanie zmeny formácie

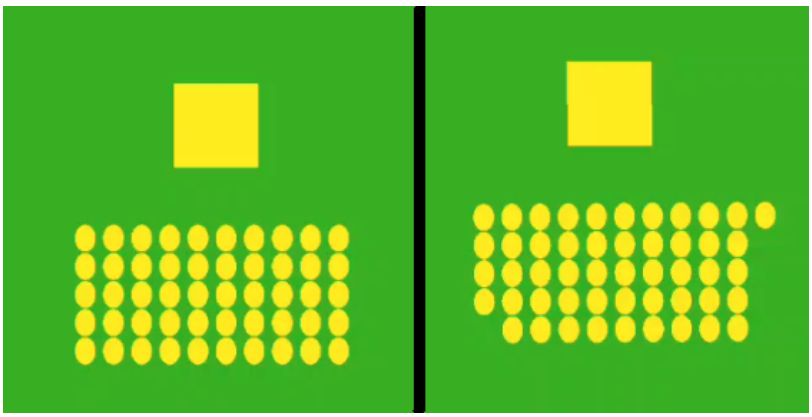
Posledné ovládanie je zamerané na zmenu formácie jednotky. Ja toto ovládanie považujem za jedno z najdôležitejších, pretože v prípade plnej hry, správne natočenie a rozpoloženie jednotiek môže znamenať rozdiel medzi výhrou a prehrou. Boli navrhnuté tri spôsoby ovládania pre zmenu formácie:

1. Podržanie prsta pre rotáciu, ďalšie kliknutie spôsobí zapnutie a vypnutie zmeny formácie.

2. Podržanie prsta pre točenie a zmenu formácie naraz.

3. Podržanie dvoch prstov pre točenie a zmenu formácie naraz.

Pre všetky typy platí, že po použití potrebného gesta sa zobrazí žltý krúžok pod prstom. Pod ním sa zobrazí žltý štvorec, ktorý ukazuje smer, kam sa vojaci pozerajú. A nakoniec je pod ním zobrazená formácia jednotiek ako žlté bodky. Hlavnou myšlienkou zmeny formácie je, že všetky možné pozície vojakov aké môžu nastať, sú dopredu vypočítané. Zobrazené sú len tie, na ktorých sa práve nachádza nejaký vojak. Pri zmene sa v podstate skryje pôvodná pozícia a zobrazí sa nová ako je ukázané na obrázku 3.12. Ak je označených viac jednotiek tak sa zoradia do radu s fixnou medzerou medzi nimi. Nachádza sa tu jedna chyba a tá je, ak má nejaká jednotka inú formáciu ako druhá tak nebude v rovnakom rade ako ostatné. Čo spôsobí to, že bude trochu pred nimi ako je zobrazené na obrázku 3.13.



Obrázek 3.12: Zobrazenie zmeny formácie.

Podržanie prsta a kliknutie pre zapnutie zmeny formácie

Prvý typ zmeny formácií je podržanie prsta. Potom je možné prstom pohybovať do rôznych smerov, pričom sa bude formácia jednotiek točiť, aby práve čelila smerom k prstu. Pre rozšírenie alebo zmenšenie radov je potrebné kliknúť prstom a následne pohybovať doprava alebo doľava. Ďalšie kliknutie vypne rozširovanie a zmenšovanie formácie a znovu zapne točenie jednotiek.

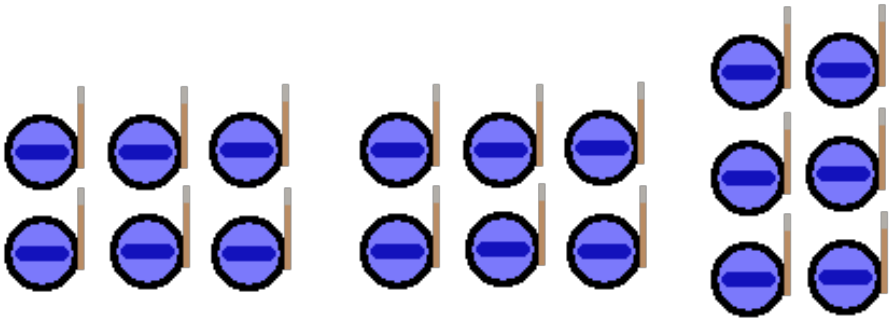
Podržanie jedného prsta

Druhý typ je podobný, tiež je potrebné podržať prst pre začatie. Potom pri pohybe prsta sa jednotka alebo jednotky točia ako aj zmenšujú a zväčšujú rad.

Podržanie dvoch prstov

Tretí typ je rovnaký ako druhý, len s jedným rozdielom a to, že treba podržať dva prsty. Tu je potom možné druhý prst, ktorý sa dotkol dotykovej plochy pustiť a fungovať len s jedným prstom. Tento spôsob ovládanie je inšpirovaný tým, ktorý používa hra ROME: Total War - Alexander[4].

[4] https://play.google.com/store/apps/details?id=com.feralinteractive.rometwalex&hl=sk&gl=US



Obrázek 3.13: Chybné zoradenie do formácie

Štruktúra

Nosnou štruktúrou celej demo hry je, že je ovládaná pomocou takzvaných manažérov. Každý manažér ovláda jednu časť hry. Napríklad taký manažér jednotiek sa stará o predávanie rozkazov správnym jednotkám. Manažér vstupu zase správne vyhodnotí užívateľov vstup. Ako ďalšie veci v štruktúrach sú takzvaní správcovia, ktorí sa starajú o jednotlivých vojakov a jednotky ako celok.

Pathfinding a pohyb jednotiek

Pathfinding, po slovensky nájdenie cesty, je veľmi dôležitá časť pohybu jednotiek. Pri výbere pathfindingových assetov, je dôležité brať na vedomie, či zvláda požadované veci. Či dokáže pracovať s 2D, aký dobrý je výpočet cesty, koľko agentov[5] zvláda a či má zabudovaný dobre spravený systém lokálneho vyhýbania. Unity nemá zabudovaný pathfinding systém, ktorý by vedel pracovať s 2D, tak som prevzal jeden asset[6], ktorý upravuje Unity pathfinding aby bol schopný zvládať aj 2D.

Pohyb jednotiek ako skupiny je inšpirovaný z jedného komentára na stránkach Unity fóra[7]. Tento spôsob presunu jednotiek sa chová tak, že jeden objekt obsahuje miesto na ktoré sa môže jednotka presunúť vo forme niečoho podobného ako je kartón vajíčok. To znamená, že sa "kartón"hýbe a vojaci si vyhľadávajú voľné pozície na ktoré sa môžu presunúť.

[5] Agent je ten, čo nasleduje vypočítanú cestu
[6] https://github.com/h8man/NavMeshPlus
[7] https://forum.unity.com/threads/how-do-i-calculate-a-new-position-for-my-units.1150091/

Kapitola 4

Implementácia

V tejto kapitole je vysvetlená moja vlastná implementácia demo hry. Pri tvorbe som sa inšpiroval príkladmi z oficiálnej dokumentácie Unity[1]. Ďalej som prebral asset, ktorý vylepšil štandardné knižnice Unity pre pathfinding od používateľa h8man na githube[2]. Tento asset je licencovaný pod MIT. Pri tvorbe minimapy som sa inšpiroval z videa[3]. Správnu implementáciu singleton návrhu som prebral z fóra[4]. Nakoniec som prebral myšlienku pohybu jednotiek ako celku z fóra Unity, od užívateľa Kurt-Dekker[5].

V texte sú použité slová korutina a komponent, korutinou sa myslí Coroutine z Unity a komponentom sa myslí component z Unity, čo je súčast každého objektu v Unity. Každý komponent je naprogramovaný ako trieda.

Najprv je vysvetlený **InputManager**, čo je manažér užívateľských vstupov a v ňom konkrétne implementácie jednotlivých ovládacích sád. Ďalej je vysvetlená implementácia zvyšných manažérov. Potom je popísaná implementácia pathfinding, pohybu jednotiek, triedy **UnitClass** a **SoldierClass**. Nasleduje implementácia boja a úplne posledná je implementácia mechanizmu obsadzovania veží a mechanizmu uzdravovania pomocou dedín.

4.1 Ovládanie

Najdôležitejšou triedou je **InputManager**, ktorý spracováva vstupy užívateľa a rozozná aké gesto použil. Každý snímok v metóde **Update()** zisťuje, či užívateľ vykonal nejakú akciu. Ako prvé sa zistí, či užívateľ klikol na užívateľský interface, a pokiaľ áno, tak tento klik ignoruje. Každý užívateľov vstup prstom je reprezentovaný ako **Touch**, čo je štruktúra z Unity. Túto štruktúru Unity automaticky vytvorí a naplní. Táto štruktúra má parameter, ktorý ukazuje v akej fáze sa práve nachádza prst. Čo je pri zisťovaný gesta dôležité, pretože v prvej fáze, ktorá sa volá **Begin** sa zapnú časovače na určenie či sa jedná o kliknutie, podržanie, alebo dvojklik prstom. V poslednej fáze **Ended** sa určí či sa jednalo o jedno z vyššie spomenutých kliknutí, podľa toho či sú stále v čase v akom sa môžu vykonať alebo či išlo o iné gesto. Ešte treba dodať, že takmer každý klik ktorý užívateľ spraví na dotykovej ploche, i keď rýchly, sa nenachádza len vo fázach **Begin** a hneď potom **Ended**. Takmer vždy

[1] https://docs.unity3d.com/Manual/index.html

[2] https://github.com/h8man/NavMeshPlus

[3] https://www.youtube.com/watch?v=28JTTXqMvOU&ab_channel=Brackeys

[4] https://gamedev.stackexchange.com/questions/116009/in-unity-how-do-i-correctly-implement-the-singleton-pattern

[5] https://forum.unity.com/threads/how-do-i-calculate-a-new-position-for-my-units.1150091/

je medzi nimi zopár fáz, buď **Moved** alebo **Stationary**, pretože užívateľ nie je schopný tak rýchlo opustiť dotykovú plochu prstom, aby nedošlo aj k týmto fázam.

Dôležitou triedou pri zachytávaní vstupov je **InputActionClass**, ktorá si drží časovače a metódy, ktoré sú určené pre zistenie či bol dodržaný čas a niekoľko premenných, potrebných pre určenie o aký vstup sa jednalo. Ďalším dôležitým aspektom je enum **Actions**, kde sú zoradené jednotlivé akcie užívateľa.

Univerzálne ovládanie

Pre pripomenutie - univerzálne ovládanie je:

1. Pohyb kamery funguje na princípe potiahnutia prstom po mape.
2. Oddialenie\Priblíženie funguje na princípe gesta pinch in\out[6].
3. Odznačenie jednotiek funguje pomocou dvojkliku prstom.

Pohyb kamery

Pohyb kamery je implementovaný ako vypočítanie vektora smerujúceho od aktuálnej pozície prsta a od prvej zaznamenanej pozície prsta. Potom sa tento vektor pripočíta k aktuálnej pozícii kamery. Toto sa môže vykonať iba ak sa prst nachádza vo fáze **Moved**, a zároveň je iba jeden prst na dotykovej ploche. Na to je určený parameter **NoCameraMove**, ktorý musí mať hodnotu **false**. Tiež sa tu pre potreby podržania prsta nastaví parameter **IHaveMoved** na hodnotu **true**, ak sa prst pohol o 0,1 jednotky. Tým pádom podržanie prsta už nebude brané ako možné gesto. Je to tu preto, lebo dotyková plocha je veľmi citlivá a teda nieje možné pri podržaní prsta s ním nepohnúť o oku neviditeľnú vzdialenosť.

