

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



**SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM**

Programozási készségeket fejlesztő játék

DIPLOMADOLGOZAT

Témavezető:
dr. Iclánzan David,
Egyetemi docens

Végzős hallgató:
Nagy Márton-Hunor

2023

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



**UNIVERSITATEA
SAPIENTIA**

Joc pentru dezvoltarea abilităților de programare

LUCRARE DE DIPLOMĂ

Coordonator științific:
dr. Iclănanz David,
Conferențiar universitar

Absolvent:
Nagy Márton-Hunor

2023

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



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

Game to develop programming skills

BACHELOR THESIS

Scientific advisor:
dr. Iclănanzán David,
Associate professor

Student:
Nagy Márton-Hunor

2023

LUCRARE DE DIPLOMĂ

Coordonator științific:
dr. Iclanțan David

Candidat: **Nagy Hunor**
Anul absolvirii: 2023

a) Tema lucrării de licență: Dezvoltarea unui joc de tip platformer 2D educațional, pentru dezvoltarea abilităților de programare.

b) Problemele principale tratate:

- Studiul bibliografic a conceptului, strategiilor și soluțiilor de gamification (eng. gamification).
- Studiul bibliografic a jocurilor educationale STEM cu accent pe soluțiile ce facilitează deprinderea cunoștințelor de programare.
- Proiectarea unui mecanism de navigare și a unor nivele de joc ce permit învățarea graduală a conceptelor predate la materia Programare I, Introducere în Programare.
- Proiectarea și realizarea jocului educațional.
- Evaluarea experienței utilizatorului și a impactului educațional cu ajutorul unor chestionare.
- Implementarea unui dashboard pentru procesarea răspunsurilor și vizualizarea rezultatelor.
- Documentarea adekvată a stadiilor de proiectare, implementare și testare a aplicațiilor.

c) Desene obligatorii:

- Schema bloc a sistemului
- Diagrame de proiectare, implementare și testare pentru aplicația software realizată.
- Captură de ecran din joc ce exemplifică dinamica jocului și componenta educațională și cum este ea rezolvată de către utilizator.
- Captură de ecran cu dashboardul de vizualizare.

d) Softuri obligatorii:

- Joc platformer 2D educațional.
- Script de analiză statistică și vizualizare a feedbackului primit de la utilizatori.

e) Bibliografia recomandată:

Elshiekh, R., & Butgerit, L. (2017). Using gamification to teach students programming concepts. *Open Access Library Journal*, 4(8), 1-7.

Simões, J., Redondo, R. D., & Vilas, A. F. (2013). A social gamification framework for a K-6 learning platform. *Computers in Human Behavior*, 29(2), 345-353.

Olsson, M., Mozelius, P., & Collin, J. (2015). Visualisation and Gamification of e-Learning and Programming Education. *Electronic journal of e-learning*, 13(6), pp452-465.

Rojas-López, A., Rincón-Flores, E. G., Mena, J., García-Peña, F. J., & Ramírez-Montoya, M. S. (2019). Engagement in the course of programming in higher education through the use of gamification. *Universal Access in the Information Society*, 18(3), 583-597.

Caponetto, I., Earp, J., & Ott, M. (2014, October). Gamification and education: A literature review. In *European Conference on Games Based Learning* (Vol. 1, p. 50). Academic Conferences International Limited.

Stott, A., & Neustaedter, C. (2013). Analysis of gamification in education. *Surrey, BC, Canada, 8(1)*, 36.

Majuri, J., Koivisto, J., & Hamari, J. (2018). Gamification of education and learning: A review of empirical literature. In *Proceedings of the 2nd international GamiFIN conference, GamiFIN 2018*. CEUR-WS.

Khaitova, N. F. (2021). History of gamification and its role in the educational process. *International Journal of Multicultural and Multireligious Understanding*, 8(5), 212-216.

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 413

Primit tema la data de: 25.04.2022

Termen de predare: 06.07.2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului
programului de studiu

Semnătura candidatului

Declarație

Subsemnatul/a Nagy Márton-Hunor, 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: 2023.06.13

Absolvent

Semnătura.....Nagy

Kivonat

A jelenlegi világban egyre fontosabb az informatika és a programozói tudás, hiszen a körülöttünk lévő gépek nagy részét már valamilyen szoftver irányítja, valamelyen szoftvert használunk kommunikáció, böngészés, dokumentumok szerkesztése, tárolása és megosztása, multimédiás tartalmak lejátszása, játékok játszása stb. szempontjából. Ezen szoftvereket emberek írják, és mivel szükség van egyre több szoftverfejlesztőre, fontos, hogy minél több személy számára elérhető legyen a programozás alapjainak megismerése. A projektem célja egy speciálisan tervezett játék létrehozása, amely segítségével a játékosok könnyedén fejleszthetik a programozáshoz szükséges logikus gondolkodást, problémamegoldó képességeket, valamint megismerhetik az alapszintű programozási műveleteket és folyamatokat.

A jelen dolgozatban arra keresem a választ, hogy hogyan lehet a legjobban, legkönnyebben és leg hatékonyabban megvalósítani a kezdők bevonását ebbe a világba a gamifikáció segítségével. A projekt másik célja az, hogy a haladóbb tudással rendelkező programozóknak is élvezetes, kihívásokkal teli legyen az alkalmazás. A megvalósított alkalmazásban a játékos megismerheti a folyamatok egymásutániságát, függvények hívását, azok kapcsolatát és sorrendjét, a ciklusok felépítését és működését, valamint a feltételutasítást. Fontos része a játéknak az egyszerűség és a minimalizmus, hogy könnyen játszható legyen.

A dolgozatban bemutatok néhány fontos játékfejlesztési elméletet, amelyek hozzásegítenek egy sikeres játék megalkotásához. Emellett szó esik különböző játékfejlesztésre alkalmas technológiákról, amelyeket érdemes használni, valamint a platformokról, amelyekre érdemes játékot fejleszteni.

Rezumat

În lumea de astăzi, competențele în domeniul IT și al programării sunt din ce în ce mai importante, deoarece mașinile din jurul nostru sunt acum controlate de un fel de software, un fel de software de comunicare, navigare, editare, stocare și partajare a documentelor, redarea de conținut multimedia, jocuri etc. Aceste programe software sunt utilizate de către oameni și cum este nevoie de tot mai mulți dezvoltatori de software, este important ca un număr cât mai mare de persoane să învețe elementele de bază ale programării. Scopul proiectului meu este de a crea un de a crea un joc special conceput care să permită jucătorilor să dezvolte cu ușurință gândirea logică și abilitățile de rezolvare a problemelor necesare pentru programare, și să învețe operațiile și procesele de bază ale programării. În cadrul acestei teze, încerc să răspund la întrebarea cum se poate realiza cel mai bine și cel mai simplu și mai eficient mod de a implica începătorii în această lume a gamificare. Un alt obiectiv al proiectului este acela de a oferi celor mai avansați programatori mai avansați să se bucure de aplicație și să o provoace. Subiectele implementate În aplicația implementată, jucătorul poate învăța despre secvențierea proceselor și apelurile de funcții, și succesiunea funcțiilor, structura și funcționarea buclelor, precum și condiți- și condiționalitatea. O parte importantă a jocului este simplitatea și minimalismul său, astfel încât să fie ușor de jucat ușor de jucat. În această lucrare, voi prezenta câteva teorii importante de dezvoltare a jocurilor care contribuie la ajuta la crearea unui joc de succes. Voi discuta, de asemenea, diferite abordări ale dezvoltării de jocuri tehnologiile care pot fi utilizate pentru dezvoltarea de jocuri și platformele care pot fi utilizate pentru a platforme pentru care se pot dezvolta jocuri.

Abstract

In today's world, IT and programming skills are increasingly important because the machines around us are now controlled by a kind of software, a kind of software for communicating, navigating, editing, storing and sharing documents. playing multimedia content, games, etc. These software programs are used by humans and as more and more software developers are needed, it is important that as many people as possible to learn the basics of programming. The aim of my project is to create a specially designed game that allows players to easily develop logical thinking and problem solving skills needed for programming, and learn the basic operations and processes of programming. In this thesis, I attempt to answer the question of how best and most easiest and most effective way to involve beginners in this world of gamification. Another goal of the project is to provide the most advanced more advanced programmers to enjoy and challenge the app. Topics implemented In the implemented app, the player can learn about process sequencing and function calls, and the sequence of functions, the structure and operation of loops, as well as the condi- and conditionality. An important part of the game is its simplicity and minimalism, so that it is easy to play easy to play. In this paper, I will present some important game development theories that contribute to help create a successful game. I will also discuss different approaches to game development technologies that can be used for game development and platforms that can be used to platforms for which games can be developed.

Tartalomjegyzék

1. Bevezető	1
2. Elméleti megalapozás	3
2.1. Játék fejlesztési elméletek	3
2.1.1. Flow-elmélet	3
2.1.2. Narratív elmélet	3
2.1.3. Játékos motiváció	4
2.1.4. Játékbalansz elmélete	4
2.1.5. Szabályok és konfliktusok elmélete	5
2.1.6. Szerződés-elmélet	5
2.1.7. Szociálpszichológiai elmélet	5
2.2. Szoftverfejlesztési módszertanok	6
2.2.1. Vízesés modell	6
2.2.2. Agile módszertanok	7
2.3. A gamifikáció és a tanulás kapcsolata	8
2.4. A programozás és a játékok	10
2.4.1. Lightbot	10
2.4.2. Robocode	11
2.4.3. Flexbox Froggy	12
2.4.4. CodeCombat	13
3. Technikai háttér	15
3.1. Videójáték történelem	15
3.2. A játékfejlesztési platformok	16
3.2.1. Konzol játékok	17
3.2.2. Arcade játékok	17
3.2.3. Személyi számítógépes játékok	17
3.2.4. Mobilos játékok	18
3.2.5. Virtualis valóság játékok	18
3.2.6. Cloud játékok	18
3.2.7. Handleod játékok	18
3.3. Az eszközök, programozási nyelvek és szoftverek	18
3.3.1. Unity	19
3.3.2. .NET	19
3.3.3. C#	19
3.3.4. Microsoft Visual Studio	19

3.3.5. Github	19
3.3.6. Egyéb technologiák	20
3.4. A játékfejlesztés lépései	20
3.5. A programozás oktatása	20
4. A rendszer specifikációja	22
4.1. Felhasználoi követelmények	22
4.2. Rendszerkövetelmények	23
4.2.1. Funkcionális követelmények	23
4.2.2. Nem funkcionális követelmények	24
5. A szoftver bemutatása	25
5.1. A játékmenet és a játék céljai	25
5.2. A nehézségi szintek és a fejlődési lehetőségek	25
5.3. Játék felépítésének	27
5.3.1. Karakter irányítása	27
5.3.2. Képernyők és pályák	30
6. Továbbfejlesztési lehetőségek	36
6.1. Programozási algoritmusok	36
6.2. Többjátékos mod	36
6.3. Több játékplatform támogatása	37
6.4. Közösség építés	37
Összefoglaló	38
Köszönetnyilvánítás	39
Ábrák jegyzéke	40
Irodalomjegyzék	43
Függelék	44
F.1. A TeXstudio felülete	44

1. fejezet

Bevezető

A modern technológia világában egyre fontosabbá válnak a programozási készségek. A programozás lehetővé teszi a szoftverek, alkalmazások és weboldalak fejlesztését, amelyek számos területen alkalmazhatóak, például a szórakoztatás, az üzleti élet, az oktatás, az egészségügy, a kommunikáció és a biztonság terén. Ennek fényében a programozási készségek rendkívül fontos szerepet játszanak és fognak játszani az emberek életében.

A programozási készségek közé tartozik a logikai, algoritmikus gondolkodás, az adatstruktúrák és algoritmusok ismerete, valamint a programozási nyelvek ismerete, és még sorolhatnánk.

A játékfejlesztés és az oktatás összekapcsolódása az utóbbi években egyre népszerűbbé válik [CEO14], mivel a játékok szórakoztató és interaktív módon lehetővé teszik az oktatási anyagok hatékonyabb megértését és elsajátítását. Az oktatási játékok lehetőséget biztosítanak a diákok számára, hogy az általuk tanult dolgokat játékos környezetben gyakorolják, így javítva a tanulási folyamat hatékonyságát és eredményességét. Ezen játékok segítségével számos téma-kört lehet lefedni, kezdve a programozástól a történelemig, a nyelvtanuláson át. A téma-körök sokszínűségéhez márten ugyanilyen sokszínű a játékok felhozatala is. Lehetnek akár online vagy offline, egyszemélyes vagy többjátékos módban is játszhatóak, és mindezek különböző típusúak lehetnek, mint például akció, kaland, ügynösségi, stratégiai, szerepjáték, és még sorolhatnánk. Ezenkívül rengeteg platformra lehetséges a megvalósításuk, ezzel is növelte elterjedésüket. A jelenséget gamifikációt nevezzük, amikor játékos formában próbálunk nehezebben érthető, unalmasabb információkat átadni, így növelve a produktivitást, motivációt és elkötelezettséget.

A játékfejlesztés [Rab05] lehetővé teszi, hogy saját videójátékot hozzunk létre. Ehhez számos különböző képességre van szükség, mint például a kreativitás, a tervezési és programozási képességek. A játékfejlesztés folyamata általában a tervezéstől kezdődik, amely során az ötletet rajzokon vagy modelleken keresztül dolgozzuk ki. A következő lépés az alkalmazásfejlesztés, amely időigényes és összetett folyamat, viszont a saját játék fejlesztése lehetőséget ad arra, hogy kreatív ötleteket valósítsunk meg, és olyan játéket alkossunk, amely másoknak élvezetet, kikapcsolódást, kihívásokat nyújthat.

Emellett a szakdolgozat a videójáték-ipar egyes aspektusainak elemzésére összponosít, és bemutatja a játékfejlesztőknek azokat a kihívásokat, amelyekkel szembe kell nézniük a játék tervezése és fejlesztése során. A cél az, hogy átfogó képet adjon a játékfejlesztés folyamatáról, a tervezéstől a megvalósításig. Emellett bemutatja az iparban használt eszközöket és technológiákat, valamint az aktuális trendeket és irányokat.

A játékfejlesztésnek számos pozitív hatása van. A játékok kreativitást és problémamegoldó képességet fejlesztenek, valamint ösztönzik a csapatmunkát és a kommunikációt, ha többjátékos módú játékokról van szó. Ezenkívül a játékok szórakoztató és motiváló élményt nyújtanak, ami segíti az embereket a kitartás és az elkötelezettség fenntartásában.

Összességében elmondható, hogy a játékfejlesztés és az oktatás összekapcsolódása lehetővé teszi az interaktív és hatékony tanulást, miközben élvezetet nyújt. A játékok által fejlesztett készségek és tudás pedig nagyban hozzájárulhatnak az emberek személyes és szakmai fejlődéséhez a modern technológia világában.

2. fejezet

Elméleti megalapozás

2.1. Játék fejlesztési elméletek

A játékfejlesztési elméletek egy széleskörű kutatási területet jelentenek, amelyek a játékok tervezésének, fejlesztésének és értékelésének alapjait és irányelveit vizsgálják. Az elméletek különböző megközelítéseket és modelleket kínálnak a játékélmény, a játékmechanikák, a játékosok motivációja és más releváns tényezők jobb megértésére.

2.1.1. Flow-elmélet

Ezt az elméletet [Lil21] Csíkszentmihályi Mihály dolgozta ki. Véleménye szerint bár milyen tevékenység okozhat "különleges" élményt az embernek, ami egyszerre vált ki sikereséget és megelégedettséget, és kíván belemerülést és koncentrációt. A Flow-élmény jellemzői közé tartozik a magas koncentráció, a teljes elmerülés és a tevékenység iránti intenzív érdeklődés. Ez az állapot egyfajta optimális teljesítmény- és élményállapot, amikor valaki egy tevékenységre teljes mértékben belemerül, élvez és magával ragadja. A Flow-élményben az egyén érzékeli, hogy képességei kihívásokkal egyensúlyban vannak, és a tevékenységhez szükséges készségei és erőfeszítései összhangban állnak a feladat nehézségével. Ez a "készség-kihívás egyensúly" az egyik kulcsfontosságú elem a Flow-élmény kialakulásához.

A Flow-állapotban levő egyén produktívabbá, kreatívabbá és elégedettebbé válhat. Emellett csökkentheti a szorongást és az idegeskedést, valamint javíthatja az egyén önértékelését és önbizalmát. A játékfejlesztésben a Flow-élmény elérése fontos szempont. A játéktervezőknek olyan játékélményeket kell teremteniük, amelyek lehetővé teszik a játékosok számára a Flow-állapotba való beleszeretést, ahol a játék kihívásai és a játékos képességei összhangban vannak, és a játékosok magas koncentrációban, elmerülve és érdeklődéssel játszanak.

