

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



**SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM**

AlgoRhythmics Universe: 3D Játékmechanikák

DIPLOMADOLGOZAT

Témavezető:

dr. Osztián Erika,
Egyetemi adjunktus
Osztián Pálma-Rozália,
Egyetemi tanársegéd

Végzős hallgató:

Kovacs Lehel-Levente

2023

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ**



**UNIVERSITATEA
SAPIENTIA**

AlgoRhythms Universe: Mecanicile jocului 3D

LUCRARE DE DIPLOMĂ

Coordonator științific:
dr. Osztián Erika,
Lector universitar
Osztián Pálma-Rozália,
Asistent universitar

Absolvent:
Kovacs Lehel-Levente

2023

SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION



SAPIENTIA HUNGARIAN UNIVERSITY OF TRANSYLVANIA

AlgoRythmics Universe: 3D Game Mechanics

BACHELOR THESIS

Scientific advisor: Student:
dr. Osztián Erika, Kovacs Lehel-Levente
Lecturer
Osztián Pálma-Rozália,
Assistant professor

Declarație

Subsemnatul/a **KOVACS LEHEL-LEVENTE**, absolvent(ă) al/a specializării **INFORMATICĂ**, promoția **2023**, cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, **TÂRGU-MUREȘ**,
Data: **2023.06.06**

Absolvent

Semnătura.....

LUCRARE DE DIPLOMĂ

Coordonator științific:
Lect. univ. dr. Osztian Erika Candidat: **Kovacs Lehel-Levente**
Anul absolvirii: **2023**

a) Tema lucrării de licență:

Proiectarea și dezvoltarea mecanicii unui joc video educativ 3D, realizat folosind motorul de joc Unity, cu scopul de a învăța începătorii cum să codeze într-un mod gamificat.

b) Problemele principale tratate:

- Realizarea unui studiu bibliografic privind mecanicile unui joc video.
- Studiul motorului de joc Unity și a tehnologiilor oferite de acesta.
- Proiectarea și realizarea unui joc video educativ 3D, cu mecanici și gameplay utilizând aceste informații.
- Documentarea corespunzătoare a importanței proiectării mecanicilor de joc.

c) Desene obligatorii:

- Diagrame de proiectare pentru aplicația software realizată

d) Softuri obligatorii:

Dezvoltarea unui joc 3D utilizând tehnologiile C# și Unity, care permite utilizatorilor să învețe noțiuni de bază în programare. Jocul prezintă diverse concepte de programare, cum ar fi structura if/else/switch, instrucțiunile repetitive, tipurile de date etc., într-un univers 3D, fiecare capitol de programare fiind asociat cu o hartă realizată în Wonderdraft. Personajul principal al jocului este Clippy, ale cărui mișcări și îndeplinirea sarcinilor în joc sunt implementate folosind scripturi și animații.

e) Bibliografia recomandată:

- [1] Game mechanics: advanced game design / Ernest Adams and Joris Dormans, 2012
- [2] Defining game mechanics / Miguel Sicart
- [3] <https://docs.unity.com/>

f) Termene obligatorii de consultații: săptămânal, preponderent online

g) Locul și durata practicii: Universitatea „Sapienția” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, laboratorul de informatică (sala 415)
Primit tema la data de: 20.05.2022.
Termen de predare: 02.07.2023.

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului
programului de studiu

Semnătura candidatului

Kivonat

A technológia rohamos fejlődésével számos új lehetőség nyílt meg előttünk, amelyek gyökeresen átalakítottak minden napirendünket. Ezek a változások nem csupán a kommunikációt és a munkavégzést érintették, hanem hatással voltak a szórakozásra és a tanulásra is.

A videójátékok interaktív szórakoztatást kínálnak felhasználóik számára, amik egy másik ember által kitalált világába kalauzolja el okét. Olyan eseményekkel tudunk interakcióba lépni, amelyeket száz évvel ezelőtt még elképzelni sem tudtunk, mivel azok irreálisnak tűntek számunkra. Azonban a modern játékfejlesztési technológiák lehetőséget nyújtanak, hogy ezeket egy virtuális környezetben megtapasztaljuk.

Képesek vagyunk harcokat szimulálni, városokat építeni, űrhajókat vezérelni a galaxis bármelyik pontján, vagy akár megfigyelni, hogyan robban fel egy bolygó. Filmek vagy akár teljes könyvek is adaptálhatóak, hogy azok cselekményét interaktívan átélhessük. Egyetlen korlát: az emberi elképzelés.

A videójátékok nem csupán egy teljesen új, végtelen szórakozási világot teremtettek, hanem azt is lehetővé teszik számunkra, hogy játékositva taníthassunk az iskolák vagy akár egyetemek keretein belül. Pénzügyektől a programozásig minden területen találhatunk legalább egy olyan játékot, amelynek célja a felhasználó fejlődése.

Dolgozatom fő témája azon szoftverek alapjait tárgyalja, amelyek interaktív szórakoztatást nyújtanak az elektronikus eszközökön. Rengeteg szoftver áll rendelkezésünkre, hogy minimális idővel és anyagi ráfordítással játékot építsünk és eladhassunk. Viszont fontos megismerni a játékkészítésnek az alapvető elemeit, a játékszabályokat, másszóval a játék mechanikákat. Ezek technikai megfontolást igényelnek annak érdekében, hogy szoftverünk sikeresen elérje a kitűzött célt.

Rezumat

Cu dezvoltarea rapidă a tehnologiei, s-au deschis multe noi posibilități în fața noastră, care au transformat în mod fundamental viața noastră de zi cu zi. Aceste schimbări nu au afectat doar comunicarea și munca, ci au avut, de asemenea, un impact asupra divertismentului și învățării.

Jocurile video oferă divertisment interactiv utilizatorilor, transportându-i într-o lume creată de altă persoană. Putem interacționa cu evenimente pe care acum o sută de ani nu le-am putut imagina, deoarece ni s-au părut ireale. Cu toate acestea, tehnologiile moderne de dezvoltare a jocurilor oferă posibilitatea de a experimenta acestea într-un mediu virtual.

Putem simula lupte, construi orașe, pilota nave spațiale în orice colț al galaxiei sau chiar observa cum explodează o planetă. Filmele sau chiar cărțile întregi pot fi adaptate pentru a trăi experiența lor în mod interactiv. Singura limită este imaginația umană.

Jocurile video nu au creat doar o lume complet nouă și infinită de distracție, ci ne permit să învățăm jucând, chiar și în școli sau universități. De la aspectele financiare până la programare, putem găsi cel puțin un joc în fiecare domeniu care își propune să dezvolte utilizatorul.

Tema principală a lucrării mele discută fundamental acestor software-uri care oferă divertisment interactiv pe dispozitivele electronice. Avem la dispoziție o mulțime de software-uri pentru a construi și vinde jocuri cu un efort și cost minim. Cu toate acestea, este important să cunoaștem elementele de bază ale creării jocurilor, regulile jocului, sau, cu alte cuvinte, mecanicile de joc. Acestea necesită o abordare tehnică pentru ca software-ul nostru să atingă cu succes scopul propus.

Cuvinte cheie: **Mecanicile jocului, Unity, Design de joc, Gameplay, 3D**

Abstract

With the rapid advancement of technology, numerous new possibilities have opened up before us, fundamentally transforming our everyday lives. These changes have not only affected communication and work but have also had an impact on entertainment and learning.

Video games offer interactive entertainment, immersing their users in a world created by another person. We can interact with events that we couldn't even imagine a hundred years ago because they seemed unrealistic to us. However, modern game development technologies allow us to experience these in a virtual environment.

We are capable of simulating battles, constructing cities, piloting spaceships anywhere in the galaxy, or even observing the explosion of a planet. Movies or even entire books can be adapted so that we can live through their stories interactively. The only limit is human imagination.

Video games have not only created an entirely new and infinite world of entertainment but also enable us to gamify education within the framework of schools and universities. In every field, from finance to programming, we can find at least one game aimed at the user's development.

The main topic of my essay discusses the foundations of software that provide interactive entertainment on electronic devices. We have a multitude of software available to build and sell games with minimal time and financial investment. However, it is important to understand the fundamental elements of game development, the rules of the game, in other words, the game mechanics. These require technical considerations in order for our software to successfully achieve its intended goal.

Keywords: **Game Mechanics, Unity, Game Design, Gameplay, 3D**

Tartalomjegyzék

1. Bevezető	10
2. Irodalmi áttekintés	11
2.1. Mechanikák	11
2.2. Tervezés	14
2.3. Játékmenet	14
3. Hasonló alkalmazások	16
3.1. SHENZHEN I/O	16
3.2. Human Resource Machine	16
3.3. Super Paper Mario	17
4. Programok, technológiák bemutatása	19
4.1. Unity játékfejlesztő motor áttekintése	19
4.2. Unity szerkesztő	19
4.3. Unity eszköztár	22
4.4. Wonderdraft	22
5. Szoftver	23
5.1. Funkcionális követelmények	23
5.2. Nem funkcionális követelmények	24
5.3. A szoftver bemutatása	24
5.3.1. Játékelemek és rendszerek	24
5.3.2. Játékmenet, vezérlés és animáció	27
5.4. Ütközés és fizika	32
5.4.1. Hang- és vizuális effektek	32
5.5. A szoftver megírásához használt könyvtárak	34
5.6. Diagramok	34
5.6.1. Use Case diagram	34
5.6.2. Osztálydiagram	35
5.6.3. Szekvencia diagram	36
6. Limitációk	38
Összefoglaló	39
Irodalomjegyzék	40

1. fejezet

Bevezető

Mik a játék mechanikák? Hogyan működnek? Miért hasznosak? Illetve hogyan tudnak segíteni a játékositott tanításban? Röviden összefoglalva, a játék mechanikák azok a rendszerek és szabályok, amelyek egy játék alapjait írják le. Ezek határozzák meg, hogy hogyan lehet játszani, milyen döntésekkel lehet hozni, illetve ezek a döntések hogyan befolyásolják a játékmenetet.

A játék mechanikák többfélék lehetnek, úgy hatásuk, mint komplexitásuk is különböző. Emiatt megértésük szükséges ahhoz, hogy jól kidolgozott mechanikákat tudunk megtervezni. Egy tanító jellegű játékban ezek az eszközök, amelyekkel a játékos a játék által kínált kihívásokat tudja megoldani. A tervezés sokfélesége miatt ajánlott sok időt belefektetni ezek jól kigondolására és összehangolására, annak érdekében, hogy természetesen legyenek a felhasználó számára.

Az eredmény eléréséhez a mechanikák használata lehet egy statikus helyes sorrendben történő szekvencia, amelyet a megfejtéshez szükséges idő szerint tudunk mérni, vagy lehet egy dinamikusabb komplex rendszer is, ahol a mechanikák kombinációival a tervezésnél is többféle eredményt tudunk elérni, ezáltal a játék témától függően ezeket az idő bonyolultság mellett, hatékonyúságuk szerint is tudjuk mérni.

Az elmúlt időszakban, az AlgoRhythmic univerzum bővítésében vettet részt, ahol egy tanító jellegű 3D játék elkészítését vállaltam. Egy olyan szoftver elkészítését túztük ki célul, amely segít elsajátítani az alapvető programozási fogalmakat, főként kezdők számára. A játék összetett és a játékmenet három részből áll: egy háromdimenziós "Felfedező" játékmódból, egy kétdimenziós "Rejtvény" játékmódból és egy hídból, amely lehetővé teszi a váltást közöttük.

A dolgozat további részében bemutatom a játék mechanikákat, valamint a tanító jellegű játék háromdimenziós játékmód és a játékmódok közti váltás tervezését, fejlesztését és működését.

