

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

A játékmenet és szabályrendszer szerepe a 2D játékokban

DIPLOMADOLGOZAT

Témavezető:
Dr. Osztián Erika,
Egyetemi adjunktus
Osztián Pálma-Rozália,
Egyetemi tanársegéd

Végzős hallgató:
Részeg Alpár

2023

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



UNIVERSITATEA
SAPIENTIA

Rolul gameplayul și regulilor în jocurile 2D

LUCRARE DE DIPLOMĂ

Coordonator Științific:
Dr. Osztián Erika,
Lector universitar
Osztián Pálma-Rozália,
Asistent universitar

Absolvent:
Részeg Alpár

2023

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



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

The role of gameplay and rule system in 2D games

BACHELOR THESIS

Scientific advisor:
Dr. Osztián Erika,
Lecturer
Osztián Pálma-Rozália,
Assistant professor

Student:
Részeg Alpár

2023

LUCRARE DE DIPLOMĂ

Coordonator științific:
Lect. univ. dr. Osztián Erika

Candidat: **Részeg Alpár**
Anul absolvirii: **2023**

a) Tema lucrării de licență:

- Rolul experienței și regulilor în jocurile 2D

b) Problemele principale tratate:

- Impactul gameplay-ului și regulilor asupra experienței jucătorului în jocurile 2D
- Importanța elementelor de design și implementarea acestor reguli în jocurile 2D
- Evoluția și inovațiile în gameplay-ul și regulile jocurilor 2D

c) Desene obligatorii:

- Diagrame de proiectare pentru aplicația software realizată

d) Softuri obligatorii:

- Scopul a fost să se realizeze elemente de joc care să poată exemplifica în mod corespunzător conceptele IT utilizatorului, în cadrul unui joc educațional 2D. Inspirat în principal de funcționarea jocurilor bazate pe programare cu blocuri, jocul și-a propus să ofere un gameplay și un sistem de reguli adecvat concepute.

e) Bibliografia recomandată:

- Brenda Brathwaite and Ian Schreiber. Challenges for game designers. Course Technology/Cengage Learning Boston, Massachusetts, 2009.
- Tracy Fullerton. Game design workshop: a playcentric approach to creating innovative games. CRC press, 2014.
- Jesse Schell. The Art of Game Design: A book of lenses. CRC press, 2008.
- Miguel Sicart. Defining game mechanics. Game studies, 8(2):1–14, 2008.
- Daniel Silber. Pixel art for game developers. CRC Press, 2015.
- Tubagus Zufri, Dodi Hilman, Octavianus Frans, et al. Research on the application of pixel art in game character design. Journal of Games, Game Art, and Gamification, 7(1):27–31, 2022. 24

f) Termene obligatorii de consultații: săptămânal, preponderent online

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 417

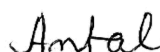
Primit tema la data de: 2022.05.07.

Termen de predare: 2023.06.21.

Semnătura Director Departament



Semnătura responsabilului
programului de studiu



Semnătura coordonatorului



Semnătura candidatului



Declarație

Subsemnatul/a RÉSZEG ALPÁR, absolvent(ă) al/a specializării
INFORMATICA, 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: 14.06.2023

Absolvent

Semnătura.....Réseg.....

Kivonat

Dolgozatom célja a 2D játékokban alkalmazott szabályrendszer és játékmenet kialakításának folyamatának részletes vizsgálata, a kezdeti koncepció felvetésétől az implementáción át egészen a megvalósított játék tesztelésig és a felmerült hibák kiküszöböléséig. Az áttekintést és az elméleti kereteket egy saját tervezésű oktatójáték létrehozásával gyakorlatban is alkalmazom.

A dolgozatban a szakirodalom által meghatározott tervezési paradigmák vizsgálatára összpontosítok, és azonosítom azokat a területeket, ahol még további fejlődésre van szükség kollektív szinten. A célom az, hogy megérthessem, hogyan alkalmazhatók ezek a paradigmák a játéktervezés gyakorlatában, és hogyan járulhatnak hozzá a játékelmény javításához.

A dolgozatban részletesen bemutatom az általam használt technológiákat, amelyek lehetővé teszik a játék fejlesztését és tesztelését. Ezután áttekintem a megvalósított szoftvert, beleértve a szabályrendszert, a játékmenetet és a grafikai megjelenítést. Emellett felismerem a jelenlegi program korlátait, és javaslatokat teszek a továbbfejlesztési lehetőségekre.

A dolgozat eredményeként olyan átfogó képet nyújtok a 2D játékok tervezésének és fejlesztésének folyamatáról, amely segít a játékfejlesztőknek és tervezőknek a szabályrendszer és játékmenet hatékony kialakításában. Továbbá, azonosítom azokat a területeket, ahol még további kutatásra és fejlődésre van szükség a játéktervezés terén.

Kulcsszavak: 2D játékok, szabályrendszer, játékmenet, tervezési paradigmák, oktatójáték, játékfejlesztés, grafikai megjelenítés, játéktervezés, fejlesztési folyamat.

Rezumat

Scopul tezei mele este de a examina în detaliu procesul de proiectare a regulilor și a gameplay-ului în jocurile 2D, de la conceptul inițial până la implementarea, testarea și depanarea jocului. Prezentarea generală și cadrul teoretic vor fi puse în practică prin crearea unui joc educațional de concepție proprie.

În această teză, mă concentrez pe examinarea paradigmelor de proiectare identificate în literatura de specialitate și identific zonele în care este necesară o dezvoltare suplimentară la nivel colectiv. Scopul meu este de a înțelege cum pot fi aplicate aceste paradigme în practica de proiectare a jocurilor și cum pot contribui la îmbunătățirea experienței de joc.

În această teză voi detalia tehnologiile pe care le folosesc pentru a permite dezvoltarea și testarea jocurilor. Voi oferi apoi o prezentare generală a software-ului implementat, inclusiv a sistemului de reguli, a gameplay-ului și a prezentării grafice. În plus, voi identifica limitările software-ului actual și voi face sugestii pentru îmbunătățiri ulterioare.

Ca rezultat al acestei teze, voi oferi o imagine de ansamblu cuprinzătoare a procesului de proiectare și dezvoltare a jocurilor 2D, care va ajuta dezvoltatorii și proiectanții de jocuri să conceapă un sistem de reguli și un gameplay eficient. În plus, voi identifica domeniile în care sunt necesare cercetări și dezvoltări suplimentare în domeniul proiectării de jocuri.

Cuvinte cheie: jocuri 2D, sistem de reguli, gameplay, paradigme de proiectare, joc educațional, dezvoltare de jocuri, reprezentare grafică, proiectare de jocuri, proces de dezvoltare.

Abstract

The aim of my thesis is to provide a detailed examination of the process involved in designing the rule system and gameplay of 2D games, starting from the initial concept proposal, through implementation, game testing, and error fixing. I apply this analysis practically by developing my own educational game.

In this thesis, I focus on exploring the design paradigms defined in the literature and investigating their practical application. Additionally, I identify areas where further collective improvement is needed. My goal is to understand how these paradigms can be applied in game design practice and how they contribute to enhancing the overall gaming experience.

I present a detailed overview of the technologies I utilize in the development and testing of the game. Subsequently, I provide an overview of the implemented software, including the rule system, gameplay mechanics, and graphical representation. Moreover, I assess the limitations of the current program and propose suggestions for further development opportunities.

As a result of this thesis, I offer a comprehensive understanding of the process of designing and developing 2D games, which can assist game developers and designers in effectively creating rule systems and gameplay mechanics. Furthermore, I identify areas that require further research and advancement in the field of game design.

Keywords: 2D games, rule system, gameplay, design paradigms, educational game, game development, graphical representation, game design, development process.

Tartalomjegyzék

1. Bevezető	10
1.1. Célkitűzések	11
2. Elméleti megalapozás	12
2.1. Hasonló alkalmazások	12
2.2. Szakirodalmi áttekintő	13
3. Tervezés	14
3.1. Játék tervezési szempontok	14
3.1.1. Játékosok	14
3.1.2. Fejlesztők	15
3.2. Játékmenet	15
3.3. Szabályrendszer	16
3.4. Verziókövetés	17
4. Programok, technológiák bemutatása	18
5. A megvalósított szoftver	19
5.1. Rendszerkövetelmények	19
5.1.1. Funkcionális követelmények	19
5.1.2. Nem funkcionális követelmények	19
5.2. A játék bemutatása	20
5.2.1. Unity felület	20
5.2.2. Scriptek	20
5.3. Diagramok	29
5.3.1. Use Case diagram	29
5.3.2. Szekvencia diagram	30
6. Limitációk	31
7. Továbbfejlesztési lehetőségek	32
Összefoglaló	33
Köszönetnyilvánítás	34
Irodalomjegyzék	35

1. fejezet

Bevezető

Kisgyerek korom óta foglalkoztattak a számítógépek, úgy fizikai összeállításuk, mint a rajtuk futó szoftverek, háttérfolyamatok, és összességében az egész digitális világ. Természetesen, mint a legtöbb alsótagozatos kisgyereket, engem is a számítógépes játékok tudtak a leginkább lekötni. A majdnem egész napot felölelő tanulás után a legjobb kikapcsolódást a játékok tudták nyújtani számomra. Akkoriban azonban még egyáltalán nem gondoltam volna, hogy nem csak szórakozásra, hanem tanulásra is alkalmasak lehetnek a játékok, sőt akár a kettő kombinálásával is hatalmas lépéseket lehet tenni egy adott irányba a tanulással.

Néhány játékkal eltöltött év után elérkezett az a pillanat az életemben, amikor a jövőmről kellett döntenem, és abban a helyzetben voltam, hogy nem volt igazi tervem a továbbtanulásomról. Ekkor jött az a hirtelen felindulás, hogy az informatika irányába induljak el. Mivel azonban nem volt előzetes tudásom a témában, eléggé nehéznek bizonyult a felzárkózás a tananyagban elvárt szinthez, és akkor éreztem azt, hogy hatalmas segítség lenne egy olyan lehetőség, amely egyszerűen, de hatékonyan tudna segíteni nekem belerázódni a programozás világába. Látva, hogy nem vagyok egyedül ezzel a problémával, mivel szaktársaim között is volt olyan, aki hasonlóan, előzetes tudás nélkül érkezett az egyetemre, azon kezdtem törni a fejem, hogy hogyan lehetne egy mindenki számára elérhető tanulási platformot létrehozni, amely a kezdetektől fogva vezeti az embert a tanulás minden fázisában. Ebből az elgondolásból kiindulva fogalmazódott meg az az ötlet a fejemben, hogy segíteni kellene valahogyan a hasonló cipőben járó társaimon, és ez az oka annak, hogy elindult ennek a szoftvernek a fejlesztése.