Oddialenie\Priblíženie kamery

Toto gesto funguje tak, že ak sa zaznamená vstup druhého prsta, ostatné možné gestá už nieje možné vykonať, okrem zmeny formácie pomocou podržania dvoch prstov. A to preto, lebo program sa už do vyhodnotení akcií jedným prstom nedostane. Aj v tomto prípade sa nastavuje parameter **IHaveMoved**. Ako prvá sa spočíta vzdialenosť od jedného prsta po druhý pomocou **Vector3.Distance()**. Potom sa porovnáva či aktuálna vzdialenosť je väčšia ako minulá a ak áno, jedná sa o priblíženie kamery. Je tu nastavené maximálne priblíženie kamery, v ktorom hra začína. Od parametru hlavnej kamery, **orthographicSize** sa odčíta hodnota 8, ktorá sa vynásobí konštantou **Time.deltaTime** aby tento pohyb bol plynulý. Podobným spôsobom sa zmenší ukazovací obdĺžnik na minimape. Na hodnotu 8 som prišiel testovaním a zdá sa ako najlepšia rýchlosť pre toto gesto. Podobným spôsobom je implementované aj oddiaľovanie kamery, kde sa hodnoty pripočítavajú a nemá obmedzenie na oddialenie kamery.

Odznačenie vojakov

Odznačenie vojakov je implementované tak, že vo fáze **Begin** sa nastaví časovač a zvýši sa počítadlo kliknutí. Vo fáze **Ended** sa zistí, že sa jednalo o kliknutie ale zatiaľ to nieje registrované ako kliknutie, pretože sa spustí korutina, ktorá až o 0,30 sekundy vyhodnotí či sa jednalo o kliknutie alebo nie. Zatiaľ užívateľ klikne druhý krát, a inkrementuje sa

[6] To je gesto, keď sa dvoma prstami približujem k sebe(in) alebo od seba(out)

počítadlo kliknutí a teraz bude na hodnote 2. Teraz sa vo fáze **Ended** zistí, že sa jedná a dvojklik a nastaví sa premenná **DoubleTap** v triede **InputActionClass** na hodnotu **true**, aby korutina ktorá je práve spustená vedela, že o tých 0,30 sekundy nastal dvojklik.

Označovanie

V tejto časti je popísaná implementácia označovania jednotiek. Konkrétne sa jedná:

1. Kliknutím na jednotku.
2. Potiahnutím prsta cez jednotku.
3. Označenie pomocou obdĺžnika.

Kliknutím

Kliknutie je implementované tak, že sa vo fáze **Begin** spustí časovač, do ktorého je možné registrovať tento vstup ako kliknutie. Následne sa po opustení dotykovej plochy tento vstup dostane do fázy **Ended**, kde sa najprv kontroluje či sa jedná o dvojklik a ak nie, potom podľa toho v akej sade je toto ovládanie implementované sa skontroluje pomocou metódy **CheckForTap()** v triede **InputActionClass** či od spustenia časovača neprebehlo viac ako 0.2 sekundy. Ak neprebehlo, tým pádom sa spustí korutina **SingleTap()**, čo je metóda z **InputManager**. V tejto korutine sa pasívne počká 0.3 sekundy. Potom, ak sa nejednalo o dvojklik, sa nastaví akcia v parametri **Action** na **Actions.OneFingerAction**. Ďalej si to už preberie **UnitManeger**. Ten si pomocou cyklu zistí či sa pozícia prsta nachádza v nejakej užívateľovej jednotke pomocou metódy **StartInFormation()** v triede **UnitClass**. Táto metóda zistí, či sa daná pozícia nachádza v jednotke, pomocou pozície rohov jednotky. Ak označí nejakú jednotku nastaví parameter **SelectingUnit** na hodnotu **true** a ďalej už tento vstup berieme ako označenie.

Potiahnutím prsta cez jednotku

Po vstupe prsta na dotykovú plochu sa podľa toho v akej je sade najprv zistí, či chce označovať alebo kresliť cestu pre jednotky, a nastaví sa príslušná premenná. Zavolá sa metóda **CreateLine()**, ktorá pridá do hry objekt čiara, ktorá je tvorená komponentom **LineRenderer** z Unity. Pomocou tohto komponentu sa kreslí čiara a to tak, že sa jej pridávajú body na mape cez ktoré má viesť. Po prekonaní predom danej vzdialenosti 0.25 jednotky prstom od pôvodnej pozície sa pridá ďalší bod do čiary. Tu sa jedná o chybu a malo by sa rátať s predošlou pozíciou a nie s prvotnou. Potom sa vždy pomocou metódy **SelectUnitByPosition()** s parametrom aktuálnej pozície z triedy **UnitManager** skúsi zistiť či sa na danej pozícii nachádza užívateľova jednotka a ak áno tak ju označí. Toto zisťovanie je implementované tak, že sa **UnitManager** v tejto metóde dopytuje každej užívateľovej jednotky metódou **StartInFormation()**, ktorej sa samozrejme dodá daná pozícia, či sa nachádza v jednotke podľa rohových pozícií danej jednotky.

Označenie pomocou Obdĺžnika

Na začiatku je potrebné zistiť, či užívateľ drží prst na dotykovej ploche. To sa zistí tak, že ak už prešiel čas potrebný pre klasifikáciu tohto vstupu ako podržanie a užívateľ nepohol prstom z miesta vstupu, tak sa zapne štvorcový objekt, ktorý ma ľavý horný roh na

mieste prvého vstupu prstu a spodný pravý roh nasleduje prst užívateľa. Ak sa raz určí, že sa jedná o toto gesto, potom sa nastaví premenná **DrawRectangle** aby tok riadenia programu ostal pri ovládaní tohto označovacieho obdĺžnika. Pri fáze **Moved** sa upravuje obdĺžnik nasledovne. Zistím aktuálnu pozíciu prsta, čo je vlastne pravý dolný roh objektu, potom vypočítam pozíciu horného ľavého rohu tak, že od x súradnice odpočítam polovicu dĺžky objektu a k y súradnici pripočítam polovicu výšky objektu. Následne vypočítam súradnice nového stredu ako $(x, y) = (\frac{1}{n}\sum_{i=0}^{n} x_i, \frac{1}{n}\sum_{i=0}^{n} y_i)$, kde x_i a y_i sú pozície horného ľavého a dolné pravého rohu. Potom vypočítam novú veľkosť toho obdĺžnika a to tak, že od y hodnoty ľavého horného rohu odčítam hodnotu pravého dolného rohu, takto získam novú výšku objektu. Obdobne zistím aj šírku. Priradím objektu pozíciu nového stredu, a novú veľkosť. Tento objekt má na sebe komponent **RectangleSelect**, ktorý pomocou metód **OnTriggerEnter2D()** a **OnTriggerExit2D()** zisťuje, či do oblasti objektu vstúpila alebo z neho vystúpila jednotka a označí ju pomocou metódy **SelectUnitByID()** z triedy **UnitClass** respektíve odznačí pomocou metódy **UnSelectUnitByID()**.

Presun Vojska

Pre presun vojska boli implementované dva spôsoby ovládania a to:

1. Kliknutím na miesto.
2. Nakreslením čiary, po ktorej sa vojsko vydá.

Kliknutie na miesto

Implementácia kliknutia už bola spomenutá vyššie, preto ju už nie je potrebné znovu uvádzať. Rozdiel nastáva v tom, ako si to preberie **UnitManager**. Ten ako bolo vyššie spomenuté, najprv zisťuje či sa jednalo o označenie a ak zistí, že nie, tak začne rozoberať daný vstup ako presun vojska. Tento proces bude vysvetlený ďalej v časti Správa jednotiek.

Nakreslenie čiary

Implementácia je zhodná s implementáciou označovanie pomocou prejdením prstu, jediným rozdielom je, že ak užívateľ začal kresliť čiaru v označenej jednotke nastaví sa premenná **FollowingLineForMove** na hodnotu **true** vo fáze **Begin**. Priebeh je rovnaký až do fázy **Ended**, kde sa pozrie na danú premennú, ktorá ak je **true** tak nastaví premennú **Action** na **Actions.FollowTheLine** aby **UnitManager** vedel, že se jedná o presun vojska.

Zmena formácie vojska

Pre implementáciu zmeny formácie jednotky, bolo treba implementovať pomocné triedy **HoldingUnitRotation** a **UnitRotation**, ktoré riešia zmenu formácie jednotky.

HoldingUnitRotation si drží zoznam tried **UnitRotation** pre každú jednotku pri ktorej práve užívateľ mení formáciu. Vytvorí jeden hlavný objekt **HoldingObject**. Táto trieda udržuje medzery medzi jednotlivými jednotkami a to tak, že keď sa zmení počet stĺpcov pri zmene formácie, tak pripočíta k x pozícii jednotlivým objektom konštantnú hodnotu. Toto pripočítanie sa neurobí pre prvú jednotku v rade.

Táto trieda taktiež zisťuje, či sa užívateľ prstom vzďaľuje od stredu hlavného objektu. Ak áno, tak presúva vojakov v jednotkách do pravej strany, teda zväčšuje posledný stĺpec, alebo pridáva nový stĺpec. Tým zároveň zmenšuje posledný rad, prípadne ho odstráni. Na to

používa metódu **ShiftToRight()** z triedy **UnitRotation**. Pri opačnom pohybe to funguje presne naopak a používa sa metóda **ShiftToRight()**.

Najkomplikovanejšou častou je, keď užívateľ dokončil zmenu formácie a má sa to preniesť na jednotlivé jednotky. Bez zabiehania do zbytočných detailov, sa v tejto časti najprv vypočítajú potrebné hodnoty pre správne presunutie nových pozícií pre jednotlivé jednotky. Najprv sa zistí veľkosť vojaka (SoldierSize), vzdialenosť medzi vojakmi a medzera medzi vojakmi (GapBetweenSoldiers), tu sa predpokladá, že medzera je rovnaká na ose x ako na ose y. Potom sa vypočíta šírka jednotky ako

```
UnitToRotate.Collumns * SoldierSize +
(UnitToRotate.Collumns + 1) * GapBetweenSoldiers
```

a pre výšku sa použije obdobný postup len sa použijú **UnitToRotate.Rows** namiesto **UnitToRotate.Collumns**. Ďalej sa vyhľadá jednotka, pre ktorú je treba zmeniť formáciu. Objekt, ktorý reprezentuje novú formáciu, sa napasuje presne na pozíciu vyhľadanej jednotky a nastaví sa správne aj jej rotácia, aby sa oba tieto objekty prekryli v pozíciach vojakov. Potom sa oranžové pozície priradia jednotlivým vojakom.

Pri zmene formácie sa na začiatku vypočítajú všetky možné pozície vojakov vo formácii a uložia sa do pamäte. Sú uložené ako dvojrozmerný zoznam, pre rýchly prístup k potrebnej pozícii. K pozícii sa pristúpi ako stĺpec, riadok. Obsadené pozície majú žltú farbu a sú viditeľné pre užívateľa, zvyšné pozície sú červené a neviditeľné pre užívateľa. Tu je potom zmena pozície vojaka jednoduchá, pretože sa len zmení farba a viditeľnosť.

Rotácia je implementovaná tak, že sa spočíta vektor smerujúci z pozície **HoldingObject** do pozície prsta. Následne sa spočíta uhol pomocou metódy **Mathf.Atan2()**, ktorej sa dodá x a y pozícia vypočítaného uhla. Potom sa to vynásobí konštantou **Mathf.Rad2Deg**, aby sa dostal uhol a odpočíta sa hodnota 90 pre korekciu. Z neznámeho dôvodu vypočítaný uhol mal pri otočení vždy o 90 stupňov viac. Tento problém vyriešilo odčítanie hodnoty 90. Ďalej som sa tým nezaoberal. Ako posledné sa spočíta rotácia typu **Quaternion** čo je Unity štruktúra, ktorá si drží informácie o rotácii objektu. Tá sa spočíta pomocou metódy **Quaternion.AngleAxis()**, ktorej sa dodá vypočítaný uhol a **Vector3(0, 0, 1)** pre určenie po ktorej ose sa má rotácia vykonať. Potom sa pomocou **Quaternion.Slerp()** vykonáva plynulé otáčanie. Táto implementácia rotácie je použitá všade, kde sa jedná o rotáciu.

