**Záródolgozat**

**Szincsák Szabolcs**

**2022**

**DSZC Beregszászi Pál Technikum**

**Platformjáték Unity-ben**

**Tartalomjegyzék**

**1. Bevezetés 4**

**2. Felhasználói dokumentáció 5**

2.1. Ismertető 5

2.2. Rendszerkövetelmények 5

2.3. Játék indítása 6

2.4 Regisztráció és Bejelentkezés 6

2.4.1. Főmenü 6

2.4.1. Beállítások 7

2.4.3. Pálya választó 8

2.5. Irányítás 8

2.6. A játékmenet 9

2.6.1. Ellenfelek 9

2.6.2. Kőfej 9

2.6.3. Tüskefej 10

2.6.4. Tüskék 10

2.6.5. Fűrész 11

2.6.6. Nyílcsapda 11

2.6.7. Élet 11

2.6.8. Pálya teljesítése 12

2.7. Mentett adatok 12

**Irodalomjegyzék 13**

**1. fejezet**

Bevezetés

A szakdolgozatom egy 2D-s platformjáték Unity-ben, C# nyelven megvalósítva. A játék célja csapdák elkerülése, akadályok leküzdése és eljutás a célba.

A platformjáték (más néven platformer vagy Jump ’n’ Run) egy videójáték-műfaj, az akciójáték alműfaja. A játékos által irányított karakternek platformokon keresztül kell ugrálnia és/vagy különböző akadályokat kell átugrania. A játékos feladata, hogy a karakterével megfelelő időben ugorjon, hogy tovább tudjon, menni vagy ne essen le. Az ugráson kívül más mozgáselem is szerepelhet, mint az úszás, mászás vagy repülés. Platformjátéknak nevezhető az olyan játék, aminek szerves részét képezi a platformokon való ugrálás.[1]

Az oldalnézetes vagy 2D-s játékok legfőbb tulajdonsága a kétdimenziós grafika. Az FPS és a TPS játékok előtt ez volt a legelterjedtebb és legváltozatosabb műfaj. Az oldalnézetes játékok gyakorlatilag minden platformon megjelentek és rengeteg típusuk van. Gyakran készültek ilyen játékok sikeres filmek és rajzfilmek alapján. A kétdimenziós grafika egyértelmű előnye, hogy nem kell nézőpontokkal, kameramozgatással foglalkozni. A játékos legtöbbször oldalnézetben látja a pályát, mozgás közben pedig a pálya együtt halad a játékfigurával. A látható világot kézzel vagy számítógéppel rajzolták.[2]

A megvalósításhoz a Unity játékmotort használom, szöveg kiírásához TextMesh Pro segítségével történik az adatbázis meg a 000webhoston létrehozott weboldalamra került fel és PHP segítségével lehet hozzá férni.

**2. fejezet**

**Felhasználói dokumentáció**

**2.1. Ismertető**

A játék egy 2D-s platformjáték. Egy sárkányt lehet irányítani és csapdákat kell kikerülni, hogy eljuss a pálya végére. A pályákon több tárggyal lehet kölcsönhatásba lépni, amikkel változtatni lehet a pályán, így lehetővé téve az előrehaladást és az előnyhöz jutást.

A játékosoknak van egy élete, amit a játék során elveszíthet. Ha az összes életük elfogy, a játékos visszakerül a kezdőpontra és újra próbálkozhat a pálya végig játszásával, de a halálszámláló növekedik, plusz egyel. A csapdákkal való ütközéskor sebződik a játékos. A víz is veszélyes hely lehet a játékos számára, ugyanis amikor bele zuhanunk elkezd sebződni, ezért jobb átugrani a vizes helyeket.

A pályák teljesítéséhez el kell jutni a célba, ami egy zászlót jelent és az csak akkor számít, ha a karakter hozzá is ér máskülönben nem ér véget a pálya. Ha ez teljesül, akkor a pályát teljesítettük.