2.1.2. Narratív elmélet

Ez az elmélet [Tam20] arra összpontosít, hogyan lehet egy játékot történettel meg tölni, és annak segítségével egy olyan élményt nyújtani, amely magával ragadja a játékosokat. A játékfejlesztőknek figyelembe kell venniük a történet és a játékmechanikák összhangját, hogy a játékosokat lebilincselő, érzelmi élményben részesítsék. A narratív el-

méletben fontos fogalom a narratív struktúra. Ez a történet szerkezetének elemeit, mint például a kezdet, középső rész és befejezés, illetve a karakterek, események és konfliktusok rendezését vizsgálja. A narratív struktúra segít a játékfejlesztőknek kialakítani a játék történetét, hogy érdekes és összetett legyen, és a játékosokat magával ragadja. A karakter fejlődés és azonosulás is fontos szerepet kap a narratív elméletben, ezáltal kialakul egy személyes azonosulás és érzelmi kötődés, amely segíti a játékélmény elmélyülését és a játékosok érdeklődésének fenntartását. A narratív elmélet fontos szerepet játszik a játékokban az interaktív narratíva terén. Az interaktív narratíva lehetővé teszi a játékosok számára, hogy befolyásolják a történet alakulását és kimenetelét a döntéseikkel és cselekedeteikkel. Ez a játékmenet és a történet közötti összefonódást eredményezi, amely fokozza a játékélményt és a játékosok bevonódását.

2.1.3. Játékos motiváció

Az egyik legismertebb elmélet a játékos motiváció terén a Self-Determination Theory (Önhatározatelmélet) [RRP06], amelyet Edward L. Deci és Richard M. Ryan dolgozott ki. Ez az elmélet azt állítja, hogy a motiváció három alapvető szükségleten alapul: az autonómia, a kompetencia és a kapcsolat. Az autonómiára vonatkozó szükséglet azt jelenti, hogy a játékosok szeretik, ha saját döntéseket hozhatnak, és befolyásolhatják a játékmenetet. A kompetenciaszükséglet azt jelenti, hogy a játékosok szeretnek kihívásokkal szembesülni, és fejleszthetik készségeiket és képességeiket a játék során. A kapcsolatszükséglet azt jelenti, hogy a játékosok értékelik a közösségi interakciókat, a csapatmunkát és a versenytársakhoz való viszonyulást.

A másik fontos elmélet Richard Bartle által kidolgozott játékostípusok elmélete [BLN⁺11], amely különböző típusú játékosokat kategorizál aszerint, hogy milyen motivációk hajtják őket. A játékostípusok közé tartoznak a teljesítésre törekvők, a felfedezők, a szociális játékosok és a pusztítók. Ez az elmélet segíti a fejlesztőket abban, hogy a játék tervezése során figyelembe vegyék a különböző játékostípusok preferenciáit és motivációjait. A játékos motiváció különböző elméleti megközelítéseket foglal magában, amelyek megpróbálják megmagyarázni, hogy mi motiválja a játékosokat, és miért élvezik a játékot.

2.1.4. Játékbalansz elmélete

Az elmélet [VLSVDHR10] azon kutatási területet jelenti, amely a játékokban a játékmenet és a játékelemek közötti egyensúlyra összpontosít. A játékbalansz kulcsfontosságú tényező a játékélmény minőségének és élvezetességének szempontjából. A játékbalansz arra törekszik, hogy elkerülje a túlzottan könnyű vagy túlzottan nehéz játékélményt, és biztosítsa a játékosok számára az optimális kihívást és sikerélményt. Az egyenlő versenyfeltételek érdekében biztosítani kell a karakterek, készségek vagy fegyverek közötti egyensúlyt, hogy a játékosok között ne legyenek túl erősek vagy túl gyengék elemek, ami így egyenlő versenyfeltételeket teremt. A játékbalansz kialakítása összetett folyamat, és több tényezőt kell figyelembe venni. A játéktervezőknek meg kell vizsgálniuk a játékosok játszási mintázatait, visszajelzéseit és reakcióit a játékra. A játékbalansz elmélete folyamatosan fejlődik és változik a játékfejlesztési iparágban.

2.1.5. Szabályok és konfliktusok elmélete

Az elmélet [Ber54] alkalmazása a játékfejlesztés területén segíthet a játékélmény és a játékmenet tervezésében. A játékokban a szabályok meghatározzák a játékosok viselkedését, az interakciókat és a célok elérésének módját. A konfliktusok pedig a játék izgalmas és kihívásokkal teli részét képezik.

A játékfejlesztés során fontos megérteni, hogy milyen szabályokat és konfliktusokat kell beépíteni a játékba annak érdekében, hogy izgalmas és élvezetes legyen. A játékoknak egyensúlyban kell tartaniuk a kihívásokat és az elérhető jutalmakat, hogy motiválják a játékosokat és fenntartsák az érdeklődésüket.

A szabályok és konfliktusok elemzése segíthet a játékfejlesztőknek abban, hogy megtervezzék a játékmenetet és az interakciókat úgy, hogy a játékosok számára kielégítő kihívásokat és élményeket nyújtsanak. Fontos megfontolni a játékbeli szabályok következetességét, egyensúlyát és érthetőségét, valamint a konfliktusok megfelelő nehézségi szintjét és diverzitását.

2.1.6. Szerződés-elmélet

Ez az elmélet [Cra85] alkalmazása a játékfejlesztés területén olyan eszközt kínál, amely segíthet a játékokban létrehozott virtuális világokban és gazdaságokban a játékosok közötti kölcsönhatások, tranzakciók és együttműködés szabályozásában.

A szerződés-elmélet a játékfejlesztésben számos területen hasznos lehet. Például a többszemélyes online játékokban a játékosok közötti kereskedelmi tevékenységek, a virtuális vagyon, tárgyak és erőforrások cseréje során a szerződés-elmélet lehetőséget ad a tranzakciók részleteinek és feltételeinek meghatározására.

A szerződés-elmélet segítségével a játékfejlesztők képesek lehetnek a játékosok közötti viselkedés szabályozására és a gazdasági egyensúly fenntartására. Ez lehetőséget ad a csalások, visszaélések és más jogosítések megelőzésére, valamint a fair és egyenlő feltételek biztosítására a játékosok között.

2.1.7. Szociálpszichológiai elmélet

Ez az elmélet [Mur78] alkalmazása játékfejlesztésben lehetőséget nyújtanak a játékélmény gazdagítására, a játékosok közötti interakciók és viselkedések megértésére, valamint a szociális dinamika hatékony kezelésére. A szociálpszichológiai elméletek figyelembe veszik az emberi viselkedés, motivációk és kapcsolatok alapvető tényezőit, és ezeket alkalmazzák a játékfejlesztési folyamat során.

Például a csoportdinamika elmélete és a közösséggépítési stratégiák segítségével a játékfejlesztők képesek lehetnek olyan játékelemeiket és funkciókat tervezni, amelyek elősegítik a közösségi élményt és a csapatmunkát. Ez lehetőséget ad a játékosoknak a kooperációra, versengésre vagy akár a szerepjátékra alapozott együttműködésre.

Az attribúciós elméletek alkalmazása a játékban segíthet a játékosok élményeinek megértésében és a sikер vagy kudarc magyarázataiban. Ez lehetőséget ad a játékfejlesztőknek arra, hogy megfelelő visszajelzést és motivációt biztosítsanak a játékosok számára, valamint az érzelmek és motivációk befolyásolásával javítsák a játékélményt.

2.2. Szoftverfejlesztési módszertanok

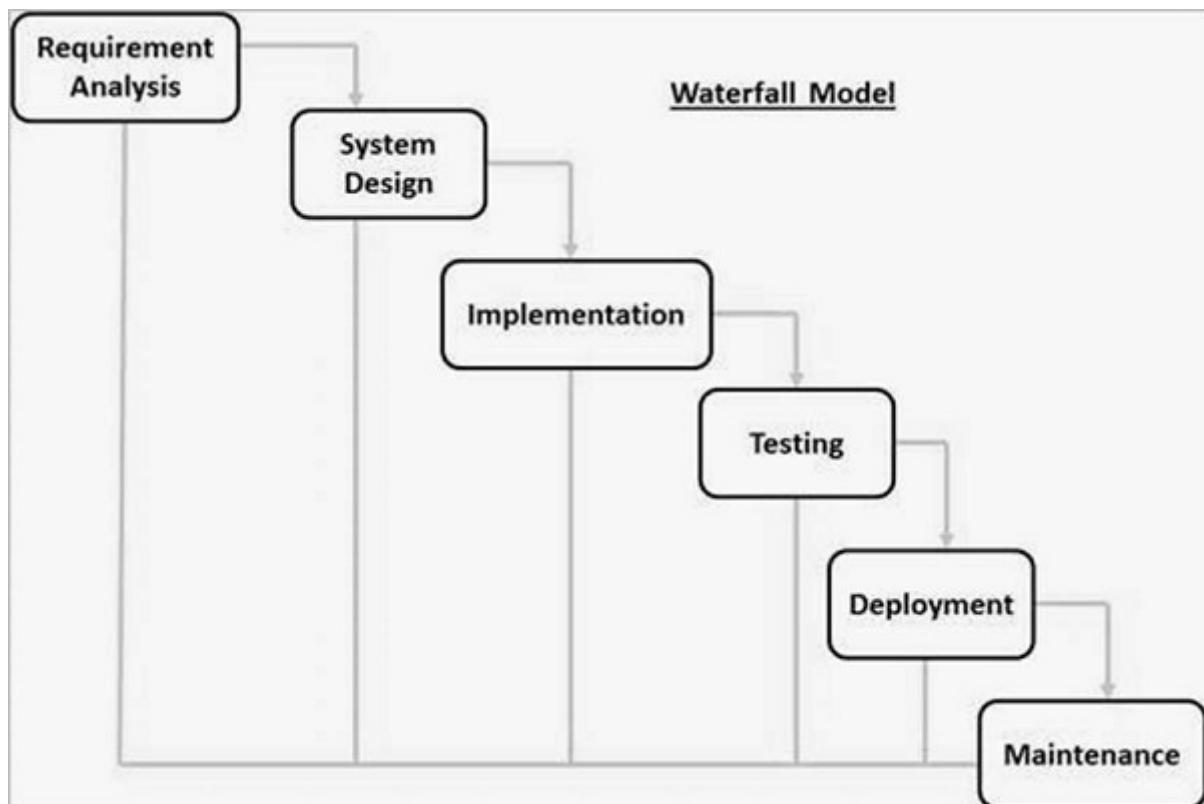
A módszertanok segítségével a fejlösstök hatékonyabban tudnak játékokat tervezni, készíteni és kiadni. A szoftverfolyamat modellje a szoftverfolyamat absztrakt reprezentációja, azaz a szoftverfolyamat egy bizonyos perspektívából adódó egyszerűsített leírása. minden modell más és más szempontból mutat be egy folyamatot. A modellek nem pontos, részletes leírásai a folyamatnak, csak nagyvonalú áttekintést adnak róla.

2.2.1. Vízesés modell

A szoftverfejlesztés történetének első publikált modellje, amelyben a fejlesztésben felmerülő tevékenységeket jól elválasztható lépésekben, lépcsősen ábrázolják, az a Vízesésmodell [BLA]. A Vízesés modell nagy hangsúlyt fektet a részletes dokumentációra. minden lépéshez részletesen dokumentálják a terveket, specifikációkat, tesztelési eljárásokat stb. Ez a dokumentáció segíti a projektmegvalósítást, az átadást és a karbantartást.

A Vízesés modell egyik hátránya, hogy nem rugalmas, a fejlesztés folyamata lineáris és előre meghatározott, így minden fázis csak akkor kezdődik el, ha az előző teljesül. Ez azt eredményezi, hogy ha a kezdeti követelmények vagy tervek hibásak vagy hiányosak, akkor ezeket csak későbbi fázisokban fedezik fel, amikor a korrekciók költségesebbek lehetnek.

Az alapvető lépések általában a következők: követelmények meghatározása, tervezés, implementáció, tesztelés, üzembe helyezés és karbantartás.



2.1. ábra. Vízesés modell folyamatai. Forrás: [[viz](#)]

2.2.2. Agile módszertanok

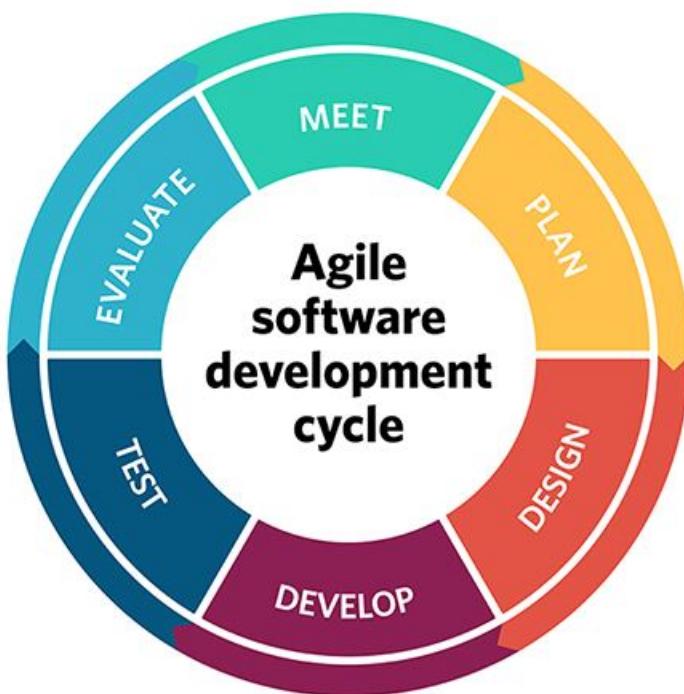
Napjainkban az egyik leggyakrabban használt módszertanok [BLA], mivel rugalmas és iteratív megközelítéseket jelentenek a szoftverfejlesztésnek, amelyek lehetővé teszik a gyors reagálást a változó követelményekre és a folyamatos visszacsatolást. Ezen módszertanok közül néhány népszerű és elterjedt változata a következő:

-Scrum: A Scrum a fejlesztési folyamatot iteratív, időboxolt időszakokra, "sprintekre" osztja, általában 1-4 hetes időtartammal. minden sprintben a fejlesztőcsapat olyan munkaelemeket választ ki és végrehajt, amelyek a legnagyobb értéket képviselik a vevő számára. A Scrumban a fejlesztőcsapat rendszeres napi találkozókat tart, visszajelzéseket kap a vevőtől, és az eredményeket rendszeresen értékeli a sprint vége felé.

-Kanban: A Kanban egy vizuális tábla alapú módszertan, amely lehetővé teszi a munkafolyamat átláthatóságát és az egyidejű feladatok korlátozását. A fejlesztőcsapat feladatai kártyákként vannak reprezentálva, amelyeket a táblán haladnak át a különböző fázisokban, például "Folyamatban", "Tesztelés alatt", "Kész", stb. A Kanban rendszeresen ellenőrzi a munkaterhelést és optimalizálja a folyamatot.

-Extreme Programming: Az Extreme Programming egy olyan módszertan, amely a fejlesztési folyamatot gyors ciklusokra, tesztelésre, folyamatos integrációra és páros programozásra alapozza.

-Lean Software Development: A Lean Software Development a lean gyártás filozófiájára épül, amely a hulladék minimalizálására, a folyamatos értékteremtésre és a hatékony erőforrásfelhasználásra összpontosít.



2.2. ábra. Agilis fejlesztés folyamatai. Forrás: [JOH23]

2.3. A gamifikáció és a tanulás kapcsolata



2.3. ábra. Gamifikáció. Forrás: [?]

A gamifikáció [Kha21] története a játékok és játékosság fogalmának mélyebb megértésével kezdődik. Az emberek évezredek óta játszanak különböző formában, legyen az az ősi társasjátékok, sportok vagy más szabadidős tevékenységek. A játékokban lévő motiváció, kihívás és ösztönzés vonzóvá tette az embereket, és a játékok természetesen felkeltették a fejlesztők és kutatók érdeklődését.

A gamifikáció fogalma azonban a 21. században vált népszerűvé. Az első említések a gamifikációról a 2000-es évek elején jelentek meg, amikor a digitális technológia és az internetes platformok elterjedtek. Az üzleti világban felmerült a kérdés, hogyan lehetne kihasználni a játékos motivációkat és mechanizmusokat más területeken, például oktatásban, egészségügyben vagy munkahelyeken.

A gamifikáció a játékok jellemzőit alkalmazza olyan területeken, mint a marketing, az oktatás vagy a munkahelyi produktivitás, hogy javítsa az emberek motivációját, elkötelezettségét és érdeklődését az adott tevékenységek iránt.

Az első gamifikált alkalmazások között voltak olyan weboldalak és alkalmazások, amelyek pontokat, ranglistákat és virtuális kitüntetéseket használtak a felhasználók motiválására. Az ilyen rendszerek célja az volt, hogy növeljék az interakciót, felhasználói hűséget és aktivitást.