Zmena formácie je implementovaná tak, že sa najprv posunie posledný rad doľava, aby nebol v strede a počítanie bolo jednoduchšie pomocou metódy **ShiftLeftLastRow()**. Implementácia tohto posunutia spočíva v spočítaní koľko žltých, teda obsadených pozícii sa nachádza v posledom rade, čo sa spočíta pomocou metódy **CountOfYellowInLastRow()**. Potom sa vie, koľko pozícií zľava treba zmeniť na žlté a koľko na červené. Následne sa spočíta koľko stĺpcov a riadkov má daná formácia pomocou **ComputeRowCollumnSize()**, a vypočítajú sa potrebné premenné na presun vojaka doľava pomocou **ComputeShiftLeftVariables()** alebo **ComputeShiftRightVariables()**, pokiaľ sa presúva vojak doprava.

Teraz sa vymenia pozície, a to tak, že sa zmení farba a viditeľnosť danej pozície. Pri presune doprava sa nájde posledný stĺpec, v ktorom nastaví poslednú pozíciu na riadku na žltú, iba ak sa nejedná o posledný riadok formácie. Ak sa jedná o posledný riadok formácie tak sa nastaví pozícia v novom stĺpci na žltú. Červenú farbu priradí pozícii v poslednom rade v poslednom stĺpci, kde sa nachádza žltá farba. Pri presune doľava je to skoro opačný proces a to tak, že žltú farbu priradí na pozíciu, ktorá sa nachádza na poslednom riadku v poslednom stĺpci, a ak je toto posledný stĺpec formácie tak farbu zmení v prvom stĺpci ďalšieho riadku. Červenú farbu priradí pozícii v poslednom stĺpci na posledom riadku.

Ako posledná vec sa zistí či sa pri tejto zmene zmenil počet stĺpcov. Ak áno, tak je to signál, že treba upraviť rozostupy medzi jednotkami. Nakoniec sa už len posledný riadok posunie do stredu pomocou metódy **CenterLastRow()**.

Tú sú tri spôsoby zmeny formácie jednotiek.

1. Podržanie prsta pre rotáciu, ďalšie kliknutie spôsobí zapnutie a vypnutie zmeny formácie.
2. Podržanie prsta pre točenie a zmenu formácie naraz.
3. Podržanie dvoch prstov pre točenie a zmeny formácie naraz.

Podržanie prsta a kliknutie pre zapnutie zmeny formácie

Pre implementáciu tohto gesta je potrebné najprv zaregistrovať podržanie prsta, čo už bolo vysvetlené vyššie. V tomto prípade sa nezapne premenná **DrawRectangle**. Nastaví sa premenná **Holding** na hodnotu **true** aby bolo toto gesto klasifikované ako podržanie prsta. Potom pomocou premennej **ShiftSoldiers**, ktorá je na začiatku nastavená na hodnotu **false**, sa určuje či sa má meniť formácia alebo rotácie jednotiek. Pri vstupe ďalšieho prsta na dotykovú plochu sa prepne táto premenná aby sa mohlo prepínať ovládanie zmeny formácie a zmeny rotácie. Zmena formácie sa vykoná pomocou metódy **Shift()** z triedy **HoldingUnitRotation**, ktorej sa dodá x hodnota pozície prsta. Pre rotáciu sa použije metóda **Rotate()**, ktorej sa dodá pozícia prsta.

Podržanie jedného prsta

Táto implementácia je zhodná s vyššie uvedenou, rozdiel je v tom, že keď bude používateľ posúvať prstom, tak sa bude zároveň volať metóda **Shift()** a **Rotate()**.

Podržanie dvoch prstov

Pre toto gesto je potrebné zistiť podržanie pomocou dvoch prstov. Implementácia je rovnaká ako pri podržaní jedného prsta, len časovač sa zapne až keď vstúpi druhý prst na dotykovú plochu. Zmena formácie je rovnaká, čiže pri posúvaní prstov sa zároveň volá **Shift()** a **Rotate()**. Pri tomto geste je možné druhý prst, ktorý sa dotkol plochy, pustiť a pracovať len s jedným. Avšak táto implementácia nieje najlepšia, lepšie by bolo aby užívateľ mohol pustiť ľubovoľný prst.

4.2 Manažéri

Manažéri sú hlavnou kostrou celej demo hry. Každý zohráva svoju dôležitú úlohu. **InputManager** bol vysvetlený spolu s ovládaním. Medzi ďalšie dôležité patria **UnitManager** a **BattleManager**, ktorým bude venovaný väčší priestor. Zvyšní manažéri sú **UiManager** a **GameManager**.

Správa jednotiek

Správu jednotiek vykonáva **UnitManager**, pomocou ktorého sa pridávajú jednotky do hry. Drží si ich v zoznamoch, kde jeden je pre nepriateľské a druhý pre užívateľské jednotky. Tento manažér zisťuje každý snímok pomocou metódy **DoAction()** či nastal užívateľský

vstup spracovaný **InputManager**, a to pomocou jeho parametru **Action**. Podľa toho čo obsahuje daný parameter, vyhodnotí a spraví užívateľov pokyn.

Teraz je potrebné uviesť triedu **CommandForPlayerUnits**, ktorá reprezentuje jednotlivé príkazy užívateľa. Trieda **UnitManager** si drží ich zoznam v premennej **PlayerCommands**. Jej hlavným cieľom je držanie informácie o akcii užívateľa, konkrétne o presune jednotiek. V každom snímku pomocou metódy **MyUpdate()** z triedy **UnitManager** sa vykonávajú tieto príkazy pomocou metódy **Move()** z triedy **CommandForPlayerUnits**. V tejto metóde sa pri sledovaní nakreslenej čiary zisťujú ďalšie body na ceste, ktoré sa predávajú jednotke, aby ich sledovala. V prípade viacerých jednotiek tu existuje objekt, ktorý sa nachádza v strede týchto jednotiek a je rodičom objektov, ktoré sa nachádzajú na pozíciach týchto jednotiek. Presun je spravený tak, že sa stredný objekt posunie na ďalšiu pozíciu na čiare, tým pádom sa automatický posunú aj objekty pod jednotkami, pretože takto funguje Unity. Následne sa jednotlivým jednotkám priradia pozície týchto objektov a oni sa začnú na ich pozíciu plynulo posúvať. Tento spôsob je použitý aj pri presune kliknutím, kde jediný rozdiel je, že hlavný objekt a jeho detské objekty sa po presunutí na pozíciu vymažú. Pri posune na ceste je potrebné držať ich v pamäti, aby sa dalo sledovať cestu. Tieto objekty sa vytvárajú v **UnitManager** a prípadne predávajú do rozkazu.

V prípade, ak je parameter **Action** rovný **Actions.OneFingerAction**, alebo **Actions.OneFingerActionMoveOnly**, tak sa tento vstup vyhodnotí ako kliknutie a treba zistiť, či označuje jednotku alebo chce spraviť presun. Najprv zistí, či sa pozícia, ktorú získa z premennej **FingerPosition** z triedy **InputManager** nachádza v neoznačenej jednotke, a ak áno, tak sa to vyhodnotí ako označenie jednotky. V opačnom prípade sa jedná o posun jednotky. Ako bolo vyššie spomenuté, vytvorí sa rozkaz a to tak, že sa najprv daná jednotka odstráni z prípadne už existujúceho rozkazu, alebo sa celý rozkaz odstráni ak bol vytvorený len pre túto jednotku. Rozkazy sa vytvárajú práve pre označené jednotky. Je to rozdelené pre jednu jednotku a zároveň pre viacero ako jednu. Je to rozdelené preto, lebo pre viac jednotiek je potrebné vytvoriť vyššie spomenuté objekty. Tento postup je rovnkaý aj keď dostane v parametri **Action** hodnotu **Actions.FollowTheLine**. Jediným rozdielom je, ako sa vytvorí rozkaz. Prekopírujú sa body vytvorenej čiary do rozkazu. Ešte sa tu nachádzajú časti, keď je parameter **Action** rovný **UnSelectAllVisible**, vtedy odznačí všetky označené užívateľové jednotky pomocou cyklu. Ak je **Action** rovný **OneFingerActionFormation** znamená to, že treba posunúť jednotku, alebo jednotky do novej formácie. Má to samostatnú vetvu preto, lebo pri tomto presune sa nebude vytvárať stredný objekt, ak sú označené aspoň dve užívateľové jednotky. K tomuto presunu sa bude chovať ako k presunu jednej jednotky. V metóde **DoAction()** sa ešte priraďuje nepriateľským jednotkám pozícia na ktorú sa majú presunúť a to tak, že z triedy **AI** sa získa na akú pozíciu sa má daná jednotka s daným **Guid** presunúť.

Metóda **MyUpdate()** vykonáva jednotlivé rozkazy a v prípade, ak boli dokončené, tak ich odstraňuje zo zoznamu rozkazov. Vykonávajú sa tu presuny nepriateľských jednotiek. Ďalšou časťou, ktorú vykonáva, je zmena farby života na kartách užívateľských jednotiek a to tak, že postupne mení farbu zo zelenej až po červenú. Tiež tu kontroluje počet vojakov v jednotlivých užívateľských a nepriateľských jednotkách a ak niektorá obsahuje 0, tak sa vymaže. V prípade nepriateľských sa pripočíta v denníku úloh počet zabitých nepriateľských jednotiek. Pri užívateľových jednotkách zistí, že ak počet užívateľových jednotiek je 0, tak užívateľ prehral.

Správa boja

Tento manažér spravuje boj jednotiek. Slúži ako prostredník medzi jednotlivými jednotkami. Pomocou jeho metód sa sprostredkuje útok jednej jednotky na druhý. To sa deje pomocou metódy **Attack()**, kde tento manažér zisťuje, či nastal útok. A to tak, že dostane ako parameter útočníka a meno brániaceho sa vojaka, kde si následne zistí pomocou metódy **GetDefender()** brániaceho vojaka a jeho jednotku. Následne sprostredkuje útok pomocou metódy **CalculateAttack()**. Ak mu táto metóda vráti hodnotu **true**, tak to znamená, že obranca bol zabitý a treba informovať jednotku aby si daného vojaka odstránila zo zoznamu jednotiek. To je implementované pomocou metódy **KillSoldier()**. V metóde **Attack()** dostane parameter **myTag**, a tým pádom vie, či brániaca jednotka je nepriateľova alebo užívateľova. Táto metóda dostane zoznam jednotiek, danú jednotku a meno vojaka. Pomocou týchto parametrov vyhľadá správnu jednotku a nechá odstrániť daného vojaka pomocou metódy **KillSoldier()** z triedy **UnitClass**.

Zvyšní manažéri

Manažér **GameManager** pomocou metódy **LateUpdate()** vykonáva akcie jednotlivých manažérov v poradí. Je to hlavný manažér, ktorý sa stará o chod hry a vyhodnocuje prehru alebo výhru užívateľa. Metóda **LateUpdate()** nastane ako posledná v každom snímku. V nej sa zavolajú nasledovné metódy v danom poradí. Najprv **DoAction()** a **MyUpdate()** z triedy **UnitManager**, nakoniec **MyUpdate()** z triedy **AI**.

Posledný manažér je **UiManager**, ktorý sa stará o užívateľský interface. Má metódy, ktoré sú naviazané na jednotlivé tlačidlá v hre. Tieto metódy potom vykonajú jednotlivé akcie, ktoré majú k dispozícii. Tiež sú tu metódy pre zmenu denníka úloh, v prípade získania alebo straty veže.