2. fejezet

Irodalmi áttekintés

2.1. Mechanikák

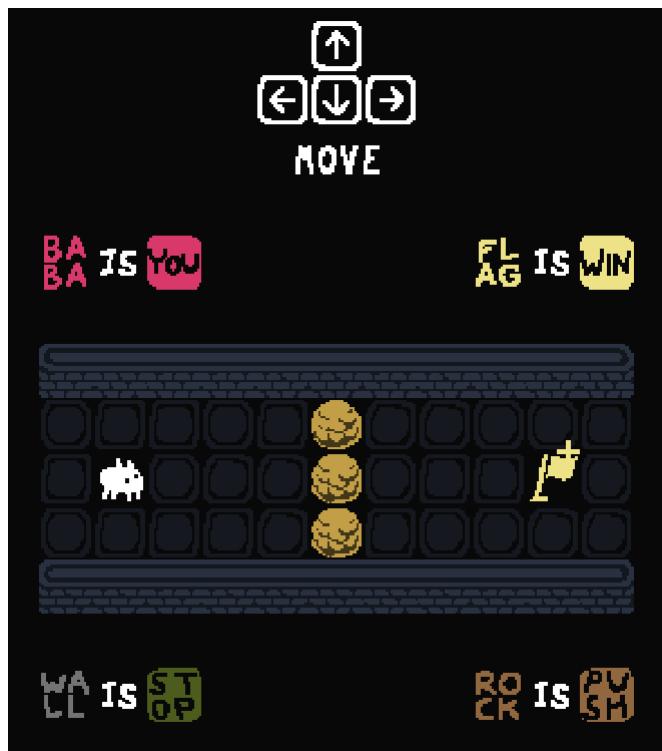
A játék mechanikák szerves része minden asztali játéknak, videojátéknak, és olyan szoftvernek, ami játéknak tekinthető. Ezek szabják meg azokat a rendszereket és szabályokat, amik megalapozzák a játék működését és az interakciókat a játékos és a játékvilág között.

Az egyik legegyszerűbb formája a játékmechanikáknak az asztali sakkban található, mint játékszabály. A játék mező egy 8x8-as fekete fehér tábla, amelyen 6 féle játék darab található, mindegyik más es más mozgással. A bánya csak vízszintesen és függőlegesen, míg a futár csak átlóban mozoghat, a ló egy különlegesebb L alakban ugrik más egységeken át. A királynő a legerősebb egység, egyszere úgy tud mozogni, mint egy bánya és mint egy futár, emiatt képes a legtöbb területet támadás alatt tartani. A király egy távolságot mozoghat bármilyen irányba, emiatt gyenge, de ö a legfontosabb a játékban mivel az elfoglalása jelenti a játszma vesztét, ezért ügyelni kell, hogy ne legyen kitéve veszélynek. A legszimplább a paraszt, aki csak egyesével mozoghat mindig csak előre, első mozgásként kettőt is léphet, viszont, ha alkalom akad rá, átlóban is támadhat. Ez a két kicsi szabály a legegyszerűbb játékdarabnak is olyan lehetőséget ad, amelynek nagy stratégiai hatása lehet a játszma során.

2017-es Nordic Game Jam-hez készült a [Baba is You](#) nevű kirakós videojáték, amit egy független finn fejlesztő demónként készített majd később továbbfejlesztett és kibővíttet, 2019-ben pedig teljes játékként kiadta PC-re és Nintendo Switch-re. A játék egyedi mechanikát tartalmaz, mégpedig a "szabályok" manipulálása a játékterületen mozgatható lapkák által. A lapkák szavakat jelentenek, átrendezésük megváltoztatja a játékkal való interakciót, illetve azt ahogyan működik a játékmenet.

A játékos egy "Baba" nevű figurát mozgat, célja ellökni a lapkákat és a tárgyakat úgy, hogy megoldja a kihívást és a célba jusson. minden szinten a játékos egy egyképernyős rejtvényt kap, amely különféle tárgyakból és lapkákból áll. (lásd 2.1 ábra) A lapkák három félénk lehetnek: a főnevek, amelyek a pályán levő objektumok, mint például a Baba, a kövek (ROCK), vagy a célzászló (FLAG); a leíró lapkák, amelyek tulajdonságokat határoznak meg, mint pld a YOU, ami a játékos által irányíthatóságot jelenti, PUSH és PULL, ami a mozgathatóságot teszi lehetővé, a STOP ami átjárhatatlanná tesz valamit, és a WIN ami a célobjektumot határozza meg, ez általában a zászló objektum; az igék, mint az IS és a HAS, ami a főneveket és a leíró lapkákat kapcsolja össze, létrehozván egy

szabályt, amely meghatározza az objektumok viselkedését, addig amíg ez a karakterlánc nincs feltörve.

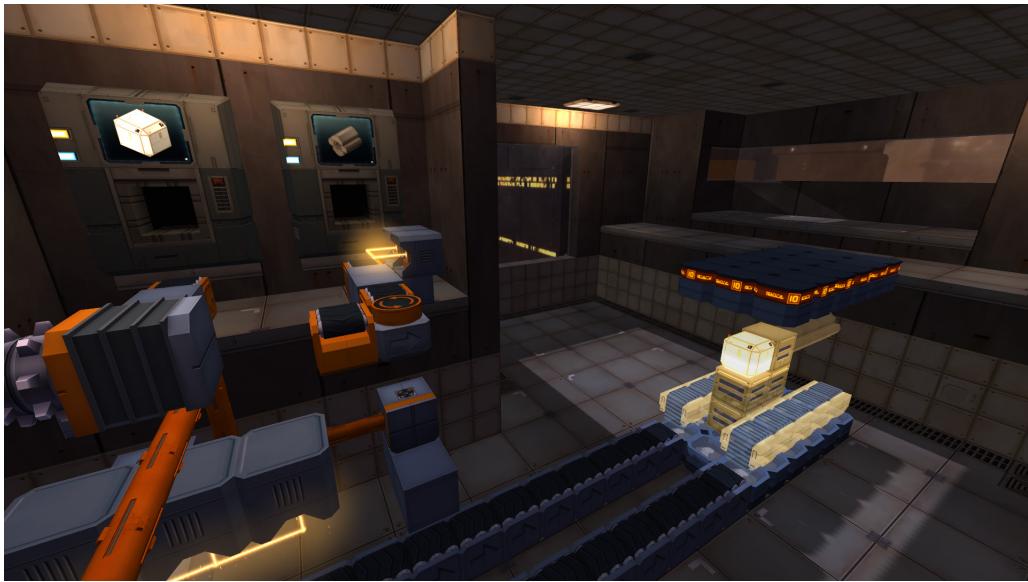


2.1. ábra. Baba is You, bevezető pálya

Egy ilyen játék pályáinak a tervezése nehéz feladat, mert a játékszabályok manipulálhatóak. Vannak olyan pályák, ahol a fejlesztő korlátokat állít fel és köteleztet arra, hogy ne tudjunk felbontani egyes szabályokat annak érdekében, hogy az előre meghatározott kihívást fejtsük meg, nem pedig a legrövidebb utat keressük. A játék végső célja, a logikus gondolkodás fejlesztése, nem pedig a versengés.

Az [Infinifactory](#) egy másik kirakós játék, amely kisebb feladatokon alapuló rejtvény készletből áll. A játékos egy idegen faj által elrabolt ember szerepét veszi fel, akit munckába állítanak, hogy felszereléseket építsen. minden rejtvényen belül a játékos feladata, hogy egy vagy több kockából álló tárgyat összeállítson és a kiszállítási pontra juttasson egy összeszerelősor segítségével, amit egyedül kell megépítsen. A pályákon találhatóak előre beállított tárgyak is, amelyek elősegíthetik vagy akadályozhatják a futószalag felépítését. (lásd 2.2 ábra) A feladat komplexitása nem csak a pályán található korlátoktól függ, hanem az előállítandó tárgyaktól is. A problémák ellenőrzésére a gyár bármikor elindítható.

Az egyetlen korlát, hogy hányféle képen lehet megoldani a feladatokat, a pálya nem-végtelen mérete. A játékos megoldását három szempont szerint lehet mérni: lábnyom (a futószalag által bezárt teljes alapterület), ciklusok (az eltelt idő amíg a követelmények teljesültek), és blokkok (a felhasznált blokkok száma). Ez a fajta pontszámrendszer arra ösztönzi a játékost, hogy javítsa és optimalizálja a befejezettként megoldásait, annak érdekében, hogy jobb, és okosabb megoldást érjen el.



2.2. ábra. Infinifactory, 6-3. pálya: Terrestrial Drone

Ebben a játékban játék mechanikának számít az interakció, az ahogyan tudunk építeni, illetve az összeszerelő kockák, amelyek együttes használatával tudjuk légyártani a feladat követelményeit.

Egy olyan játékban, mint a [Cities skylines](#), egy "sandbox" város építő játék, ahol az elvárt cél egy város építése és fenntartása, nincs konkrét végállapota. Vannak kívánatos állapotok, amelyeket a játékosok megpróbálhatnak elérni, de nem korlátozza semmi se, hogy ezt hogyan érjék el, követelmény sincs, ami elvárja, hogy ezeket elérjék. Az ilyen szimulációkban, a mechanikák csak lehetőségeket nyújtanak saját kitalált célok eléréséhez. Ezek lehetnek egy modern nagyváros építése, vagy akár rombolása is természetes vagy mesterséges katasztrófák által.

A játékmechanikákat tehát a játékrendszer elemeinek meghatározására használjuk, hogy leírjuk, hogyan lépnek kapcsolatba a játékosok a játék világával, szabályaival, állapotaival, stratégiáival és céljaival. Azokat a funkciókat értjük, amiket a játékos bizonyos interakciói által végzünk el. Ez egy pragmatikus meghatározása a mechanikáknak.

Miguel Sicart a "Defining Game Mechanics" című cikkében a játékmechanikákat próbálja megfogalmazni más fogalmazásokkal, illetve saját gondolataival. Ö egy formálisabb módon próbálja leírni, hogy mik a játék mechanikák.

Eleinte a játékmechanikákat egymondatban úgy írja le, hogy "Az ügynökök által hivatkozott módszerek", amit az objektum orientált programozás paradigmából kölcsönöz. A játék elemeket objektumoknak állítja be, a mechanikákat pedig ezek eljárásainak. Egyes mechanikákat kontextus függőnek nevez, például a "Gears of War" akciójátékban, ahol a fedezékhez szükség van a közelben egy olyan objektum, ami mögé el lehet bújni. Ebben az értelemben a mechanikákat korlátozhatja a játékvilág.

Az objektum orientált terminológiának köszönhetően, a mechanikákat leképezhetjük a beviteli eszközök bemenetéhez is, ezáltal az interakciókat absztrakt modellező nyelv használatával is leírhatjuk.

Kutatása során azt is elmondja, hogy "A játékmechanikát bármely ügynök hívhatja, legyen az ember vagy a számítógépes rendszer része. Például az AI-ügynökök számos

módszerrel is kapcsolatba léphetnek a játékvilággal. Esetenként ezek a módszerek eltérnek az emberi játékos számára elérhető módszerektől, aminek elemzésre méltó következményei lehetnek". Fontos, hogy a mechanikák használhatóak legyenek a gép által is, ezáltal univerzálisak legyenek.

Végezetül ebben a meghatározásban elmondja a szabályok és a mechanikák közötti ontológiai különbséget is. A játékmechanikák a játék állapotával való tényleges interakciójával foglalkozik, míg a szabályok lehetőséget adnak, ahol ez az interakció lehetséges. [Sic08]

2.2. Tervezés

A játékmechanikák készítésé fontos, hogy még a játékmenet kialakítása előtt történjen. Az egyetlen módja annak, hogy lássuk ezek hogyan működnek az, hogy kipróbaljuk őket. Ezért ajánlott először prototípusokat készíteni, hogy ezeket fejlesztés közben alaposan tudjuk megtervezni és felépíteni. Mivel a játék egy komplex rendszer fontos, hogy az alapvető mechanikák kiegyensúlyozottak és jól meghatározott összhangban legyenek egymással. Az egyensúlyt könnyedén tudjuk felbontani új mechanikák bevezetésével, vagy meglevő módosításával. Amint készen vannak a mechanikák, elkezdődhet a játékmenet tervezése.