Ahogy a gamifikáció elterjedt, a tervezők és kutatók jobban megértették az emberi motivációkat és a játéktervezés pszichológiáját. A gamifikáció már nem csak a pontok és

ranglisták egyszerű alkalmazása volt, hanem egy komplexebb tervezési megközelítést tett lehetővé. Az egyre fejlettebb technológiák és adatelemzési módszerek lehetővé tették a személyre szabottabb és dinamikusabb gamifikációs rendszerek kialakítását.

Az újabb kutatások, tervezési módszerek és technológiák folyamatosan fejlődnek a gamifikáció terén. Az mesterséges intelligencia és gépi tanulás technológiái lehetővé teszik a gamifikáció személyre szabottabbá tételeit, az adaptív rendszerek kialakítását és az elő visszajelzések optimalizálását. Az ilyen rendszerek képesek monitorozni és értékelni a felhasználói viselkedést, valamint az adatok alapján testre szabott jutalmakat és kihívásokat biztosítani.

Emellett az augmented reality (kiterjesztett valóság) és virtual reality (virtuális valóság) technológiák is új dimenziókat nyitnak meg a gamifikáció terén. Ez lehetővé teszi a felhasználók számára, hogy interaktív és élménydús játékélményt éljenek át, amely valós és virtuális elemeket kombinál.

A gamifikáció nemcsak az egyéni felhasználókat célozza, hanem társadalmi és közösségi szinten is hatékony lehet. Például a közösségi média platformok játékosított mechanizmusokat alkalmaznak, amelyek ösztönöznek az embereket a tartalom megosztására, interakcióra és közösséggépítésre. Ezenkívül a gamifikáció alkalmazása a fenntarthatóság terén is fontos lehet, hogy ösztönözze az embereket az energiahatékonyság, hulladékcsökkenést vagy környezetvédelem terén.

A programozás oktatása napjainkban egyre nagyobb hangsúlyt kap, hiszen az informatika területe folyamatosan fejlődik és bővül. Ebben a folyamatban kiemelt szerepet játszik a gamifikáció, amely a játékos tanulási módszerek bevezetését jelenti az oktatásba.

A gamifikáció lényege, hogy a játékokból ismert mechanizmusokat, elemeket és design technikákat alkalmazza olyan területeken, amelyek eredetileg nem játékos jellegűek, így a tanulás során is. A programozás oktatásában a gamifikáció alkalmazása nemcsak a tanulók motivációját növeli, hanem a tananyag könnyebb megértését is elősegíti. Mivel lehetővé teszi a diákoknak, hogy játékusan és interaktív modon szerezenek tudást.

A gamifikáció azonban nem csak a játékokat jelenti, hanem azokat az elemeket is magában foglalja, amelyek motiváló hatással lehetnek a diákokra. Például az eredmények, az elismerés, a pontok és a rangsorok kialakítása, a küldetések és a kihívások, amelyek megoldásához tudásra van szükség.

Anélkül, hogy a hibák negatív hatással lennének a tanulásukra vagy az érzelmeikre a gamifikáció lehetővé teszi a diákok számára, hogy szabadon kísérletezzék, hibákat tegyenek, és tanuljanak a sikertelenségekből.

A gamifikáció segítségével a programozás oktatásában új módszerekkel találkozhatunk, mint például a játék-alapú tanulás, a versenyek és a küldetések teljesítése. Ezek a megoldások segítenek abban, hogy a programozást ne csak mint száraz tudományt, hanem mint érdekes és izgalmas tevékenységet ismerjék meg a tanulók. Az informatika területén a gamifikáció alkalmazása mellett szól az is, hogy a játékok fejlesztése és az informatikai rendszerek tervezése között számos hasonlóság felfedezhető, így a játékos tanulási módszerek bevezetése valójában összhangban van a későbbi munka során elvárható készségekkel és kompetenciákkal.

Fontos megjegyezni, hogy a gamifikáció nem csak a játékot jelenti, hanem a játéktervezési elemek és mechanizmusok alkalmazását más kontextusokban. A sikeres gamifikáció megfelelő tervezést, megértést és célkitűzéseket igényel, hogy a felhasználók motivációját és elkötelezettségét valóban növelni lehessen.

2.4. A programozás és a játékok

A programozás és a játékok [CBD16] szorosan összekapcsolódnak, hiszen a játékok fejlesztése során rengeteg kódolás és programozás szükséges. A játékokat általában különböző programnyelveken írják, például C++, C#, Java, Python stb.

A programozás segítségével a játékfejlesztők megtervezhetik és megvalósíthatják a játékmenetet, a grafikát és a hanghatásokat. A játékfejlesztőknek a programozáson kívül ismeretekre van szükségük a játéktervezés, a grafika, a zene és a hanghatások területén is.

A játékok programozása összetett feladat, és a fejlesztés során rengeteg probléma adódhat, amelyeket meg kell oldani a játék működésének biztosítása érdekében. A játékfejlesztőknek figyelembe kell venniük a hardver korlátait, optimalizálniuk kell a kódot, és teszteket kell végezniük a játék megfelelő működése érdekében.

A programozás azonban nem csak a játékfejlesztők számára fontos, hanem a játékosok számára is lehetőséget nyújt arra, hogy egyedi játékokat hozzanak létre. Az utóbbi években egyre népszerűbbek lettek a játéképítő programok, amelyekkel a felhasználók könnyedén és egyszerűen készíthetnek saját játékokat, akár programozási ismeretek nélkül is.

Ezek az eszközök általában felhasználóbarát felülettel rendelkeznek, amely lehetővé teszi a játékterületek létrehozását, karakterek és tárgyak elhelyezését, a játékmenet szabályainak definiálását és még sok más funkció használatát. Így a játéképítő programok lehetővé teszik az emberek számára, hogy kreatív ötleteiket valóra váltsák, és saját játékvilágokat alkossanak, akár hobbi szinten vagy akár hivatásos játékfejlesztőként.

Ezek az eszközök gyakran tartalmaznak előre elkészített sablonokat és erőforráskat, mint például karaktermodellek, animációk, hanghatások stb., amelyeket könnyedén használhatnak a felhasználók. Így azok, akik nem rendelkeznek mélyebb programozási ismeretekkel is lehetőséget kapnak arra, hogy saját játékot készítsenek.

A játéképítő programok általában vizuális szerkesztőfelülettel rendelkeznek, ahol a felhasználók egyszerűen húzhatnak és ejthetnek elemeket a játékba, és beállíthatják a tulajdonságaikat. Ez lehetővé teszi a játékosok számára, hogy létrehozzanak egyedi pályákat, kalandokat vagy akár teljes játékélményeket, és megosztassák azokat másokkal.

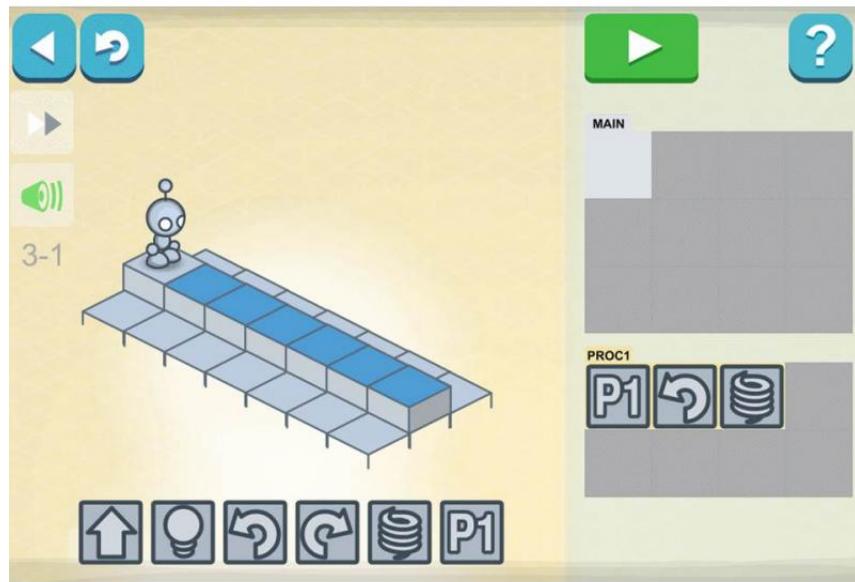
A játéképítő programok nemcsak a játékok létrehozását teszik könnyebbé, hanem lehetőséget adnak a játékosoknak ismeretek megszerzésére a programozás alapjairól. Néhány játéképítő eszköz lehetőséget nyújt a blokk alapú programozásra, ahol a felhasználók logikai blokkokat helyezhetnek el és kapcsolhatnak össze, hogy vezéreljék a játékbeli eseményeket és viselkedést. Ez segít megérteni a programozási logikát és elveket, miközben szórakoztató és interaktív módon alkotnak.

Az ilyen játéképítő eszközökkel a játékosok lépésről lépésre megtanulhatják a programozás alapjait, és fokozatosan növelhetik a képességeiket. Ezáltal felkeltik érdeklődésüket a programozás iránt, és inspirációt adnak a további tanuláshoz és kreatív projektjeikhez.

2.4.1. Lightbot

Jelenleg több olyan játék is van amelyik a gamifikáció segítségével probálja a programozást tanitani. Az én játékomat a Lightbot [Yar14] nevű játék inspirálta amelynek célja a programozási alapelvek és logikai gondolkodás megismertetése gyerekekkel és kezdő

programozókkal. A játékban a játékosnak irányítania kell egy kis robotot, hogy megoldjon különböző feladványokat és teljesítse a szinteket.



2.4. ábra. Lightbot. Forrás: [lig]

A Lightbot játékban a játékosnak parancsokat kell kiadnia a robotnak, hogy az elvégezze a feladatokat. A parancsok közé tartozhatnak a robot mozgatása előre, hátra, jobbra vagy balra, valamint speciális parancsok, mint például a lámpák be- vagy kikapcsolása. A játékosnak stratégiákat kell kidolgozni, terveznie kell a parancssorozatokat, és meg kell találnia a legoptimálisabb megoldásokat a feladványokra.

A Lightbot játék során a játékos fokozatosan tanulja meg a programozási fogalmakat, például a vezérlési szerkezeteket, ciklusokat és feltételes utasításokat. Az alapvető logikai gondolkodásra és problémamegoldó képességekre is épít, miközben interaktív és szórakoztatónak környezetben fejleszti a programozási készségeket.

A Lightbot játék különböző nehézségi szinteket és feladványokat kínál, amelyek fokozatosan növelik a kihívást és az igényességet. Ez lehetővé teszi a játékosok számára, hogy lépésről lépésre tanuljanak és gyakoroljanak, miközben egyre összetettebb programozási feladatokat oldanak meg.

A Lightbot játék nemcsak szórakoztatónak, de pedagógiai értékkel is rendelkezik. Segít a logikai gondolkodás, a problémamegoldás és a programozási készségek fejlesztésében. Emellett támogatja a kreativitást és az innovatív gondolkodást a játékosokban.

2.4.2. Robocode

A Robocode [Har04] egy izgalmas programozós játék és keretrendszer, amelyben a játékosok saját virtuális robotokat, vagy "botokat" tervezhetnek és irányíthatnak. A cél az, hogy ezeket a robotokat programozással olyan módon vezessük, hogy legyőzzék az ellenfeleket egy szimulált csatárában.

A játékot Java vagy C# nyelven írt kód segítségével lehet irányítani. A botoknak különböző képességeik vannak, például mozgás, célpontok kiválasztása, támadások végrehajtása és különböző stratégiák alkalmazása. A játékosoknak feladata, hogy minél okossabb és hatékonyabb botokat hozzanak létre, amelyek képesek navigálni a csatáren, elkerülni a lövedékeket, támadni az ellenfeleket és túlélni a csatákat.

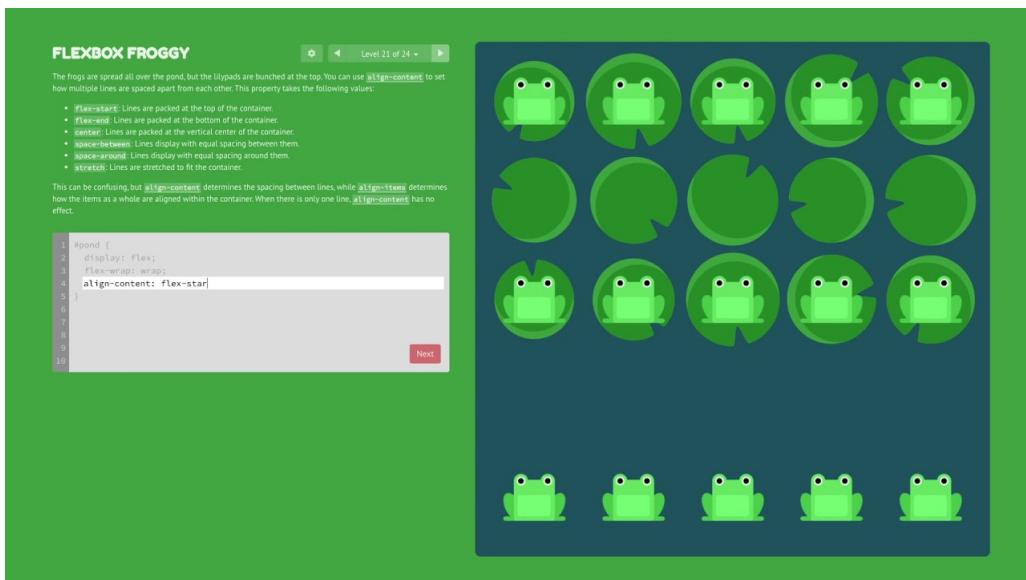
A csatáter egy kétdimenziós rácsos környezet, ahol a botok egymás ellen küzdenek. A botok rendelkeznek radarokkal, amelyekkel érzékelik az ellenfeleket, valamint fegyverekkel, amelyekkel lövedékeket tudnak kilőni. A játék során a botok folyamatosan kommunikálnak egymással, hogy információkat cseréljenek, például az ellenfelek helyzetéről vagy az észlelt veszélyekről.

A Robocode nemcsak egy játék, hanem egy oktatási eszköz is. A játék lehetővé teszi a programozási és mesterséges intelligencia koncepcióinak gyakorlati alkalmazását. Kezdőknek segít megérteni a programozás alapjait, míg haladóknak lehetőséget ad különböző algoritmusok és stratégiák kipróbálására.

A Robocode-ban versenyeket is lehet rendezni, ahol a játékosok botjaikat küldik csatába egymás ellen. Ezek a versenyek lehetnek online vagy helyi rendezvények, és különböző kategóriákban vagy korlátozásokkal rendelkezhetnek.

2.4.3. Flexbox Froggy

A Flexbox Froggy [Stu23a] egy interaktív játék, amely segít megismerni és gyakorolni a CSS flexbox koncepcióit. A játék online elérhető és ingyenesen játszható a böngészőben.



2.5. ábra. Flexbox Froggy. Forrás: [fle]

A játék azzal a feladattal indul, hogy el kell helyezni egy vagy több béka ikont egy liliomra a játéktéren. A béka ikonokat a flexbox tulajdonságok segítségével kell pozicionálni és rendezni a liliomon belül.

A játék során a játékosnak különböző feladatokat kell megoldania, amelyekben a flexbox különböző tulajdonságait kell alkalmaznia, például a justify-content, align-items,

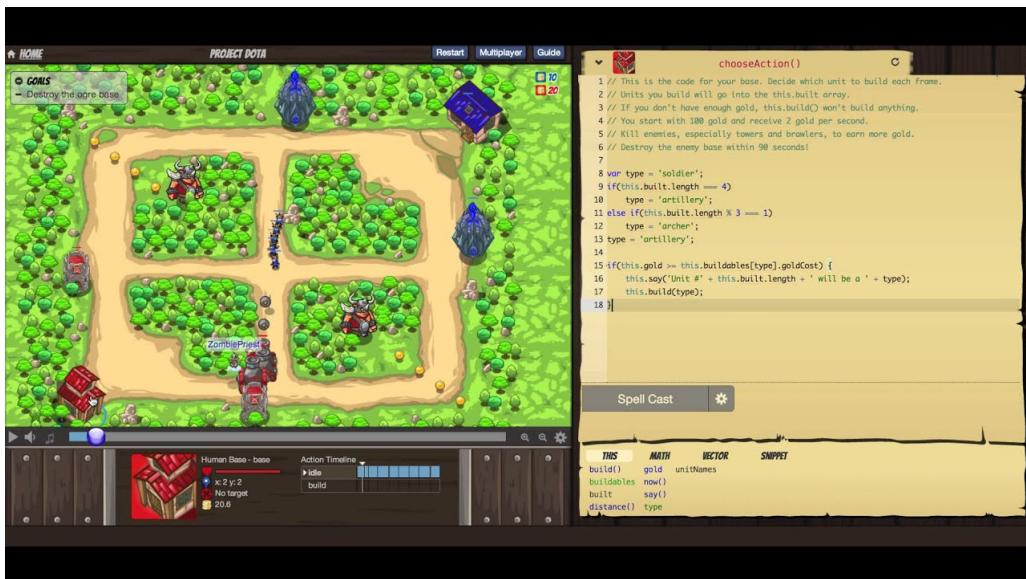
flex-direction, stb. A feladatok során a játékosnak úgy kell elhelyeznie a béka ikonokat, hogy a megfelelő sorrendben és elrendezésben legyenek a liliomon.