Mindenképpen szerettem volna összevonni a számítógépes játékok által nyújtott szórakoztató élményt, a céltudatos tanulás elősegítésével. Koncepcióm alapját az képezte, hogy fokozatos nehezítések és gyakorlás által megtaníthassam, illetve megszerettessem a játékosokkal a programozás világát, és bemutassam, hogy az elsőre akár unalmasnak tűnő, értelmetlen szavak halmaza mennyi mindenre képes, mekkora alkotási teret ad annak, aki egy kis energiát fordít arra, hogy megértse a működését. Szerettem volna egyszerre új és régi játékelemeket felhasználni, és egy kis kihívás elé is állítani magamat, hogy a mostanra már mindennapjaim szerves részét képező programozást mennyire tudom kreatívan, de felhasználóbarát és a továbbiakban is felhasználható módon átadni.

1.1. Célkitűzések

A konkrét céloom tehát egy olyan játék megalkotása lett, amely segítségével bárkit bevezethetünk a programozás világába, ami a könnyed szórakozás mellett értékes tudással is felruházza a játékosokat, mindezt pedig egy olyan köntösbe helyezi, amely később egyszerűen felhasználhatóvá teszi az elsajátított tudást.

Mivel összetett játékot szerettünk volna csinálni, fontosnak tartottuk, hogy jól legyen felosztva a munka egymás között. Ezt figyelembe véve külön választottuk a 3D illetve 2D részt. Az én feladatom a 2D részen belül a játékmenet és a szabályrendszer implementációja volt, a továbbiakban tehát ezeknek a megvalósítását részletezem, az előzetes kutatást a témában, az akadályokat amik hátráltattak, és természetesen a sikereket amiket elértünk a fejlesztés során.

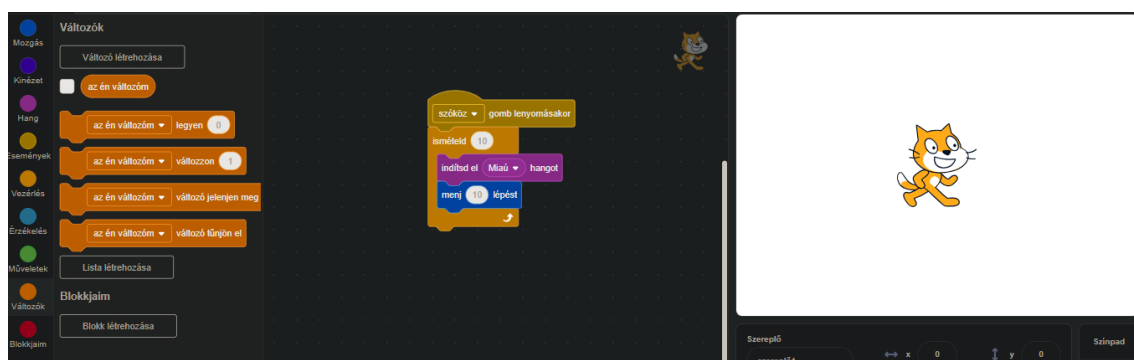
2. fejezet

Elméleti megalapozás

2.1. Hasonló alkalmazások

A blokk alapú programozás nem egy újkeletű dolog, több ismert megvalósítása is létezik már, így több referenciánk is volt arra, hogyan is kellene felépíteni a játékunkat. Ennek ellenére viszont egyik sem tartalmazta konkrétan azt amit mi szerettünk volna nyújtani.

A legelterjedtebb a már meglévő alkalmazások közül a Scratch, amely egy böngészőben futtatható felülettel rendelkezik, több nyelven is elérhető, és egy nagyon jó lehetőség, amit gyerekeknek illetve kezdő programozóknak fejlesztettek ki. A blokkok segítségével irányíthatunk különböző karaktereket, és így animációkat hozhatunk létre, ahogyan a [2.1 ábrán](#) is látható. Ehhez hasonló a Tynker illetve Blockly felépítése is, utóbbiban különböző feladatokat kell megoldanunk, például végigvezetni egy karaktert egy labirintuson, hasonló blokkokkal mint amilyenek a Scratch-ben találhatóak. Viszont ezek a programok nem egy adott programozási nyelvre fókuszálnak, csupán az általános programozói gondolkodás megtanulására szolgálnak.



2.1. ábra. A Scratch felhasználói felülete

Vannak olyan megvalósítások is, amelyek főleg olyanoknak tudnak szórakozást nyújtani akik már jártasabbak a programozásban, ilyen például a Human Resource Machine, ahol ugyanúgy blokkok segítségével kell irányítanunk a karakterünket, viszont itt sem a konkrét programozáson van a lényeg, sokkal inkább a logika elsajátításán.

Ezeket a játékokat figyelembe véve döntöttem úgy, hogy hasznos lenne egy olyan megvalósítás, ami konkrétan egy programozási nyelvet tanít meg.

A játékmenet egy másik fő inspirációja nem egy programozással foglalkozó, de tanulásra kifejlesztett program volt, a Duolingo. A játék ellenőrzés részét, amely során ellenőrizzük, hogy az adott blokkok a megfelelő pozícióba lettek-e helyezve, ez a program ihlette.

2.2. Szakirodalmi áttekintő

Számos szakirodalmi kutatás fogalmazta meg azt a problémát, hogy nincs egy pontosan meghatározott módja a játékok megtervezésének, így mindenki csak próbál rájönni a legkézenfekvőbb technikára, amelyet alkalmazva a legkevesebb probléma léphet fel a tervezés során, ez pedig magába foglalja azt, hogy legyen egy jól meghatározott játékmenetünk, amelyet követni szeretnénk.

Tracy Fullerton [Ful14] könyvében a játékmenet és a szabályrendszer kiemelt szerepet kap. Gyakorlati útmutatót nyújt a játéktervezéshez, ahol a szerző interaktív és játékos megközelítést alkalmaz a játékmenet kialakításában, és bemutatja, hogyan lehet a szabályokat dinamikus és élvezetes játékélménnyé formálni.

Brenda Brathwaite és Ian Schreiber [BS09] könyve olyan kihívásokat és gyakorlati feladatokat kínál, amelyek segítségével a játéktervezők elmélyíthetik a játékmenet és a szabályrendszer megértését. A játékmenet kialakítása és finomítása mellett a szerzők számos szabályrendszeri témát is érintenek, beleértve a konfliktust, az egyensúlyt és a progressziót.

Jesse Schell [Sch08] könyvében átfogóan beszél a játéktervezésről, a játékmenetet a játékosok interakcióinak és az általuk átélt élményeknek a tervezésére összpontosítja, valamint bemutatja, hogyan kell jól meghatározott és kiegyensúlyozott szabályokat kialakítani.

Következtetésképpen arra jutottam, hogy egy minőségi játék tervezésének első lépése az előre pontosan megtervezett szabályrendszer kialakítása, majd a játékmenet ráépítése a szabályrendszer kereteire.

3. fejezet

Tervezés

3.1. Játék tervezési szempontok

A játékunk tervezési szakaszában először csupán 3D játékmenetben gondolkodtunk, de jobban átgondolva az egész koncepciót, a kezdetektől fogva a legkézenfekvőbb megoldást szerettük volna nyújtani a programozás tanulásához. Ehhez pedig mi lenne találóbb, mint hogy magát a tanulási folyamatot egy olyan környezetben folytassuk le, amelyhez nagy valószínűséggel a későbbiekben is kötődni fogunk. Ezért lett bevezetve a 2D világ, és ezért az egészet a Visual Studio Code fejlesztői környezet Pixel Artosított verziójában láthatjuk. Ezzel már a kezdetektől fogva kialakítunk egy köteléket a játékos és a valós programozási világ között. Ezen az úton tovább haladva, arra törekedtünk, hogy megmutassuk a programozás szépsége mellett a Pixel Art nyújtotta lehetőségeket.

3.1.1. Játékosok

Egy 2022-es kutatás [ZHF⁺22] kimutatta, hogy a Pixel Art még mindig hatalmas befolyást gyakorol a játékosok közösségére. Miután pedig az első lépcső a 3D világ a játékunkban, kontrasztba próbáltuk helyezni a modernebb grafikai megvalósításokat, a legelsőként használt megoldásokkal. Erre pedig tökéletes a Pixel Art. Ahogyan a Pixel art for game developers [Sil15] könyvben olvashatjuk, nem is csupán egyetlen oka van annak, hogy a játékosok ennyire szeretik a mai napig az ilyen stílusú játékokat.

Elsősorban ott a nosztalgiafaktor, amely annak ellenére, hogy főleg az idősebb korosztálynál van jelen, segíthet fenntartani a közösséget az által, hogy a Pixel Art szerelmesei belevezetik a közösség fiatalabb tagjait ebbe a világba. Ez mellett látható, hogy az utóbbi években a játék műfajok egyre inkább szétváltak, és ma már számos különböző típusú játék áll rendelkezésre, az elérhető élményeknek és esztétikáknak soha nem volt még ennyi fajtája. Az eltelt idő alatt számos elfogadott stílus alakult ki a különböző kategóriákban, és úgy tűnik, hogy a Pixel Art időtálló maradt.

Ezt bizonyítja többek között az is, hogy az ilyen stílusban készült játékok közül több is megtalálható a Steam toplistáján. A Stardew Valleynek (3.1 ábra) naponta átlagosan 30-40 ezer aktív játékosa van, a Terraria napi játékoszáma úgyszintén 35 ezer fölötti. Ezek alapján el kell ismernünk, hogy a Pixel Art stílus ikonikus megjelenése miatt különleges élményt nyújt mindenki számára.