Ernest Adams és Joris Dormans a "Game mechanics" című könyükben elmondják, hogy a prototípuskészítésnek egy kritikus szempontja a megfelelő fókusz megtalálása. A prototípusok tapasztalatot kell gyártsanak nekünk azért, hogy megtanuljuk melyik ötlet jó, és melyiket kell majd újra gondolni. Ha a játéknak csak egyetlen aspektusra összpontosítunk, hatékonyabban tudjuk majd a játéknak ezt a részét kifejleszteni az elvárásoknak megfelelően. "A prototípus fókusza befolyásolja a prototípus technika megválasztását" [AD12]

Először egy technikai demó előállításán kell dolgoznunk. Meg kell keresnünk a legnehezebb és legújszerűbb aspektusát a játéknak, amit megpróbálunk elkészíteni, hogy bebizonyítsuk, hogy képesek vagyunk kezelní a szükséges technológiát és elkészíteni a mechanika prototípusát. Ellenkező esetben megtörténhet, hogy túl becsültük azt amire képesek vagyunk, vagy azt amire a kiválasztott technológia képes, és ajánlott újra terveznünk saját képességeink keretein belül és máskepp közelítsük meg azt, amit akarunk elkészíteni, vagy ha ez egy meghatározott alapvető mechanika, más technológiát válasszunk a prototípus fejlesztéshez.

2.3. Játékmenet

A játékmenet a játék elemek közös működését jelenti, ez egy sajátos mód, egy minta, ahogyan a játékosok interakcióba lépnek egy játékkal. Magába foglalja a játék szabályokat, a játékos és a játék közti kapcsolatot, a cselekményt, a kihívásokat és azt ahogyan ezeket leküzdjük. "... a játékmenet azok az összetevők, amelyek egy kifizetődő, magával ragadó kihívásokkal teli élményt alkotnak, amely arra készíteti a játékosat, hogy visszatérjen még többért ... [A játékmenet] nem egy nagyszerű vizuális karakterből származik, nem a legmodernebb technológiából és a gyönyörűen megjelenített művészettelből." [Ox104]

"Alapvető játékmenet" alatt bizonyos alapvető játékmechanikáakra utalunk, amelyek meghatározzák magának a játéknak az általános jellemzőit. Az előzőleg megemlített Infinitifactory játékmenete az építészet és tervezés általi rejtvények megoldása Egy lövöldözős vagy verekedős videojáték alapvető játékmenete az, hogy az ellenfél úgy üssük, hogy ők ne üssenek vissza. A póker alapvető játékmenete az, hogy lapkombinációkat összeállítunk a közös és a saját kezünkben levő lapokkal, hogy megnyerjük a tétet, amit pénzzel vagy zsetonnal fogadtunk. A golf alapvető játékmenete, hogy a labdát a kijelölt helyre üssük. Ezeknek a játékoknak eltér a céljuk a játék menettelől. Az alapvető játékmenet azt határozza meg, hogy a játékos számára mi a játék, míg a játékmechanika a játék részeit határozza meg.

A játékmechanikákat programozási vagy általános tervezési szempontból azonban fel lehet fedni az alapvető játékmenet dekonstruálásával. Például a Wube Software által készített [Factorio](#) nevű játékban az alapvető játékmenet az automatizált gyárak felépítése azzal a céllal, hogy egyre összetettebb tárgyakat állítsanak elő, egy idegen bogarakkal teli polygon. Végső cél egy rakéta kilövése. Itt a játékmenet lebontható építészetre, készítésre, pusztításra, lövöldözésre. Emiatt a játékmechanika egy alacsony szintű részletnek írható le, ami egy mérnöki koncepció, míg a játékmenet ezek összesége, ami egy tervezési koncepció.

A játékmenet tervezése fontos szerepet játszik annak érdekében, hogy egy olyan környezetet alakítsunk ki, amiben a játék mechanikákat azokra a célokra használhatjuk, amikre ezeket tervezéskor szántuk.

A játszhatóság a játékmenet azon merő száma, amivel a játék minőséget és játszható időtartamát mérjük.

A játszhatóságot különböző attribútumok és tulajdonságok jellemzik a játék élmény mérésére. Ide tartozik az elégedettség, tanulás, hatékonyság, elmélyülés, motiváció, érzellem és szocializáció. "A játszhatóság értékelési módszerei a játékokat a tervezés javítása érdekében célozzák meg, míg a játékosélmény-értékelési módszerek a játékosokat a játék fejlesztése érdekében célozzák meg." [Gro09]

3. fejezet

Hasonló alkalmazások

Ebben a fejezetben azokat az alkalmazásokat szeretném bemutatni, amik a projekt legnagyobb inspirációinak számítanak. Ezek már létező programok melyek olyan elemeket tartalmaznak, amik segítettek nem csak a tervezés, hanem a fejlesztés során is, hogy egy más szemszögből is lássuk azt, ami hasonlít ahhoz, amit elszeretnénk érni. Sok más alkalmazásból is használtunk fel ötleteket, de ezek azok amiket ugy éreztem hogy érdemes röviden bemutatni.

3.1. SHENZHEN I/O

SHENZHEN I/O egy Zachtronics nevű indie cég által fejlesztett programozó és kirakós videójáték, amelyben a játékosok egy elektronikai mérnök szerepét töltik be, aki a kínai Shenzhen városba emigrált, hogy a Shenzhen Longteng Electronics technológiái vállalatánál dolgozzon. Feladata termékek fejlesztése áramkörök felépítésével és ezek futtatásához szükséges kód megírásával.

A játék Assembly-ben tanít párhuzamosan futó modulok programozására, illetve ezek összecsatolására, kötésekre. A feladatok egyre komplexebb rendszerek előállítását követeli a korlátozott méretű munkaasztalon. Igy arra készítet bennünket, hogy optimizálunk, és csak arra összpontosítunk amire szükségünk van a feladat megoldására. (lásd [3.1 ábra](#))

Tervezési szempontból a játék kódolás és logika mellett az objektum orientált gondolkodásra is megtanít. Sokféle modul áll rendelkezésünkre, köztük logikai operátorok is.

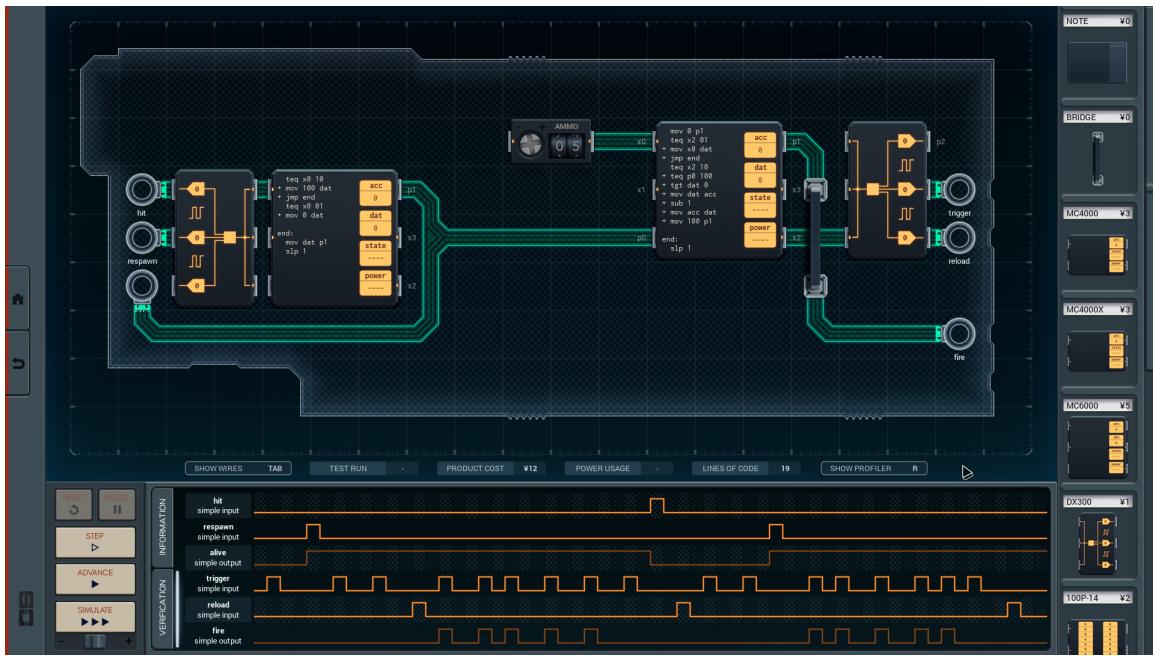
A játék dinamikus programozási képessége, és a rejtvények feladat szerű játékos bemutatása egy remek ötlet, amire mi is odafigyeltünk a tervezéskor.

3.2. Human Resource Machine

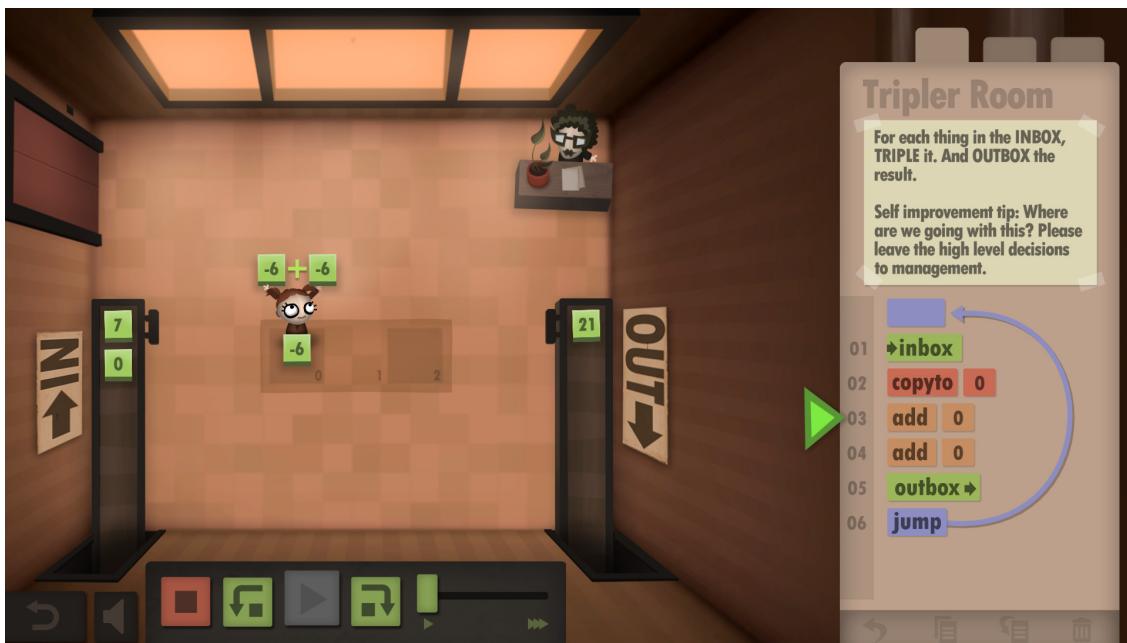
A Human Resource Machine egy másik programozó és kirakós videójáték. Itt a cél egyszerű parancsokkal utasítások listáját állítsuk össze, amit a munkások elvégeznek.

Az előző játékhöz képest itt egyszerűbb a programozás, mivel nem kell modulok kialakításán is gondolkodni, elég egy utasítás szekvencia összeállítása. (lásd [3.2 ábra](#))

Ebből a játékból merítettük a kód blokkok létezését, és ezek húzogatós használatát.