**2.2. Rendszerkövetelmények**

A játék asztali számítógépen játszható. A részletes rendszerkövetelmények megtalálhatók a Unity honlapján.[3] A játék futtatátásához a következőek a minimum követelmények:

* Operációs rendszer: Windows 7 (SP1+), Windows 10 and Windows 11
* Videókártya: DX10, DX11, DX12 képes.
* Processzor: x86, x64 architektúrájú és támogatja SSE2 művelethalmazt.
* Merevlemez: 1024MB szabad hely
* Memória: 4GB
* További követelmények: A hardvergyártó hivatalosan támogatott illesztőprogramokat

**2.3. Játék indítása**

A játék indításához nem szükséges telepítés. A játék mappáját a számítógép merevlemezére kell másolni, ezután a mappában található Trapland.exe fájl indításával kezdhetünk el játszani.

**2.4.** **Regisztráció és Bejelentkezés**

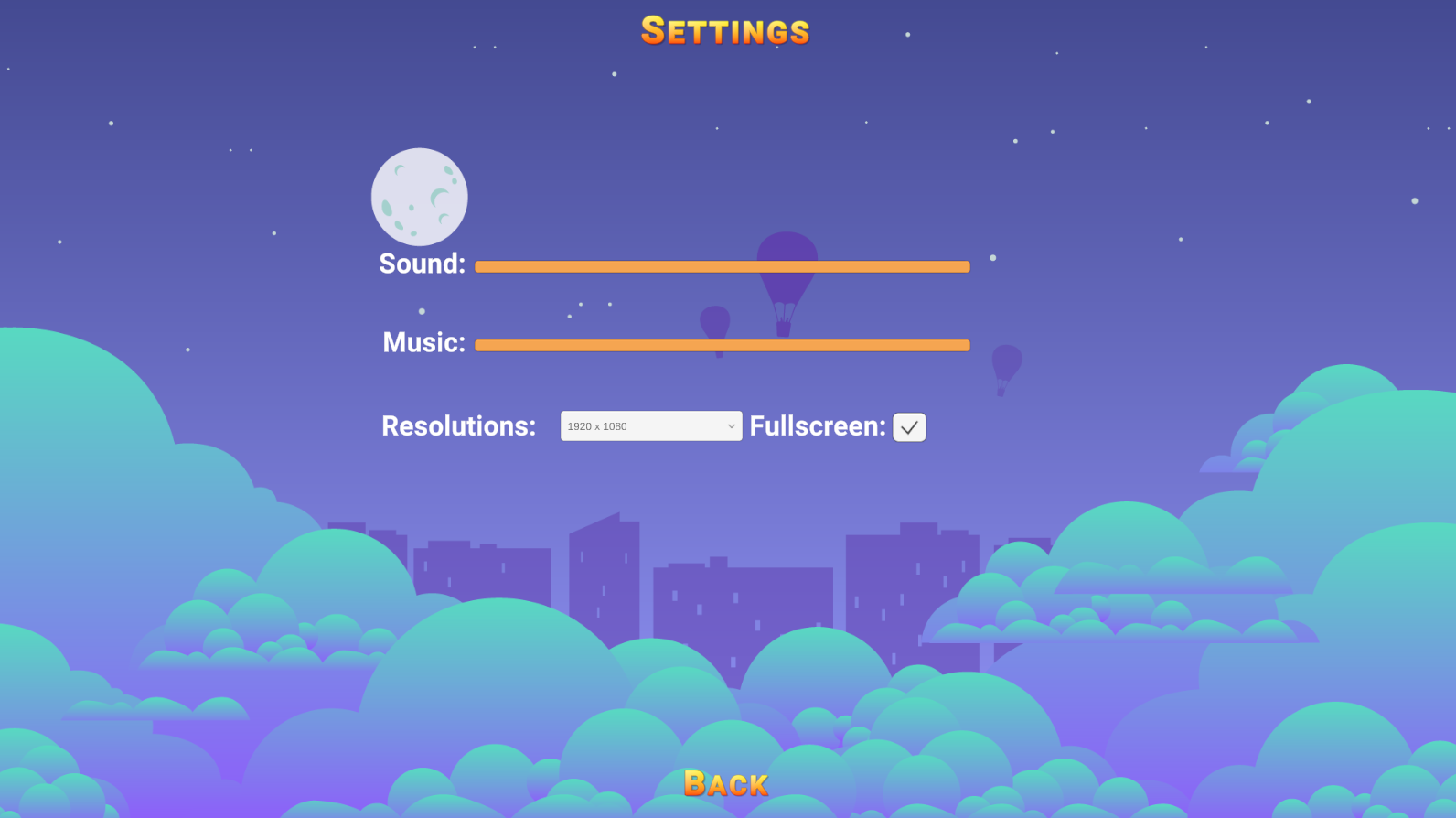
A játék indítása után a Regisztráció és Bejelentkezésnél találjuk magunkat. Itt regisztráció után be lehet jelentkezni és elkezdeni a játékot.