3.1. ábra. A Stardew Valley grafikája

3.1.2. Fejlesztők

A játékosok igényei mellett figyelembe kellett vennünk azt is, hogy tapasztalat hiányában nem feltétlenül kellene túlvállalnunk magunkat a játék grafikájával kapcsolatban. Mivel a Pixel Artnak éppen abban rejlik az előnye, hogy nem igényel hatalmas előzetes tudást, így egyértelmű volt számunkra, hogy ez lesz az a megközelítés, ami a legkézenfekvőbb. Az említett [Sil15] könyv alapján rengeteg hasznos dolgot megtudtam a stílus kivitelezésével kapcsolatban, ez nagyban hozzájárult a design elkészítéséhez. A könyvet Daniel Silber írta, aki művészi pályáját játékok tervezésére cserélte, így tapasztalatai nyomán rengeteg hasznos, értékes és a szakmában is használt tanáccsal látott el.

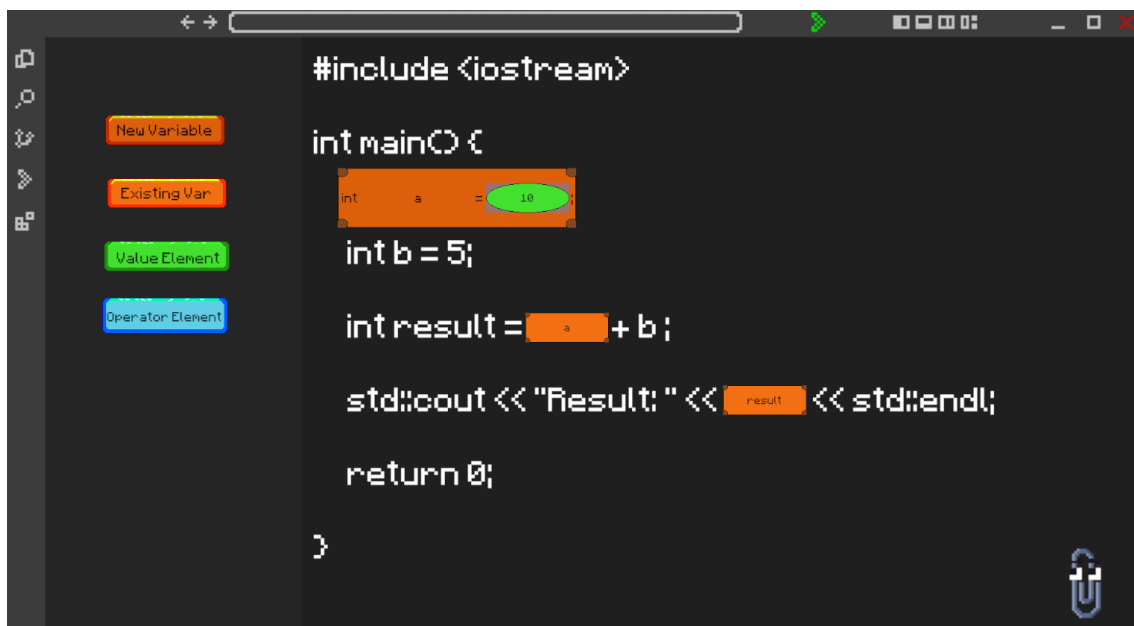
3.2. Játékmenet

A játékmenet helyes tervezése általánosságban azonos sémára épül, és elsősorban a játékosok igényeit veszi figyelembe. Brenda Brathwaite és Ian Schreiber könyvükben [BS09] megfogalmazták az alapvető kérdéseket, amiket fel kell tennünk magunknak egy játék megtervezése előtt. Ezek a kérdések magukba foglalják a játék alapvető mechanikáit, mint például: Hogyan lehet győzni? Miért akar valaki a játékkal játszani? Mit tud és kell elvégezzen a játékos?

Ezen kérdések mentén indultam el én is, viszont rájöttem arra, hogy mivel oktatási célokra szeretnénk felhasználni a játékunkat, így néhány kérdést át kell fogalmaznunk. A játék célja (a tanulás) ki volt tűzve, így a legfontosabb kérdések a következők lettek: Hogyan tudunk egyszerűen, de hatékonyan tanítani? Hogyan kössük össze a már ismert programozási környezeteket a játékunkkal? Hogyan tudunk a játékosnak megadni a tanulás mellé egy olyan játékelményt, ami a folytatásra buzdítja őket? Ezeket összegezve és átgondolva kirajzolódott egy kép a végső célról: 3D felfedezés, összekötve a 2D blokk alapú, puzzle-es játékmenettel. Így ennek kellett megalkotni a világát.

Az elfogadott és implementált megoldás a következőképpen épül fel: Az egymásra épülő, 3D környezetben felépített fejezeteken belül található 3-3 puzzle, amelyeket megoldva léphetünk a következő fejezetbe. Minden fejezet hozzáférést biztosít több és több, egyre mélyebb programozási fogalomhoz és technikához. Ezeket a 3D részen összegyűjtve tudjuk felhasználni őket a puzzle-ekben, így elmélyítve a tudásunkat a különböző területeken. A puzzle egyszerű játékmenei elemeket tartalmaz, a megfelelő gombbal létrehozhatunk egy adott típusú blokkot, majd ezt az egérrel húzva behelyezhetjük az annak megfelelő helyre. A helytelenül behelyezett blokkok a gombok alatti részre ugranak, majd onnan újra felhasználhatóak lesznek.

A puzzle játékmeneivel szerettük volna elérni, hogy hasonlóan legyen megjelenítve, mint ahogyan egy fejlesztői környezetben látjuk a kódot, így a blokkok és az előre megadott kódrészletek együttesen képezik a végső kódot, amit a játékos lát (3.2 ábra). Ezután, ha a megfelelő helyre bekerültek a megfelelő blokkok, a játékos a zöld, futtatás gombra kattintva ellenőrizheti, hogy helyes megoldást adott-e a feladatra, és ha ez teljesül, megkapja a következő feladványt. A harmadik feladvány helyes megoldása után átkerül a következő 3D pályára, ahol újabb blokkokat gyűjthet össze, és újabb puzzle kihívások várják. Mindezekben a játékos segítségére áll Clippy, aki a jobb alsó sarokban található, és tippekkel járul hozzá a sikeres feladatmegoldáshoz, valamint az első puzzle során ő magyarázza el a játékosnak az irányítást is.



3.2. ábra. Egy kész puzzle

3.3. Szabályrendszer

Miguel Sicart [Sic08] cikkje több kutatást is figyelembe vesz, így több szemszögből is vizsgálja a kapcsolatot a szabályrendszer és játékmenei fogalmak között. A legszélesebb körben elfogadott definíciók alapján a játékmenei több elemből áll, célok, szabályok,

stratégiák, ezek közül a szabályrendszerhez kapcsolódik a játékmenet azon specifikus része, amely meghatározza mit tehetünk és mit nem a játékunkban.

A 2D részben elsősorban a játék nyújtotta lehetőségek szabályait kellett meghatározni. Ezek a következők: csak az összegyűjtött blokkokat lehessen felhasználni, ezekből is csak egy bizonyos mennyiséget, a blokkok mozgatását az egérrel megfogva lehessen elvégezni, a blokkok ne akadályozzák egymást a mozgásban. Ezután pedig a programozás szabályait kellett beépíteni a játékba, amely során, többek közt, a következő pontokat kellett figyelembe venni: a változók nevei nem kezdődhetnek számmal és nem tartalmazhatnak szóközőket, a változókhoz csak a saját típusukkal megegyező értékeket lehet hozzárendelni, egy névvel csak egy változó deklarálnak. Fontos volt, hogy konkrét szabályrendszer köré építsük fel a játékot, mivel ehhez kell alkalmazkodnia a játékosnak, és ennek elrontásával akár akaratlanul is rossz irányba terelhetjük őt a tanulás során.

3.4. Verziókövetés

Projektünkhöz a mindannyiunk által jól ismert GitHub verziókövető szolgáltatásait használtuk. Minden funkció implementálásához külön branchet hoztunk létre, majd miután megoldás született egy-egy problémára, merge-eltük a master branchre, így mindig a legfrissebb működő verzió volt a fő branchen. A repositorynk elérhető [ide](#) kattintva.

4. fejezet

Programok, technológiák bemutatása

C#

A C# egy Microsoft által fejlesztett modern, objektumorientált programozási nyelv. Többek között alkalmazásfejlesztésre, webfejlesztésre és játékfejlesztésre is használják. A Unity scriptjei is C# nyelvet használnak, így a funkciók fejlesztéséhez ezt használtam, előzetes ismereteim mellé így még nagyobb tudásra tettem szert ebben a nyelvben. A nyelv funkcionális programozási paradigmákat is tartalmaz, így lambdákat is felhasználhattam a kódban, valamint a LINQ is segítségemre volt az adatok kezelésében.

Unity

Mivel a játékmotorok világában kevésbé mozogtam otthonosan, így egy olyan irányt szerettem volna elkezdni, amelyet később is felhasználhatok, viszont egyidejűleg belekezdeni is könnyű, erre pedig tökéletes választás volt a Unity Technologies által fejlesztett játékmotor. A Unity [weboldala](#) rengeteg tanuláshoz hasznos információval lát el, amelyeket már a kezdetektől felhasználhattam a saját játékom fejlesztésében. Ezeken felül egyszerűen lehet kezelni vele a 2D és 3D világok közötti váltást is, amely fontos szempont volt a játékunkban.

Aseprite

Az Aseprite egy grafikai tervezőprogram, amely rengeteg eszközhöz biztosít hozzáférést, és egyszerű lehetőséget nyújt a Pixel Art grafikai stílus megvalósításához. A felhasználói felület rám eső részét ennek a programnak a segítségével terveztem meg, többek közt például a blokkokat létrehozó gombokat, magukat a blokkokat, valamint a blokkokat tároló ItemSlotokat is. Sokrétű szerkesztési és exportálási lehetőségei miatt az egyik leghasznosabb program a grafikai tervezőeszközök közül.