3.1. ábra. Shenzhen I/O, "Tag, you're it" pálya megoldva



3.2. ábra. Human Resource Machine, "Tripler room" pálya

3.3. Super Paper Mario

A Super Paper Mario már nem egy programozó és kirakós videojáték, viszont sokat segített a játék 3D, felfedező részének a megtervezésén. Egy egyszerű játék, ahol végig ugrálunk egy előre elkészített pályán, megoldjuk a rejtvényeket, legyőzzük ellenfeleinket, és bevégezzük a pályát.

Viszont ennek a játéknak van egy kulcsfontosságú jellemzője, ami felkeltette az érdeklődésemet és olyan inspirációt adott, amelyek segítségével meg tudtam tervezni a játékunk egyik legjelentősebb funkcióját: A felfedező 3D és a programozó 2D játékmód közti átmenetet.

A Super Paper Mario világában minden karakter egy papír. A játékos által irányított karakter, Mario, különleges képességet kap, amivel a dimenziók között tud ugrani, ezáltal a pályákon olyan részeket és rejtelyeket tud felfedni, amik másképp elérhetetlenek egy 2D figura számára. (lásd 3.3 ábra)



3.3. ábra. Super Paper Mario, 2D és 3D játékmód

4. fejezet

Programok, technológiák bemutatása

Dolgozatom ezen részének a célja, hogy átfogó képet adjon azokról a programokról és technológiákról, amelyeket a Unity-ben készült videójáték fejlesztése során használtam. Ezek az eszközök és platformok fontos szerepet játszódtak a játékmechanikák létrehozásában, és a játék különböző részeinek a fejlesztésében.

4.1. Unity játékfejlesztő motor áttekintése

A [Unity](#) egy többplatformos játékmotor, amelyet először 2005 júniusában jelentettek be, és adtak ki az Apple Worldwide Developers Conference rendezvényen. A motort azóta fokozatosan kibővítették, hogy a különféle asztali, mobil, konzol és virtuális valóság platformokat is támogassa. Használata elterjedt az Android és iOS mobiljátékok fejlesztésénél, mivel könnyen használható kezdő fejlesztők számára egyszerű játékok elkészítéséhez. Ugyanakkor népszerű is az indie játékok fejlesztésében, ahol sok elkepesztően nagy és jó minőségű játékok készültek.

A motor lehetővé teszi a háromdimenziós és kétdimenziós játékok elkészítését is. Szkripteléshez Mono-t használ, ami egy C# programozó nyelvben írt szoftver keretrendszer.

A kétdimenziós játékok esetében sprite-okat használ, ami egy grafikus objektum, amelyet a számítógépes grafikában használnak. A háromdimenziós játékok esetében nem csak a textúrák használata elérhető számunkra, hanem a fejlettebb grafikus funkciók, mint a mipmap-ek, a bump-mapping, tükrözés-leképezés, parallaxis-leképezés, képernyő térelzárás, dinamikus árnyékok árnyéktérképekkel és egyéb utófeldolgozási effektusok is. Emellett elérhető két külön renderelési folyamat is, a High Definition Render Pipeline (HDRP) és az Universal Render Pipeline (URP). A HDRP renderelési folyamat több grafikus funkciót tartalmaz, amit csak modern számítógépes hardver támogat.

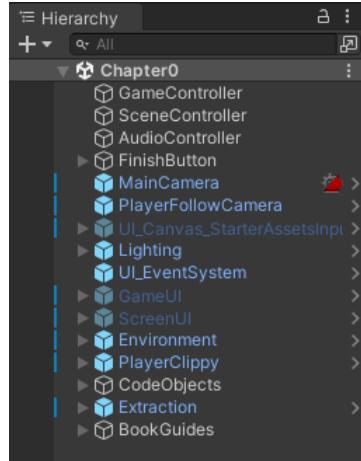
4.2. Unity szerkesztő

A Unity játék fejlesztő motornak egy nagyszerű vizuális szerkesztő felülete van sokféle eszközökkel, amelyek lehetővé teszik a játékterek, objektumok és interakciók könnyed létrehozását.

A Unity szerkesztő fontosabb elemei a következőek:

- Hierarchia ablak

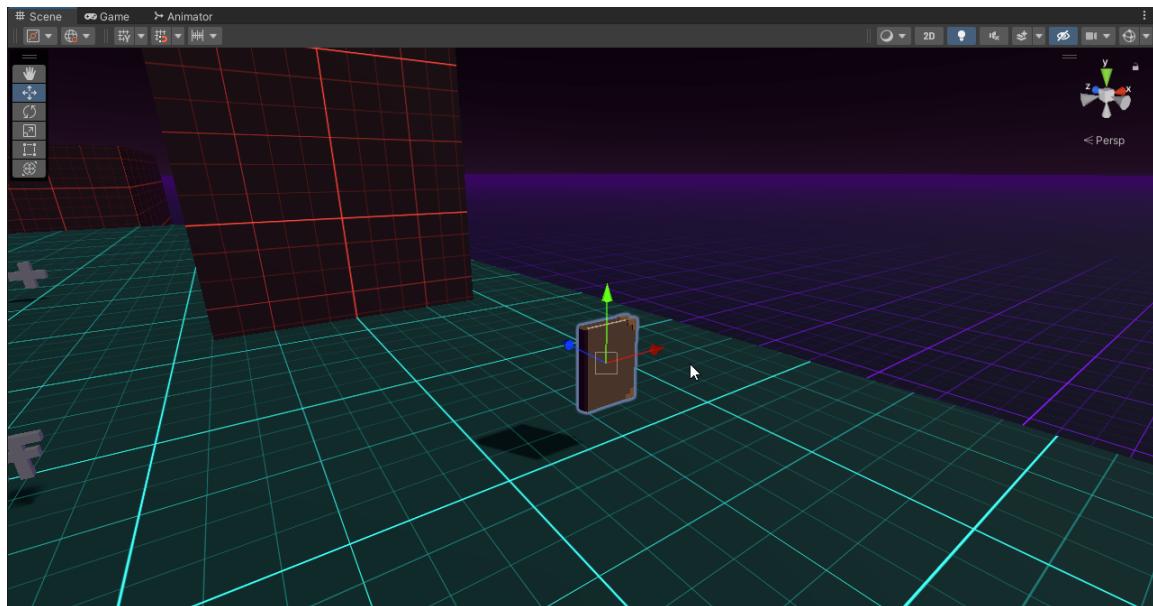
Ez az ablak egy fastruktúrát jelenít meg a játékban szereplő objektumok hierarchiájáról. Itt látható és kezelhető az összes objektum, beleértve a játéktér, a modellek, a fények, hangok, és egyéb elemek. Az objektumokat könnyedén átrendezhetjük, csoportosíthatjuk és elnevezhetjük. (lásd 4.1 ábra)



4.1. ábra. Unity Hierarchy ablak

- Színpad ablak

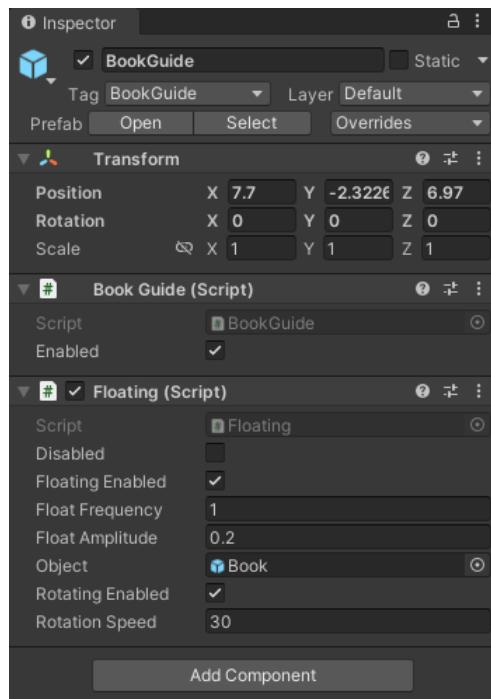
Ez az ablak egy nézetet mutat a játéktérünkről. A térben navigálni és közelíteni is lehet, így könnyen felfedezhetjük a játékteret. Ugyanitt manipulálni is lehet a játékban levő objektumok helyzetét, forgatását és skáláját gizmo-k segítségével. (lásd 4.2 ábra)



4.2. ábra. Unity Színpad ablak

- Inspektor ablak

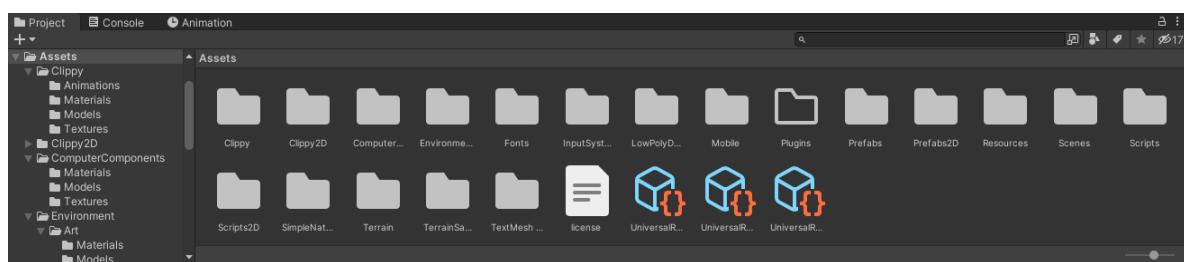
Az Inspektor ablak megjeleníti a kiválasztott objektum tulajdonságait és komponenseit. Az objektum tulajdonságai közé tartozik a pozíció, a forgatás, a skála és az anyag, ami szerint vannak renderelve. A komponensek viszont további specifikus viselkedést, tulajdonságokat és funkciókat adnak hozzá az objektumhoz. Ezek köze tartoznak a fizikai tulajdonságok, az ütközés, az események kezelése és az animációvezérlés. Ugyanakkor a komponensek lehetnek általunk megírt szkriptek is amikkel pontosabb és dinamikusabb tulajdonságokat és funkciókat adhatunk a játék objektumoknak. (lásd 4.3 ábra)



4.3. ábra. Unity Inspektor ablak

- Projekt ablak

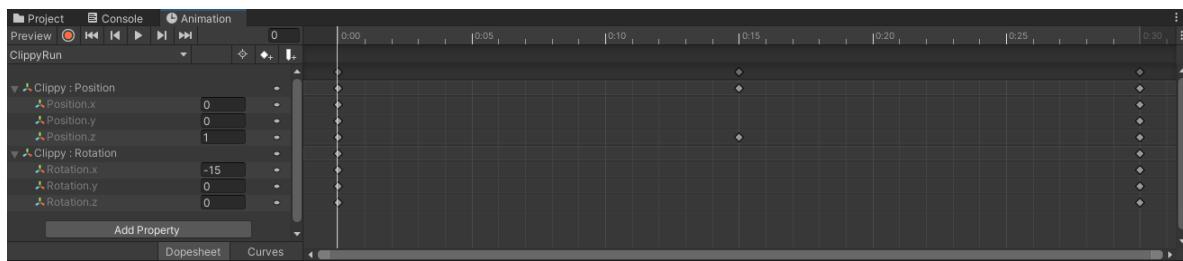
A Projekt ablakban található és kezelhető a projekt összes fájljai és erőforrásai. Ezek a modellek, textúrák, hangok, animációk, scriptek, terep fájlok, anyagok, és egyéb elemek, amelyeket importáltunk vagy létrehoztunk. Az ablak egy teljes fájlkezelőként működik, itt navigálhatunk a mappákban, kereshetünk, rendezhetünk, szűrhetünk, létrehozhatunk, importálhatunk és törölhetünk is. (lásd 4.4 ábra)



4.4. ábra. Unity Projekt ablak

- Animáció ablak

Az Animáció ablak lehetővé teszi az animációk létrehozását és szerkesztését. Itt láthatjuk az idő vonalat, a kulcspozíciókat és az animáció görbüйт is. Ugyanitt beállíthatjuk a sebességet, az átmeneteket, a hurkokat és egyéb paramétereket is. Az ablakban animációkra rögzíthetjük az objektum tulajdonságait és komponenseit, például a helyzetet vagy a forgatást, és manipulálhatjuk azokat az idő vonalon keresztül. Ezt a játékos által irányított Clippy 3D karakternek az animáció készletének tervezésére használtam. (lásd 4.5 ábra)



4.5. ábra. Unity Animáció ablak

- Szkriptelés

A Szkriptelést külső kód szerkesztőben végezzük, ehhez a Unity projektfájlokat generál az intellisense működéséhez. Ez lehet például a Visual Studio Code, vagy akár a JetBrains Rider is. Miután elmentettük a fájlokat és visszatérünk a Unity szerkesztőhöz, ez automatikusan újra fordítja a szkript fájlokat, hogy ezeket már használni tudjuk.