A Flexbox Froggy egy interaktív és vizuális módja a flexbox koncepcióinak tanulásának. A játék segít megérteni, hogyan működik a flexbox és hogyan lehet rugalmasan elrendezni és pozícionálni elemeket a weboldalon. A játék egyre nehezebb feladatokkal halad előre, így lehetőséget ad a gyakorlásra és elmélyítésre.

A játékot könnyen kezelhető felület jellemzi, ahol a játékos egyszerűen húzhatja és rendezheti az ikonokat a liliomon belül. Emellett a játék részletesen magyarázza el a flexbox tulajdonságait és hogyan kell őket használni.

2.4.4. CodeCombat

A CodeCombat [Stu23b] egy programozást oktató játék, amely lehetővé teszi a játékosoknak, hogy játék közben sajátítsák el a programozás alapjait. Ez egy interaktív platform, amely kalandjátékként van felépítve, és a játékosoknak különböző kihívásokat kell megoldaniuk a kód írásával.



2.6. ábra. CodeCombat

A játékban a játékosok egy fantasztikus világban találják magukat, ahol különböző karakterekkel és ellenfelekkel találkoznak. A cél az, hogy a karaktereket irányítsák, és megoldják a feladatokat a megfelelő programozási kódok segítségével. A játékosok különböző programozási nyelveket használhatnak, mint például Python, JavaScript, vagy CoffeeScript.

A CodeCombat játékélménye interaktív és dinamikus. A játékosoknak számos kihívással kell szembenézniük, például akadályokat átugrani, ellenfeleket legyőzni, aranyat gyűjteni és tárgyat használni a kaland során. Mindezt a megfelelő programozási logika és algoritmusok alkalmazásával kell megtenni.

A játék fokozatosan vezeti be a programozási fogalmakat és technikákat, így kezdők és haladók is megtalálhatják a számukra megfelelő kihívásokat. A CodeCombat egyedül-

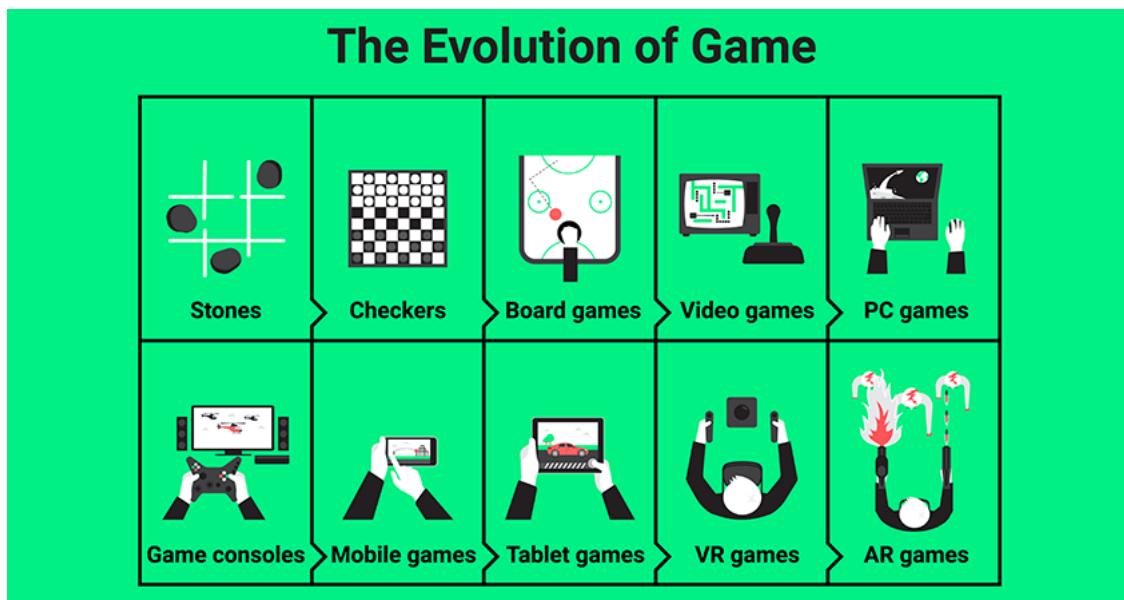
állóan kombinálja a játékélményt és a valós idejű programozási gyakorlást, amely segít a játékosoknak fejleszteni a problémamegoldó és kritikus gondolkodási képességeiket.

A játék számos pályát és küldetést kínál, amelyek közül néhány speciális témákat is feldolgoz, mint például mesterséges intelligencia, adatstruktúrák vagy robotika. Emellett a játék közösségenek része lehet, ahol a játékosok megoszthatják kódjaikat, tanácsokat és tapasztalatokat cserélhetnek egymással.

3. fejezet

Technikai háttér

3.1. Videojáték történelem



3.1. ábra. Játék fejlődés. Forrás:[gam]

A videojátékok hosszú utat tettek meg napjainkig, és folyamatosan fejlődtek a technológiával együtt. A történelem során olyan mérföldkövek jelentek meg, amelyek forradalmasították a szórakoztatónapot és az emberek szabadidejét. Az elsőként számon tartott videojáték 1947-ben jelent meg Cathode Ray Tube Amusement Device [Ivo15a], amelyet a második világháborúban használt radar kijelzők ihlettek. Az első helyezés bizonyos szempontok miatt viszont kétségbe vonható, mivel a játék nem használ semmilyen számítógép-generált grafikát, sem programot, illetve a létrehozásához sem volt szükség számítógép- vagy memóriaegységre. Így az első helyért versenyezik még az OXO játék [Ivo15b] is, amely az amőba játék megfelelője.

Nemsokkal később kezdetét vette a korai árkád játékkonzolok generációja, amelyet 1971-1977-es évek foglalnak magukba, és amely korszak első áttörő játéka volt a Pong [Low09]. 1977-től megjelentek a második generációs konszolok, amelyekben újdonságnak

számított az általános célú mikroprocesszor és a játékkazetta. Az ekkori három fő konzol az Atari 2600, az Intellivision és a ColecoVision voltak.

Az idő műlásával egyre szerteágazóbbá vált a játékipar, rengeteg új műfaj jött létre, például a kalandjátékok, szerepjátékok, stratégiajátékok, szimulátorok és még sorolhatnám. A hardverek fejlődése is két fő részre szakadt. Megjelentek ugyanis a személyi számítógépek, amelyek egészen más ütemben fejlődtek, mint a konzolok. A konzolok világában folyamatosan jöttek ki az új generációk, amelyekben minden volt egy kis újítás. Harmadik generáció: 8-bites játékok, negyedik generáció: 16-bites játékok, ötödik generáció: 32-64-bites játékok, illetve megjelent a 3D technológia és a CD-ROM is. Jelenleg a nyolcadik generációtól tartunk a konzolvilágban, és valószínűleg nem ez lesz az utolsó. A személyi számítógépek felépítésükön kifolyólag nehéz volna generációkra bontani a fejlődésüket, miután az egyes alkatrészek egymástól függetlenül, folyamatosan fejlődtek.

A mobiltelefonos játékok története az 1990-es évek elején kezdődött, amikor az előző egyszerű játékokat kezdték készíteni a kis képernyős mobiltelefonokra. Az 1997-ben megjelent Snake [PS06] játék volt az első népszerű mobiltelefonos játék, amelyet a Nokia gyártócége készített.

A 2010-es években a mobiljátékok már olyan nagyszabású és komplex játékokká váltak, amelyek versenyezni tudtak a hagyományos konzol- és számítógépes játékokkal. Az okostelefonok kijelzőjének növekedése, a processzorok és a grafikus chippek javulása lehetővé tette, hogy a játékfejlesztők olyan nagyobb terjedelmű és komolyabb grafikával rendelkező játékokat készítsenek, mint például az Asphalt, a PUBG, a Fortnite vagy a Minecraft.

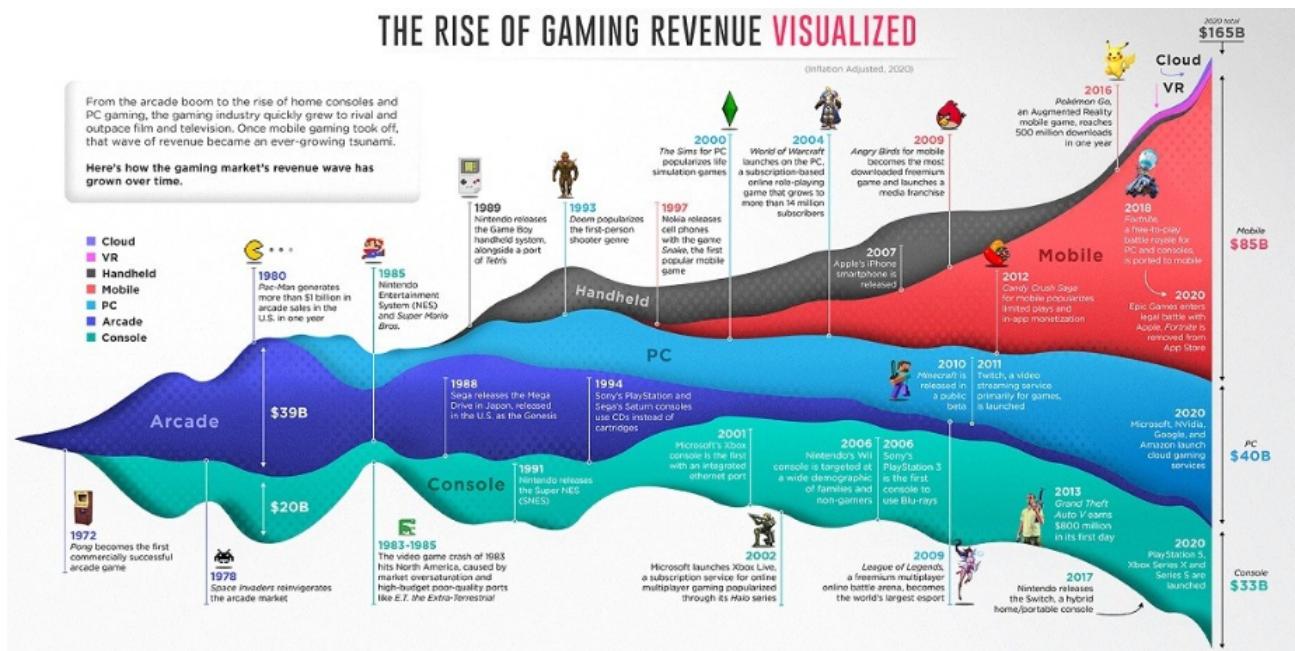
Mára ez az ágazat a videojátékipar legnagyobbja lett, köszönhetően a mobiltelefonok széles körű elterjedésének. Megfigyelhetünk azonban trendeket és technológiai szemléletváltozásokat, amelyek kihatnak napjaink rendszereire és játékaira egyaránt.

3.2. A játékfejlesztési platformok

A videojáték-ipar [vga], mint neve mutatja, a videojátékok fejlesztésével, marketingjével és eladásával foglalkozó gazdasági szektor. Önmagában véve is számtalan munkakört foglal magába, és emberek ezreinek ad munkát világszerte. Az évek során rengeteg eszköz jelent meg, amelyekre különböző típusú játékokat terveztek. Vannak eszközök, amelyek csak videojátékozásra készültek, illetve vannak, amelyeknek elsődleges funkciójuk nem az volt, de könnyedén lehet őket játékra is használni. Az alábbi ábra jól vizualizálja, hogyan változtak az évek során a videojáték felhasználási területei, hogyan alakultak ki, illetve hogyan vesztek feledésbe bizonyos játékfelhasználási területek.

Ez az ábra áttekintést nyújt arról, hogy változtak a videojátékok használati módjai az évek során. Korai időszakokban, például az 1980-as években, főként otthoni konzolokon és számítógépeken játszottak. Ezt követően megjelentek a hordozható játékkonzolok, amelyek lehetővé tették a játékot útközben is. A mobiltelefonok fejlődésével pedig megjelentek a mobiljátékok, amelyek ma már nagyon népszerűek. Emellett számos más platform létezik, mint például a VR (virtuális valóság) és az AR (kiterjesztett valóság), amelyek új dimenziókat adnak a játékélménynek.

Fontos megjegyezni, hogy a videojáték-ipar rendkívül dinamikus és gyorsan változó. Technológiai újítások és a felhasználói igények folyamatosan alakítják az ipart, és új lehetőségeket teremtenek a játékfejlesztők és a játékosok számára egyaránt. A folyamatos



3.2. ábra. A játékfejlesztési platformok változása. Forrás:[[jpv](#)]

fejlődés és az új trendek követése elengedhetetlen ahhoz, hogy az iparban versenyképes maradjon az egyéni fejlesztők és a nagyobb játékstúdiók egyaránt.

3.2.1. Konzol játékok

A videojáték-konzol olyan elektronikus eszköz, amely videojelet vagy képet ad ki egy videojáték megjelenítéséhez, amelyet egy játékvezérlővel lehet játszani. Ezen konzonzolok használatához tehát elengedhetetlen egy megjelenítő eszközhöz, tévéhez való csatlakoztatás illetve egy játékvezérlő. A videojáték-konzolok az othoni számitogépek egy szpeciális a videojátékozásra szánt formái amelyekből hiányzik a testre szabhatóság és a nyers számítási teljesítmény cserébe viszont megfizethetőb.

3.2.2. Arcade játékok

Az arcade videojátékok a játéktermi videojátékokat fedik le. A legtöbb játéktermi videojáték érmével működik, játéktermi szekrényben van elhelyezve, és a játéktermekben más típusú játéktermi játékokkal együtt található. Az 1990-es évek végéig az árkád videojátékok voltak a videojáték ipar legnagyobb és legfejlettebb technológiai szempontból fejlett szegmense. Napjainkban már csak Japánban, Kinában és Koreában erős a játéktermek ipara.

3.2.3. Személyi számitogépes játékok

A személyi számitogépes játékok ahogy azt a neve is mutatja olyan játékok amelyeket egy személyi számítógép segítségével lehet játszani. Ezen számitogépeket jellemzője, hogy tetszés szerint lehet fejleszteni a hardvert és szoftvert is így optimalizálva az adott

játékhoz. Az othoni számítogépes játékok az 1983-as videojátek-krachot váltak népszerűvé. 2020-as felmérések szerint a PC játekszektor volt a második legnagyobb kategória az összes platformon közül, mintegy 1,75 milliárd játékossal.

3.2.4. Mobilos játékok

A mobilos játékok olyan játékok, amelyeket jellemzően mobil telefonon játszanak. Napjainkban a mobil telefon a leg elterjedtebb digitális eszköz amely a legtöbb ember számára már elérhető söt szükség szerű. 2022-ben mintegy 6,64 milliárd okeos telefon felhasználó volt világ szerte. Ezen adatok fényében érthető, hogy egy ennyire elterjedt eszköz hatalmas piacot jelent a mobilos játékoknak is. Legelső mobilos játék 1994-ben jelent meg és azota ez a szektor lett a videójáték első számu kategoriája, világszerte mintegy 2,2 milliárd aktiv mobil játékossal.

3.2.5. Virtualis valóság játékok

A virtuális valóság játék vagy VR-játék a virtuális valóság (VR) hardverén játszott videojáték. A legtöbb VR-játék a játékos elmerülésén alapul, jellemzően fejre szerelt kijelzőegység vagy headset és egy vagy több kontroller segítségével. A headset jellemzően két sztereoszkópikus kijelzőt biztosít a felhasználó szeme előtt a 3D térszimulálása érdekében. Az első fogyasztói VR-termék, az Oculus Rift 2013-as megjelenésével bár nem lett széles körben használt a VR technologia, azota is jelennek meg újabb termékek.

3.2.6. Cloud játékok

A felhőalapú játék, az online játék egy olyan típusa, amely videojátékokat futtat távoli szervereken, és közvetlenül a felhasználó eszközére streameli azokat, vagy köznyelvben a játék távoli lejátszása a felhőből. Szemben áll a hagyományos játékokkal, amelyeknél a játék helyben fut a felhasználó videojáték-konzolján, személyi számítógépen vagy mobil-eszközén.

3.2.7. Handleod játékok

A kézi játékkonzol, vagy egyszerűen csak kézi konzol egy kicsi, hordozható, önálló videojáték-konzol, beépített képernyővel, játékvezérlővel és hangszórókkal. A kézi játékkonzolok kisebbek, mint az otthoni videojáték-konzolok, és egy egységen tartalmazzák a konzolt, a képernyőt, a hangszórókat és a vezérlőket, így az emberek magukkal vihetik és bármikor és bárhol játszhatnak velük.

3.3. Az eszközök, programozási nyelvek és szoftverek

A játékfejlesztéshez számos eszköz, programozási nyelv és szoftver áll rendelkezésre, amelyek lehetővé teszik a játékok megtervezését, fejlesztését és tesztelését. Az alábbiakban néhány ilyen eszkösről és nyelveről lesz szó.