**2.4.1 Főmenü**

A sikeres bejelentkezés után a főmenüben találjuk magunkat. Itt láthatjuk a játék címét (Trapland). Ezen a felületen érhetőek el a program fő funkciói.

* Play gombra kattintva elkezdhetjük a játékot miután kiválasztottuk melyik pályát akarunk menni bár a legelején csak az első kezdő pálya lesz elérhető.
* Settings gombra kattintva a beállításokat lehet elérni.
* Log Out gombra kattintva kiléphetünk jelenlegi felhasználóból.

**2.4.2 Beállítások**



**2.1. ábra. Settings**

A beállításokat (Settings, 2.1. ábra) megnyitva változtathatunk a játék megjelenésén. Négy beállítás érhető el: teljes képernyős mód be - ki kapcsolása, felbontás beállítása, zene hangerejének beállítása, játék hangerejének beállítása.

Teljes képernyős mód és ablakos mód között válthatunk a Fullscreen feliratú jelölőnégyzet segítségével. Ha bejelöljük akkor a játék teljes képernyős módra vált, ha megszüntetjük a bejelölést, akkor ablakos módra.

A Resolutions feliratú legördülő menüben állítható be a felbontás. A monitor által elérhető összes felbontás közül választhatunk. A kisebb felbontások elől találhatóak a listában, a nagyobbak a végén. A legjobb választás az a felbontás amire a monitorunk is be van állítva, de a jobb teljesítmény érdekében választhatunk kisebb felbontást.

A Sound feliratú csúszkával lehet alítani a játék hangerejét.

A Music feliratú csúszkával lehet alítani a játék alatt szóló zenét.

**2.4.3 Pályaválasztó**



**2.2. ábra. Levels**

Bejelentkezés után a Play gombra kattintva elérhetővé válik a szintválasztó (Levels, 2.2. ábra), ahol regisztrálás után elérhető az első szint, ami többnyire csak a játék mechanikáit mutatja be és a különböző csapdákat. Egy pálya teljesítésekor az adatbázisba kerül mentésre a felhasználónév, amivel regisztráltak, az idő, amibe került teljesíteni a pályát és a halálok számát.

**2.5 Irányítás**

A karakterünket a Nyílbillentyűk és WASD billentyűk segítségével irányíthatjuk, de támogatja a fél-kontrolleres irányítás is.

* 🡨: A balra nyíl és „a” billentyű lenyomásával karakter elkezd balra menni.
* 🡪: A jobbra nyíl és „d” billentyű lenyomásával karakter elkezd jobbra menni.
* ↑: A Space billentyű lenyomásával a karakterünk ugrik egyet.
* Játék megállítása: „Esc” billentyűvel lehet megtenni.