4.3 Vojaci, jednotky a hľadanie cesty

Táto časť bude pojednávať o tom, ako sú jednotliví vojaci a jednotky v ktorých sa nachádzajú uložené v pamäti. Ich triedy, ktoré sa o ne starajú, ako je implementovaný ich pohyb a rotácia.

Trieda pre vojakov

SoldierClass je trieda, ktorá drží informácie o vojakovi. Drží si informácie ako počet životov, hodnota útoku a obrany, Unity objekt vojaka a ďalšie potrebné informácie pre správne riadenie vojaka. Pomocou metódy **ComputeRotation()** sa spočíta rotácia vojaka a metóda **RotateSoldier()** vykoná rotáciu. Poslednej metóde sa dodá rýchlosť otáčanie. Pohyb je implementovaný tak, že sa najprv vypočíta vzdialenosť od vojaka k pozícii, ku ktorej sa má dostať pomocou metódy **Vector3.Distance()**, a ak je táto vzdialenosť menšia než **Time.deltaTime * Rýchlosť Pohybu**, tak vojak prišiel do cieľa a nastaví sa parameter **FinishMoving** na hodnotu **true**. V opačnom prípade sa k pozícii vojaka pripočíta hodnota, ktorá sa vypočíta nasledovne. Najprv sa vypočíta vektor smerujúci od pozície vojaka do pozície kam sa má presunúť, potom sa tento vektor normalizuje a vynásobí konštantou **Time.deltaTime** a **MovementSpeed**, čo je rýchlosť pohybu. Tento spôsob pohybu je použitý všade, kde je nutný plynulý pohyb.

Hľadanie cesty

Unity má svoje vlastné triedy pre hľadanie cesty, avšak nepodporuje 2D. Preto som prevzal, ako som vyššie spomenul, úpravy týchto tried, aby som bol schopný používať hľadanie cesty. Z tejto implementácie používam len vytvorenie plochy, po ktorej sa môže chodiť. Tá sa vytvorí tak, že rodičovský objekt **NavMeshSurface2d** bude obsahovať komponent **NavMeshSurface2d**, ktorému sa nastavia parametre **Collect Objects** na **Children** a **Use Geometry** na **Physics Colliders**. Toto nastavenie spôsobí, že všetky detské objekty, ktoré tento objekt obsahuje budú zarátané buď ako prekážky alebo plochy pre chodenie. Druhý parameter spôsobí, že plocha týchto objektov bude vypočítaná pomocou **Collider** komponentov. Každý detský objekt, ktorý sa má zarátať musí obsahovať tieto dva komponenty. **NavMeshModifier**, ktorý určí či sa jedná o prekážku, alebo miesto po ktorom sa môže chodiť a nejaký typ **Collider**. Následne sa potom v Unity editore vypočíta priestor, po ktorom sa môže chodiť.

Ďalšiou časťou je komponent **NavMeshAgent**, ktorý dokáže samostatne posúvať objekt ku ktorému je pripojený a samostatne nasledovať cestu ktorú si vypočíta. Lenže bol nevyhovujúci, pretože nedokáže zarátať lokálne vyhýbanie pohybujúcich sa objektov a občas sa pri pohybe blízko statického objektu s ním prekryje. Preto bola implementovaná trieda **UnitMovementClass**.

Táto trieda spracováva moje vlastné hľadanie cesty a to tak, že si nechá pomocou **NavMeshAgent** vypočítať cestu, ktorá je uložená ako pozície typu **Vector3** v poli. Túto cestu potom upraví, aby po ceste nedošlo k prekrytiu so statickým objektom. Ďalšie funkcie už neboli implementované. Mali by pravdepodobne vlastné triedy a boli by to funkcie pre lokálne vyhýbanie, kontrolovanie či na ceste po ktorej ide jednotka sa nenachádzajú prekážky, a ak áno, podľa toho sa im vyhnúť.

Úprava pozícií na ceste je implementovaná tak, že sa každý bod na ceste skontroluje pomocou metód triedy **ChangeUnitSizeClass**. To sa zisťuje spôsobom, že sa na pozícii na ceste vytvorí objekt, ktorý bude natočený tak, akoby jednotka na túto pozíciu prišla. Tento objekt je o niečo väčší než jednotka sama a pomocou metódy **Physics2D.OverlapBoxAll()** zistí, či sa neprekrýva s nejakým statickým objektom, a ak áno, tak vypočíta vektor zo smeru nájdenej prekážky k pozícii objektu, normalizuje ho, zväčší a jeho nový koniec nastaví ako novú pozíciu objektu. Tento proces sa počíta v cykle **While**, kde pokiaľ nenájde, že sa objekt s niečim prekrýva, tak cyklus skončí. Inak stále počíta. Tento cyklus sa môže zacykliť v prípade, že sa objekt nachádza medzi dvoma prekážkami a nezmestí sa medzi ne. Preto bolo navrhnuté, že ak po nejakom čase cyklus neskončí, v tom prípade sa jednotka zmenší. Nebolo to implementované celé, čiže sa to nepoužilo. Toto bolo prvé kontrolovanie. Teraz sa objekt otočí do smeru v akom by pokračoval do ďalšieho bodu a znovu sa skontroluje či sa s niečim neprekrýva.

Ďalšou funkciou triedy **UnitMovementClass** je pohyb **Agenta**, teda celej jednotky rovnako ako sa hýbe vojak. Táto rotácia je však instantná, teda nieje to plynulý pohyb. Vypočítanie rotácie sa spraví v metóde **ComputeAgentRotationPosition()** tak, že sa vypočíta vektor od agenta k ďalšiemu bodu na ceste a jeho hodnoty x a y sa vynásobia 10 aby bol tento bod ďalej. Potom sa zistí nový koniec, a na tento koniec sa jednotka otočí.

Táto trieda tiež pohybuje vojakmi a teda sa stará o celý pohyb jednotky aj s vojakmi. Spôsob pohybu jednotky je taký, že sa pohne agentom a potom sa pohne vojakmi. Na začiatku pohybu sa najprv vojaci otočia do smeru kam majú ísť, a až potom sa budú hýbať a točiť zároveň. Keď agent dokončí svoj pohyb, tak sa otočí do takého smeru, v akom bola

jednotka než sa začala hýbať. V tomto konečnom štádiu vojaci dokončia svoj pohyb a potom sa taktiež otočia, aby správne smerovali a až teraz nastane koniec presunu.

Trieda pre jednotky

UnitClass je trieda, ktorá reprezentuje jednotku a drží si informácie o svojich vojakoch v zozname **Soldiers** pozícií, na ktoré sa môže vojak posunúť **Coordinates**, a ďalšie potrebné informácie pre správne riadenie jednotky.

Celá jednotka je v Unity reprezentovaná tak, že premenná **Agent** je objekt reprezentujúci jednotku, a jeho detskými objektami sú **DirectionPoint**, čo je objekt ktorý sa nachádza ďaleko pred jednotkou. Na tento objekt sa vojaci pri rotácií otáčajú. Ďalej všetky pozície v jednotke, na ktoré sa môže vojak posunúť, sú uložené v **Coordinates**. Majú fixný počet a každý vojak sa presunie práve na jednu z týchto pozícií, a tým tvoria formáciu jednotky.

Ďalší samostatný objekt **SoldiersObjectChild**, ktorý slúži na to, aby boli všetci vojaci detskými objektami jedného objektu a aby bol poriadok v Unity editore. Tento objekt je detským objektom **SoldiersObject**. Tento spôsob uloženia je takýto kvôli tomu, aby sa lepšie pracovalo s vojakmi.

Táto trieda dostáva informácie o rozkaze z triedy **UnitManager** a to tak, že najprv je zavolaná metóda **StartMoving()**, ktorej sa dodá pozícia na ktorú sa má jednotka presunúť. V tejto metóde sa pomocou komponentu **NavMeshAgent** vypočíta trasa použitím metódy **SetDestination()**, ktorej sa dodá pozícia na akú sa má presunúť. Potom priradí potrebné parametre triede **UnitMovementClass**.

Metóda **PerformMove()** je v každom snímku volaná z **UnitManager**. V tejto metóde, pokiaľ jednotka nedokončila pohyb, sa najprv zavolá metóda **RotateUnit()** pre vykonanie rotácie, pokiaľ je umožnená. Potom metóda **MoveUnit()** pre vykonanie pohybu, taktiež, ak je umožnená a ak sa má otočiť do originálnej pozície, tak sa použije metóda **RotateToOrigin()**. Tieto metódy sú z triedy **UnitMovementClass**. Ich umožnenie je sledované pomocou parametrov **DoRotate**, **DoMove** a **RotateToOrigin** z triedy **UnitMovementClass**.

4.4 Spôsob boja a zvyšné mechanizmy

Táto časť vysvetľuje implementáciu začatia boja, priebeh boja, spôsob obsadzovania veží a uzdravovacích dedín.

Boj jednotiek

Boj jednotiek začína pomocou triedy **RadiusForAttack**, ktorá každý snímok zisťuje, či sa v vopred danom rádiuse nenachádza nepriateľská jednotka. Táto trieda je použitá ako komponent pre objekt **Agent** z triedy **UnitClass**. Funguje to na tom spôsobe, že pokiaľ nenašla nepriateľa, tak bude volať metódu **Physics2D.OverlapBoxAll()**. Ak nájde, že sa v rádiuse nachádza objekt s názvom „Soldier“, čo je objekt vojaka a zároveň tento vojak nieje vlastným vojakom, buď z tejto alebo inej priateľskej jednotky, tak začne prípravu k boju. Táto príprava označí pomocou parametra „EnemyFound“, že našla nepriateľa a teda už nebude ďalej vyhľadávať. Potom nastaví parameter „Attacking“ jednotky, ku ktorej je pripútaná na hodnotu **true**, priradí nepriateľský objekt v parametre **FightTarger** typu **UnitClass** a začne boj pomocou metódy **Fight()**, parametere **Attacking**, **FightTarger** a metóda **Fight()** sú z triedy **UnitClass**.

Metóda **Fight()** zavolá metódu **Attack()** pr každom svojom vojakovi. Potom si uloží pozíciu pred bojom, aby sa mohla na ňu po boji vrátiť. Tento návrat nieje implementovaný. Následne zavolá korutinu **KeepAgentInSoldiers**, ktorá sa opakuje každých 0,2 sekundy, pokiaľ premenná **FightTarger.Agent** je rozdielna od **null**. Táto korutina udržiava pozíciu **Agenta** v strede jednotiek. Metóda **Attack()**, ktorá je z triedy **SoldierClass**, zapne parameter **Attacking** na hodnotu **true** a zavolá metódu **Attack()** z komponentu **AttackingComponent** pre začatie boja.

Priebeh boja vojakov je riadený komponentom **AttackingComponent**, ktorý je pripojený k objektu každého vojaka. Tu sa každý snímok v metóde **Update()** zisťuje či má daná jednotka útočiť pomocou parametru **Attacking** z triedy **SoldierClass**. No najprv je zavolaná metóda **Attack()**, ktorá si uloží pozíciu vojaka a zavolá metódu **FindEnemy()** pre nájdenie najbližšieho nepriateľského vojaka. Táto metóda pomocou **Physics2D.OverlapCircleAll()** hľadá v rádiuse jednotky pre útok, ak nejakú nájde, tak si ju uloží a spočíta k nej vzdialenosť. Potom nastaví parameter **MoveToAttack**, aby mohol začať presun vojaka k nepriateľovi. Pokiaľ nenájde žiadneho nepriateľa, tak najprv zistí, či sa má približovať k už predtým nájdenému nepriateľovi pomocou parametra **GettingCloser**, ak áno tak spočíta pozíciu na presun a rotáciu pomocou metód **ComputePositionToMove()** a **ComputeRotation()** z **SoldierClass**. Inak zavolá korutinu **RepeatFindMethod()**, ktorá následne zavolá metódu **Attack()** po 0,1 sekunde.