3.3.1. Unity

A Unity [Uni23] egy videójáték-motor, amelyet a Unity Technologies fejleszt. A Unity segítségével háromdimenziós illetve kétdimenziós videójátékokat, ezen kívül egyéb interaktív jellegű tartalmakat lehet létrehozni, például építészeti látványterveket vagy valós idejű háromdimenziós animációkat. Többek között előnye, hogy a szoftver képes nagyméretű adatbázisokat kezelní, kihasználni a kölcsönhatások és animációk képességeit, előre kiszámított vagy valós idejű világítást biztosítani. Továbbá használható geometriai eszközcsomagok továbbítására, illetve viselkedési elemek hozzáadására egyes objektumokhoz. Az Unity lehetővé teszi a különböző platformokra (pl. Windows, macOS, Android, iOS stb.) történő exportálást, és számos beépített funkcióval rendelkezik, mint például az AI, a fizika, a hang és a grafika.

3.3.2. .NET

A .NET keretrendszer [.NE23] (kiejtése: "dot net") a Microsoft által kifejlesztett saját szoftver keretrendszer, amely elsősorban a Microsoft Windows rendszeren fut. Tartalmaz egy nagy osztálykönyvtárat, az úgynevezett Framework Class Library-t (FCL), és több programozási nyelven keresztül biztosítja a nyelvi interoperabilitást. A .NET keretrendszerre írt programok egy szoftveres környezetben, a Common Language Runtime (CLR) nevű környezetben futnak. A CLR egy olyan virtuális gép, amely olyan szolgáltatásokat nyújt, mint a biztonság, a memória kezelés és a kivételkezelés. Az FCL és a CLR együtt alkotja a .NET keretrendszert.

3.3.3. C#

A C# nyelv [C23] a C programozási nyelvből fejlődött ki, amit a Microsoft végzett, a .Net keretrendszer által. Alapja a C++ és Java programozási nyelvek. A felhasználói felületet valósítottam meg általa, ugyanis kényelmes és könnyen kezelhető. Egy Windows Forms App-ot létrehozva a dizájn részt könnyedén meg lehet valósítani. Mindazonáltal pedig a C++ programmal való összekötés se bizonyult lehetetlen feladatnak.

3.3.4. Microsoft Visual Studio

A Microsoft Visual Studio [Stu23c] egy a Microsoft által fejlesztett, ingyenesen letölthető alkalmazás, mindezt Windows, Linux és macOS felületekre. Számtalan lehetőséget kínál a fejlesztők részére, mint a hibakeresés, szintaxis kiemelés, kód kompilálás és beépített verziókövetés – Git. Lényegében ebben a környezetben fejlesztettem a teljes softwaret. Továbbá azért itt valósult meg minden, mert könnyedén kompilálható és futtatható a kód.

3.3.5. Github

A GitLab [Git23] egy olyan web alapú verziókövető rendszer, amely fejlesztés során egy Git alapú adattárat nyújt a fejlesztők számára. Mindazonáltal, hogy egy központi adattároló, lehetőség adódik akár a párhuzamos fejlesztésre is.

3.3.6. Egyéb technologiák

A szoftverem megvalósításához nem használtam ezen technologiákat de érdemes megemlíteni ezeket is. Unreal Engine, GameMaker Studio, Construct, C++, Java, Python, Adobe Photoshop

3.4. A játékfejlesztés lépései

A játéktervezés és programozás alapelvei fontosak a sikeres játékfejlesztéshez. Ezek közé tartozik többek között:

-Felhasználói élmény: A játékosok élményének javítása az elsődleges célja a játéktervezésnek és a programozásnak. A játékfejlesztőknek meg kell találniuk a megfelelő egyensúlyt a kihívás és a szórakozás között, hogy a játékosoknak élvezetes legyen a játék.

-Játékmenet: A játékmenet meghatározása és megtervezése kulcsfontosságú a játék sikeréhez. A játékmenetnek logikusnak és érthetőnek kell lennie, hogy a játékosok könnyen megtanulhassák a játékot és élvezhessék annak kihívásait.

-Hang és zene: A hanghatások és a zene fontos szerepet játszanak a játékosok élményében. A hangok és a zenei kompozíciók segítenek a játékosoknak érzelmi kapcsolatot kialakítani a játékkal és hozzájárulnak a hangulat megteremtéséhez.

-Grafika: A játékok grafikai megjelenése fontos a játékosok számára. A játékgrafikának lenyűgözőnek és vonzónak kell lennie, hogy felkeltse a játékosok érdeklődését és bevonja őket a játékba.

-Programozás: A programozás az alapja a játékfejlesztésnek. A programozás során a játékfejlesztők megvalósítják a játékmenetet, az AI-t, a grafikát és a hanghatásokat. A játékfejlesztőknek több programozási nyelvet kell ismerniük, hogy hatékonyan fejleszhessék a játékot.

-Tesztelés: A tesztelés kulcsfontosságú a játékfejlesztés során. A tesztelés során a játékfejlesztők különböző tesztekkel ellenőrzik a játék működését és megbizonyosodnak arról, hogy a játék megfelelően működik és nincsenek hibák benne.

3.5. A programozás oktatása

A programozási oktatás [RRR03] az utóbbi években egyre fontosabbá vált, mivel az informatika és a technológiai iparágak folyamatosan fejlődnek és bővülnek. A programozási oktatás célja, hogy az emberek megtanulják a kódolás és a számítógépes programozás alapjait, hogy képesek legyenek tervezni, fejleszteni és megvalósítani különböző szoftvereket és alkalmazásokat. Az alapvető programozási ismeretek megszerzése és a programozás alapjainak megértése fontos ahhoz, hogy valaki hatékonyan tudjon fejleszteni szoftvereket és alkalmazásokat. A programozás alapjai lehetővé teszik a problémamegoldó gondolkodást, a hatékony kódírást és a minőségi szoftverfejlesztést. Ez a tudás alapot biztosít a továbbfejlődéshez és az új technológiák megértéséhez a programozás terén. A programozás alapjait a következő területek foglalják magukba:

-Algoritmusok és adatszerkezetek: Az algoritmusok a lépések sorozatai, amelyeket a számítógép követ, hogy megoldja a problémákat. Az adatszerkezetek olyan módszerek és eszközök, amelyek segítik az adatok tárolását, rendezését és kezelését a programban. A

programozás alapjaival kapcsolatos első lépés az algoritmusok és adatszerkezetek megértése és használata.

-Programozási nyelvek: A programozási nyelvek olyan formális nyelvek, amelyeket a számítógépek értelmezni tudnak. Az alapvető programozási nyelvek közé tartozik a C, C++, Java, Python, JavaScript stb. A programozás alapjai közé tartozik a programozási nyelvek szintaxisának és szerkezetének megértése, valamint a megfelelő nyelv kiválasztása a probléma megoldásához.

-Változók és adattípusok: A változók olyan nevek, amelyeket az adatok tárolására használunk a programban. Az adattípusok meghatározzák az adatok jellegét és tárolási módját, például számok, szövegek, logikai értékek stb. A programozás alapjai közé tartozik a változók deklarálása, értékadása és az adattípusok kezelése.

-Vezérlési szerkezetek: A vezérlési szerkezetek segítenek a program folyamatának irányításában és vezérlésében. Ilyenek például a feltételes utasítások (pl. if-else, switch), ciklusok (pl. for, while) és elágazások (pl. break, continue). A programozás alapjai közé tartozik a vezérlési szerkezetek megértése és helyes használata.

-Függvények és modulok: A függvények olyan blokkok, amelyek specifikus feladatakat hajtanak végre, és újrafelhasználhatók a programban. A modulok olyan fájlok vagy kötegek, amelyekben függvények és más programozási elemek vannak csoporthoz. A programozás alapjai közé tartozik a függvények létrehozása, meghívása és az adatok átadása közöttük.

-Hibakeresés és hibakezelés: A programozás során gyakran előfordulhatnak hibák. A programozás alapjai közé tartozik a hibák azonosítása, hibakeresés és a megfelelő hibakezelési technikák alkalmazása. Ez magában foglalhatja hibakódok kezelését, kivételek használatát vagy hibajelentések generálását.

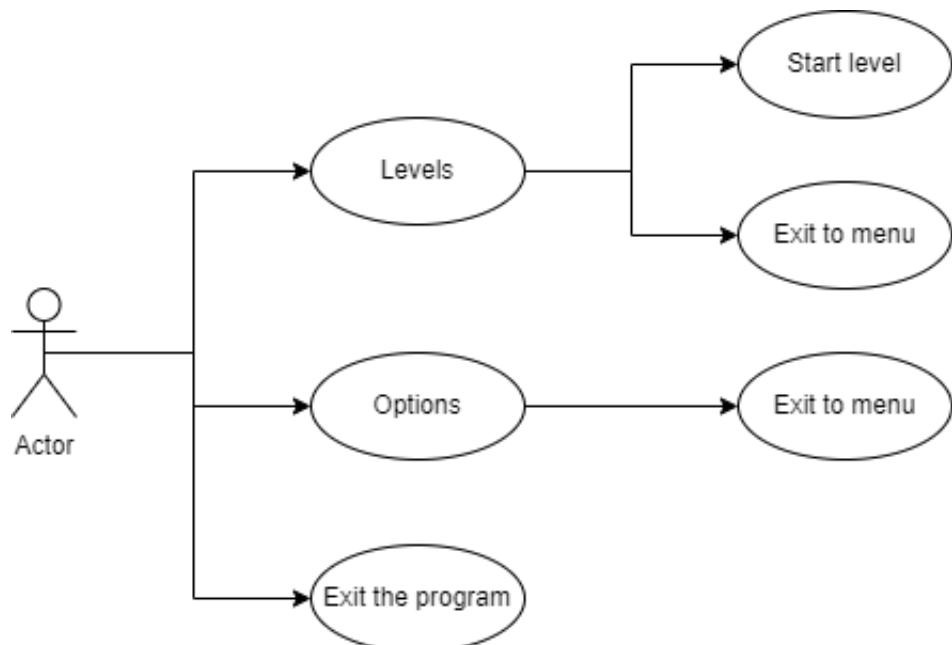
-Objektumorientált programozás (OOP): Az OOP egy programozási paradigmát jelent, amely az objektumokra épül. Az objektumok adatainkat tárolják (mezők) és műveleteket végeznek (metódusok). Az OOP alapelvei közé tartozik az öröklődés, az absztraktiós, a polimorfizmus és az adat elrejtése. Az OOP lehetővé teszi a programok strukturáltabbá és könnyebben karbantarthatóvá tételeit.

4. fejezet

A rendszer specifikációja

4.1. Felhasználoi követelmények

A program elindításával egy felhasználói felületet, angolul user interface (UI) jelenik meg számunkra. A felhasználói felület akár egy hozzá nem értő felhasználó számára is könnyen átlátható, használható mivel szerkezete egyszerű ugyanakkor minden információt megjelenít. Mindez megttekinthető a 4.1 ábrán is, ami egy használati eset(Use Case diagram)



4.1. ábra. Használati eset diagram

A használati eset diagram valójában megjeleníti számunkra, hogy mit is kell tudnia a rendszernek, valamint milyen funkciói legyenek a megtervezett rendszernek. A rendszer, lényegében a megvalósítandó software. Az Actor egy szerepkört reprezentál, esetünkben ez nem más, mint a felhasználó. A használati esetekről elmondható az, hogy egy viselkedési mintát tükröznek. Hárrom féle relációs kapcsolat lett betervezve a programba:

1. Általánosítás: esetünkben használati eset – és használati eset között van jelen. Viszont akár Actorok között is megvalósítható lehet.

2. Include: csak használati esetek közt lehetséges, akkor valósul meg, ha egyik kiterjed a másikra. Tehát igénybe veszi, és mindenig igénybe veszi a tulajdonságait.

3. Extend: kibővítést jelent. Az egyik használati eset kiterjeszti egy másik használati esetre a tulajdonságait.

A fentiek alapján elmondható, hogy mindezt a szemléletesség és áttekinthetőség jellemzi.

4.2. Rendszerkövetelmények

A rendszerkövetelmények olyan követelmények vagy specifikációk, amelyek meghatározzák, hogy egy számítógépes rendszer milyen hardver- és szoftvereszközökre van szüksége a megfelelő működéshez. Ezek az előírások segítenek abban, hogy a felhasználók vagy fejlesztők meghatározzák, hogy a számítógépes rendszerük alkalmas-e egy adott alkalmazás vagy szolgáltatás futtatására. Két tipusut különböztetünk meg a funkcionális és a nem funkcionális rendszerkövetelmények.

Az általános rendszerkövetelmények általában a következőket tartalmazzák: hardverkövetelmények, operációs rendszerkövetelmények, szoftverkövetelmények, hálózati követelmények.

4.2.1. Funkcionális követelmények

A funkcionális rendszerkövetelmények olyan követelmények, amelyek a rendszer által elvégzett konkrét funkciókra és feladatokra vonatkoznak. Ezek meghatározzák, hogy a rendszernek milyen feladatokat kell elvégeznie, milyen funkciókat kell ellátnia, milyen műveleteket kell végrehajtania stb.

-Interfész: A játék egy felhasználóbarát interfést biztosít, amely lehetővé teszi a játékos számára a játék irányítását és navigálását.

-Célok és kihívások: A játékban definiált célok és kihívások vannak, amelyek elérésére a játékosnak törekednie kell.

-Játékmenet: A játék tartalmaz egy játékmenetet, amely lehetőséget nyújt a felhasználónak a játékhöz való csatlakozásra és részvételre.

-Jutalmazás és fejlődés: A játékban jutalmazási rendszer van, amely elismeri a játékos eredményeit és lehetőséget nyújt a fejlődésre, szintek révén.

-Irányítás és kezelés: A játék irányítása és kezelése fontos szerepet játszik a játékélmény javításában. Az alkalmazás biztosítja az egyszerű és könnyen tanulható irányítási rendszert, amely lehetővé teszi a játékosok számára, hogy élvezzék a játékot.

4.2.2. Nem funkcionális követelmények

A nem funkcionális rendszerkövetelmények olyan követelmények, amelyek a rendszerrel kapcsolatos jellemzőket és tulajdonságokat határozzák meg, amelyek nem a rendszer által végrehajtott konkrét funkciókkal vagy feladatok. Ezek az követelmények a rendszer tulajdonságaira vonatkozhatnak, mint például a megbízhatóság, a teljesítmény, a biztonság, az elérhetőség, a karbantarthatóság stb.

-Ajánlott operációs rendszer: Windows 7, 8, 10 mindezek 32 és 64 Bit-es változatai.

-Programozási környezet szempontjából Microsoft Visual Studio-ban készült a software teljes egésze.

-Programozási nyelvek nézőpontjából C# .

-Keretrendszer szempontjából Unity, .NET.

Mindezek amennyiben nem teljesülnek, a rendszer részben, vagy akár teljesen használhatatlannak mondható.

5. fejezet

A szoftver bemutatása

5.1. A játékmenet és a játék céljai

A játék egyik célja természetesen az minden más játék esetén, hogy a játekosok szorakozzanak és élvezék a játékal eltöltött idöt. Mevel azonban ez egy oktató jellegű játék így igyekszik hasznosabb funkciokat is teljesíteni. A játékmenet és a játék célja lehetővé teszi a játékosok számára, hogy valós idejű interakcióba lépjenek a játékkörnyezettel és annak elemeivel. Az ilyen és ehez hasonló játékok célja általában az, hogy a játékosok fejlesszék a logikai gondolkodásukat, a probléma megoldó képességüket és a kreativitásunkat.

A játék tervezése és fejlesztése során igyekeztem olyan játékelemeket és feladatokat beépíteni a játékmenetbe, amelyek segítenek ezeknek a képességeknek a fejlesztésében. minden pályán a játékosnak meg kell oldania egy logikai feladványt vagy egy programozási kihívást, hogy továbbléphessen a játékban egy következő szintre ahol nehezebb feladat vár rá.

Az általam írt játékban a fő hangsúly, hogy a játékos észre vegye és össze kapcsolja megfelelően azokat a feladatokat amelyek a pálya elvégzéséhez szükségesek. Ezen folyamatok nem mindig egyértelműek illetve nem minden lehet elvégezni egyetlen fügvényel. Ennek érdekében a játékosnak más fügvényeket, ciklusokat is használnia kell amelyek így egy összetettebb feladat sort eredményeznek. A játék során a játékosnak azonosítania kell a problémát, amelyet meg kell oldania a szint végrehajtásához.

A pályák elkészítése során igyekeztem valamennyire a programozási alapismeretekhez szükséges fogalmakat, gyakorlatokat implementálni. Ezek közé tartozik a feltétel utasítás, paraméter nélküli fügvény hívás és a ciklus utasítás.