Egyszerre csak egyet tudunk ugrani, dupla ugrás nem lehetséges, de elő lehet készíteni a következő ugrást, ha ugrás közbe megnyomjuk újra az ugrás gombot amint a földre ér a karakterünk egyből amint lehetséges ugrani fog. Földre éréskor újra ugorhatunk, illetve ugrás közben irányíthatjuk a leérkezést a balra és a jobbra nyíllak lenyomásával.

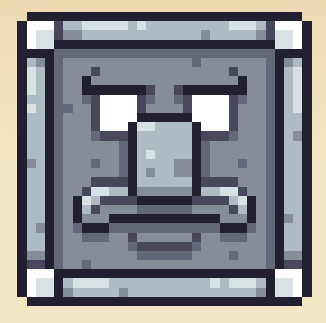
**2.6. A játékmenet**

A játék végső célja, hogy minél hamarabb eljussunk a célba. A cél a karakter, minél gyorsabb és biztonságosabb eljutása a célig. Az út viszont akadályokkal teli, kezdve a vérszomjas csapdákkal és a mély szakadékokkal, sok megpróbáltatás vár a cél előtt. AZ elején könnyebb, de ahogy haladunk tovább egyre nehezebb és nehezebb lesz az út.

**2.6.1. Ellenfelek**

A csapdákkal nagyon kell vigyázni mert végzetesek lehetnek. Hozzájuk érve a karakterünk sebződik és meghal ezzel vissza kerülve a pálya legelejére. Szakadékokkal is vigyázni kell mert ha lezuhanunk meghalunk és kezdhetjük a pályát a legelejéről közbe időt veszítve így gátolva a minél gyorsabb célba jutást.

**2.6.2. Kőfej**



**2.3. ábra. Kőfej**

Ha érzékeli, hogy alatta elhalad, egy játékos a magasból nagy gyorsasággal ránk zuhan, így megölve minket gátol a tovább jutásban, de ha elég gyorsak vagyunk, ezt könnyen el lehet kerülni.

**2.6.3. Tüskefej**

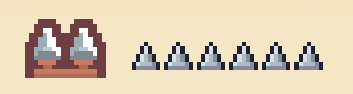
**A képen játék látható

Automatikusan generált leírás**

**2.4. ábra. Tüskefej**

Hasonlóan testvéréhez kőfejhez ő is érzékeli, ha egy játékos elhalod mellette csak ő nem csak lefele irányba képes érzékelni karakterünket, hanem mind a négy irányba képes karakterünket keresni és támadni.

**2.6.4. Tüskék**



**2.5. ábra. Tüskék**

Ha hozzájuk érűnk sebződik a karakterünk és egyből meghalunk ezzel növelve a halál számlálót

**2.6.5. Fürész**

A képen szöveg, molnár látható

Automatikusan generált leírás

**2.6. ábra. Fürész**

Képes jobbra és balra mozogni, de csak egy megadott távolságig ezzel nehezítve a pálya teljesítését. Ha a játékosunk hozzá ér egyből meghal így kezdhetjük az elejétől.

**2.6.6. Nyílcsapda**



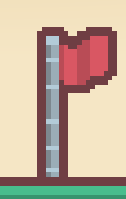
**2.7. ábra. Nyílcsapda**

Folyamatosan lövöldözi ki magából a nyilakat, ha pedig az egyik nyíl eltalálja a játékosunkat meghaltunk és kezdhetjük elölről.

**2.6.7. Élet**

Karakterünknek mindössze egy élete így nagyon oda kell figyelni mihez érünk hozzá. Két esetbe veszíthet életet a karakterünk. Az első, hogy szakadékba lépve leesni a pályáról, ebben az esetben az összes életünket elveszítjük, és kezdhetjük elölről a pályát. Második pedig, ha hozzá érünk a különböző csapdákhoz.