V metóde **Update()**, už je povolený útok, takže prvá vec, ktorá sa musí vykonať je dostatočné priblíženie k nepriateľskému vojakovi. To je implementované tak, že sa najprv aktualizuje kam sa má vojak posunúť a potom sa zavolajú metódy **RotateSoldier()**, **MoveSoldier()** pre vykonanie rotácie a pohybu vojaka z triedy **SoldierClass**. Ďalej sa skontroluje, či je nepriateľ už v rádiuse pomocou metódy **EnemyInAttackingRadius()**, ak áno, tak v ďalšom snímku začne vojak útočiť pomocou metódy **AttackEnemy()**. Metóda **EnemyInAttackingRadius()** vyhodnotí, že je dostatočne blízko vtedy, keď sa tento konkrétny nepriateľ nachádza v rádiuse do 0,5 jednotiek. Na to použije metódu **Physics2D.OverlapCircleAll()**.

Samotný útok je implementovaný ako korutina, ktorá sa opakuje každú sekundu, pokiaľ má vojak dovolené útočiť pomocou parametra **Attacking** z triedy **SoldierClass**, alebo ak je parameter **Enemy** rozdielny od **null**. Celkový útok je vytvorený tak, že volá metódu **Attack()** z triedy **BattleManager** a pokiaľ mu nevráti hodnotu **true**, teda že zabil nepriateľa, tak sa celá korutina opakuje. V prípade, že nepriateľ už nieje v rádiuse, tak bude opakovať postup vyššie a teda priblíženie k cieľu. Pri ukončení korutiny pomocou zmeny parametra **Enemy** z triedy **SoldierClass** je boj celkovo ukončený. V druhom prípade zavolá metódu **Attack()** pre začatie hľadania nového nepriateľa.

Priradenie rozkazu jednotke, ktorá práve bojuje nieje možné, pretože to nebolo implementované. Teraz je to urobené tak, že sa nepodarí priradiť rozkaz jednotke pretože v metóde **PerformMove()** z triedy **UnitClass**, ktorá ako bolo vyššie spomenuté, že riadi jednotku, je kontrola či jednotka bojuje pomocou parametru **Attacking** a ak je ten parameter rovný **true**, tak rovno nastaví parameter **FinishOrder** na **true** a teda rozkaz je hneď dokončený.

Ukončenie boja je implementované tak, že keď v korutine **KeepAgentInSoldiers()** bude parameter **FightTarger.Agent** rovný **null**, tak sa korutina už znovu nezavolá, ale zavolá sa metóda **StopAttack()**, kde sa komponentu **RadiusForAttack** nastaví parameter **EnemyFound** na **false**, tak isto sa nastaví parameter **Attacking** na **false** a u všetkých vojakoch sa nastaví parameter **Attacking** na **false**.

Obsadzovanie veží

Obsadzovanie veží sa zapína pomocou komponentu **CollisionDetectorTower**, ktorý je priradený detekčnému kruhu okolo veže. Tento komponent pomocou metód **OnTriggerEnter2D()** a **OnTriggerExit2D()** zistí, či do detekčného kruhu vstúpila nepriateľská, alebo užívateľova jednotka. Podľa toho zavolá metódu **PlayerEnter()** alebo **EnemyEnter()**, respektíve **PlayerLeaving()** alebo **EnemyLeaving()**, podľa toho či sa jednalo o užívateľove alebo nepriateľské jednotky. Tieto metódy sú z komponentu **TowerCapture**, ktorý je pripojený na prvého rodiča, čo má pod sebou ostatné časti veže ako text, obsadzovací obdĺžnik, detekčný kruh a samotnú vežu.

TowerCapture komponent v korutine **Capture()** je podľa toho, či je viac nepriateľských alebo užívateľových jednotiek v detekčnom kruhu, podľa toho tento mechanizmus obsadzovania veží funguje. Ak je viac nepriateľských, tak obsadzuje nepriateľ a naopak. Funguje to tak, že najprv musí odstrániť nepriateľské obsadenie, čiže ak je napríklad obsadzovací obdĺžnik veže červený, znamená to že veža je pod kontrolou nepriateľa a teda je najprv nutné znížiť tento vplyv na 0 a až potom je možné obsadzovať vežu. Obsadzovanie funguje tak, že sa napĺňa obsadzovací obdĺžnik pre užívateľa zelenou farbou, a počet percent stúpa. Keď dosiahne 100 % tak užívateľ obsadil vežu. Rovnaký proces je implementovaný aj pre nepriateľa, len má opačné farby. Tento proces sa opakuje každé 0,2 sekundy.

Uzdravovacie dediny

Tieto dediny sú objekty, ktoré majú na sebe komponent **VillageHealing**, ktorý pomocou metód **OnTriggerEnter2D()** a **OnTriggerExit2D()** z Unity zistí, či niekto vstúpil do rádiusu pre uzdravovanie. Potom zistí, aká to bola jednotka, či nepriateľova alebo užívateľova, a každé dve sekundy zavolá metódu **HealAndReviveUnit()** z triedy **UnitClass** pre danú jednotku. Táto metóda postupne vojakom pridáva osem bodov života pokiaľ nedosiahnu plný počet životov a oživuje nové jednotky a to tak, že znovu vytvorí nového vojaka a položí ho na neobsadenú pozíciu v koordinátoch v jednotke, až do maximálnej kapacity jednotky.

Kapitola 5

Vyhodnotenie experimentu

Demo verzia hry obsahuje štyri sady ovládania, pri ktorých sa sledovala ich intuitívnosť a jednoduchosť. Výsledkom tohto experimentu je odpoveď, ktoré ovládanie sa ukazuje ako najlepšia voľba pri tvorbe strategických hier pre dotykové zariadenia. Toto zisťovanie bolo vykonané pomocou dotazníkov. Pre tvorbu dotazníka som sa rozhodol použiť štandardizované dotazníky ako SUS a UEQ, ktoré budú popísané ďalej. Dotazník obsahuje 2 sady otázok, kde jedna sada je z SUS a druhá z UEQ, pre každú jednu sadu ovládania. Na konci sa nachádza môj vlastný dotazník, ktorý sa pýta tri jednoduché otázky pre každý typ ovládania. Pre označovanie, pohyb a zmenu formácie sa pýta, ktoré ovládanie bolo najjednoduchšie, najrýchlejšie a najefektívnejšie. Počet mojich účastníkov experimentu bolo šesť. Skrátenú verziu dotazníka je možné nájsť v prílohách.

Experiment prebiehal tak, že som najprv určil v akom poradí má účastník vyskúšať jednotlivé sady ovládania. Každý účastník mal vlastné, unikátne poradie jednotlivých sád. Na začiatku som účastníkovi ukázal ako sa používa univerzálne ovládanie a potom pri každej sade aktuálne ovládane. Následne som mu podľa scenára hovoril čo má robiť. Po každej sade vyplnil dotazník vzťahujúci sa k danému spôsobu ovládania. Na konci vyplnil môj dotazník, kde som mu pripomenul, z akej sady boli jednotlivé ovládania.

5.1 Dotazníky

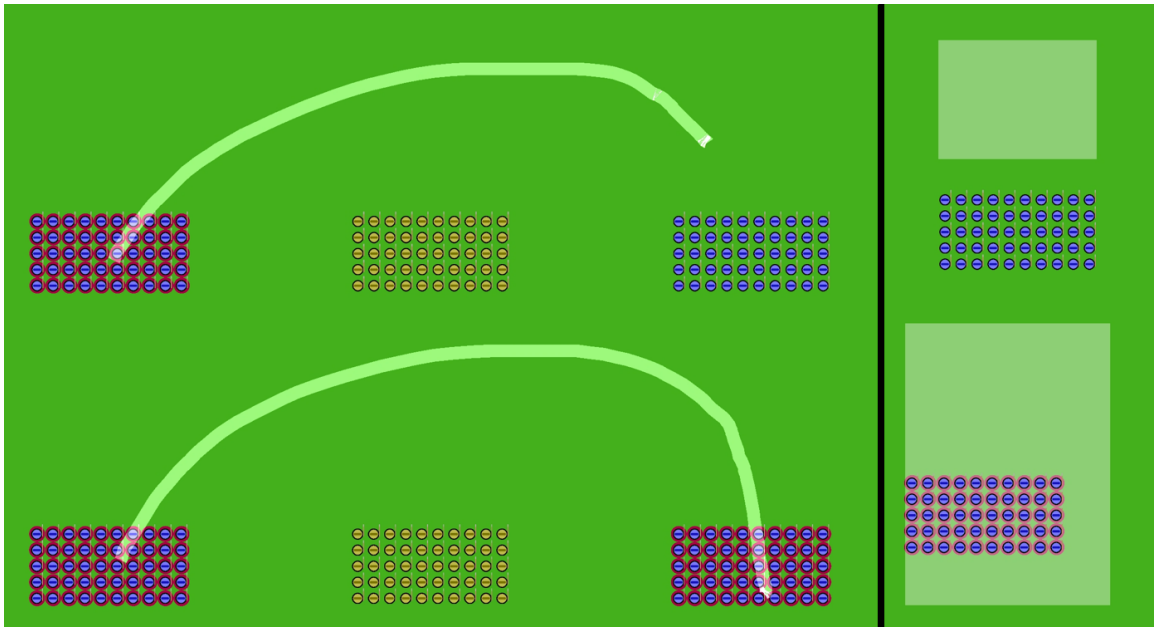
Dotazník sa pýta na jednotlivé sady ovládania tejto demo hry. Tieto sady sú štyri. Každá obsahuje tri základné spôsoby ovládania a to označenie jednotiek, presun vojska a zmenu formácie. Pre označenie boli zvolené 3 spôsoby:

1. Kliknutie na jednotku.
2. Presun prstom cez jednotku.
3. Označovací obdĺžnik.

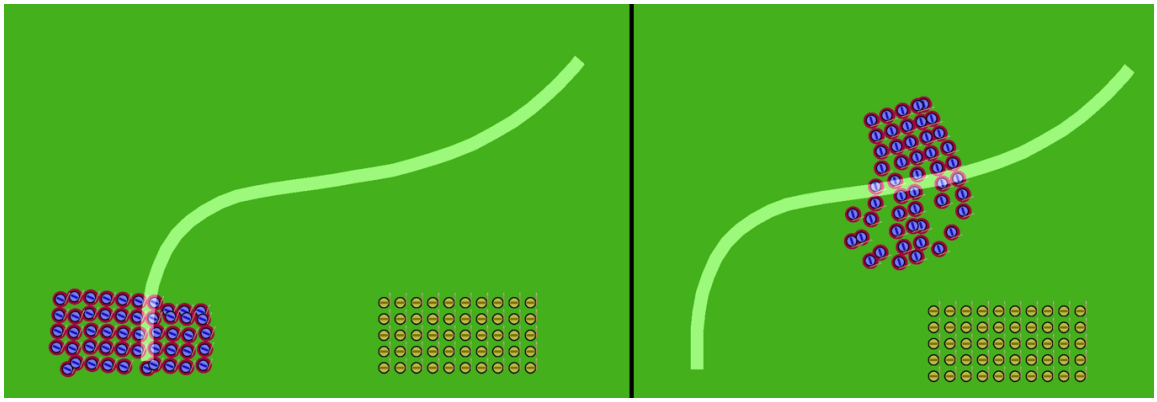
Kliknutie je jasné, ostatné dva typy sú ukázané na obrázku 5.1.

Pre presun vojska boli zvolené len 2 spôsoby presunu, nakoľko na iné možné typy som ani nenarazil. Tu je ich vymenovanie:

1. Kliknutie na miesto.
2. Nakreslenie cesty, po ktorej sa vojsko presunie.



Obrázek 5.1: Označovanie pomocou presunu prstom cez jednotku a obdĺžnikom.