5.2. A nehézségi szintek és a fejlődési lehetőségek

A nehézségi szintek és fejlődési lehetőségek egy programozási képességeket fejlesztő játékban fontos szerepet játszanak abban, hogy a játékosok fokozatosan fejlődjenek és javítsák a programozási képességeiket. A játékosoknak érezniük kell, hogy képesek haladni és fejlődni, miközben az egyre nehezebb kihívásokat állítják előjük. A játéktervezőknek lehetőséget kell adniuk a játékosoknak, hogy felfedezzék a játékot és saját ütemben haladjanak, miközben megadják a megfelelő ösztönzést a fejlődésre.

Egy jól tervezett programozási játékban általában több nehézségi szint áll rendelkezésre, amelyek fokozatosan nehezednek ahogy a játékosok haladnak a játékban. Az első szintek általában egyszerűbbek és a játékosoknak alapvető programozási koncepciókat kell megtanulniuk, mint például a változók, a ciklusok és a függvények használata. Ahogy a játékosok haladnak, a nehézségi szintek és a feladatok bonyolultabbak lesznek, és a játékosoknak olyan kihívásokkal kell szembenézniük, mint a problémamegoldás, az optimalizálás és az algoritmusok tervezése.

A fejlődési lehetőségek egy programozási játékban szintén fontosak. A játékosoknak lehetőséget kell adni arra, hogy gyakorolják és fejlesszék a programozási képességeiket, például azzal, hogy javítják a korábban megoldott feladatokat vagy saját programokat írnak. Emellett a játéktervezőknek lehetőséget kell adniuk a játékosoknak, hogy kreatívak legyenek és saját ötleteiket valósítsák meg a játékban. A játékosoknak lehetőséget kell adni arra, hogy kísérletezzenek és próbáljanak ki új dolgokat, és a játéktervezőknek lehetőséget kell adniuk arra, hogy ösztönözzék a játékosokat a kreativitásra és az innovációra.

5.3. Játék felépítésének

5.3.1. Karakter irányítása

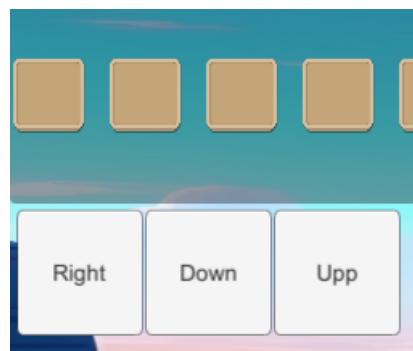
A játékban bár programozási készségeket szeretnénk fejleszteni mégis ugy van megvalositva a játék felépítése és a karakter irányítása, hogy a játékosunk konkrét kodot nem kell irjon. A játék mechanikája 3 fontos dologból tevődik össze emely segítségével irányítani lehet a karaktert. Ezek a következők :

-Leltár: A leltárba lehet betenni azokat a mozgásokat és speciális műveleteket amelyek a karakter mozgását segítik elő, ezen elemek pozícióját akár meg is lehet változtatni berakási sorrendtől függetlenül. A leltár mérete pályánként változó ezzel is növölve egyes pályák nehézségi szintjét. Illusztráció a 5.1-as ábrán látható.



5.1. ábra. Leltár

-Mozgások: A mozgásokat gombok segítségével lehet hozzá adni a leltárhoz. minden egyes gomb lenyomással az annak megfelelő mozgás adódik hozzá a leltárhoz ez addig ismételhető amíg a leltárunk meg nem telik. Illusztráció a 5.2-as ábrán látható.



5.2. ábra. Mozgások bevitel

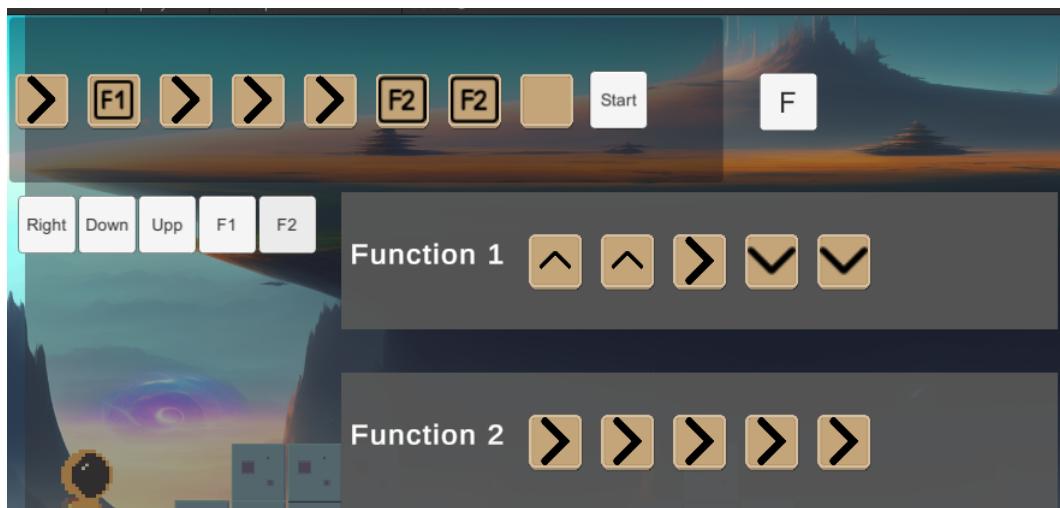
-Speciális műveletek: A speciális műveletek szükségesek az összetettebb illetve a kis leltárral rendelkező pályák elvégzéséhez. Ezek a műveletek az alapszintű programozási tudást probálják fejleszteni. Jelenleg 3 műveletet különböztetünk meg, a paraméter nélküli fügvény hívás, a ciklus utasítást és a feltétel utasítást.

A paraméter nélküli fügvény hívás az olyan pályák elvégzésében segít ahol az alapvető leltárunk tul keves, így a fügvényünk leltárjába helyezhetjük azokat az utasításokat amelyeket a karakter akkár többször is végre tud hajtani ez csupán attól számit, hogy hányszor hívjuk meg a fügvényt a leltárunkban. A fügvényünk leltárjának a merete is véges tehát ide sem tehetünk be a megadott méretnél több műveletet. Illusztráció a 5.3-as ábrán látható.



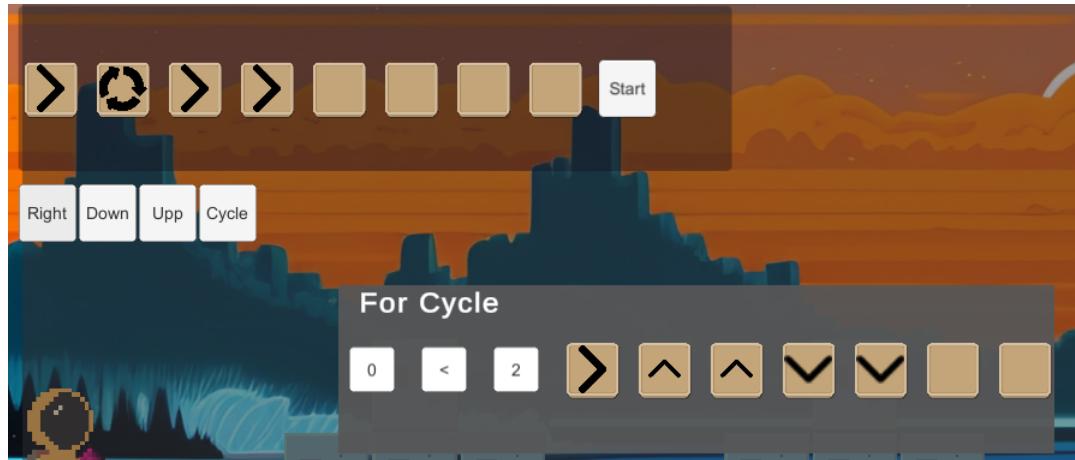
5.3. ábra. Fügveny használata

Természetesen vannak pályák ahol több fügvényt is hozzá lehet adni ezzel is összetettebbé téve a játékot. Illusztráció a 5.4-as ábrán látható.



5.4. ábra. Fügveny használata 2

A ciklus utasítás az olyan pályák elvégzésében segit ahol karakterünk egymás után többször is meg kell ismételje ugyanazokat a folyamatokat. Ennek segítségével mehatározhatjuk, hogy a karakterünk hanyszor ismételje meg a ciklus utasítás saját leltárjába bevitt feladatokat. Ebben az esetben is a leltár mérete mehatározott. Illusztráció a 5.5-as ábrán látható.



5.5. ábra. Ciklus használata

A feltétel utasítás az olyan pályák elvégzésében segit ahol a pályának van legalább egy olyan része ahol valamilyen feltétel kell teljesülni egy akadály legyözéséhez. A játékosunk kiválaszthatja az adott feltételeket amelyek a játék futása során ha telyesülnek akkor a karakterünk végre hajtja a kiválasztott mozgást. Illusztráció a 5.6-as ábrán látható.



5.6. ábra. Feltétel utasítás használata

A játékban a játékosnak a fentebb említett mozgásokat és speciális műveleteket kell felhasználnia annak érdekében hogy teljesítjen tudja a szinteket. A játékosnak stratégiákat kell kidolgozni, terveznie kell a parancssorozatokat, és meg kell találnia a legoptimálisabb megoldásokat a feladványokra.

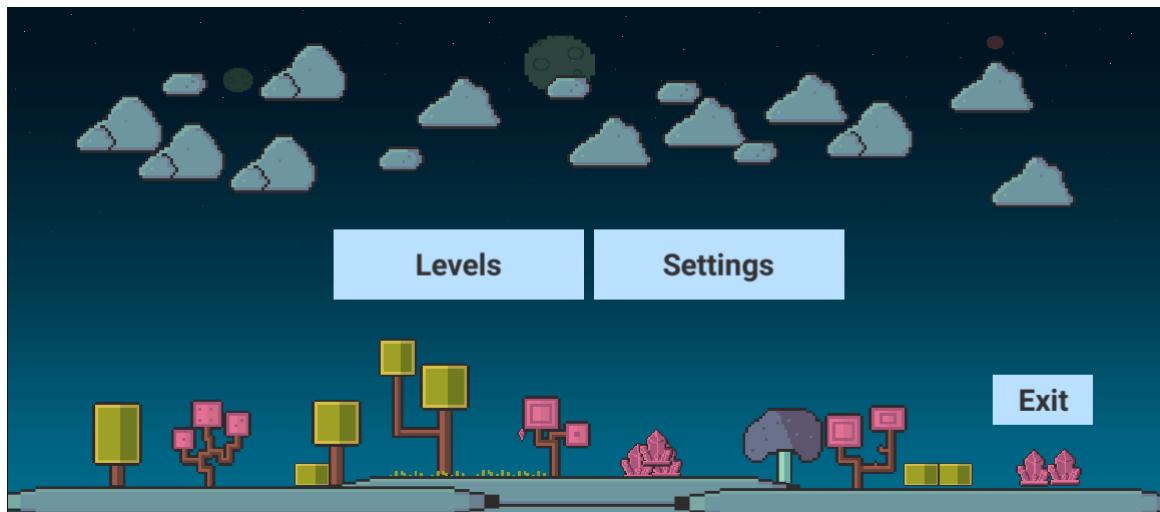
5.3.2. Képernyők és pályák

Kezdő képernyő 5.7-es ábra, ez a képernyő fogadja a felhasználat a játék elindítása után. A képernyön csupán három funkció közül lehet választani, ezek a következők:

-Levels: Az opció kiválasztása után uj képernyöre kerülünk ahol a pályák közül lehet választani.

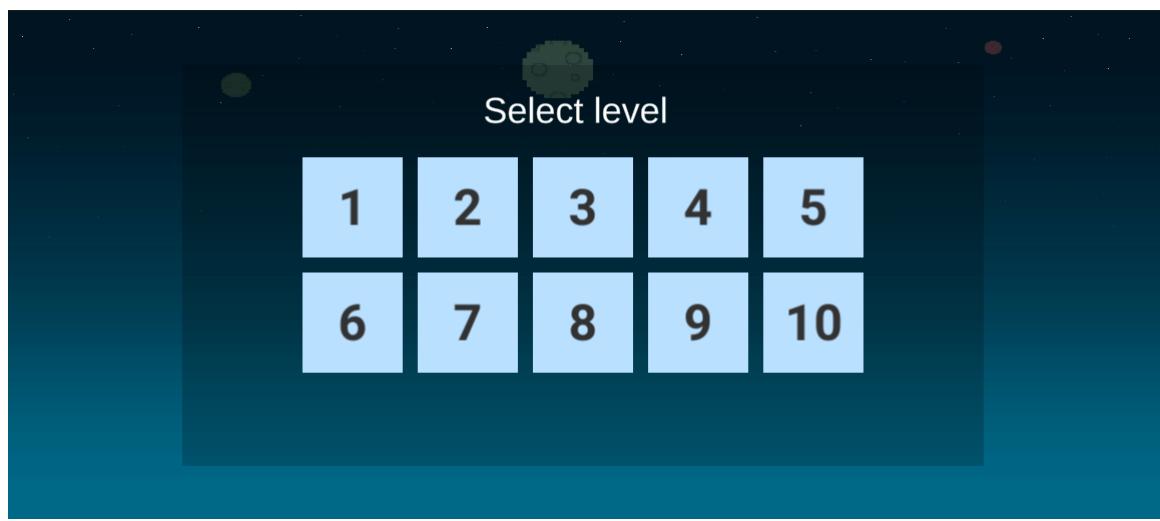
-Settings: Az opció kiválasztása után uj képernyöre kerülünk ahol a játék alap beállásait lehet módosítani mind például a hangerő.

-Exit: Az opció kiválasztása után bezárodik az alkalmazás.



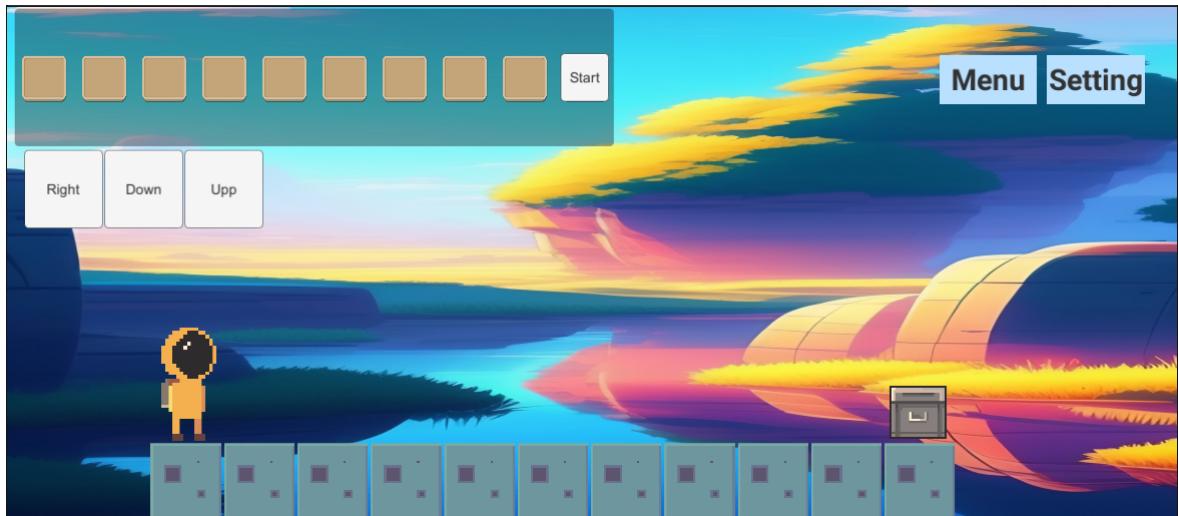
5.7. ábra. Kezdő képernyő

Pálya választó ablak 5.8-es ábra, ezen a képernyön lehet választani a különböző pályák közül. minden szám egy-egy pályát takar és nehézségi szint szerint növekednek.



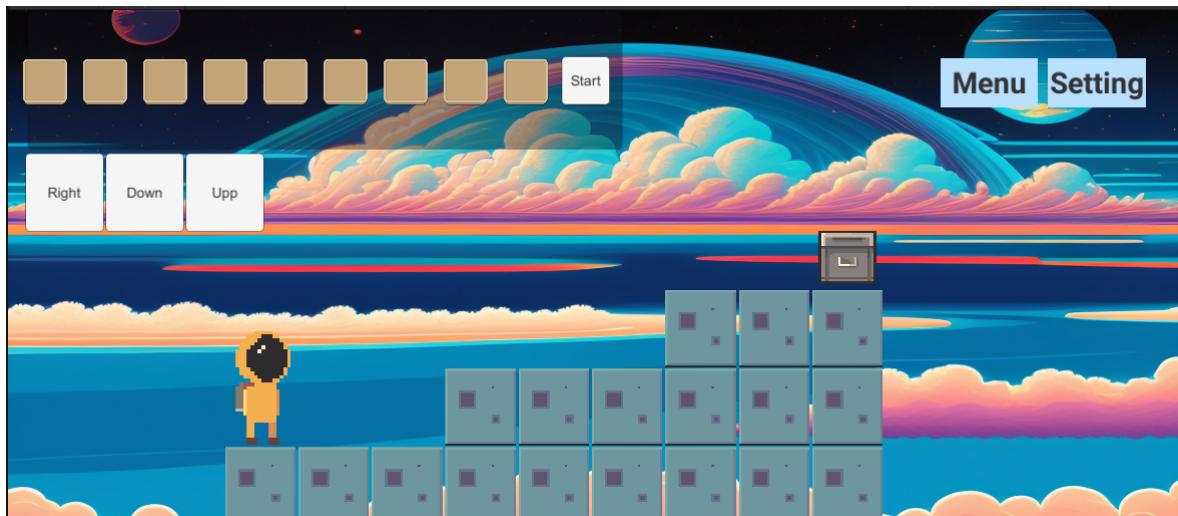
5.8. ábra. Pálya választó menü

Első pálya: A játék első pályáján minden ahogyan az látható a ??-es ábrán, még nincsenek speciális műveletek, csupán egyetlen mozgást kell használnunk mégpedig az előre fele mozgást.