**2.6.8. Pálya teljesítése**



**2.8. ábra. Zászló**

A pályát csak is akkor teljesítettük, ha eljutottunk a zászlóig és hozzá értünk ekkor a jelenlegi idő és halálszámláló mentésre kerül.

**2.7. Mentés**

Egy pálya végig játszása utána mikor hozzá érünk a zászlóhoz egy távoli adatbázisba kerülnek mentésre az elért pályaszint, idő és a halál számláló. Így ezek az eredmények a játék újraindításakor vagy újra a számítógépre másolása után is megőrződnek.

**3. fejezet**

**Fejlesztői dokumentáció**

**3.1. Felhasznált technológiák**

**3.1.1. Unity**

A Trapland több technológiát is felhasznál működéséhez. Az Unity játékmotor, MySQL adatbázis és PHP kódok mind a program megfelelő működéséhez elengedhetetlen.

A Unity egy játékmotor, amelynek beépített IDE-jét a Unity Technologies fejlesztette ki. Számos platformra készíthető vele játék, köztük Windows, Mac, Linux, Android, Xbox One, PS4 platformokra. Három dimenziós, kétdimenziós, VR és augmented reality játékok is készíthetőek vele. Elérhető Windows-on, Mac-en és Linuxon, a használata ingyenes. [4]

**3.2. Architektúra**

A program architektúráját 3 rétegre bontjuk:

1. UI (User Interface) - felhasználói felület.
2. BL (Business Logic) - üzleti logika
3. DB (Database) – adatbázis

A felhasználói felület a felhasználóval való kommunikációért felelős. Egy jó felhasználói felület felhasználóbarát élményt nyújt, a felhasználó könnyen ráérez a program használatára.[5] Unity-ben a UIElements egy felhasználói felület eszközkészlet felhasználói felület készítéséhez. Az Event System közvetíti a felhasználói interakciókat a vizuális elemek felé.

Az üzleti logika (BL) ebben az esetben inkább játék logikának (GL, Game Logic)

nevezhető. Az alkalmazás funkcionalitását vezérli. Ide tartoznak a C# nyelven íródott scriptek, amik a játékot vezérlik.

A harmadik réteg az adatok tárolásával foglalkozik. A játék során elért eredmények tárolásáért és írásáért a 000webhost.com által üzemeltetett és fenntartott távoli szerverről elérhető MySQL adatbázis felelős.

**3.3. MonoBehaviour osztály**

A MonoBehaviour osztályban [6] található legfőbb metódusok, amiket én is használok a játékban, a következőek:

* **Awake:** Akkor hívódik meg ez a metódus, amikor a script töltődik.
* **Start:** A script indulásakor előszőr ez fut le, még az Update metódus előtt.
* **Update:** Minden képkockában meghívódik.
* Komponensekhez kötődő metódusok: például OnCollisionEnter2D, OnTriggerEnter2D, OnTriggerExit2D. Ezek a collider-ek és rigidbody-k érintkezésénél használt metódusok. Void a visszatérési értékük, és paraméterként azt a collider-t kapják meg, amivel az ütközés történt.

**3.4. Unity definíciók**

**3.4.1. GameObject és komponensek**

**GameObject:** A GameObject-ben (játékobjektumokban) rejlenek a játék legkritikusabb részei. Minden elem egy jelenetben egy játékobjektum, ezek önmagukban képtelenek cselekedni, mivel komponensekre van szükségük. Emellett a GameObject-ek menthetőek sablonokba (**Prefab**), amelyeket magunknak elkészített GameObject-eket és tulajdonságukat a projekt bármely részére beilleszthető és felhasználható. Fő előnye az, hogy a sablon tulajdonságait minden példány felveszi, a sablonban történő változásokat is beleértve.