Obrázek 5.2: Presun vojska pomocou nakreslenia cesty.

Nakreslenie cesty je možné vidieť na obrázku 5.2.

Na zmenu formácie boli zvolené tri typy zmeny formácie, kde dve boli moje vlastné a jedno prebrané z hry ROME: Total War - Alexander[1]. Tu je ich zoznam:

1. Podržanie prsta pre rotáciu, ďalšie kliknutie spôsobí zapnutie a vypnutie zmeny formácie.
2. Podržanie prsta pre točenie a zmenu formácie naraz.
3. Podržanie dvoch prstov pre točenie a zmeny formácie naraz.

S týmito spôsobmi ovládania boli vytvorené 4 sady ovládania, ktoré je možné vidieť na obrázku 5.3.

[1] https://play.google.com/store/apps/details?id=com.feralinteractive.rometwalex&hl=cs&gl=US

	Sada 1	Sada 2	Sada 3	Sada 4
Označenie	Kliknutím na jednotku.	Pretiahnutím prstom cez jednotky, treba začat v jednotke.	Podržaž prst a vybrat jednotky pomocou obdĺžnika.	Pretiahnutím prstom cez jednotky, treba začat v jednotke.
Presun	Kliknutim na miesto.	Nakreslením čiary z jednoky.	Kliknutim na miesto.	Kliknutim na miesto.
Zmena Formácie	Podžať jeden prst a točit pre smerovanie jednotky, potom kliknutím ďalším prstom pre zväčšenie a zmenšenie formácie.	Podržanie jedného prsta pre točenie a menenie formácie jednotky.	Podržat dva prsty pre točenie a menenie formácie jednotky. Je možné potom druhý prst pustiť.	Podžať jeden prst a točit pre smerovanie jednotky, potom kliknutím ďalším prstom pre zväčšenie a zmenšenie formácie.

Obrázek 5.3: Štyri sady ovládania

Scenár experimentu

Pre správne vyhodnotenie experimentu bolo dôležité, aby každý účastník spravil presne dopredu dané kroky, a aby každý mal rovnaký zážitok. Preto som napísal scenár, ktorým si každý účastník prešiel. Tu je spomínaný scenár:

- Vyberte dve jednotky rytierov, ktorých práve vidíte (modré jednotky).
- Obsaďte s nimi vežu a potom ich odznačte.
- Označte zostávajúcu jednotku sedliakov (hnedý vojak) a zlikvidujte pohybujúceho sa nepriateľa. Ak sa vám to nepodarí tak použite rytierov.
- Odznačte jednotky a označte jednotky v dolnej ľavej časti mapy.
- Zoraďte ich do formácie tak, aby mohli prejsť priesmykom a zlikvidujte nepriateľa strážiaceho vchod do priesmyku.
- Označte všetkých rytierov a obsaďte druhú vežu a zároveň zlikvidujte nepriateľa.
- Zoraďte jednotky do 2 radov, kde prvý rad sú rytieri a druhý sú sedliaci.
- Zlikvidujte posledných nepriateľov a obsaďte poslednú vežu.

Dotazník SUS

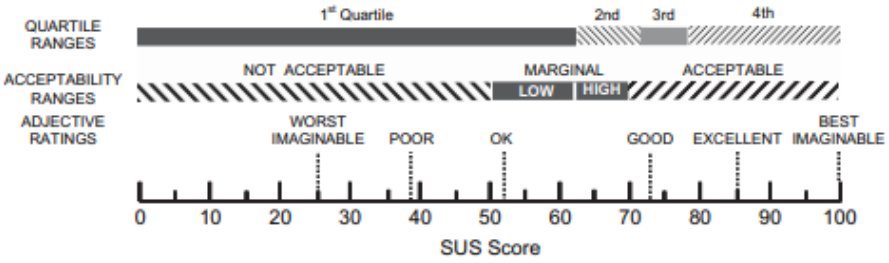
Prvý typ dotazníka je SUS, po slovensky Mierka použiteľnosti systému. Štandartný SUS dotazník pozostáva z týchto desatich otázok, kde párne položky sú formulované záporne a nepárne kladne [11].

1. Myslím, že by som tento produkt často rád používal.
2. Zistil som, že je tento systém zbytočne komplikovaný.
3. Myslím si, že je tento systém ľahké používať.

4. Myslím si, že by som potreboval technickú podporu aby som bol schopný používať tento systém.
5. Zistil som, že rôzne funkcie systému sú dobre spracované.
6. Myslím si, že je tento systém príliš nekonzistentný.
7. Myslím si, že väčšina ľudí by sa tento systém naučila rýchlo používať.
8. Myslím si, že systém je ťažkopádny na používanie.
9. Pri používaní systému sa cítim sebavedome.
10. Myslím si, že sa potrebujem naučiť veľa vecí predtým, než môžem začať s týmto systémom.

Účastník tohto dotazníku dostane na výber z päť bodovej škály, ktorá začína od 1 ako veľmi nesúhlasím až po 5, ktoré je označené ako veľmi súhlasím. Pre vypočítanie skóre, je najprv potrebné posunúť hodnoty na škálu 0 až 4. To sa spraví tak, že od párnych otázok sa odčíta číslo 1 a od nepárnych sa od čísla 5 odčíta hodnota ktorá bola na danej otázke priradená. Následne sa tieto hodnoty sčítajú a potom vynásobia číslom 2,5. Takto sa získa hodnota od 0 po 100 [11].

Výsledné skóre je možné reprezentovať na stupnici univerzitného známkovania, kde 90 až 100 je A a podobne. Z tohto je možné vyvodiť, že skóre 70 a viac je možné označiť ako celkom vyhovujúce. Čím sa ide vyššie tým je hodnotenie lepšie, kde skóre nad 90 je označované ako špičkové. Na obrázku 5.4 je možné vidieť hodnotiacu škálu SUS dotazníku [5].



Obrázek 5.4: SUS hodnotiaca škála [5]

Dotazník UEQ

Hlavným cieľom dotazníku ESQ, user experience questionnaire, po slovensky Dotazník skúseností používateľov, je rýchle a okamžité vyhodnotenie používateľových skúseností. Tento dotazník skúma 6 aspektov pomocou 26. otázok. Tieto aspekty si rozdeľujú týchto 26 otázok. Medzi tieto aspekty patrí atraktivita, učebnosť, účinnosť, ovládateľnosť, stimulácia a novosť [15].

Na atraktívnosť sa dá pozerať ako na celkový dojem užívateľa z produktu. Medzi pragmatické aspekty patria učebnosť, účinnosť, ovládateľnosť a môže sa povedať, že určujú technickú kvalitu produktu. Stimulácia a novosť zobrazujú, ako najradšej by tento produkt užívateľ používal [15]. Tento dotazník je možné zadarmo stiahnuť z webu[2], kde je aj excel

[2] https://www.ueq-online.org/

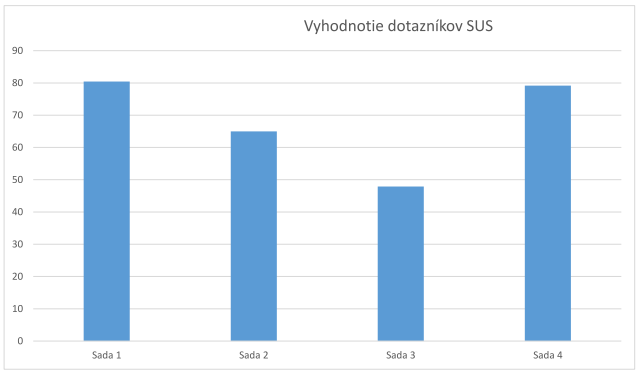
tabuľka pre jeho vyhodnotenie, tiež zadarmo. Jeho hodnotenie nadobúda hodnôt -2 až 2, kde čím vyššia hodnota, tým je na tom daný aspekt lepšie.

5.2 Vyhodnotenie

Na pamäťovom médiu sú uložené excel tabuľky, v ktorých sú výsledky dotazníkov, ich výpočet a konečné grafy.

SUS dotazník

Môj predpoklad bol, že sada 1 a sada 2 si povedú celkovo najlepšie. Podľa hodnotenia dotazníka SUS to aj tak vyšlo. Podľa toho ako sa reprezentuje výsledok skóre SUS sada č. 1 a 4 majú skoro vynikajúce hodnotenia, dosahujú hodnôt 80, čo udáva, že ich spôsoby ovládania boli pre užívateľov jednoduché. Sada 2 dosahuje priemerné hodnotenie, čo je pravdepodobne dané, že všetko bolo riadené kreslením čiar. Najhoršie dopadla sada č. 3 pri ktorej som to aj očakával, nakoľko jej spôsob označovania a zmena formácie boli podľa môjho názoru najťažšie a najnepríjemnejšie. Na tomto obrázku 5.5 je zobrazený daný graf.

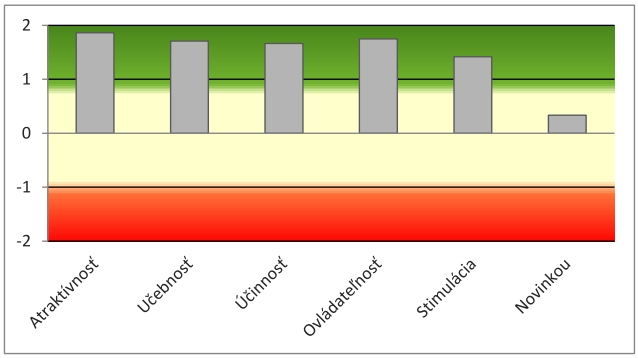


Obrázek 5.5: Vyhodnotenie SUS časti dotazníkov.

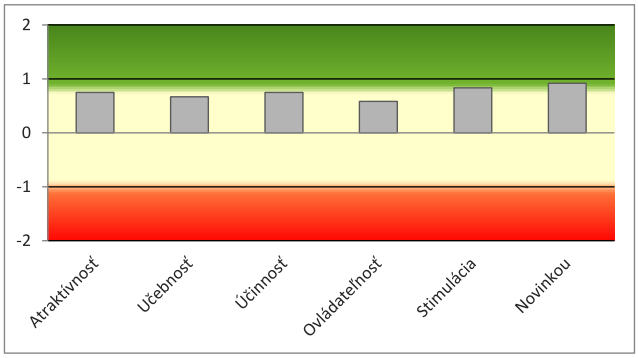
UEQ dotazník

Výsledky tohto dotazníka je možné pozorovať na štyroch grafoch, pre každú sadu zvlášť. Pri sade 1 sú skoro všetky hodnoty väčšie ako 1, čo značí, že je celkovo veľmi dobre hodnotená. Atraktívnosť dokonca skoro dosahuje maximálnu hodnotu, čo je veľmi dobré, ale technické aspekty sú podla mňa dôležitejšie. Tie dosahujú taktiež veľmi dobré čísla. Novosť má skoro podpriemernú hodnotu, ale to nieje až také prekvapenie, nakoľko je táto sada už zaužívaná a ničím neprekvapí. Môžem teda hodnotiť, že sada 1 má skvelé výsledky a teda je doporučená ako hlavný spôsob ovládania strategických hier, hneď za sadou 4. Na obrázku 5.6 je možné vidieť graf.

Pri sade 2 je už možné vidieť, že kvalita tejto sady ovládania klesá. Jej hodnotenie je priemerné a všetky jej aspekty majú hodnoty pod úrovňou 1. Nie je doporučené aby sada 2 bola hlavným ovládaním. Zaujímavosťou by mohlo byť, že má hodnotu novosť väčšiu ako má sada 1, čo však v konečnom dôsledku nie je až tak prekvapujúce, keď pri uvedomení, že sada 1 má klasické ovládanie s ktorým sa používateľ stretne skoro pri každej strategickej hre na dotykové plochy. Na obrázku 5.7 je vidieť výsledný graf tejto sady.