5. fejezet

A megvalósított szoftver

5.1. Rendszerkövetelmények

5.1.1. Funkcionális követelmények

- A felhasználók a program elindítása után interakcióba tudnak lépni a menüben található gombokkal.
- A felhasználók szabadon válthatnak a feloldott fejezetek között.
- A játékosok egy fejezetbe belépve interakcióba léphetnek a világban elhelyezkedő objektumokkal, a karakterüket irányítva.
- Szabadon válthatnak a 2D és 3D felületek között.
- A 2D felületen interakcióba léphetnek a gombokkal, valamint a létrehozott objektumokkal.

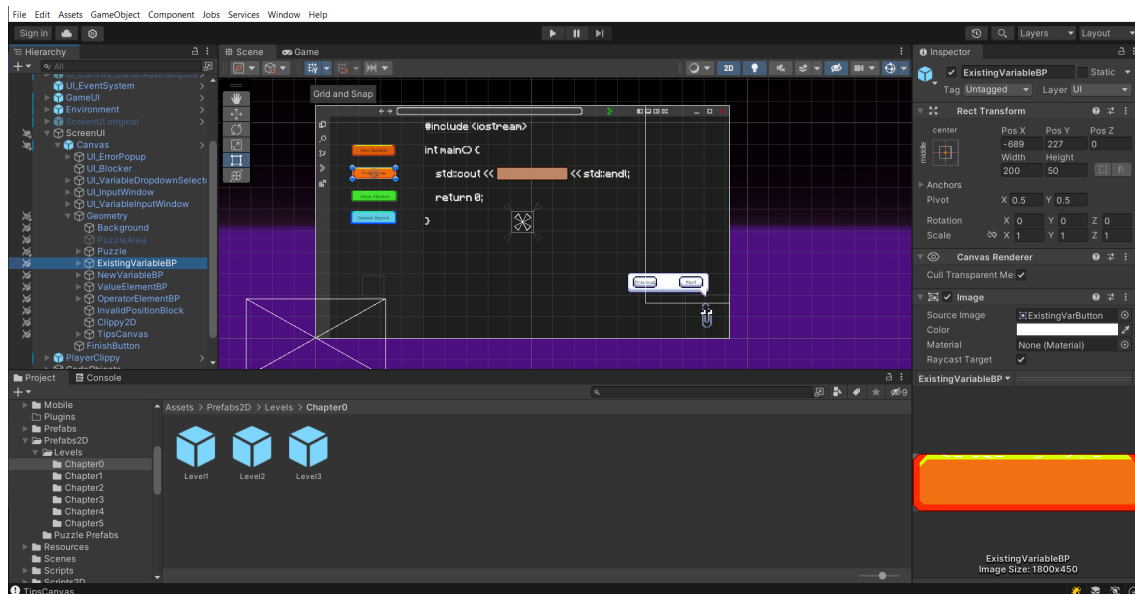
5.1.2. Nem funkcionális követelmények

- A játék futása során a rendszernek gyorsan kell reagálnia a felhasználói interakciókra, például a puzzle darabok mozgatására.
- A játéknak kompatibilisnek kell lennie a különböző Windows operációs rendszerekkel, például Windows 7, Windows 8 és Windows 10.
- A játéknak felhasználóbarát kezelőfelülettel kell rendelkeznie, a felhasználók könnyedén tudjanak navigálni a menük között, könnyen megérthessék a játékszabályokat és egyszerűen léphessenek interakcióba a blokkokkal.
- A játék helyesen kell kezelje a különböző felbontásokat és képernyőméreteket, hogy a játékosok különböző méretű képernyőkön is élvezhessék a játékot.

5.2. A játék bemutatása

5.2.1. Unity felület

A Unity kezelőfelületén rengeteg beállítás érhető el a projekttel kapcsolatban, így a programozáshoz használt alkalmazás mellett ebben töltöttem a legtöbb időt. Egy fejezet felépítése ebben a kezelőfelületben az [5.1 ábrán](#) látható. Nagyon fontos az is, hogy a legtöbb hivatkozás ezen a felületen állítható be, így az objektumokat helyesen kellett összekötni, és a változtatásokat is folyamatos felügyeletet igényeltek.



5.1. ábra. A Unity kezelőfelülete

5.2.2. Scriptek

Interface-ek

Az IBlock interface a polimorfizmus megvalósításában játszik szerepet a projektben. Mivel egyes pozíciókba helyezhetünk operátor és érték blokkokat is, így szükséges volt, hogy az ezeket a blokkokat leíró osztályok egy közös interface-től öröklődjenek, így hozzásegítve a programot a helyes működéshez. Az interface saját implementálandó metódusokkal nem rendelkezik.

Enumok

- ExpressionE: A kifejezések enumja, segít, hogy nyilvántartsuk az elérhető kifejezéseket, és könnyedén hozzáadhassunk újakat.
- ItemSlotType: Ez az enum az azonos prefabet használó ItemSlot-ok megkülönböztetésére szolgál, beállítása a Unity szerkesztőfelületén történik.

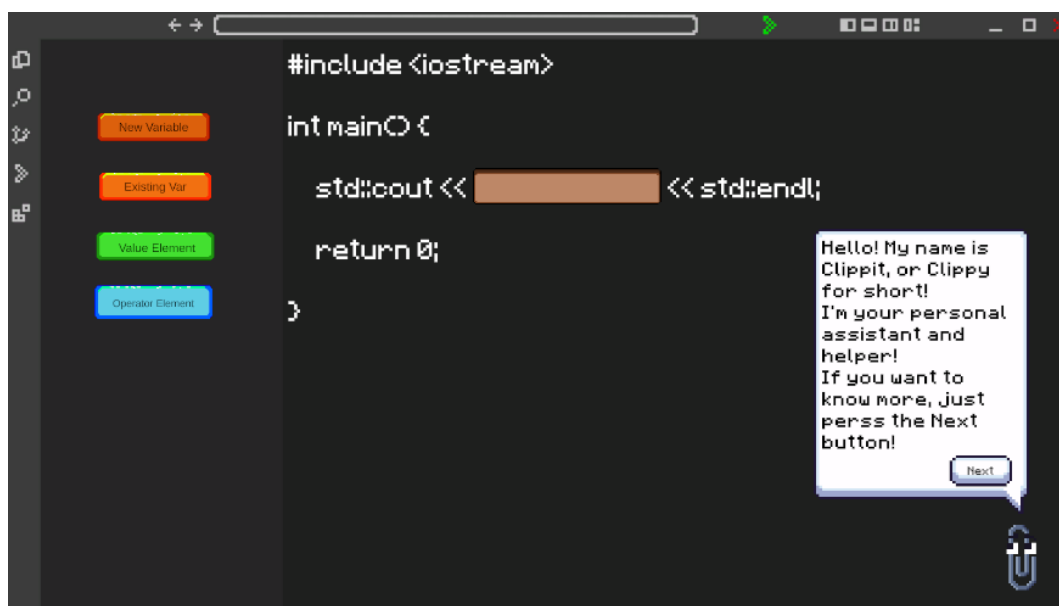
- **ObjectBlockTypeE:** Ez az enum a blokkok megkülönböztetésében játszik szerepet, minden új blokk tartalmaz egy szövegdobozt, amelybe beleíródik a típusa, a későbbi ellenőrzés elvégzéséhez.
- **OperatorE:** Az operátorok típusait tartalmazza ez az enum, ezek kódbéli felismerésében segít, valamint a 3D részben összegyűjtött objektumok 2D formájukká alakításában.
- **TypeE:** Az elérhető típusokat tartalmazza, az operátorokhoz hasonlóan a változóblokkok nyilvántartásában segít.

Modellek

- **Expression:** Egy kifejezés objektumot ír le, a kifejezés értékét egy getter és egy setter metódussal módosíthatjuk valamint kérhetjük le.
- **Operator:** Egy operátor objektumot ír le, egy getter és egy setter metódust tartalmaz, amelyek az operátor típusát térítik vissza, illetve állítják be.
- **Variable:** Egy változó objektumot ír le, egy getter és egy setter metódust tartalmaz, amelyek a változó típusát térítik vissza, illetve állítják be.

Funkcionalitások

Miután a játékos elvégezte feladatát a 3D világban, és a megfelelő gombkombinációt használva áttért a játék 2D felületére, a legelső puzzle során először Clippyvel találja szemben magát, aki az eddigiek ellenére most már nem a játékos által irányított karakter, itt már egy segítő társként van jelen, aki tanácsokkal látja el a játékost, miután elmagyarázta az alapvető irányítását ennek a felületnek (5.2 ábra).

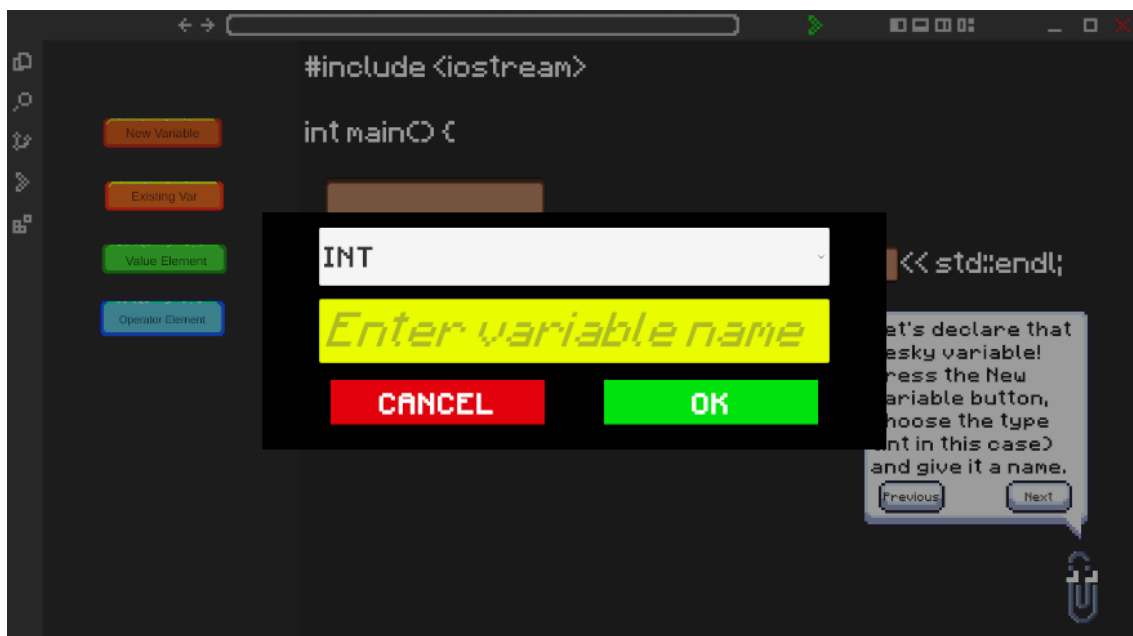


5.2. ábra. Első találkozás a 2D felülettel

A játékos Clippyvel való interakciója után a feladvánnyal találja szemben magát, valamint a képernyő bal oldalán azon gombokkal, melyek segítségével blokkokat hozhat létre, hogy aztán azokat felhasználva elvégezhesse az adott pálya feladatát. A gombok, ahogyan az 5.2 ábra is mutatja, sorrendileg a következők:

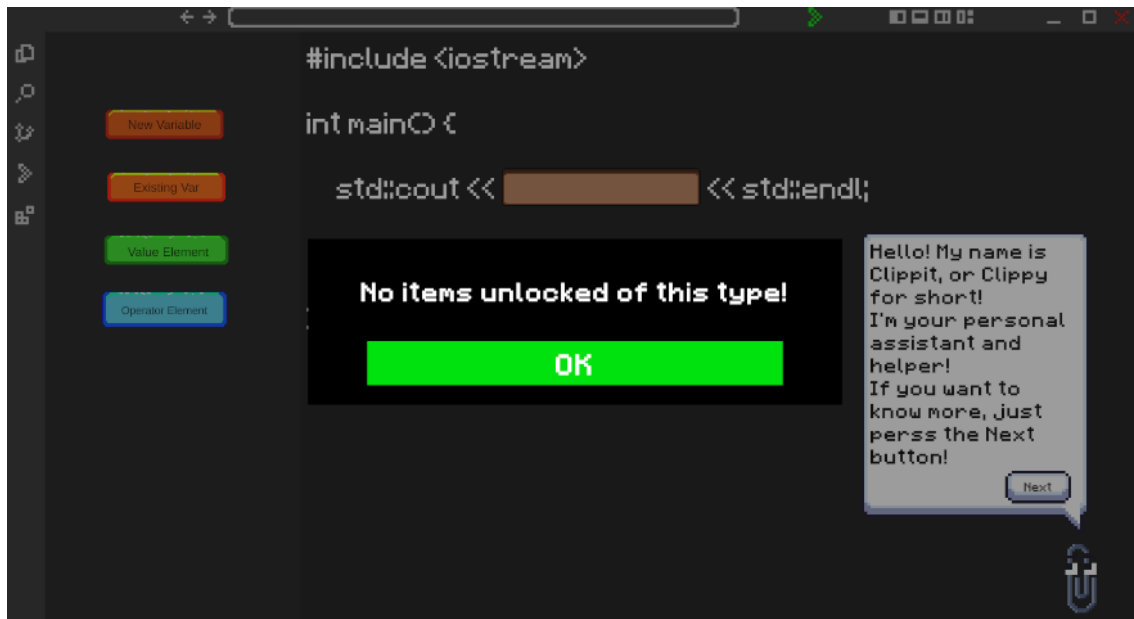
- New Variable: Ennek a gombnak a segítségével egy új változót hozhatunk létre, a típusát az összegyűjtött típusokból tudjuk kiválasztani egy legördülő menüben, míg a nevét az alatta található szövegdobozba kell beírunk (5.3 ábra). Ha még nem gyűjtöttünk össze egyetlen típusblokkot sem a 3D részben, egy felugró ablak jelzi ennek szükségességét (5.4 ábra). A típusblokkok összegyűjtés után egy listában tárolódnak el, innen kerülnek bele a típusválasztás során megjelenő ablakba.

Ez a gomb a NewVariable.cs scripttel rendelkezik, amely kattintás érzékelésekor megjelenít egy felugró ablakot. A felugró ablak kezeli a legördülő menüt, valamint a bemenet helyességének ellenőrzését is. Ha a bemeneti adatok helyesek, létrehozza a megfelelő prefabból az új változó blokkját. A létrehozott változók nevei és típusai egy Dictionaryben tárolódnak el, a kulcs a változók neve lesz.



5.3. ábra. Változót deklaráló blokk létrehozása

- Existing Var: Ez a gomb egy már deklarált változóból hoz létre egy példányt, amely csak a változó nevét tartalmazza, így felhasználható a kódban műveletek elvégzésére, illetve kiíratásoknál. Ennek a scriptje csak a létrehozni kívánt változó nevét kéri be egy ablakban (5.5 ábra), és ha előzetesen létre lett hozva egy megadott nevű változó, elkészíti a blokkot.
- Value Element: Ezen gomb lenyomásakor, ha már oldottunk fel típust, egy típusválasztó legördülő menü jelenik meg, melyből miután kiválasztottuk a típust, egy új ablakban beírhatjuk a megfelelő értéket, amelyet szeretnénk adni a blokknak. A bemenet helyességét itt is a bemeneti ablak ellenőrzi, az adott típusú értékblokkoknak nem tudunk helytelen értéket adni.

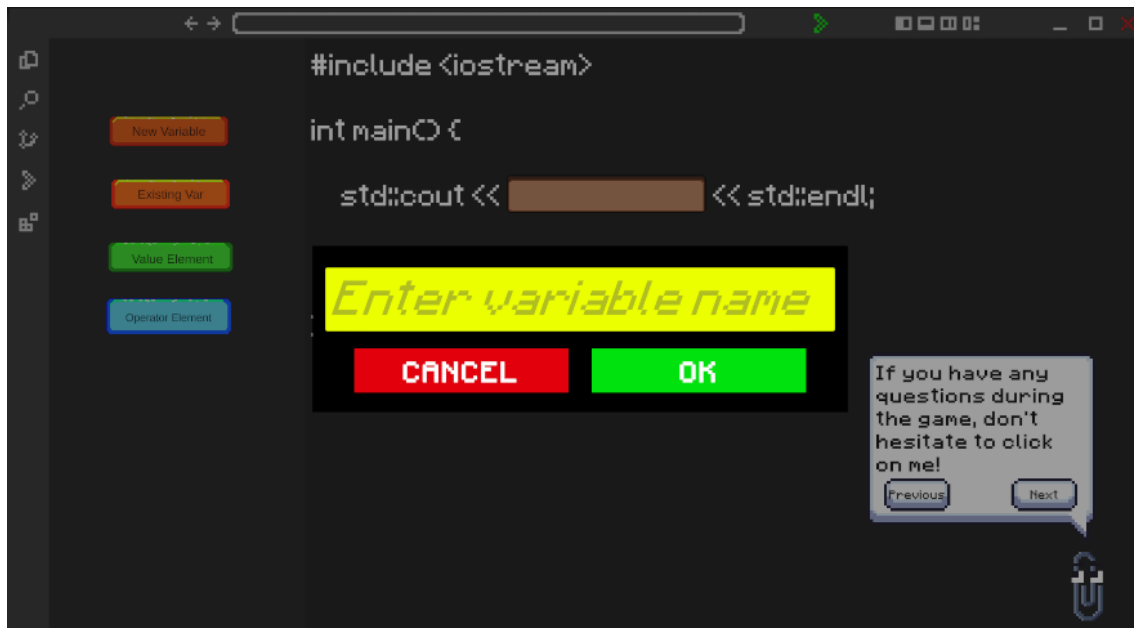


5.4. ábra. Felugró ablak hibaüzenettel

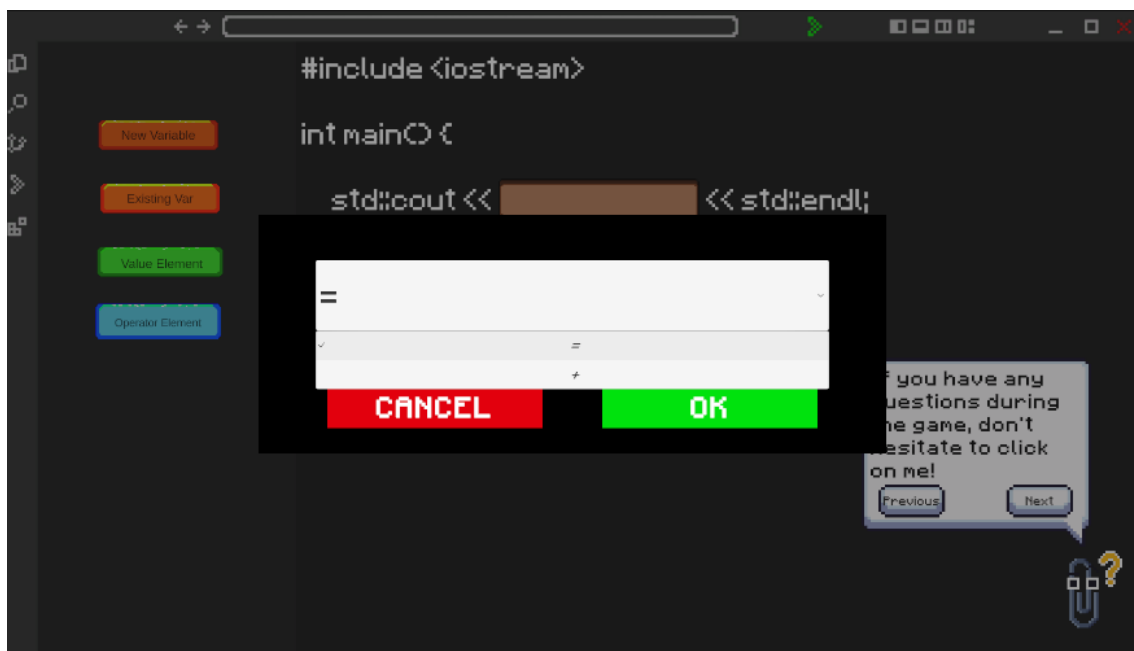
- Operator Element: A feloldott operátorokból ennek a gombnak a segítségével tudunk blokkot létrehozni, egy legördülő menüből kiválasztva a kívánt operátort (5.6 ábra). Ha még nem oldottunk fel egyet sem, az 5.4 ábrához hasonló felugró ablakot kapunk, a problémáspecifikus üzenettel.
- Expression Element: A különböző kifejezéseket ezzel a gombbal érhetjük el, mint például if, else, switch. Az operátorokhoz hasonlóan itt is egy legördülő menüből válasszthatjuk ki az igényelt kifejezést, viszont itt egy 3D blokk több kifejezést is felold, példának okáért az if kifejezés összegyűjtése feloldja az else és else if kifejezéseket is (5.1 kódrészlet).

```
case "if":
    newExpression.SetExpression(ExpressionsE.IF);
    unlockedExpressions.Add(newExpression);
    var expr2 = new GameObject("Expression").AddComponent<Expression>();
    expr2.SetExpression(ExpressionsE.ELSE);
    unlockedExpressions.Add(expr2);
    var expr3 = new GameObject("Expression").AddComponent<Expression>();
    expr3.SetExpression(ExpressionsE.ELSEIF);
    unlockedExpressions.Add(expr3);
    break;
```