**Komponens:** A GameObject-ek funkciói a hozzá kapcsolt komponensektől függ. A Unity rengeteg beépített komponenst tartalmaz, de a sajátunkat is elkészíthetjük. Néhány fontosabb komponens:

* A Transform komponens az objektum pozícióját, méretét és elforgatásának mértékét határozza meg. Minden GameObject alapvető komponense, nem törölhető belőle.
* Sprite Renderer komponens a karakter vagy játékelem képét jeleníti meg, amit sprite-nak hívunk.
* RigidBody2D a Unity Physics Engine-t használja a karakter mozgatásához. Ez szükséges ahhoz, hogy a karakter a fizika szabályainak megfelelően mozoghasson, ezzel élethűbbé válik a karakter mozgása.
* Collider-ek (CapsuleCollider2D, BoxCollider2D, CirlcleCollider2D) a más objektumokkal való érintkezést, ütközést teszik lehetővé. Triggernek jelölve a collider áthaladhat más collider-eken.
* Animator segítségével irányítjuk az animációkat. Az animációk működését egy irányított gráf formájában adhatjuk meg. Az animációkat átmenetekkel kapcsolhatjuk össze, az átmenetekhez feltételek adhatóak.
* Text komponenssel szövegdobozt jeleníthetünk meg.
* Image komponenssel adható kép az objektumhoz.
* Button komponenssel az objektum gombként funkcionál. Megadható hozzá függvény, ami meghívódik a gombra kattintáskor.
* Scriptek is hozzáadhatók komponensekként. Ezzel saját funkciókat adhatunk a GameObjectnek.

**3.4.2. Scene elemek**

**Scene:** A Scene-ek, magyarul színhelyek, a játék környezetét és menüit tartalmazzák. Színhelyekre gondolhatunk úgy, mint egy pályára, elhelyezhetők benne akadályok, tájelemek, dekorációk, illetve a menü és egyéb felületek.[7]

**Camera:** A kamera egy olyan eszköz, amely megjeleníti a világot a játékos számára. Testre szabható, scriptelhető, hogy szinte bármilyen hatást elérhető legyen vele.[8]

**Canvas:** A Canvas egy GameObject egy Canvas komponenssel. Minden UI elemnek ebben a GameObject-ben kell lennie. Ha még nincs Canvas a Scene-ben, egy UI elem hozzáadásával automatikusan létrejön.

**Event System:** Az Event System közvetíti a billentyűzet, egér és egyéb inputokat az objektumok felé.

**Grid:** Egy GameObject Grid komponenssel, ami a GameObject-ek elhelyezésében segít, cellákra osztja fel a Scene-t

**Tilemap:** Tile Asset-ek, azaz a pályát felépítő platformok tárolására és kezelésére alkalmas. Collider komponens adható hozzá. Grid komponenssel együtt, vagy egy Grid gyerek objektumaként használható.

**3.5. Scene elemek**

**Irodalomjegyzék**

**[1]**  <https://hu.wikipedia.org/wiki/Platformj%C3%A1t%C3%A9k>

**[2]** [**https://hu.wikipedia.org/wiki/Oldaln%C3%A9zetes\_akci%C3%B3j%C3%A1t%C3%A9k**](https://hu.wikipedia.org/wiki/Oldaln%C3%A9zetes_akci%C3%B3j%C3%A1t%C3%A9k)

**[3]** [**https://docs.unity3d.com/Manual/system-requirements.html**](https://docs.unity3d.com/Manual/system-requirements.html)

**[4]** [**https://en.wikipedia.org/wiki/Unity\_(game\_engine)**](https://en.wikipedia.org/wiki/Unity_(game_engine))

**[5]** [**https://techterms.com/definition/user\_interface**](https://techterms.com/definition/user_interface)

**[6]** [**https://docs.unity3d.com/ScriptReference/MonoBehaviour.html**](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)

**[7]** [**https://docs.unity3d.com/Manual/CreatingScenes.html**](https://docs.unity3d.com/Manual/CreatingScenes.html)

**[8]** [**https://docs.unity3d.com/Manual/class-Camera.html**](https://docs.unity3d.com/Manual/class-Camera.html)