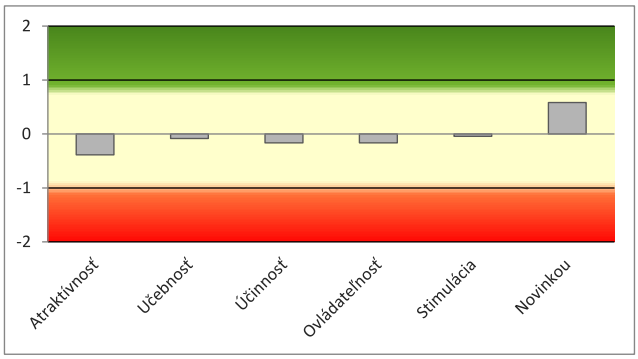


Obrázek 5.6: Vyhodnotenie UEQ dotazníku pre sadu 1.



Obrázek 5.7: Vyhodnotenie UEQ dotazníku pre sadu 2.

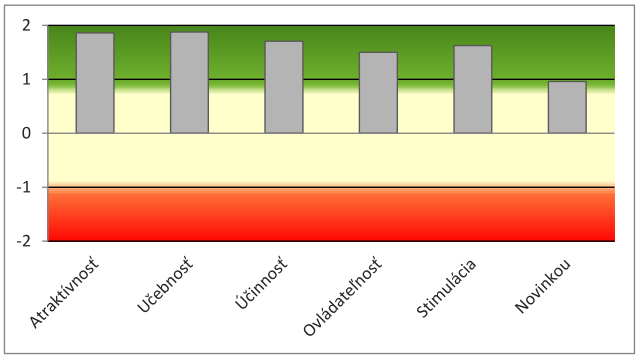
Sada 3 dopadla najhoršie spomedzi ostatných sád. Jej hodnoty okrem novosti, boli záporné. Čo jasne ukazuje, že daná sada ovládania nieje vhodná pre použitie samostatne. Prečo vyšla novosť veľmi dobre oproti ostatným hodnotám mi je záhadou, ale moja hypotéza je, že to predsa len bolo niečo nové oproti klasickému klikaniu. Na obrázku 5.8 je vidieť výsledný graf tejto sady.



Obrázek 5.8: Vyhodnotenie UEQ dotazníku pre sadu 3.

Sada 4 podobne ako sada 1 dopadla veľmi dobre, je to pravdepodobne kvôli tomu, že boli skoro rovnaké až na označovanie jednotiek. Práve toto označovanie zdvihlo aspekt novosti podstatne vyššie ako pri sade 1. Celkovo táto sada vyšla najlepšie spomedzi ostatných sád

a preto ju doporučujem ako hlavnú sadu ovládania. Na obrázku 5.9 je vidieť výsledný graf tejto sady.

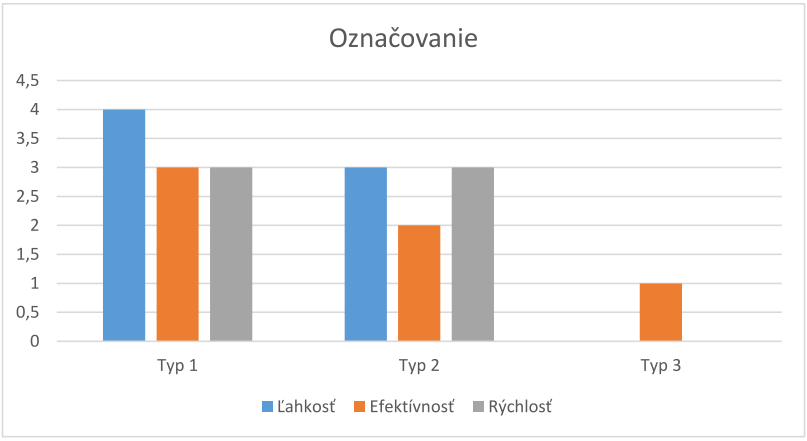


Obrázek 5.9: Vyhodnotenie UEQ dotazníku pre sadu 4.

Môj dotazník

Môj vlastný dotazník bol postavený na hodnotení jednotlivých spôsobov ovládania, jeho výsledky ma prekvapili. Počítal som ho tak, že pre každé zaškrtnutie som vložil do tabuľky hodnotu 1, zvyšné mali hodnoyu 0. Urobil som priemer pre každú otázku, čiže maximálna hodnota pre danú otázku je 6 a najnižšia 0. Na koniec, keďže niektoré sady zdieľajú spôsoby ovládania, tak som vypočítal priemer jednotlivých ovládaní. Na grafoch, ktoré budú ukázané, tak ľavá osa ukazuje koľko účastníkov zvolilo tento spôsob pre danú otázku.

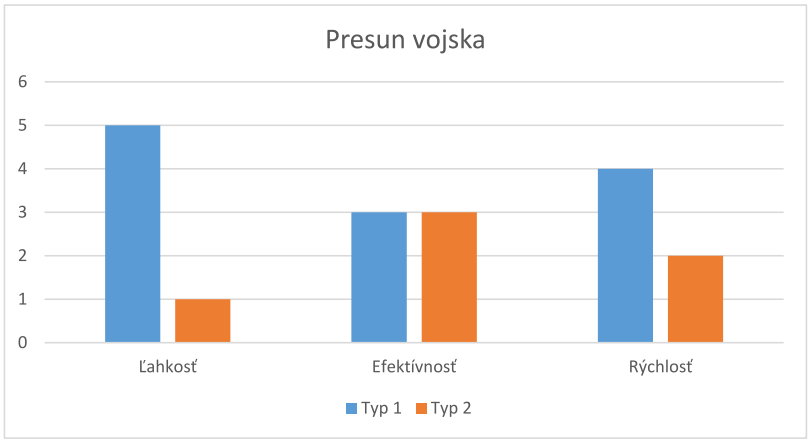
Označovanie dopadlo podľa očakávaní, a to tak, že klikanie bude najlepšie hodnotené. Zaujímavo vyšlo, že aj označovanie prejdením prstom dopadlo celkom dobre. Posledný typ označovania dopadol veľmi nízko, totiž iba jeden účastník mu dal hodnotenie, že je efektívny. Podľa výsledkov môžem teda doporučiť aby strategická hra pre dotykové zariadenia obsahovala hlavne označovanie pomocou kliknutia ale aj pomocou potiahnutím prsta. Na obrázku 5.10 je možné vidieť výsledný graf.



Obrázek 5.10: Graf vyhodnotenia 3 otázok pre označovanie.

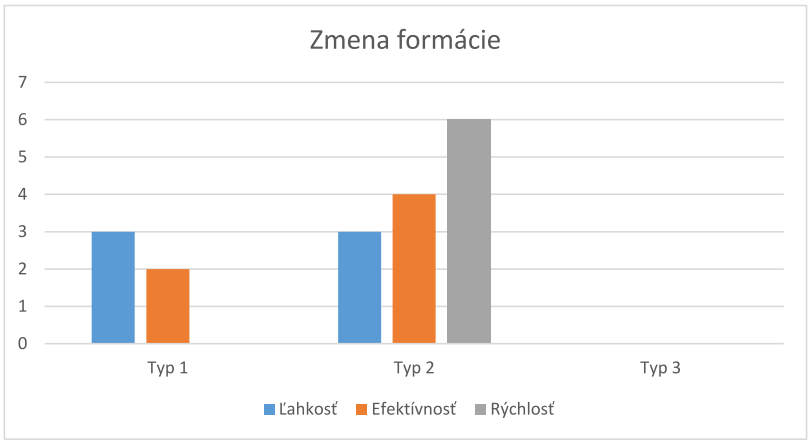
Presun vojska dopadol podľa očakávaní, v tom, že najlepší presun je prosté kliknutie, ktoré zvolilo 5 zo 6 účastníkov, pričom môžem skonštatovať, že pre jednoduchosť treba

zvoliť kliknutie. Pri efektívnosti oba spôsoby dopadli rovnako a pri rýchlosti bolo rýchlejšie klikanie. Tieto výsledky môžeme interpretovať, tak, že hra by v prvom rade mala mať presun pomocou kliknutia ale ako druhotný spôsob je dobré zvoliť kreslenie cesty. Na obrázku 5.11 je možné vidieť výsledný graf.



Obrázek 5.11: Graf vyhodnotenia 3 otázok pre pohyb.

Mňa najviac prekvapilo hodnotenie spôsobov ovládania pre zmenu formácie. Očakával som, že ovládanie zo sady 1 a 4 5.3 bude na tom najlepšie ale nebolo tomu tak. Najlepšie dopadlo ovládanie zo sady 2, teda zmena formácie podržaním prstu, kde je možné rovno točiť a upravovať rady. Vo všetkých aspektoch bolo skoro najlepšie. Všetkých 6 účastníkov sa zhodlo, že bolo najrýchlejšie a najefektívnejšie ovládanie zo všetkých sád. Podľa účastníkov je rovnako ľahké na používanie ako ovládanie zo sady 1 a 4. Spôsob ovládania pomocou dvoch prstov nedostalo žiadny bod. Zaujímave je, že toto ovládanie používa hra ROME: Total War - Alexander[3]. Môžem hodnotiť, že ovládanie zo sady 2 je najlepšou voľbou pre zmenu formácie jednotiek. Znovu na obrázku 5.12 je graf hodnotenia.



Obrázek 5.12: Graf vyhodnotenia 3 otázok pre zmenu formácie.

[3] https://play.google.com/store/apps/details?id=com.feralinteractive.rometwalex&hl=cs&gl=US

5.3 Zhrnutie hodnotenia

Celkovým zhrnutím experimentu dochádzam k záveru, že podľa dotazníkov SUS a UEQ sú sady 1 a 4 najlepšími spôsobmi, ako ovládať strategické hry. V oboch dotazníkoch ukázali veľmi dobré výsledky a preto je možné ich takto hodnotiť. Keď zarátam posledný dotazník môžem vytvoriť novú sadu ovládania, ktorá je podľa dát tou najlepšou možnou z ponúkaných. Ako bolo vidieť v poslednom dotazníku, ovládanie a označovanie sedelo so sadami 1 a 4, teda je vhodné použiť obidve. Pri presune vojska v konečnom dôsledku ostáva ovládanie zo sád 1 a 4, pretože bolo najľahšie a najrýchlejšie. No ovládanie pre zmenu formácie vojska bolo podľa dát najlepšie v sade 2 a to treba zarátať. Konečne nová sada ovládania, ktorá by mala byť prítomná pri strategických hrách na dotykové plochy by mohla vyzerať nasledovne:

1. Označovanie:

   Kliknutie na jednotky.

   Prejdením prstom cez jednotky.

2. Presun vojska:

   Kliknutím na miesto.

3. Zmena formácie:

   Podržať prst pre točenie jednotky a zmeny formácie.

Podľa výsledkov by toto malo byť hlavným ovládaním. Hlavným sa myslí, aby iné spôsoby ovládania, ktoré by sa mohli prekrývať s vyššie uvedenými, neboli implementované.

Kapitola 6

Záver

Cieľom tejto práce bolo preskúmať rôzne spôsoby ovládania strategických hier pre dotykové zariadenia. Zhrnúť tieto poznatky a využiť ich pri vlastnom návrhu demo verzie hry na ktorej bol spravený experiment. Súčasne bolo potrebné navrhnúť a implementovať rôzne spôsoby ovládania a vytvoriť z nich sady, na ktorých sa bude testovať.