5.1. kódrészlet. Több expression feloldása



5.5. ábra. Változó blokk létrehozása

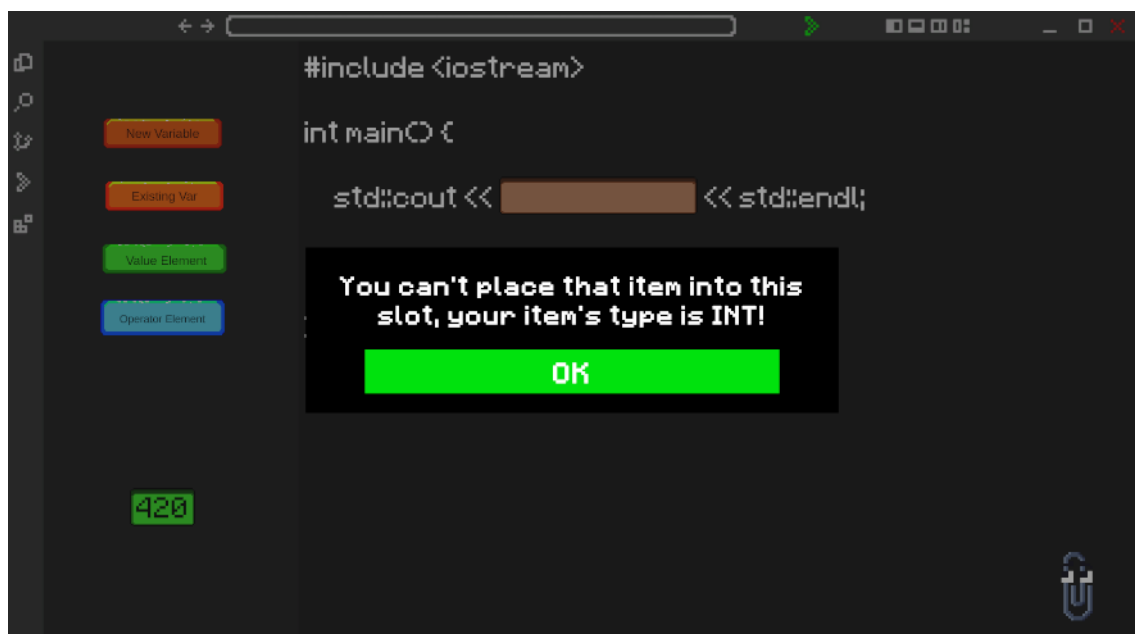


5.6. ábra. Operátor kiválasztása blokk létrehozásához

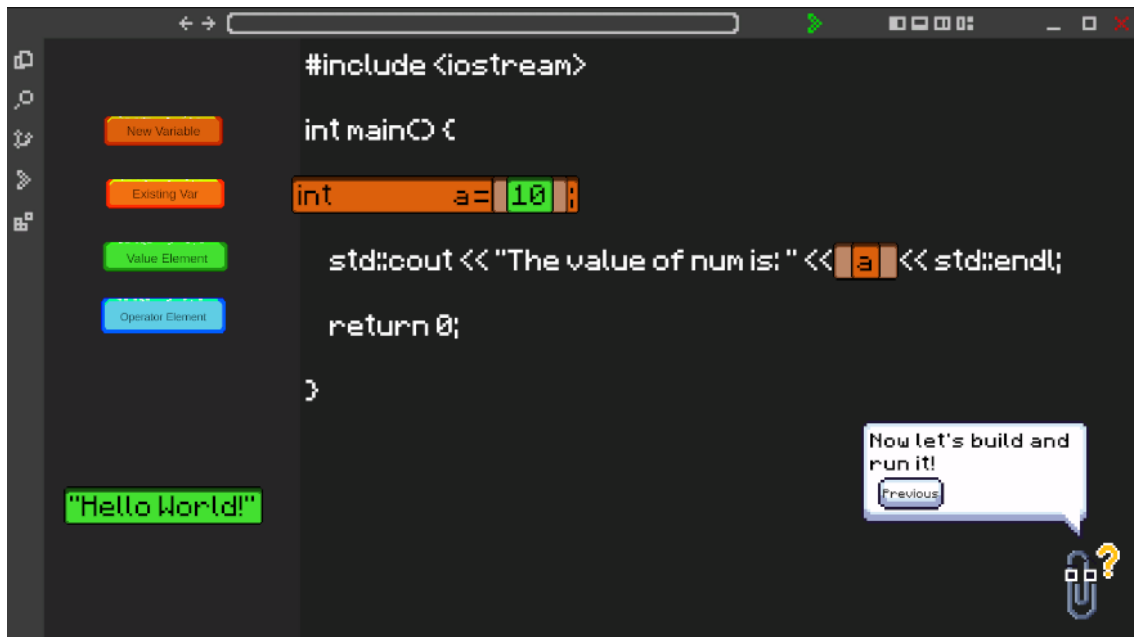
Miután a játékos létrehozott egy blokkot, a legfontosabb dolog, amit tehet vele, hogy az egér segítségével elmozgatja azt a képernyőn. Ezt a mozgást a CombineElements script valósítja meg. A CombineElements osztály implementálja a következő interface-eket: IEndDragHandler, IBeginDragHandler, IDragHandler, IPointerDownHandler, IPointerUpHandler, ezek a játékos interakcióinak kezelésében segítenek, beépített interface-ei a Unitynek.

Az egérrel való húzás helyes megjelenítéséhez először kiszámoljuk a húzott elem és az egér pozíciója közötti különbséget (ofszet), majd a húzás során letiltjuk, hogy a blokk interakcióba léphessen más blokkokkal. Az objektum mozgatása során folyamatosan számoljuk az ofszet és az egér pozíciójának összeadásával, hogy hová húzta a blokkot a játékos. Az egérgomb felengedésekor ellenőrizzük, hogy éppen egy ItemSlot fölött található-e az objektumunk.

Ha a húzott blokkunkat egy kódrészletnek kijelölt helyen tesszük le, elkezdődik az ellenőrzés, amely azt figyeli meg, hogy az adott helyre lehelyezhető-e a blokk. Mivel minden pozíciónak meg van előre határozva, hogy milyen típusú blokkot kell odahelyeznünk (változó deklaráció, értékblokk, operátor vagy változónév), így ha ezek megegyeznek, a blokk behelyezésre kerül a pozícióba. Máskülönben a gombok alatti területre kerül a blokk, ahonnan a játékos újra elérheti (5.8 ábra), és egy hibaüzenet fogadja a játékost (5.7 ábra).



5.7. ábra. Hibás elhelyezés hibaüzenete



5.8. ábra. A helytelen blokkok pozíciója ("Hello World!")

A CombineElements script együttműködik az ItemSlot scripttel az elemek helyességének ellenőrzéséhez. Az IDropHandler interface ItemSlot osztály által implementált OnDrop metódusa megvizsgálja a ráhúzott objektum típusát és az ItemSlot típusát is, ha ezek megegyeznek, akkor a húzott blokk új szülőobjektuma az adott ItemSlot lesz, és együtt mozognak, míg szét nem választjuk őket. Az OnDrop függvény egy switch-case függvénnyel vizsgálja az ItemSlot típusát, majd ellenőrzi, hogy ezzel megegyező objektumot akarunk-e belehelyezni.

Az 5.2 kódrészletben az új változó deklarálását ellenőrző esetet láthatjuk. Miután megbizonyosodtunk, hogy az érintett ItemSlot egy új változó deklarálását várja, ellenőrizzük, hogy az éppen húzott blokk Objektum típusú-e. Jelen kontextusban az objektum típus a már deklarált változóneveket, az operátorokat és az értéktípusú blokkokat foglalja magába, hátrahagyva így a változót deklaráló blokkot. Amennyiben objektumot szeretnénk belerakni az ItemSlotba, egy hibaüzenetet kapunk (5.7 ábra) és a blokk a hibásan elhelyezett blokkoknak kijelölt pozícióba kerül (5.8 ábra). Ha azonban új változót deklarálunk a blokkunkkal, az ItemSlot gyerekobjektumává tesszük a húzott blokkot, így összekötve azokat.

```
case ItemSlotType.NewVarPlace:
    Debug.Log("OnDrop Variable");
    if (IsObject(draggedObject))
    {
        Debug.Log("Not a valid object to place here!");
        _uiErrorPopup.Show("You can't place that item into this slot!");
        draggedObject.GetComponent<RectTransform>().position =
            invalidPositionBlock.transform.GetComponent<RectTransform>().position;
        draggedObject.transform.SetParent(invalidPositionBlock.transform);
    }
```