4.3. Unity eszköztár

A [Unity eszköztár](#) egy online piactér, ahol széles választékban találhatók különböző más felhasználók által készített erőforrások, amelyek segíthetnek a Unity-ban készült projekt fejlesztésében. Ez a piactér azzal a céllal készült, hogy a felhasználók megosztani tudják díjmentesen vagy fizetősen az általuk készült erőforrásokat, amiket a fejlesztők megvásárolhatnak és felhasználhatnak saját projektükben, ezáltal felgyorsítván a munkafolyamatot.

Az eszköztárnak egyszerű felülete van kereséssel és szűréssel, és közvetlenül integrálódik a Unity szerkesztővel is, így a kívánt erőforrásokat a projektbe importálhatjuk.

4.4. Wonderdraft

A [Wonderdraft](#) egy fantáziatérkép készítő eszköz, amely képes különböző típusú valósághű világokat létrehozni csupán néhány kattintással.

Ennek egy ingyenes [Ralia](#) nevű színes témáját használtam a térkép megrajzolásához.

5. fejezet

Szoftver

Ebben a fejezetben a szoftver általam megvalositott részeiről fogok részletesebben írni, mégpedig a játék háromdimenziós oldalának a játék mechanikáiról, a kétdimenziós játékmódba való átmenetről, illetve az ezek segítségével kialakult játékmenetről is.

5.1. Funkcionális követelmények

- **A játék indítása:** A játék elindításához a felhasználó a "Play Game" gombra kell kattintson, hogy a fejezetek térképre jusson. Itt az elérhető fejezetek közül egyiket kiválasztva, rákattint az ennek megfelelő gombra.
- **Karakter irányítása:** Miután egy pálya betöltött, a karakterét a játékos a **WASD** billentyű gombokat használva tudja irányítani, az e körül forgó kamerát pedig az egér mozgatásával. A karakterrel szaladni is lehet a **SHIFT** billentyű gomb lenyomva tartásával. Az akadályok kikerüléséhez pedig ugrani is tudunk a **SPACE** billentyű gombbal.
- **Kód objektummal való interakció:** A **TAB** billentyű gomb lenyomásával a kurzor felszabadul, így a felhasználó interakcióba léphet a kód objektumok fölött felugró eszköztipp felülettel, ahol a "Magyarázat lejátszása" (hang jel) vagy a "Példa kimutatása" (programkód jel) gombokra kattinthat.
- **Kód objektumok összegyűjtése:** A karakterrel a földön található kód objektumokat érintéssel felvehetjük. Ezek a játékos körül fognak majd lebegni.
- **Kód objektumok kivonása:** A játékos által összegyűjtött kód objektumokat egy kivonónál tudjuk leadni. A kivonó fölött lebegő billentyű gomb megnyomásával tudunk majd ezzel interakcióba lépni.
- **Játékmód váltás:** A játékmódok között a **CTRL+TAB** billentyű gomb kombinációval tudunk váltani.
- **Játék szüneteltetése/folytatása:** Szüneteltetni a játékot az **ESC** billentyű gomb megnyomásával tehessük meg. Ez egy kisebb menüt fog megjeleníteni, ahonnan folytathatjuk a játékot, vagy visszaléphetünk a start menühöz. A billentyű gomb ismételt megnyomásával is folytathatjuk a játékot.

5.2. Nem funkcionális követelmények

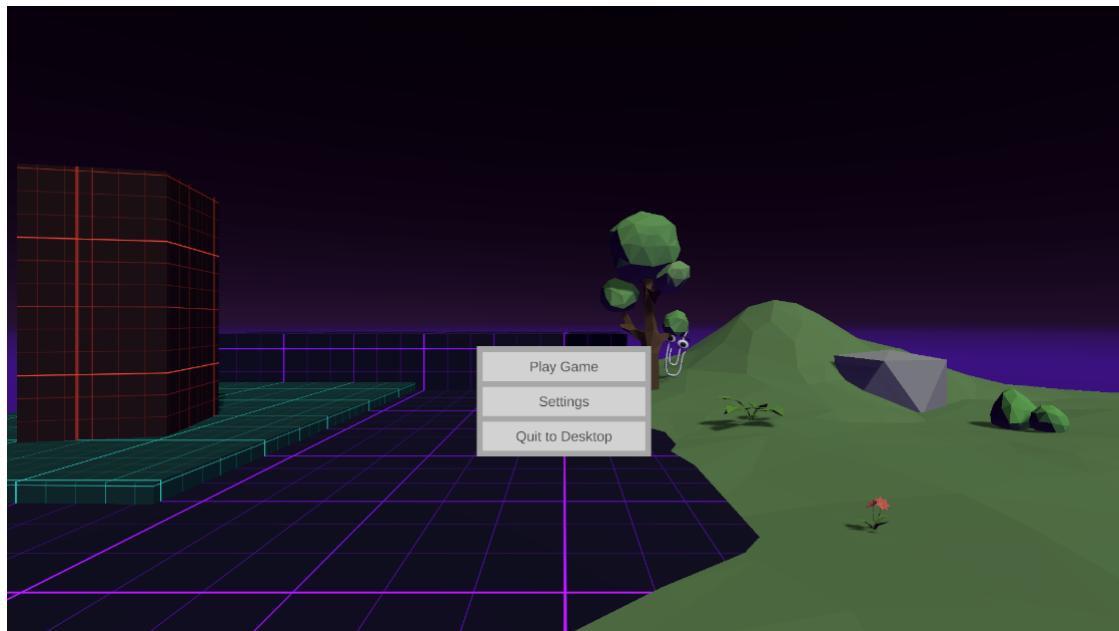
A játék helyes működéséhez a használandó számítógépnek a következő feltételeket kell teljesítenie:

- Operációs rendszer: Windows 10/11
- Processzor: 2.0GHz, 64-bit
- Grafika: 1920 x 1080
- Grafikus API: DirectX11
- Memória: 256MB
- Tárhely: 300MB
- További követelmények: Egér, billentyűzet

5.3. A szoftver bemutatása

5.3.1. Játékelemek és rendszerek

A játék egyszerű menüket és interfészket tartalmaz. Indításkor, a legelső kép, amit látunk az egy statikus háromdimenziós környezet, ahol a start menüt látjuk előtérben (lásd 5.1 ábra). Itt elindíthatjuk a játékot, módosíthatjuk a beállításokat vagy kiléphetünk a játékból.



5.1. ábra. Start menü

A játék elindításakor egy térképet láthatunk, ahol a különböző fejezeteket érhetjük el. (lásd 5.2 ábra)

Ehhez egy singleton osztályt írtam, a LevelController-t ami észbe tartsa a feloldott fejezeteket, illetve képes betölteni őket a SceneManager játék objektum LoadScene metódusával.

A még nem megnyitott fejezetek szürkével jelennek meg. Ezt a start menü frissíti a térkép betöltését követően, a LevelController segítségével.

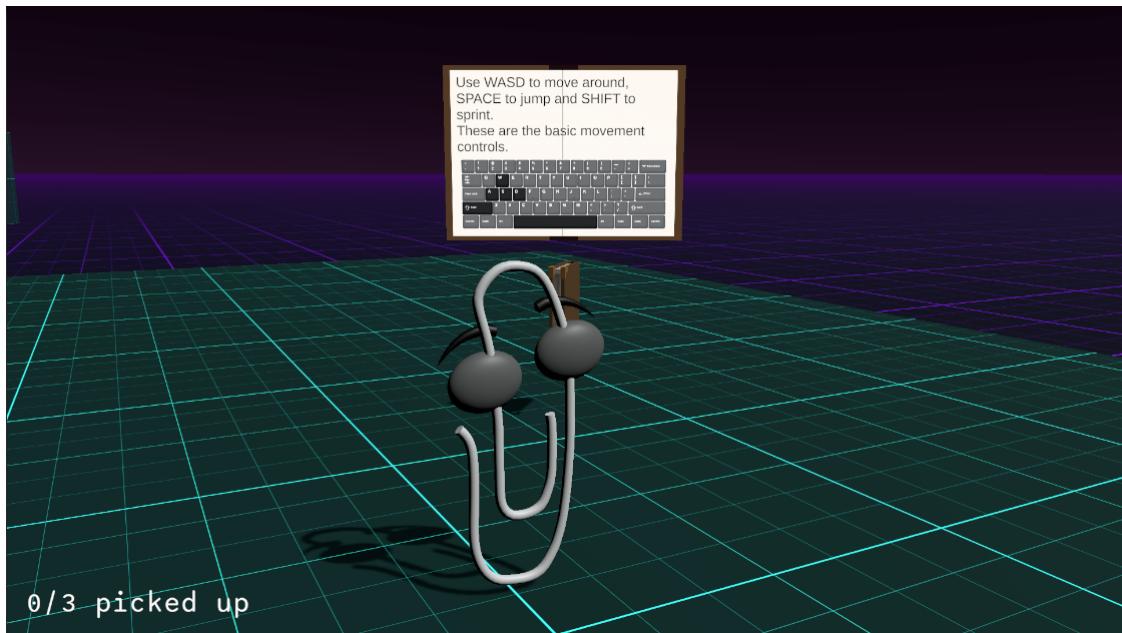


5.2. ábra. Fejezetek térkép

A játékon belül kétféle felhasználói felületet ismerhetünk meg. Az egyik a játszás-hoz szükséges részleteket jeleníti meg, illetve a menüt, amivel visszatérhetünk a start menühöz, a másik pedig a 2D kirakósnak készít egy vászont.