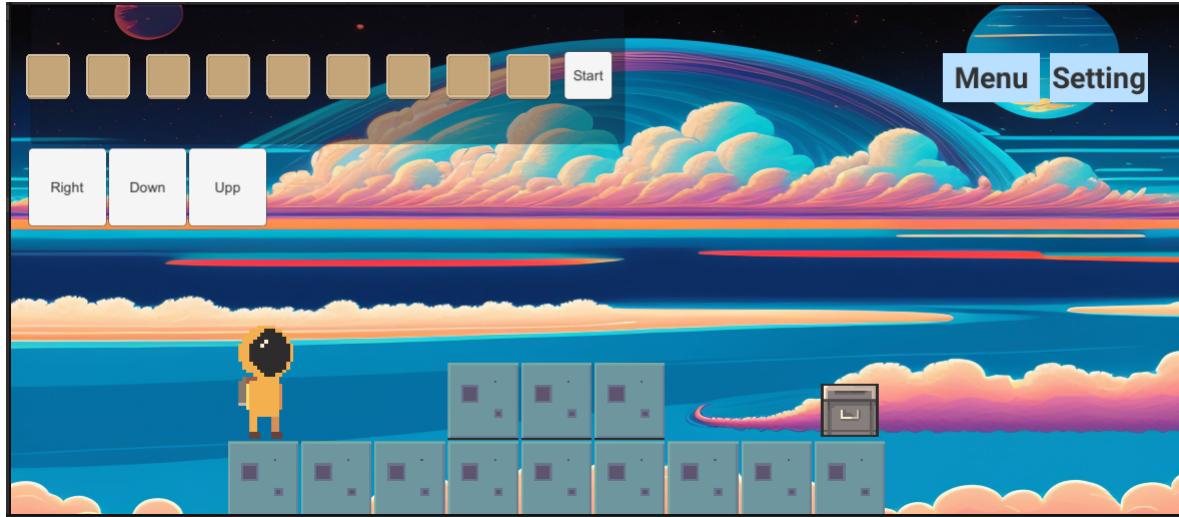
5.9. ábra. Első pálya

Második pálya: A játék második pályáján minden ahogyan az látható a 5.10-es ábrán, még nincsenek speciális műveletek, itt már a pálya teljesítéséhez két különböző mozgást is használnunk kell, mégpedig az előre fele mozgást és a felfelé ugrást.



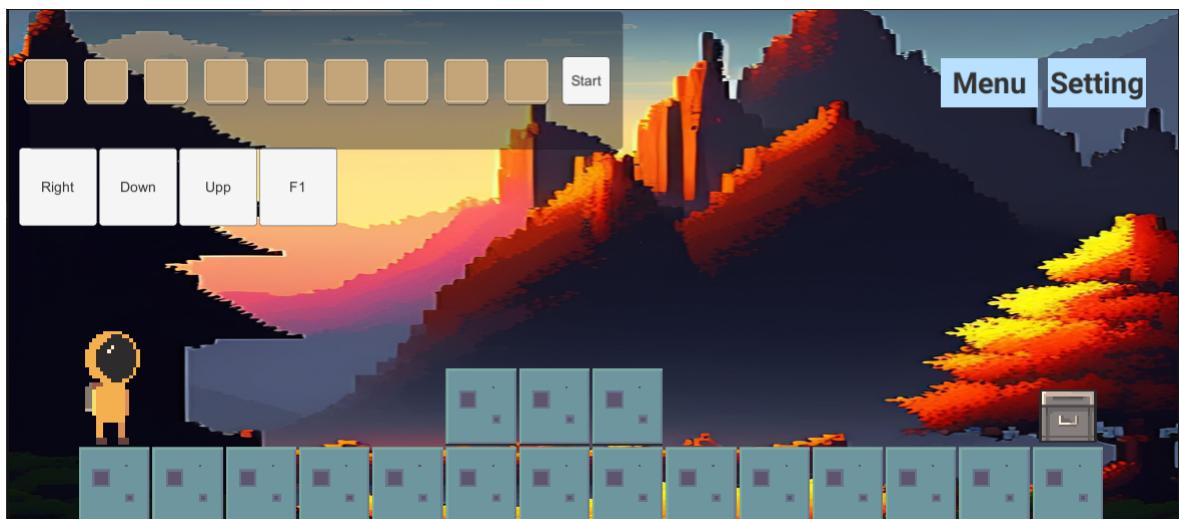
5.10. ábra. Második pálya

Harmadik pálya: A játék harmadik pályáján minden ahogyan az látható a 5.11-es ábrán, még itt se nincsenek speciális műveletek, itt már a pálya teljesítéséhez három különböző mozgást is használnunk kell, mégpedig az előre fele mozgást, a felfelé ugrást és a lefelé ugrást



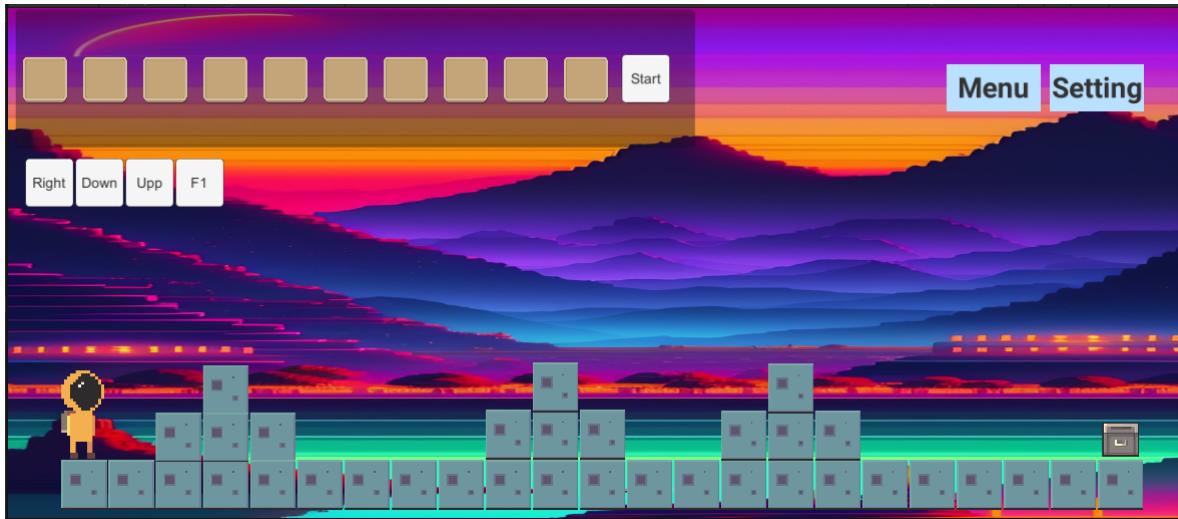
5.11. ábra. Harmadik pálya

Negyedik pálya: A játék negyedik pályáján minden ahogyan az látható a 5.12-es ábrán, már van egy speciális művelet, a három mozgás melett megjelent már egy paraméter nélküli fügvény hívás 5.3 is. Ezen a pályán elég csupán egyszer meghivnunk a fügvényt a szint teljesítéséhez.



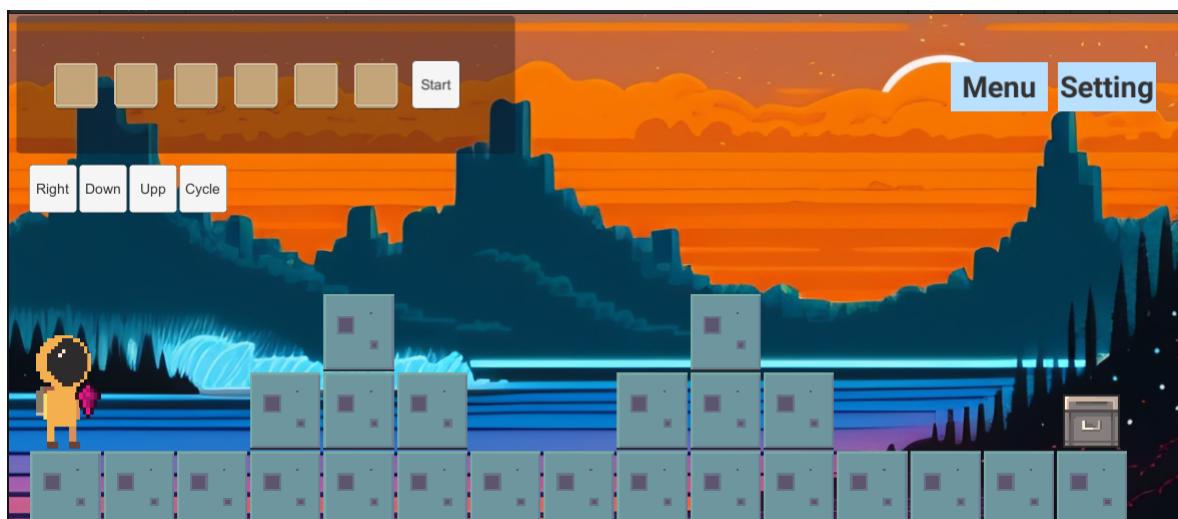
5.12. ábra. Negyedik pálya

Ötödik pálya: A játék ötödik pályáján minden ahogyan az látható a 5.12-ös ábrán, az előző negyedik pályához hasonlóan itt is a mozgások melett ott van a paraméter nélküli fügvény hivás. Ezen a pályán viszont már többször is meg kell hívni a fügvényt a pálya hossza miatt.



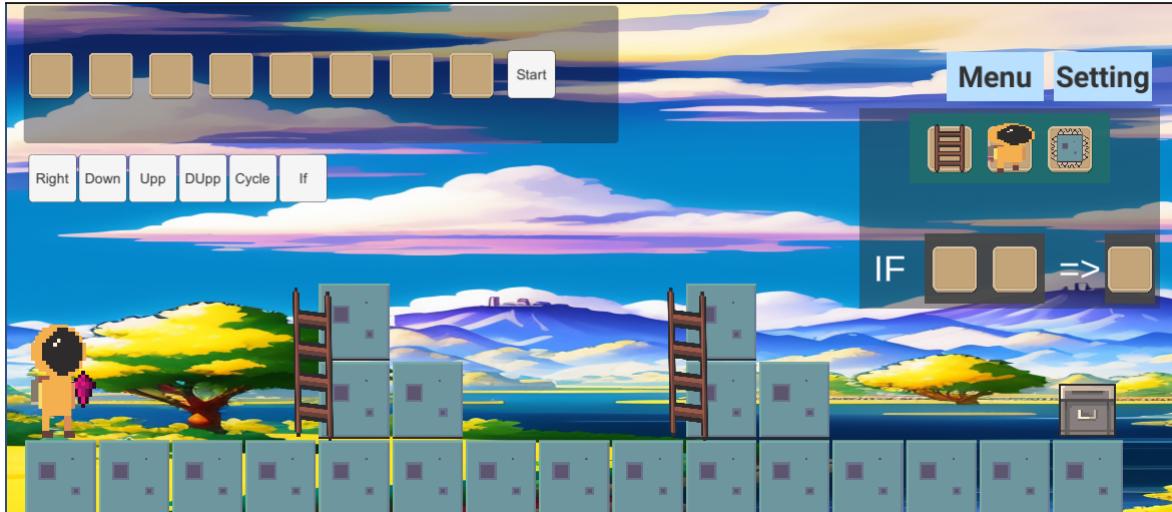
5.13. ábra. Ötödik pálya

Hatodik pálya: A játék hatodik pályáján minden ahogyan az látható a 5.14-ös ábrán, már van egy új speciális művelet, a három mozgás melett megjelent már egy ciklus utasítás 5.5 is. Igy a játékosunk az ismétlődő pálya szakaszokat felmérvén csupán meg kell határozza a mozgásokat és azok ismétlődésének számát.



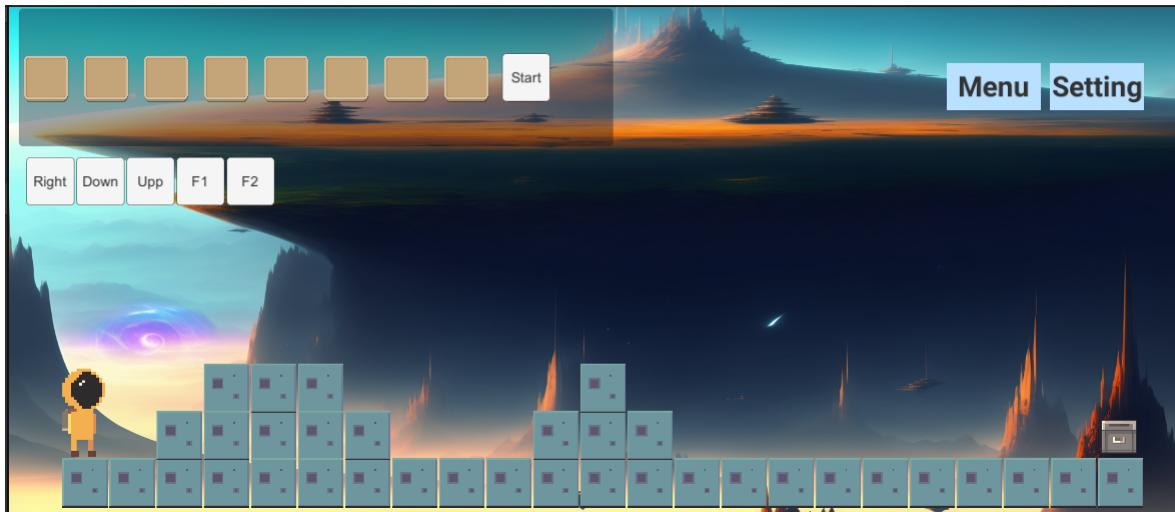
5.14. ábra. Hatodik pálya

Hetedik pálya: A játék hetedik pályáján mind ahogy az látható a 5.15-es ábrán, már van egy új speciális művelet, a három mozgás melett megjelent és ciklus utasítás melett megjelent a feltétel utasítás is. Igy a játékosunk feltételek megadásával ki tud választani egy megelelő mozgást. Ahogy az a 5.6-es ábrán látható.



5.15. ábra. Hetedik pálya

Nyolcadik pálya: A játék nyolcadik pályáján mind ahogy az látható a 5.16-as ábrán, vissza térünk a paraméter nélküli fügvény hiváshoz viszont itten már két fügvényt és ezok kombinációját is használhatjuk. Ahogy az a 5.4-es ábrán látható.



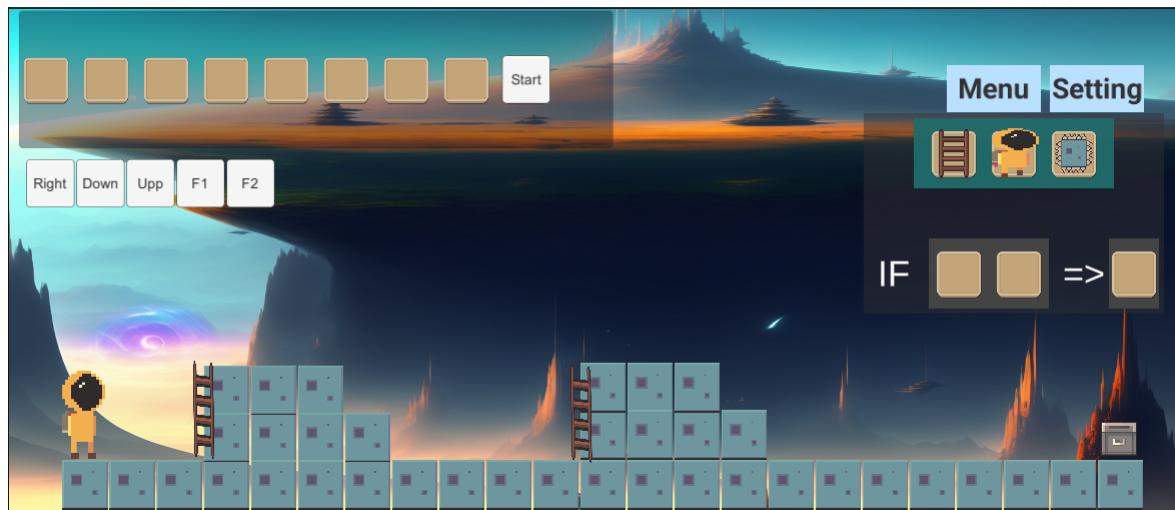
5.16. ábra. Nyolcadik pálya

Kilencedik pálya: A játék nyolcadik pályáján mind ahogy az látható a 5.17-es ábrán, újra két paraméter nélküli fügvény hívást használhatunk illetve itt a ciklus utasítás is megjelent mellettük, ezzel gyarapítva a pálya elvégzésének lehetőségeit.