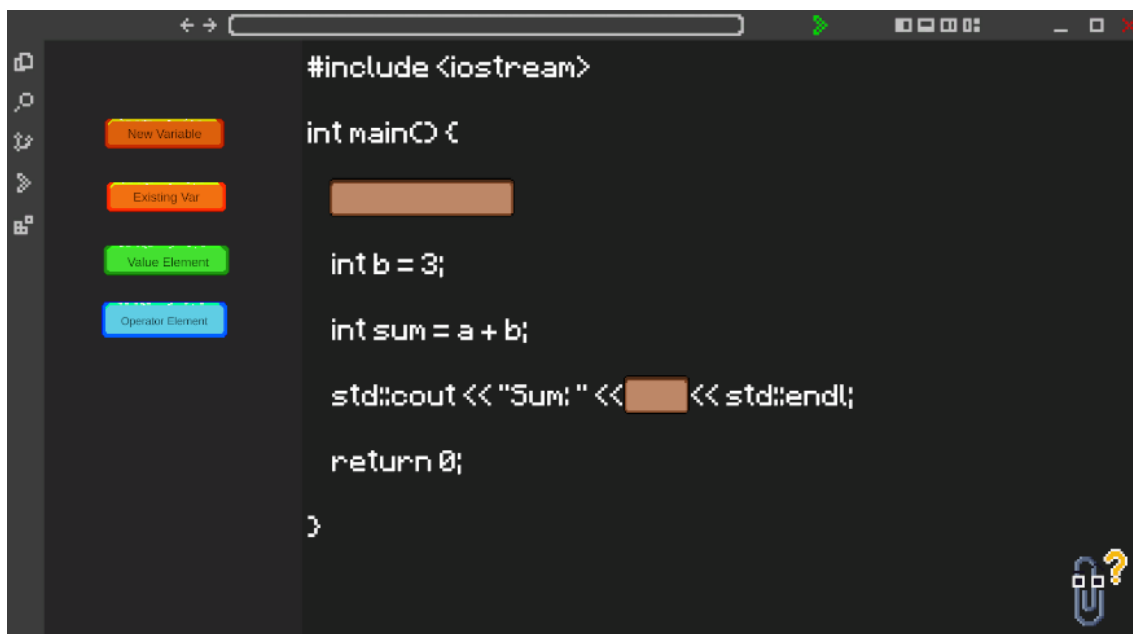
```

else
{
    draggedObject.transform.parent = transform;
    draggedObject.GetComponent<RectTransform>().position =
        GetComponent<RectTransform>().position;
    SetStoredObject(draggedObject);
}
break;

```

5.2. kódrészlet. ItemSlot ellenőrző eset

A játékos feladata teljesítéséhez tehát blokkokat hoz létre, és azokkal kiegészítve a megadott kódrészletet, összeállítja a végleges kódot. Mindebben segítségére van Clippy, ha a játékos a kurzort Clippyre mozgatja, egy kérdőjel jelenik meg mellette (5.9 ábra), ezzel sugallva, hogy ha kérdése merült fel a feladat során a játékosnak, itt választ kaphat rá. A segítséget tartalmazó szöveg egy szövegbuborékban jelenik meg, hasonlóképpen, ahogyan az 5.2 ábra is mutatja.



5.9. ábra. Clippy hover animációja

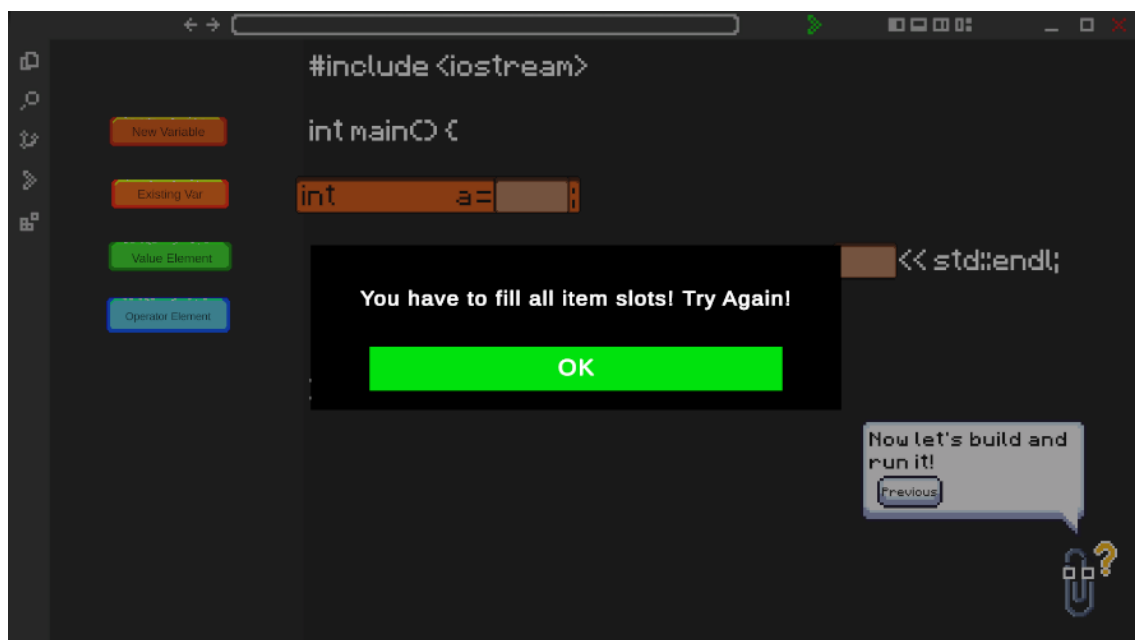
Miután a játékos befejezte az aktuális puzzle megoldását, "futtatja" a kódot a Visual Studio Code-ból is ismert zöld futtatás gombra kattintva. Ekkor, mivel az ItemSlotok csak a megfelelő blokkokat tartalmazhatják, az 5.3 kódrészlet ellenőrzi, hogy minden ItemSlot tartalmaz-e objektumot, mivel egy kódrészlet hiánya kompilálási hibát váltana ki. Az IsAllSlotFilled végigiterál a 2D elemeket tartalmazó Canvason, és ellenőrzi, hogy mindenik ItemSlot elembe került-e behelyezésre neki megfelelő elem. Amennyiben minden ilyen blokk fel lett töltve, a FinishPuzzle script kiadja az utasítást, és betöltődik a következő puzzle, ha pedig az aktuális puzzle a fejezeten belül az utolsó volt, feloldja a soron következő Chaptert, és átirányítja a játékost annak 3D pályájára. Ellenkező esetben egy hibaüzenet jelenik meg a játékosnak, amely jelzi, az összes kijelölt helyre szükséges blokkot tennie (5.10 ábra).

```

private bool IsAllSlotFilled(Transform parent)
{
    var children = parent.GetComponentInChildren<ItemSlot>();
    Debug.Log($"COUNT: {children.Length.ToString()}");
    var filledSlots = new List<bool>();
    foreach (var child in children)
    {
        if (child.GetStoredObject() != null)
        {
            filledSlots.Add(true);
        }
        else
        {
            filledSlots.Add(false);
        }
    }
    if (filledSlots.Count == children.Length &&
        !filledSlots.Contains(false))
    {
        return true;
    }
    return false;
}