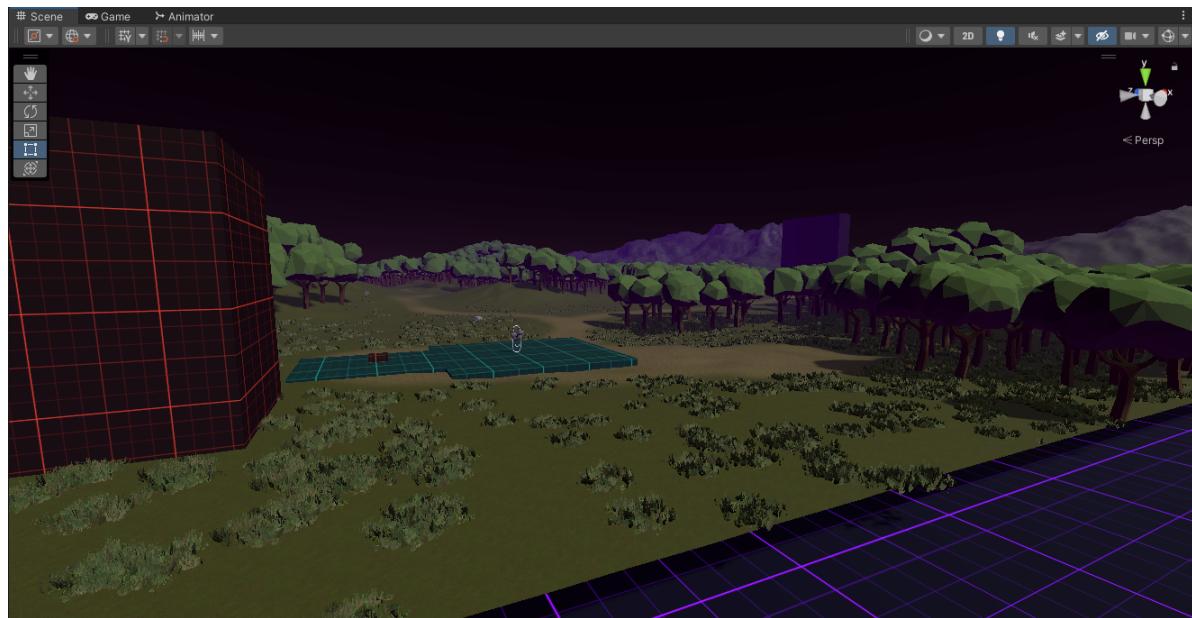
Minden pálya egy retrohullám stílusú világban játszódik le, ezek pedig kézileg voltak összeállítva.

Az első pálya egy bevezető a játék mechanikáiba (5.3 ábra). Itt lebegő útikönyvek magyarázzák el nekünk a játék elemeit.



5.3. ábra. Bevezető pálya, lebegő útmutató könyv

A további pályák cselekményei egy kézileg formált "Terrain" területén játszódik le, amit a beépített terepesközökkel készítettem. Ezek kinézetének a tervezése egy memória darabot ábrázol, azt az érzést keltve mintha a felhasználó egy memória szeleten játszana. Ezt a jelenséget kihasználva biztosítsuk azt, hogy folytonosan előre haladjunk a játék által nyújtott virtuális világban. (lásd 5.4 ábra)



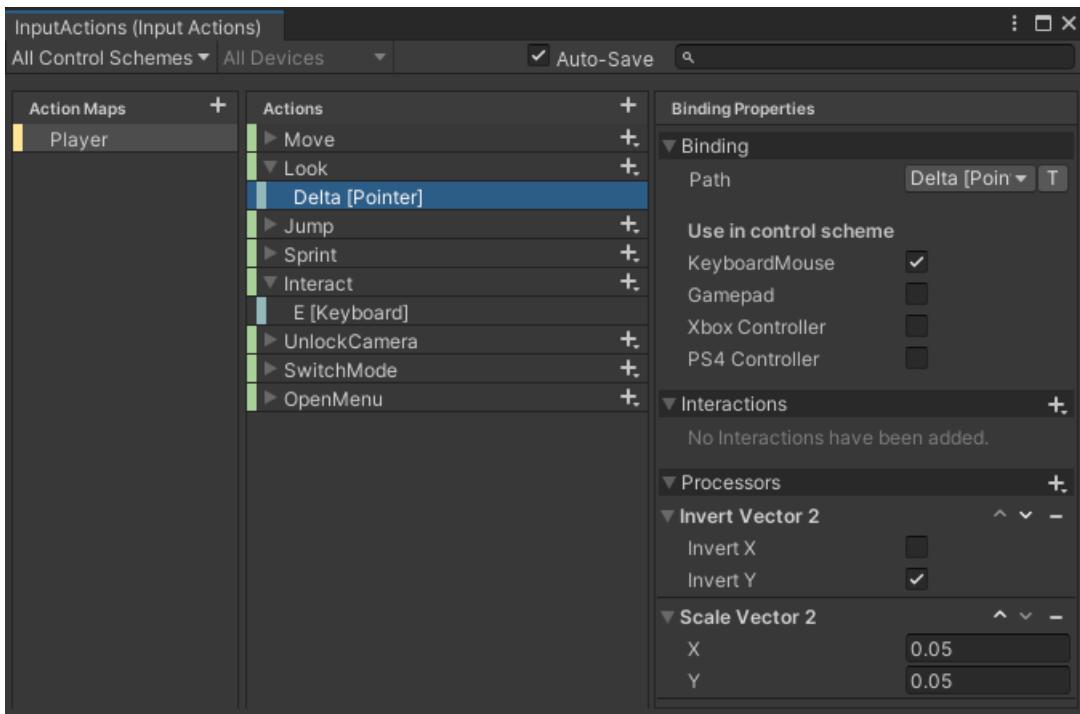
5.4. ábra. Első fejezet pálya

5.3.2. Játékményet, vezérlés és animáció

A Unity Input System egy kiterjedt rendszer a bemenet kezeléséhez. Az Input System a beviteli műveleteket úgynevezett Input Actions-ként kezeli, ami meghatározza a beviteli eszközök konfigurálását és kezelését. A beviteli események bekövetkezésekor az Input System automatikusan értesíti a megfelelő komponenseket vagy metódusokat, hogy ezek reagáljanak az eseményre.

A Unity Technologies által készített harmadik személyű karakter vezérlő kezdő csomagra építve valósítottam meg a kamera vezérlést, a karakterrel való mozgást és ugrást, a játék világgal való interakciókat és a játékmódok közti perspektívaváltást.

Egy Input Action Asset-ban minden Action-t külön konfigurálhatunk egy beviteli eszköz bemenethez vagy ennek egy módosított állapotához, amit például a CTRL gombbal való közös lenyomásával érünk el. (lásd 5.5 ábra)



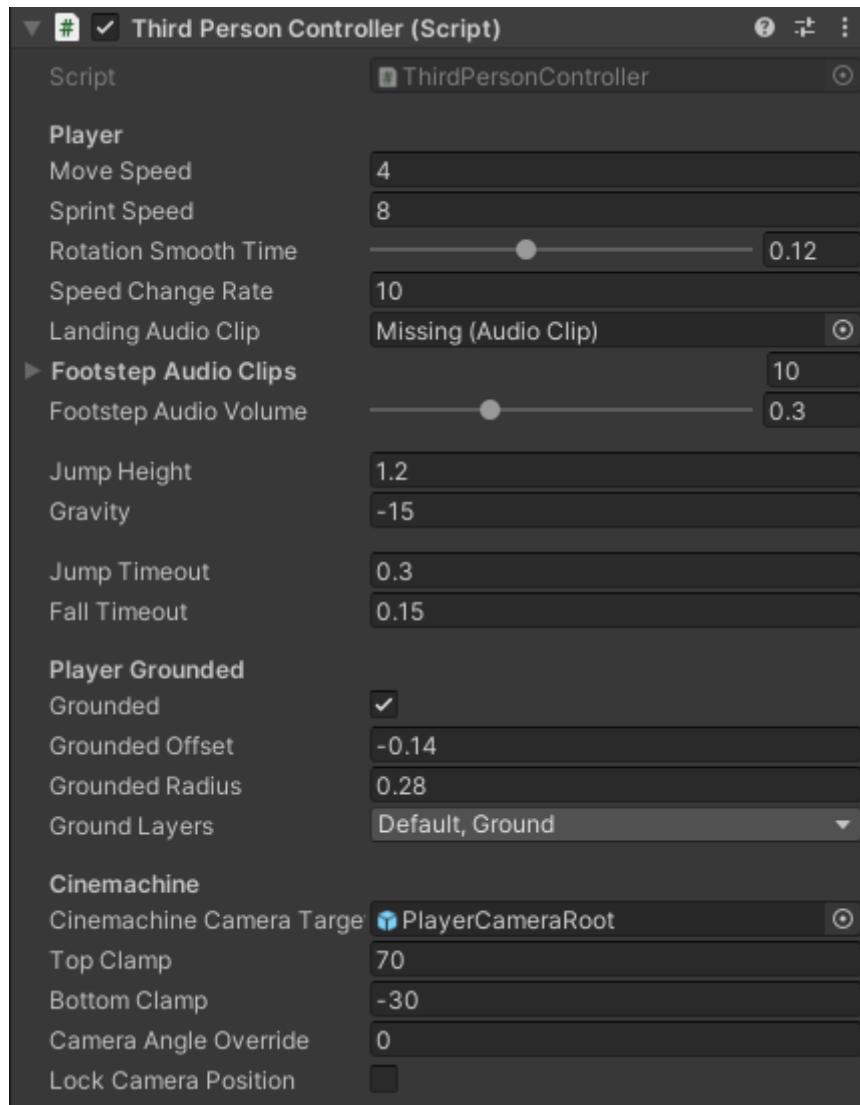
5.5. ábra. Unity Input Actions ablak

A szoftver jelenleg csak egeret és billentyűzetet támogat, mint beviteli eszköz és az 5.1 táblázatban ábrázolt beviteli térképet implementálja.

Bevitel	Leírás
Karakter mozgatás	WASD
Kamera mozgatás	Egér
Ugrás	Space
Sprint	Bal Shift
Kamera feloldása	Tab
Játékmod váltás	Ctrl+Tab
Menü	Esc

5.1. táblázat. Beviteli térkép

A felhasználónak a játékkal való interakcióját az InputController szkript végzi esemény függvények segítségével. A karakter mozgást pedig a ThirdPersonController végzi. Ennek számos beállításai vannak, amikkel a karakter járását, szaladását, forgását, ugrását és kamerájának mozgását lehet konfigurálni. (lásd 5.6 ábra)



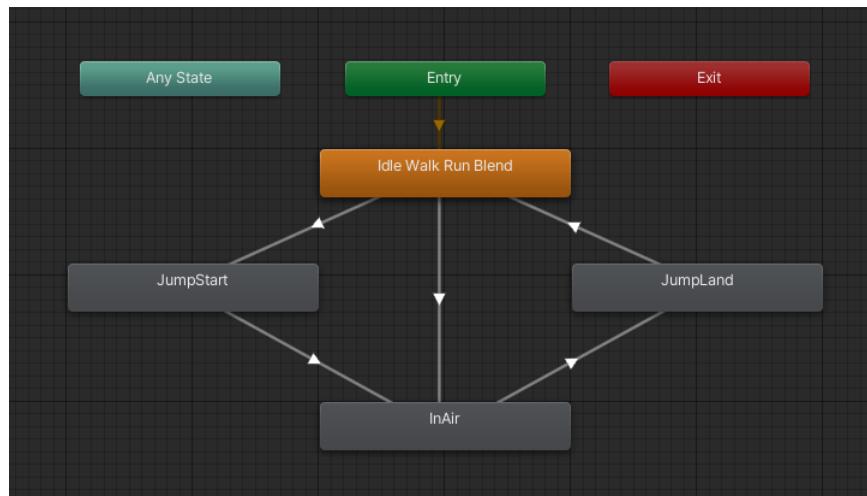
5.6. ábra. Third Person Controller komponens

Emellett a szkript elvégez egyéb számításokat is, amik paraméterként szolgálnak az animációs vezérlőnek. (lásd 5.7 ábra)



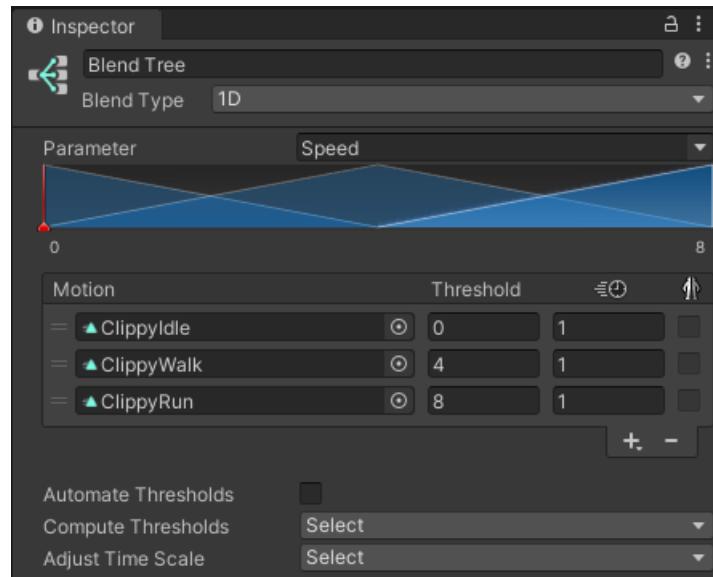
5.7. ábra. Animátor vezérlő paraméterei

Az animációk sorrendjét és ezek közti átmenetet az animátorban határozzuk meg egy animációs fa segítségével. (lásd 5.8 ábra)



5.8. ábra. Animációs fa

Itt vannak egyszerű és összetett állapotok is. Az alábbi ábrán látható az "Idle Walk Run Blend" ami a járás állapotait egybe keveri azért, hogy az animációkat sebesség szerint válasszuk ki. (lásd 5.9 ábra)



5.9. ábra. "Idle Walk Run Blend" összetett animációs állapot

A játék világgal való interakcióért a ClippyController a felelős. Ez végzi el a közelben levő játék objektumok eszköztippjeinek megjelenítését címkék segítségével.