Výsledkom tejto práce je demo verzia hry, ktorú je možné si zahrať štyrmi rôznymi sadami ovládania a obsahuje jednu misiu, ktorú si môže užívateľ zahrať. Na pamäťovom médiu je uložené video, ktoré prezentuje kľúčové vlastnosti výsledného riešenia. Najdôležitejším výsledkom je vyhodnotenie experimentu z ktorého vznikla nová sada ovládania, ktorá je podľa výsledkov intuitívna, jednoduchá a strategická hra pre dotykové zariadenia by ju mala obsahovať.

Literatura

[1] Amin, D. a Govilkar, S. Comparative study of augmented reality SDKs. *International Journal on Computational Science & Applications*. 2015, zv. 5, č. 1, s. 11–26.

[2] Ayer, S. K., Messner, J. I. a Anumba, C. J. Augmented Reality Gaming in Sustainable Design Education. *Journal of Architectural Engineering*. 2016, zv. 22, č. 1, s. 04015012. DOI: 10.1061/(ASCE)AE.1943-5568.0000195. Dostupné z: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29AE.1943-5568.0000195.

[3] Azuma, R. T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*. August 1997, zv. 6, č. 4, s. 355–385. DOI: 10.1162/pres.1997.6.4.355. Dostupné z: https://doi.org/10.1162/pres.1997.6.4.355.

[4] Bademosi, F., Blinn, N. a Issa, R. R. Use of augmented reality technology to enhance comprehension of construction assemblies. *J. Inf. Technol. Constr.* 2019, zv. 24, č. 4, s. 58–79.

[5] Bangor, A., Kortum, P. T. a Miller, J. T. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction*. Taylor & Francis. 2008, zv. 24, č. 6, s. 574–594. DOI: 10.1080/10447310802205776. Dostupné z: https://doi.org/10.1080/10447310802205776.

[6] Beijing Puppy Robotics Co., L. *Hachi Infinite M1, the First-Ever AI-Powered Interactive Touchscreen Projector, Launches on Amazon*. 2020. [Online; naposledny navšívené Apríl 15, 2021]. Dostupné z: https://www.prnewswire.com/news-releases/hachi-infinite-m1-the-first-ever-ai-powered-interactive-touchscreen-projector-launches-on-amazon-301120290.html.

[7] Benko, H., Jota, R. a Wilson, A. MirageTable: Freehand Interaction on a Projected Augmented Reality Tabletop. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2012, s. 199–208. CHI '12. DOI: 10.1145/2207676.2207704. ISBN 9781450310154. Dostupné z: https://doi.org/10.1145/2207676.2207704.

[8] Billinghurst, M., Clark, A. a Lee, G. A Survey of Augmented Reality. *Foundations and Trends® in Human–Computer Interaction*. 2015, zv. 8, 2-3, s. 73–272. DOI: 10.1561/1100000049. ISSN 1551-3955. Dostupné z: http://dx.doi.org/10.1561/1100000049.

[9] Conor Cawley, Flickr, Gabriel Jorby. *Seriously, Why Don't We Have Real Holograms Yet?* 2018. [Online; naposledny navšívené Apríl 16, 2013]. Dostupné z: https://tech.co/news/seriously-no-real-holograms-yet-2018-01.

[10] Kesim, M. a Ozarslan, Y. Augmented Reality in Education: Current Technologies and the Potential for Education. *Procedia - Social and Behavioral Sciences*. 2012, zv. 47, s. 297–302. DOI: https://doi.org/10.1016/j.sbspro.2012.06.654. ISSN 1877-0428. Cyprus International Conference on Educational Research (CY-ICER-2012)North Cyprus, US08-10 February, 2012. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877042812023907.

[11] Lewis, J. R. a Sauro, J. The Factor Structure of the System Usability Scale. In: Kurosu, M., ed. *Human Centered Design*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 94–103. ISBN 978-3-642-02806-9.

[12] Milgram, P. a Kishino, F. A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems*. December 1994, vol. E77-D, no. 12, s. 1321–1329.

[13] Piumsomboon, T., Day, A., Ens, B., Lee, Y., Lee, G. et al. Exploring enhancements for remote mixed reality collaboration. In:. November 2017, s. 1–5. DOI: 10.1145/3132787.3139200.

[14] puppyrobot. *Tech Specs*. 2021. [Online; naposledny navšívené December 12, 2021]. Dostupné z: http://www.hachismart.com/en/hachiinfinite.

[15] Schrepp, M., Hinderks, A. a Thomaschewski, J. Applying the User Experience Questionnaire (UEQ) in Different Evaluation Scenarios. In: Marcus, A., ed. *Design, User Experience, and Usability. Theories, Methods, and Tools for Designing the User Experience*. Cham: Springer International Publishing, 2014, s. 383–392. ISBN 978-3-319-07668-3.

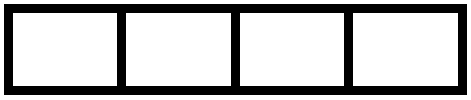
[16] Vávra, P., Roman, J., Zonča, P., Ihnát, P., Němec, M. et al. Recent Development of Augmented Reality in Surgery: A Review. *Journal of Healthcare Engineering*. Hindawi. Aug 2017, zv. 2017, s. 4574172. DOI: 10.1155/2017/4574172. ISSN 2040-2295. Dostupné z: https://doi.org/10.1155/2017/4574172.

Příloha A

Dotazník

Dotazník pre vyhodnotenie sady ovládania
demo hry

Vaše poradie výberu ovládacej sady je



Vopred Vám ďakujem za účasť.

Dnešný dátum: .....

	Sada 1	Sada 2	Sada 3	Sada 4
Označenie	Kliknutím na jednotku.	Pretiahnutím prstom cez jednotky, treba začať v jednotke.	Podržať prst a vybrať jednotky pomocou obdĺžnika.	Pretiahnutím prstom cez jednotky, treba začať v jednotke.
Presun	Kliknutím na miesto.	Nakreslením čiary z jednotky.	Kliknutím na miesto.	Kliknutím na miesto.
Zmena Formácie	Podržať jeden prst a točiť pre smerovanie jednotky, potom kliknutím ďalším prstom pre zväčšenie a zmenšenie formácie.	Podržanie jedného prsta pre točenie a menenie formácie jednotky.	Podržať dva prsty pre točenie a menenie formácie jednotky. Je možné potom druhý prst pustiť.	Podržať jeden prst a točiť pre smerovanie jednotky, potom kliknutím ďalším prstom pre zväčšenie a zmenšenie formácie.

Univerzálne ovládanie:

- Drag pre presun po mape.
- Pinch in/out pre priblíženie/oddialenie kamery.
- Dvojklik pre odznačenie vojakov.

Scenár:

1. Vyberte dve jednotky rytierov, ktorých práve vidíte (modré jednotky).
2. Obsaďte s nimi vežu a potom ich odznačte.
3. Označte zostávajúcu jednotku sedliakov (hnedý vojak) a zlikvidujte pohybujúceho sa nepriateľa. Ak sa vám to nepodarí tak použite rytierov.
4. Odznačte jednotky a označte jednotky v dolnej ľavej časti mapy.
5. Zoraďte ich do formácie tak, aby mohli prejsť priesmykom a zlikvidujte nepriateľa strážiaceho vchod do priesmyku.
6. Označte všetkých rytierov a obsaďte druhú vežu a zároveň zlikvidujte nepriateľa.
7. Zoraďte jednotky do 2 radov, kde prvý rad sú rytieri a druhý sú sedliaci.
8. Zlikvidujte posledných nepriateľov a obsaďte poslednú vežu.

Vždy zaškrtnite jednu z možností. Rozhodujte sa prosím spontánne. Nepremýšľajte nad svojim rozhodnutím príliš dlho, aby ste naozaj sprostredkovali svoj prvotný dojem.

	veľmi nesúhlasím	nesúhlasím	neviem	súhlasím	veľmi súhlasím
Myslím, že by som toto ovládanie rád používal.	☐	☐	☐	☐	☐
Zistil som, že je toto ovládanie zbytočne komplikované.	☐	☐	☐	☐	☐
Myslím si, že je toto ovládanie ľahké používať.	☐	☐	☐	☐	☐
Myslím si, že by som potreboval technickú podporu aby som bol schopný toto ovládanie používať.	☐	☐	☐	☐	☐
Zistil som, že rôzne funkcie tohto ovládania sú dobre spracované.	☐	☐	☐	☐	☐
Myslím si, že je toto ovládanie príliš nekonzistentné.	☐	☐	☐	☐	☐
Myslím si, že väčšina ľudí by sa toto ovládanie naučila rýchlo používať.	☐	☐	☐	☐	☐
Myslím si, že toto ovládanie je ťažkopádne na používanie.	☐	☐	☐	☐	☐
Pri používaní tohto ovládania sa cítim sebavedome.	☐	☐	☐	☐	☐
Myslím si, že sa potrebujem naučiť veľa vecí predtým, než môžem začať používať toto ovládanie.	☐	☐	☐	☐	☐

Položky dotazníka sú dvojice protikladných vlastností, ktorými posudzujete sadu ovládania. Krúžky medzi vlastnosťami reprezentujú jednotlivé stupne medzi protikladmi.

	1	2	3	4	5	6	7	
obťažujúci	□	□	□	□	□	□	□	**pútavý**
nepochopiteľný	□	□	□	□	□	□	□	**pochopiteľný**
nápaditý	□	□	□	□	□	□	□	**tuctový**
intuitívny	□	□	□	□	□	□	□	**neintuitívny**
hodnotný	□	□	□	□	□	□	□	**menejcenný**
nudný	□	□	□	□	□	□	□	**vzrušujúci**
nezaujímavý	□	□	□	□	□	□	□	**zaujímavý**
nepredvídateľný	□	□	□	□	□	□	□	**predvídateľný**
rýchly	□	□	□	□	□	□	□	**pomalý**
moderný	□	□	□	□	□	□	□	**tradičný**
obmedzujúci	□	□	□	□	□	□	□	**podporujúci**
dobrý	□	□	□	□	□	□	□	**zlý**
zložitý	□	□	□	□	□	□	□	**jednoduchý**
odpudzujúci	□	□	□	□	□	□	□	**potešujúci**
bežný	□	□	□	□	□	□	□	**špičkový**
nepríjemný	□	□	□	□	□	□	□	**príjemný**
spoľahlivý	□	□	□	□	□	□	□	**nespoľahlivý**
motivujúci	□	□	□	□	□	□	□	**demotivujúci**
spĺňajúci očakávania	□	□	□	□	□	□	□	**nespĺňajúci očakávania**
neefektívny	□	□	□	□	□	□	□	**efektívny**
jasný	□	□	□	□	□	□	□	**mätúci**
nepraktický	□	□	□	□	□	□	□	**praktický**
prehľadný	□	□	□	□	□	□	□	**neprehľadný**
príťažlivý	□	□	□	□	□	□	□	**nepríťažlivý**
sympatický	□	□	□	□	□	□	□	**nesympatický**
konzervatívny	□	□	□	□	□	□	□	**inovatívny**

Vždy zaškrtnite jednu z možností. Rozhodujte sa prosím spontánne. Nepremýšľajte nad svojim rozhodnutím príliš dlho, aby ste naozaj sprostredkovali svoj prvotný dojem.

Môžete zvoliť viac ako jednu možnosť.

	Sada 1	Sada 2	Sada 3	Sada 4
Najľahšie označovanie jednotiek.	☐	☐	☐	☐
Najefektívnejšie označovanie jednotiek.	☐	☐	☐	☐
Najrýchlejšie označovanie jednotiek.	☐	☐	☐	☐
Najľahší presun vojska.	☐	☐	☐	☐
Najefektívnejší presun vojska.	☐	☐	☐	☐
Najrýchlejší presun vojska.	☐	☐	☐	☐
Najľahšia zmena formácie jednotiek.	☐	☐	☐	☐
Najefektívnejšia zmena formácie jednotiek.	☐	☐	☐	☐
Najrýchlejšia zmena formácie jednotiek.	☐	☐	☐	☐