```

5.3. kódrészlet. Blokkok jelenlétét ellenőrző függvény

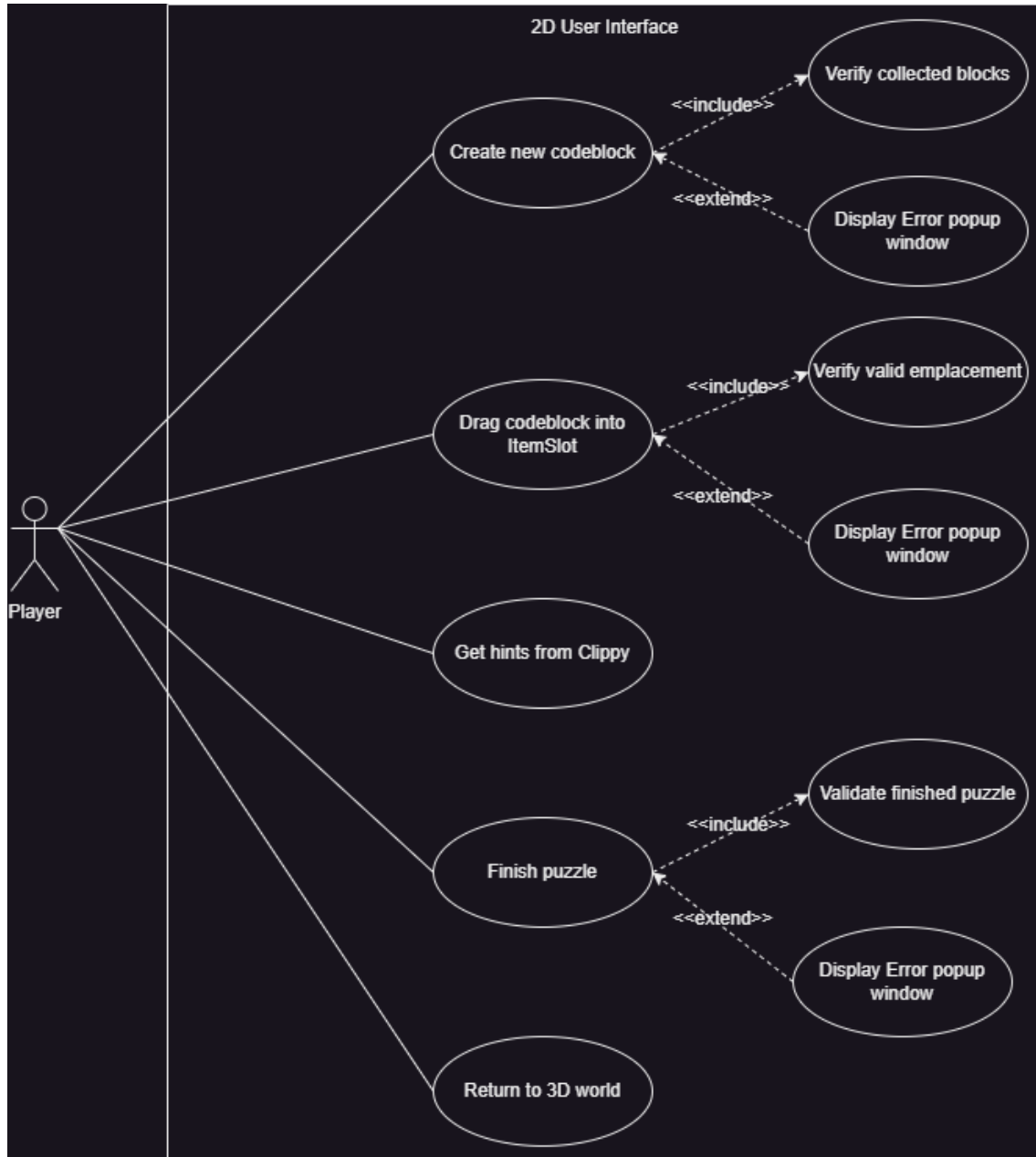


5.10. ábra. Helytelen megoldás hibaüzenete

5.3. Diagramok

5.3.1. Use Case diagram

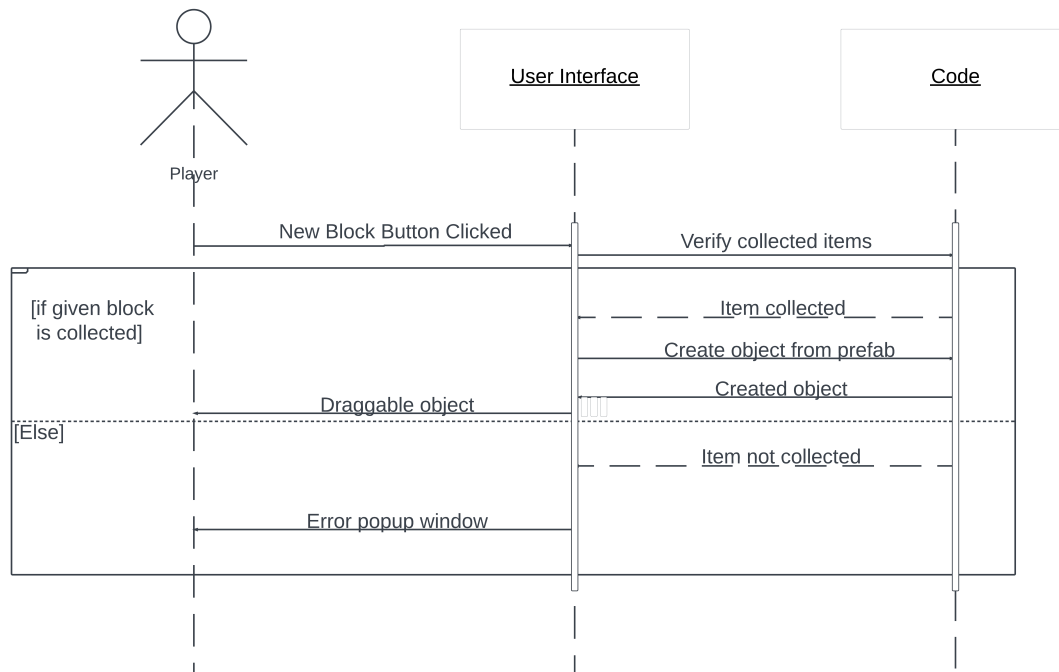
Az 5.11 ábrán látható Use Case diagram a játékos 2D világgal való interakciók lehetőségeit írja le.



5.11. ábra. A 2D világ use case diagramja

5.3.2. Szekvencia diagram

Az 5.12 ábrán látható szekvenciadiagram egy új blokk létrehozásának folyamatát írja le, az interakciókat, amelyeket a játékos végez a felhasználói felülettel, és hogy ez hogyan vetül le a kódra.



5.12. ábra. Új blokk létrehozása

6. fejezet

Limitációk

Alkalmazásunk jelenleg még számos limitációval rendelkezik, ezért még oktatásban való használatra nem alkalmas. Ennek ellenére úgy érezzük, jó úton haladunk a fejlesztésével, így a jelenlegi kipróbálásra alkalmas tesztverzió is rendelkezik egy kezdetleges tanulási folyamattal.

Mivel idő hiányában kevés pálya készült el, nem tudtuk részletesen és hosszasan bemutatni a programozás minden aspektusát. Igyekeztünk minél szerteágazóbb területet lefedni, és mindenből egy kis információt belefoglalni a játékba.

Az alkalmazásban jelenleg előfordulhatnak hibák, és optimalizálásra lehet szükség a felhasználói élmény javítása érdekében. A stabilitás, a sebesség és az általános teljesítmény további fejlesztése fontos szempont a játék további használhatósága szempontjából.

Ezeket a limitációkat felismerve és megértve, további fejlesztésekkel és javításokkal lehetőség nyílik az alkalmazásunk továbbfejlesztésére és jobb felhasználói élmény nyújtására.

7. fejezet

Továbbfejlesztési lehetőségek

Véleményünk szerint, jelenlegi állapota ellenére, játékunk hatalmas és értékes potenciállal rendelkezik a jövőre nézve. Igyekeztünk úgy felépíteni a játékot, hogy könnyedén továbbfejleszthető legyen, valamint a kibővítésére is legyen lehetőség.

Az egyik legkézenfekvőbb továbbfejlesztési irány a több programozási nyelv bevezetése lenne. Mivel jelenleg csak a C++ nyelvre koncentráltunk, annak ellenére, hogy ez egy széles körben ismert és használt nyelv, fontosnak tartjuk, hogy megadhassuk a lehetőséget a játékosoknak a választásra, ezzel egyidőben pedig nagyobb közönséget is elérhetünk játékunk, hiszen aki más programozási nyelvet szeretne megtanulni, annak is tudnánk lehetőséget nyújtani.

Egy következő lehetőség a továbbfejlesztésben a játékunk felhasználói felületének többnyelvűsítése. Jelenleg csak angol nyelven elérhető a játék, és bár a programozásban nagyon fontos az angol tudás, nyitni szeretnénk azok fele is, akiknek még nem megy magasabb szinten az angol nyelv, de szeretnék a programozás világába belekóstolni.

Ezek mellett a játékunkba beépíthető lenne egy pontozási rendszer is, amely segítségével a játékosok megoszthatnák milyen gyorsan sikerült elvégezniük a feladatokat. Ezzel bevezethetnénk egy versengést is a játékosok között, ez pedig pozitívan is befolyásolhatja tanulásukat.

Mivel a mai világban már mindenki zsebében található egy okostelefon, így biztosan lennének olyan játékosok is, akik egy hosszabb utazás során akár szeretnék tanulással eltölteni az idejüket, így a játék mobiltelefonokra való átfejlesztése is segítené, hogy nagyobb körben elterjedjen. Mindehhez a Unity egyszerű fejlesztési rendszerével nagyban hozzájárul.

Összefoglaló

A dolgozatom célja az volt, hogy ismertesse a játékmenet és szabályrendszer szerepét a 2D játékokban, valamint bemutassa egy saját játék fejlesztését.

A szakirodalom kutatásával bemutattam, hogy a játékmenet és szabályrendszer kulcsfontosságú szerepet játszanak a játékelmény és a játék fejlődése szempontjából. A játékmenet meghatározza a játékosok tevékenységeit és döntéseit, míg a szabályrendszer szabályokat és korlátokat állít fel, amelyekhez a játékosoknak alkalmazkodniuk kell. Ezáltal a játékmenet és szabályrendszer közvetlen hatással vannak a játékosok motivációjára, elkötelezettségére és szórakoztatottságára.

A saját játékunk fejlesztésének célja pedig az volt, hogy egy olyan játékot hozzunk létre, amely nem csak szórakoztató, de értékes tudással is felruházza a játékosokat, bevezetve őket a programozás világába. Emellett fontos szempont volt, hogy a játék könnyen használható legyen, és lehetővé tegye az elsajátított tudás gyakorlati alkalmazását.

Az implementáció során törekedtem a játékmenet és a szabályrendszer kiváló megvalósítására. Részletesen bemutatom a lépéseket, amelyeket megtettünk a játék kialakítása során, és azokat a kihívásokat, amelyekkel szembe kellett néznünk. Az eredményekről is beszámolok, amelyeket a fejlesztés folyamán elértünk.

Összességében, a szakdolgozatban bemutatott játékmenet és szabályrendszer implementációja részletesen ismerteti a 2D játékokban betöltött szerepüket. A játékunk lehetőséget teremt a programozás világának felfedezésére, miközben szórakoztató és interaktív élményt nyújt a játékosoknak. Reményeink szerint a játék sikeresen hozzájárul a programozási készségek fejlesztéséhez, és könnyen alkalmazható tudást ad a játékosok kezébe.

Végezetül ismertettem a játékunk jelenlegi limitációit, emellett pedig továbbfejlesztési lehetőségeket is feltártam dolgozatomban.

Köszönetnyilvánítás

Szeretnék mély tisztelettel és hálával köszönetet mondani mindazoknak, akik támogattak és segítettek engem diplomamunkám elkészítésében.

Első és legfontosabb köszönetemet szeretném kifejezni Osztián Pálma-Rozália egyetemi tanársegédnek, aki a diplomamunkám témavezetője volt. Kivételes szakmai tudása és fáradhatatlan elkötelezettsége révén vezetett végig ezen az izgalmas úton. Hozzáértése, értékes tanácsai és támogatása nélkül nem sikerülhetett volna ez a munka. Hálás vagyok az iránymutatásáért és a folyamatos bátorításáért.

Köszönetet szeretnék mondani az egyetem összes tanárának, biztatásuk és segítségük nélkül nem jutottam volna ilyen szintű szakmai tudáshoz.

Nem feledkezhetek meg a családom és barátaim támogatásáról sem. Köszönet nekik a türelemért, a biztatásért és az örökös hitért bennem.

Végül, köszönet illeti az összes olyan személyt, aki közvetlenül vagy közvetve hozzájárult a diplomamunkám elkészítéséhez.

Irodalomjegyzék

- [BS09] Brenda Brathwaite and Ian Schreiber. *Challenges for game designers*. Course Technology/Cengage Learning Boston, Massachusetts, 2009.
- [Ful14] Tracy Fullerton. *Game design workshop: a playcentric approach to creating innovative games*. CRC press, 2014.
- [Sch08] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC press, 2008.
- [Sic08] Miguel Sicart. Defining game mechanics. *Game studies*, 8(2):1–14, 2008.
- [Sil15] Daniel Silber. *Pixel art for game developers*. CRC Press, 2015.
- [ZHF⁺22] Tubagus Zufri, Dodi Hilman, Octavianus Frans, et al. Research on the application of pixel art in game character design. *Journal of Games, Game Art, and Gamification*, 7(1):27–31, 2022.