Az `UiUtils`-ban implementált metódusokkal az eszköztippek kifestjük és adatokkal feltöljtük (lásd 5.13 ábra). Az `Utility` osztály metódusaival pedig lekérjük a közelben levő objektumokat címke szerint, és a felállított eszköztipp felületet a játékos kamerájának irányába forgassuk.

```
foreach (GameObject obj in objects) {
    var lookPos = obj.transform.position - target;
    lookPos.y = 0;

    var rotation = Quaternion.LookRotation(lookPos);
    obj.transform.rotation = rotation;
}
```

A kód objektumokat feltudjuk szedni érintéssel. Maximum 3-at hordozhatunk magunkkal, a képernyő bal alsó sarkában pedig láthatjuk, hogy hányat hordunk és hogy hányat hordozhatunk (lásd 5.10).



5.10. ábra. Felvett kód objektumok

Ugyanezen az 5.10 ábrán látható, hogy a felszedett, de még nem kivont objektumok, Clippy körül lebegnek. Ezt a következőképpen oldja meg a ClippyController:

```
// Rotate picked up items around Clippy
foreach (GameObject obj in pickedUp) {
    float actualTheta = objectToRotAngle[obj] + (orbitDegreesPerSec *
        Time.deltaTime);
    objectToRotAngle[obj] = actualTheta;

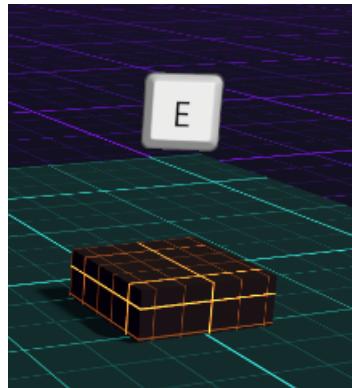
    float theta = actualTheta * Mathf.Deg2Rad;

    Vector3 center = transform.position;
    float x = Mathf.Cos(theta) * orbitDistance + center.x;
    float y = center.y + orbitHeight;
    float z = Mathf.Sin(theta) * orbitDistance + center.z;

    obj.transform.position = new Vector3(x, y, z);
}
```

Az `actualTheta` változó fokban tárolja a jelenlegi szöget, ahol az objektumnak lennie kell. Ezt átalakítjuk sugárrá, majd trigonometria képletekkel kiszámoljuk a háromdimenziós pozíciót a világban, ahova az objektumot tennünk kell.

Leadni kívánt kód objektumokat egy kivonónál adhassuk le (lásd 5.11 ábra).



5.11. ábra. Kivonó

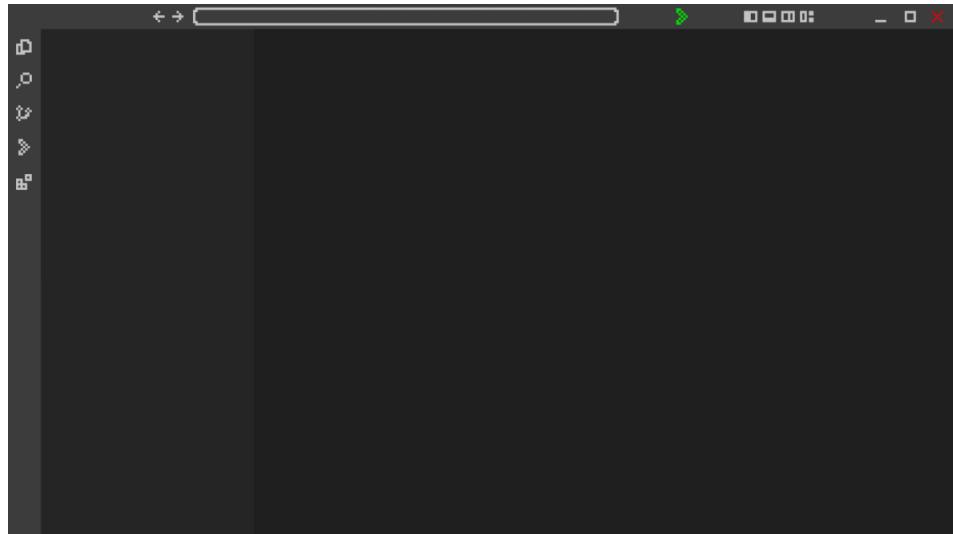
A kivonóhoz való lebegést pedig a következő aszinkron függvény végzi el:

```
public static IEnumerator moveOverSpeed(Transform transform, Vector3 target,
    float speed, float delay=0) {
    yield return new WaitForSeconds(delay);

    while (transform.position != target) {
        transform.position = Vector3.MoveTowards(transform.position, target,
            speed * Time.deltaTime);
        yield return new WaitForEndOfFrame();
    }
}
```

Játék szinten létezik egy GameController singleton osztály, ami implementálja a játékmódok közti ugrást, és képes szüneteltetni is a játékot.

A játékmódok közti váltás egy animációval történik a Windows operációs rendszerben megoldott nyitott programok közti váltáshoz hasonlóan, amit az Alt+Tab billentyű parancssal csinálunk. A 2D-s nézetmódba váltáshoz a kamera passzív rezgését megállítjuk, és a perspektívát is kicseréljük. (lásd [5.12](#) ábra)



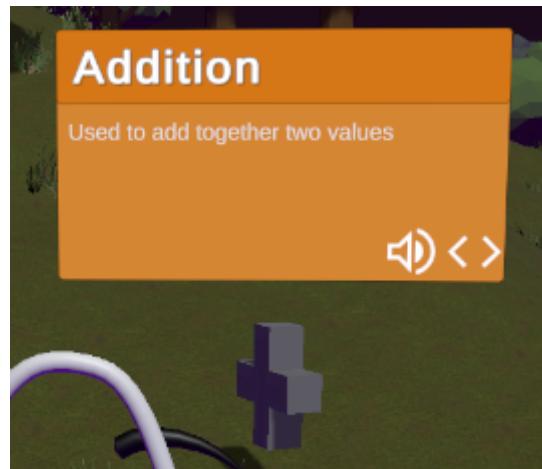
5.12. ábra. 2D nézetmód üres munkaasztallal

5.4. Ütközés és fizika

A játékon belüli ütközéseket és fizikákat a beépített komponensek segítségével oldottam meg. A Unity rendelkezik ütközésdetekcióval, ütközési alakzatokkal, Rigidbodies-el, ami felelős az objektumok fizikai viselkedéséért. Az objektumok fizikai tulajdonságokkal is rendelkeznek, ahol ennek a fizikai tulajdonságait tudjuk beállítani, mint például a súrlódás és a rugalmasság.

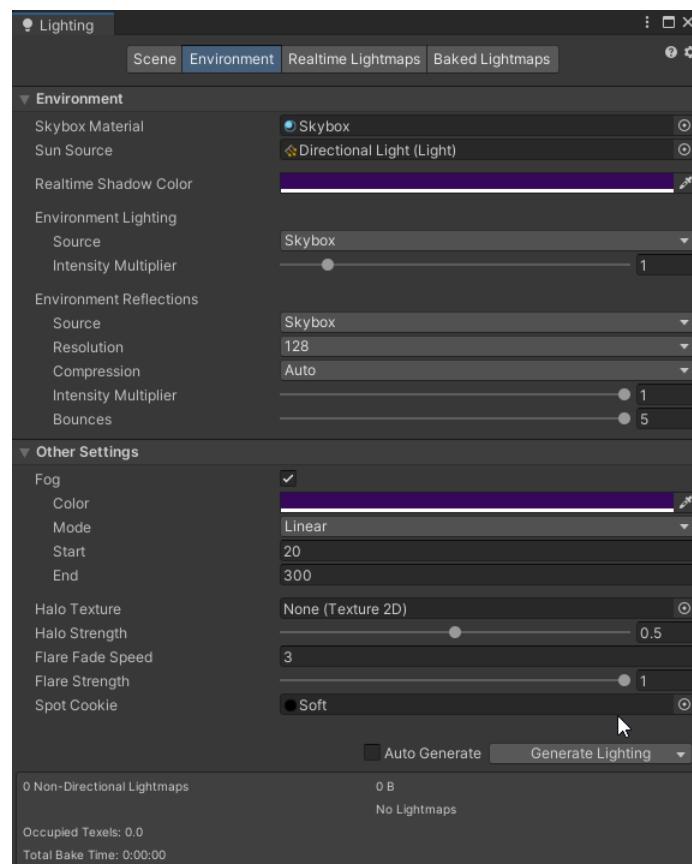
5.4.1. Hang- és vizuális effektek

A játék hanghatásokat is tartalmaz, ezek a kód objektumok eszköztippjein található gombnyomással játszhatok le (lásd [5.13](#) ábra). Szerepük a kód objektum elmagyarázására szolgálnak.



5.13. ábra. Kód objektum eszköztipp

A játékvilág megvilágítását egy irányított fényforrás, az árnyékolást pedig az URP renderelési folyamat konfigurálása állítja be. Az ég, illetve a távolban látszó köd színét és kinézetét pedig a Színpad környezeti fény beállításai állítják be. (lásd 5.14 ábra)



5.14. ábra. Unity Színpad Fény beállítás ablak

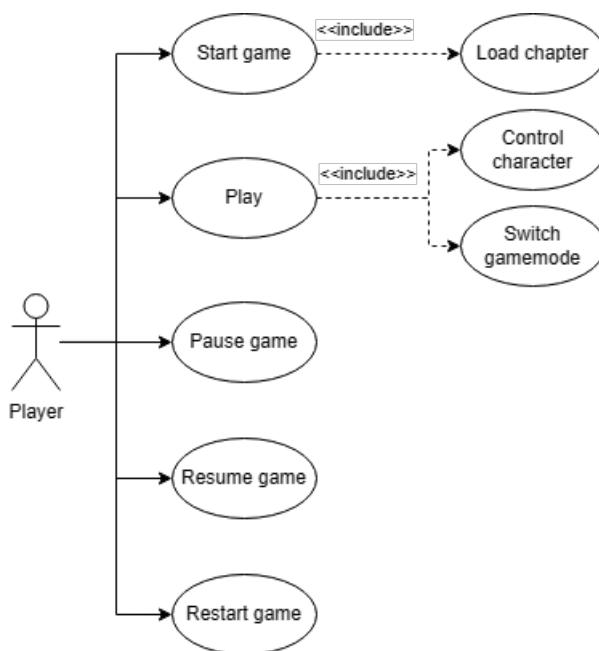
5.5. A szoftver megírásához használt könyvtárak

A fejlesztéshez a Unity játékmotorba beépített könyvtárakat használtam. Ez elég-séges volt a szoftver minden részének megvalósításához és nem volt szükség egyéb külső könyvtárakra.