5.17. ábra. Kilencedik pálya

Tizedik pálya: A játék nyolcadik pályáján mind ahogy az látható a 5.18-es ábrán, újra két paraméter nélküli fügvény hívást használhatunk illetve itt a feltétel utasítás is megjelent, így a játékosunk feltételek megadásával ki tud választani egy megelelő mozgást.



5.18. ábra. Tizedik pálya

6. fejezet

Továbbfejlesztési lehetőségek

6.1. Programozási algoritmusok

A játékban lehetne több olyan pálya is amely alap programozási algoritmusok megoldására épül minden például:

-Keresési algoritmusok:

Lineáris keresés: Az adatsor elemeit egymás után ellenőrzi, amíg megtalálja a keresett elemet vagy eléri az adatsor végét.

Bináris keresés: Gyorsan megtalálja a keresett elemet rendezett adatsorokban, mivel a keresési tartományt folyamatosan felezve halad előre.

-Rendezési algoritmusok:

Buborékrendezés: Az algoritmus egymás mellett lévő elemeket hasonlít össze és cserél, amíg az adatsor rendezetté válik.

Gyorsrendezés: Rekurzív algoritmus, amely a pivot elem segítségével osztja fel az adatsort kisebb és nagyobb részekre, majd rekurzívan rendez.

Ezen algoritmusok ismereteinek fontossága az, hogy ezek az algoritmusok alapvető elemei a programozásnak, és széles körben alkalmazzák őket valós világbeli problémák megoldására. Az algoritmusok megértése és hatékony használata segít a hatékonyabb és optimálisabb kódolásban, az algoritmusok megfelelő választásában és a problémamegoldási képesség fejlesztésében. A programozók, akik ismerik és jól használják ezeket az algoritmusokat, képesek hatékony és optimalizált programokat írni és hatékonyan megoldani a különböző problémákat. Megértésük és alkalmazásuk segít a kód optimalizálásban, a hatékony algoritmusok kiválasztásában és az általános programozási készségek fejlesztésében. Az ismeretek bővítése és a gyakorlatban való alkalmazásuk lehetővé teszi a programozónak, hogy hatékonyabb és rugalmasabb kódokat írjon, és hatékonyan megoldja a valós világ problémáit a szoftverfejlesztés terén.

6.2. Többjátékos mod

Az általm irt szoftver egyik és talán leg hasznosabb tovább fejlesztése az lenne ha a játék online több játékos is tudna játszani egymás ellen. minden játékos saját pályákat tudna létre hozni amit ugy tudna publikálni ha ő maga is képes végig vinni. A játékos salyát pályája végén a megszerzett vagyona lenne amit más játékosok eltudnak rabolni ha sikerül teljesíteniük a pályat. Igy minden játékos abban igyekezne, hogy minnél

nehezebb pályát tudjon létre hozni ezzel csökkentve a rablások számát. A játékosok meghatározhatnák a pálya hosszát a bevitt mozgások számát illetve hogy hány segéd fügvényt használhatnak fel az adott pálya telyesítése érdekében

A játékosok rangsorolva lennének aszerint, hogy ki mennyi kincset tudott elrabálni illetve ki hány pályát tudott sikeresen telyesíteni. Köztudott hogy a kompetitív játékok lehetőséget nyújtanak a játékosoknak az érzelmi intelligencia, a stratégiai gondolkodás és a döntéshozatal fejlesztésében.

6.3. Több játékplatform támogatása

A jelenlegi alkalmazás androidos környezeten fut mivel jelenleg ez a leg elterjedtebb felhasználoin környezet a játékos világban. Ennek legföbb oka a mobil telefonok rendkívül széles körben való elterjedése. A mai világban a mobil telefon már a minden napok részévé vált a legtöbb ember életében amely hiány jelentősen megnehezítené a minden napokat. Bár a mobil telefon nem játszásra volt kitalálva mégis az emberek töbsége unalom üzes vagy szorakozás céljából valamilyen játékot ki ki probál telefonján.

A jelenlegi játékom játszását leginkább pc s játékként vagy esetleg web oldalon játszható játékként tudnám elképzelni. Ezen területek véleménye szerint nem vonzanának annyi játékost be mind a mobilos alkalmazás de ezek sem elhanyagolhatóak.

6.4. Közösség építés

A játék egyik tovább fejlesztett verziója alkalmas lehetne közösség építésre is. Ez két modon valósulhatna meg. Vagy egyéni, privát beszélgetéseket lehhetne kezdeményezni más játékosokkal. Vagy csapatokat, klánokat lehetne létre hozni amelyek pontszáma a tagok sikerétől és fejletségi szintjükkel függne.

Összefoglaló

A modern technológia világában a programozási készségek egyre fontosabbá válnak, mivel lehetővé teszik a szoftverek, alkalmazások és weboldalak fejlesztését, amelyek számos területen alkalmazhatóak. A programozási készségek logikai, algoritmikus gondolkodást, adatstruktúrák és algoritmusok ismeretét, valamint programozási nyelvek ismeretét foglalják magukban. Az utóbbi években egyre népszerűbbé válik a játékfejlesztés és az oktatás összekapcsolódása, mivel a játékok interaktív és szórakoztató módon segítik az oktatási anyagok hatékonyabb megértését és elsajátítását. Az oktatási játékok lehetőséget nyújtanak a diákok számára, hogy játékos környezetben gyakorolják az általuk tanult dolgokat, ezáltal javítva a tanulási folyamat hatékonyságát és eredményességét. A játékfejlesztés lehetővé teszi saját videójátékok létrehozását, amelyek fejlesztéséhez kreativitás, tervezési és programozási képességekre van szükség. A játékfejlesztés számos pozitív hatással jár, mint például a kreativitás és problémamegoldó képesség fejlesztése, a csapatmunka és kommunikáció ösztönzése, valamint szórakoztatás és motiváló élmény nyújtása. Összességében elmondható, hogy a játékfejlesztés és az oktatás összekapcsolódása lehetővé teszi az interaktív és hatékony tanulást, miközben élvezetes élményt nyújt. A játékok által fejlesztett készségek és tudás nagyban hozzájárulhatnak az emberek személyes és szakmai fejlődéséhez a modern technológia világában.

Mindent összevetve még fontosnak tartottam a programkód helyesnek vélt eltárolását és verziókövetését is. Így a programkódot, egyebek mellett a vele járó dokumentációt is egy GitLab repository-n is eltároltam, melynek fejlődését commit-ok által nyomon lehet követni (<https://github.com/huni98asd/-1lamvizsga2023.git>)

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezető tanáromnak, dr. clăanzan David-nak, aki szakértelmével, hasznos magyarázataival és a konzultációk során biztosított elengedhetetlen tanácsaival hatalmas segítséget nyújtott szakdolgozatom elkészüléséhez. Köszönöm barátaimnak, tanáraimnak és mindeneknek, akik még akkor is bátorítottak, amikor az alagút vége közel sem volt. Köszönettel tartozom továbbá szüleimnek, és családom minden egyes tagjának, akik támogatása és szeretete nélkül, talán ez a szakdolgozat nem jöhett volna létre. Köszönöm nekik, hogy tanulmányaim során végtelen türelemmel és megértéssel támogattak, és minden helyzetben mellettem álltak.

Ábrák jegyzéke

2.1. Vizesés modell folyamatai. Forrás: [viz]	6
2.2. Agilis fejlesztés folyamatai. Forrás: [JOH23]	7
2.3. Gamifikáció. Forrás: [?]	8
2.4. Lightbot. Forrás: [lig]	11
2.5. Flexbox Froggy. Forrás: [fle]	12
2.6. CodeCombat	13
3.1. Játék fejlödés. Forrás:[gam]	15
3.2. A játékfejlesztési platformok változása. Forrás:[jpv]	17
4.1. Használati eset diagram	22
5.1. Leltár	27
5.2. Mozgások bevitel	27
5.3. Fügveny használata	28
5.4. Fügveny használata 2	28
5.5. Ciklus használata	29
5.6. Feltétel utasítás használata	29
5.7. Kezdő képernyő	30
5.8. Pálya választó menü	30
5.9. Első pálya	31
5.10. Második pálya	31
5.11. Harmadik pálya	32
5.12. Negyedik pálya	32
5.13. Ötödik pálya	33
5.14. Hatodik pálya	33
5.15. Hetedik pálya	34
5.16. Nyolcadik pálya	34
5.17. Kilencedik pálya	35
5.18. Tizedik pálya	35
F.1.1A TeXstudio L ^A T _E X-szerkesztő	44

Irodalomjegyzék

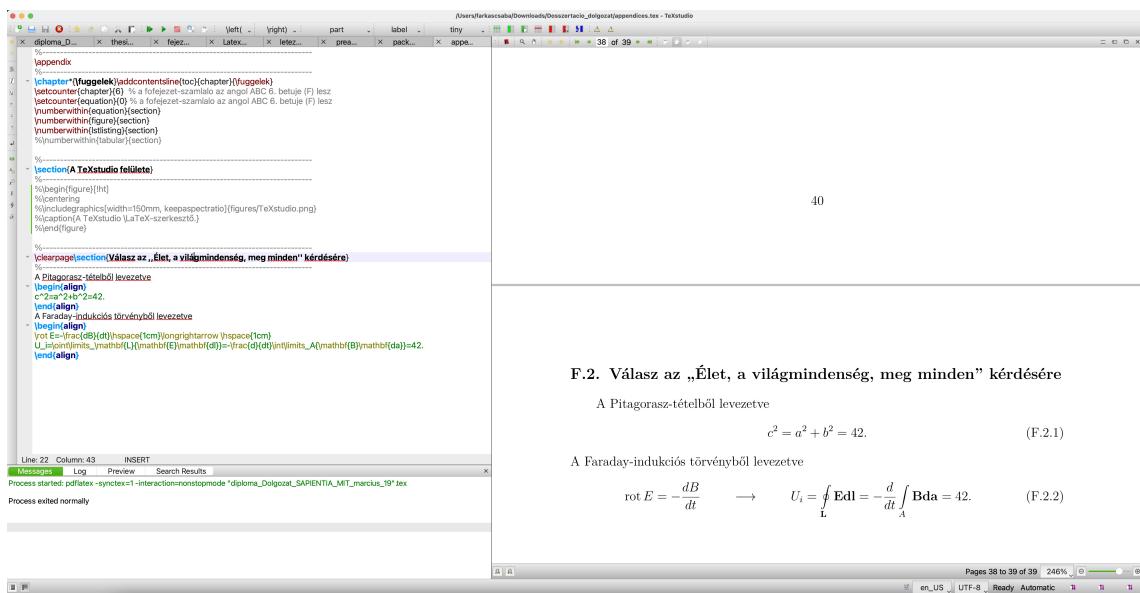
- [Ber54] Jessie Bernard. The theory of games of strategy as a modern sociology of conflict. *American Journal of Sociology*, 59(5):411–424, 1954.
- [BLA] SZÉKELY BLANKA. Székely blanka–erdeiné késmárki-gally szilvia: A vizesés és agilis projektmenedzsment módszertan alkalmazása a vezetésben waterfall and agile project methodologies application in management. *VSZI kötetek*, page 82.
- [BLN⁺11] Chris Bateman, Rebecca Lowenhaupt, Lennart E Nacke, et al. Player typology in theory and practice. In *DiGRA conference*, pages 1–24. Citeseer, 2011.
- [C23] C. Available at. <https://learn.microsoft.com/en-us/dotnet/csharp/>, 2023.
- [CBD16] Sébastien Combéfis, Gytautas Beresnevičius, and Valentina Dagienė. Learning programming through games and contests: overview, characterisation and discussion. *Olympiads in Informatics*, 10(1):39–60, 2016.
- [CEO14] Ilaria Caponetto, Jeffrey Earp, and Michela Ott. Gamification and education: A literature review. In *European Conference on Games Based Learning*, volume 1, page 50. Academic Conferences International Limited, 2014.
- [Cra85] Vincent P Crawford. Dynamic games and dynamic contract theory. *Journal of Conflict resolution*, 29(2):195–224, 1985.
- [fle] Flexbox-froggy. <https://codepip.com/games/flexbox-froggy/>.
- [gam] Evolve now or become gaming history: “development of gaming and it’s years”. <https://www.augray.com/blog/history-of-gaming/>.
- [Git23] GitHub. Available at. <https://github.com/>, 2023.
- [Har04] Ken Hartness. Robocode: using games to teach artificial intelligence. *Journal of Computing Sciences in Colleges*, 19(4):287–291, 2004.
- [Ivo15a] James D Ivory. A brief history of video games. In *The video game debate*, pages 3–5. Routledge, 2015.

- [Ivo15b] James D Ivory. A brief history of video games. In *The video game debate*, pages 1–21. Routledge, 2015.
- [JOH23] J.R. JOHNIVAN. Agile software development methodology principles. <https://project-management.com/agile-software-development-methodologies/>, 2023.
- [jpv] 50 years of gaming history. <https://www.ebinary.it/tag/gaming/>.
- [Kha21] Nazokat Fayzullayevna Khaitova. History of gamification and its role in the educational process. *International Journal of Multicultural and Multireligious Understanding*, 8(5):212–216, 2021.
- [lig] Lightbot. <https://www.roboticsforschools.eu/roboquests-level1/138-procedures-loops-and-logical-thinking-all-you-need>
- [Lil21] Dominek Dalma Lilla. A flow mint a pozitív pszichológia jelenléte az oktatásban. *Eruditio-Educatio*, 16(4):72–82, 2021.
- [Low09] Henry Lowood. Videogames in computer space: The complex history of pong. *IEEE Annals of the History of Computing*, 31(3):5–19, 2009.
- [Mur78] J Keith Murnighan. Models of coalition behavior: Game theoretic, social psychological, and political perspectives. *Psychological Bulletin*, 85(5):1130, 1978.
- [.NE23] .NET. Available at. <https://dotnet.microsoft.com/en-us/>, 2023.
- [PS06] Jussi Parikka and Jaakko Suominen. Victorian snakes? towards a cultural history of mobile games and the experience of movement. *Game Studies*, 6(1), 2006.
- [Rab05] Steve Rabin. *Introduction To Game Development (Game Development)*. Charles River Media, Inc., 2005.
- [RRP06] Richard M Ryan, C Scott Rigby, and Andrew Przybylski. The motivational pull of video games: A self-determination theory approach. *Motivation and emotion*, 30:344–360, 2006.
- [RRR03] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172, 2003.
- [Stu23a] Microsoft Visual Studio. Available at. <https://flexboxfroggy.com/#hu>, 2023.
- [Stu23b] Microsoft Visual Studio. Available at. <https://codecombat.com/>, 2023.
- [Stu23c] Microsoft Visual Studio. Available at. <https://visualstudio.microsoft.com/>, 2023.

- [Tam20] Pólya Tamás. A videojátékok mint történetmesélő mászókák és a médiaszöveg-fogalom: Az olvasat átértelmeződése és a narratíva másodrendűsége egy ergodikus médiumban. *Médiakutató*, 21(1):93–104, 2020.
- [Uni23] Unity. Available at. <https://unity.com/>, 2023.
- [vga] Video game platforms. https://en.wikipedia.org/wiki/Video_game.
- [viz] Sdlc - waterfall model. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.
- [VLSVDHR10] Giel Van Lankveld, Pieter Spronck, H Jaap Van Den Herik, and Matthias Rauterberg. Incongruity-based adaptive game balancing. In *Advances in Computer Games: 12th International Conference, ACG 2009, Pamplona Spain, May 11-13, 2009. Revised Papers 12*, pages 208–220. Springer, 2010.
- [Yar14] Danny Yaroslavski. How does lightbot teach programming. *Retrieved January, 29:2016*, 2014.

Függelék

F.1. A TeXstudio felülete



F.1.1. ábra. A TeXstudio L^AT_EX-szerkesztő.

F.2. Válasz az „Élet, a világmindenség, meg minden” kérdésére

A Pitagorasz-tételből levezetve

$$c^2 = a^2 + b^2 = 42. \quad (\text{F.2.1})$$

A Faraday-indukciós törvényből levezetve

$$\text{rot } E = -\frac{dB}{dt} \quad \rightarrow \quad U_i = \oint_{\text{L}} \mathbf{E} \cdot d\mathbf{l} = -\frac{d}{dt} \int_A \mathbf{B} \cdot d\mathbf{a} = 42. \quad (\text{F.2.2})$$