5.6. Diagramok

Ebben az alfejezetben azokat a diagramokat fogom bemutatni, amik a szoftver architektúráját mutassák be.

5.6.1. Use Case diagram



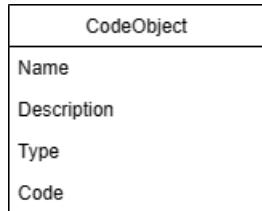
5.15. ábra. Use case diagram

- A játékos a start menüben indíthatja el a játékot. Itt egy térképen tudja majd betölteni az általa kiválasztott pályát.
- Miután kiválasztott egy pályát és ezt betöltötte a szoftver, a beviteli eszközökkel irányítani tudja a karakterét, illetve képes a 2D és 3D játékmódok között váltani.
- A játékon belüli menüvel szüneteltetni tudjuk a játékot. A szünetelt játékot pedig folytatni.
- A szünetelt játékot újra is lehet indítani, ha visszamegyünk a start menűhöz és ujra kiválaszuk ezt.
- Egy pálya akkor számít befejezetnek amikor elvégeztük a kitűzött feladatot. Ezután megnyílik a következő pálya, amit a start menüből is el fogunk tudni érni.

5.6.2. Osztálydiagram

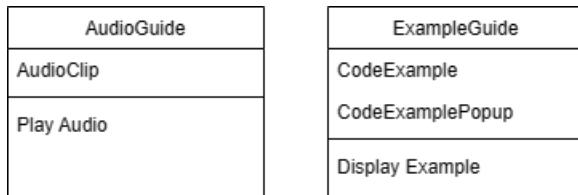
A szoftverben vannak olyan osztályok, amelyek modellként viselkednek, különleges funkciókkal ruházva fel a nekik megfelelő játék objektumokat.

Az egyik ilyen alapvető modell a kód objektum (lásd 5.16 ábra).



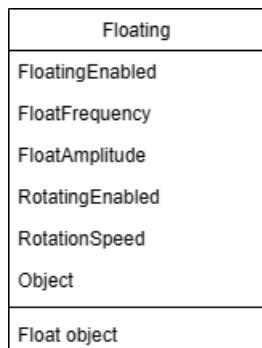
5.16. ábra. Kód objektum

A hangos útmutató és a példa útmutató konfigurálásával a kód objektumot opcionális funkciókkal egészítik ki (lásd 5.17 ábra). Hiányuk megváltoztatja a kód objektum eszköztipp felületét.



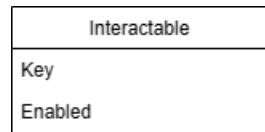
5.17. ábra. Útmutató kiegészítő modellek

Az objektumok lebegéséhez egy külön osztályt terveztem, ami a lebegéshez `sin` függvényt használ, illetve lehetőséget ad a frekvencia és az amplitúdó beállítására is. Ezt akármilyen játék objektumra lehet csatolni mint komponens.



5.18. ábra. Lebegő objektum modell

Az objektumokkal való interakcióhoz szükséges billentyű gombot az interaktív modellben tudjuk konfigurálni. Ezt a ClippyController használja, hogy a megfelelő viselkedést végezze el.

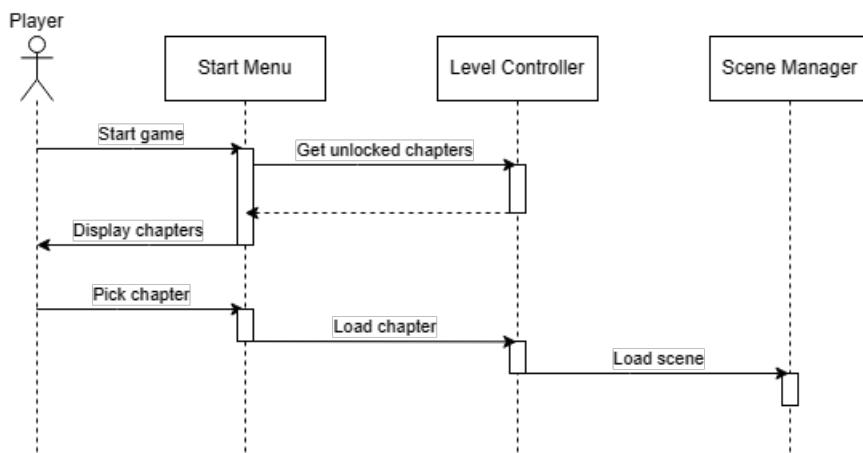


5.19. ábra. Interaktív modell

5.6.3. Szekvencia diagram

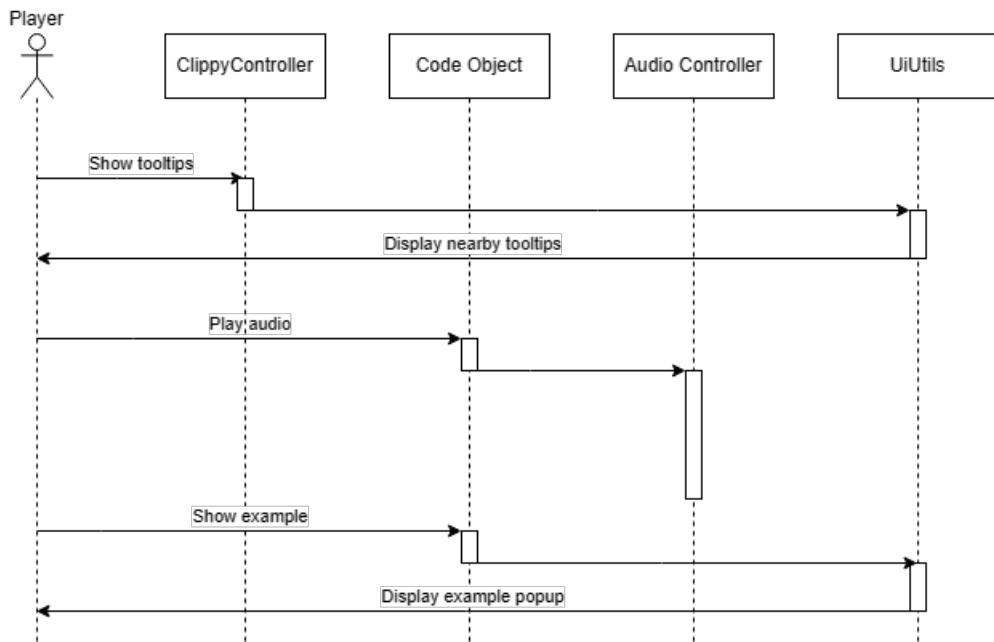
Ebben az alfejezetben a játék leghasználthatóbb 3 elemét fogom bemutatni szekvencia diagramok segítségével.

Az első ilyen elem a start menü, ahol a játékot tudjuk elindítani egy pálya kiválasztásával. (lásd 5.20)



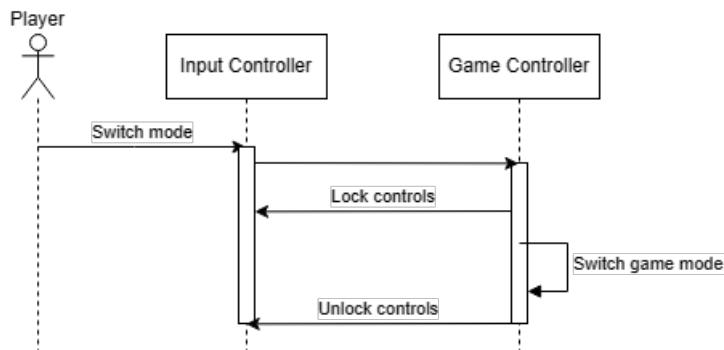
5.20. ábra. Szekvencia diagram, menü

A kód objektumokkal való interakció az, ami minket, mint tanuló, megtanít a programozás felépítő részeire szöveg, hang és példa segítségével. (lásd 5.21)



5.21. ábra. Szekvencia diagram, kód objektum

A játékmódok közti váltás a játékmenet azon eleme, amivel a játék kettő módja között tudunk váltani. Mindkettő mód tartalmaz olyan kitűzött feladatokat, amiknek a megoldása szükséges a pálya befejezéshez. A váltás hibamentes működéséhez szükségünk van, hogy a bevitelt lezárjuk, majd feloldjuk.



5.22. ábra. Szekvencia diagram, játékmód váltás

6. fejezet

Limitációk

A projekt dinamikája és a Unity játékmotor által szolgáltatott rengeteg funkciója miatt, nincsenek technológiai korlátok se a tervezés se a fejlesztés szempontjából.

Viszont a szoftver még nem alkalmas az újabb hallgató generációk oktatásában, mivel még vannak hiányos fejezetek és hiányos funkciók, amelyek szükségesek a játékmenet teljesé tételehez, illetve olyan optimizálási lehetőségek, amelyek segítenek a játékélményt simítani és tökéletesíteni.

Összefoglaló

Dolgozatom főbb témái a játék mechanikák meghatározása és tervezése voltak, valamint egy általam fejlesztett videójáték bemutatása. Az említett játék egy összetett, tanító jellegű 3D videojáték.

Kezdetben meghatároztam, hogy mik a mechanikák és tanulmányoztam, hogy más játékok hogyan valósítottak meg őket. Tárgyaltam a klasszikus sakkról, néhány kirakós és építkező játékokról is. Ezek a játékok számomra egyedi, kiemelkedő elemekből állnak, amelyek segítettek a kutatásomat a játék mechanikák és a játékmenet közti kapcsolatok meghatározásában (lásd 2. fejezet). Kutatásom során számos jól kidolgozott cikket és könyvet is idéztem.

Ezt követően bemutattam azt a három játékot, amelyek közvetlenül befolyásolták a különböző rendszerek tervezésékor hozott döntéseimet. Emellett beszéltem azokról a programokról is, amelyek segítették a célomnak kitűzött szoftver megvalósításában. Itt megemlítettem egy játék fejlesztő motort és bemutattam az ezt implementáló szerkesztőt, egy online piactért erőforrások beszerzésére, és egy fantáziatérkép készítő eszközt is.

Ezek után részletesebben megvizsgáltuk a fejlesztett szoftver játékmenetét és építő elemeit. Két pálya került említésre, egy bevezető pálya, ahol a játék alapvető funkcióit sajátítjuk el, valamint egy olyan pálya, amely alapot teremt a későbbi fejezetek elkészítéséhez. A vezérléshez egy kiterjedt bemenet kezelőt használtam, amitől a játékos a világgal való interakciója függ. A játékmenet legfontosabb elemeiről diagramok is készültek.

A továbbfejlesztési lehetőségek közé tartoznak a limitációk megoldása, például a hiányos fejezetek (2-9) és további funkciók implementálása a szoftverbe, illetve olyan játékelemek alapos tervezése, amelyek kiterjesztik a játékmenetet. Eredeti elképzélések köze tartozott a pályák zónáinak a lezárása egy kiválasztott rejtvény megoldásáig, valamint AI Vírusok, amiket egy bazooka rakéta vetővel lehet megállítani annak érdekében, hogy megelőzzük a memória szeletek megsemmisítését.

Irodalomjegyzék

- [AD12] Ernest Adams and Joris Dormans. *Game mechanics: advanced game design*. New Riders, 2012.
- [Gro09] Breaking New Ground. Innovation in games, play. *Practice and Theory-Proceedings of DiGRA*, 2009, 2009.
- [Oxl04] Kevin Oxland. *Gameplay and design*. Pearson Education, 2004.
- [Sic08] Miguel Sicart. Defining game mechanics. *Game studies*, 8(2):1–14, 2008.