Záródolgozat

Szincsák Szabolcs

2022. 04.

DSZC Beregszászi Pál Technikum

2D Platformjáték Unity-ben

Tartalomjegyzék

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 futtatásához 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

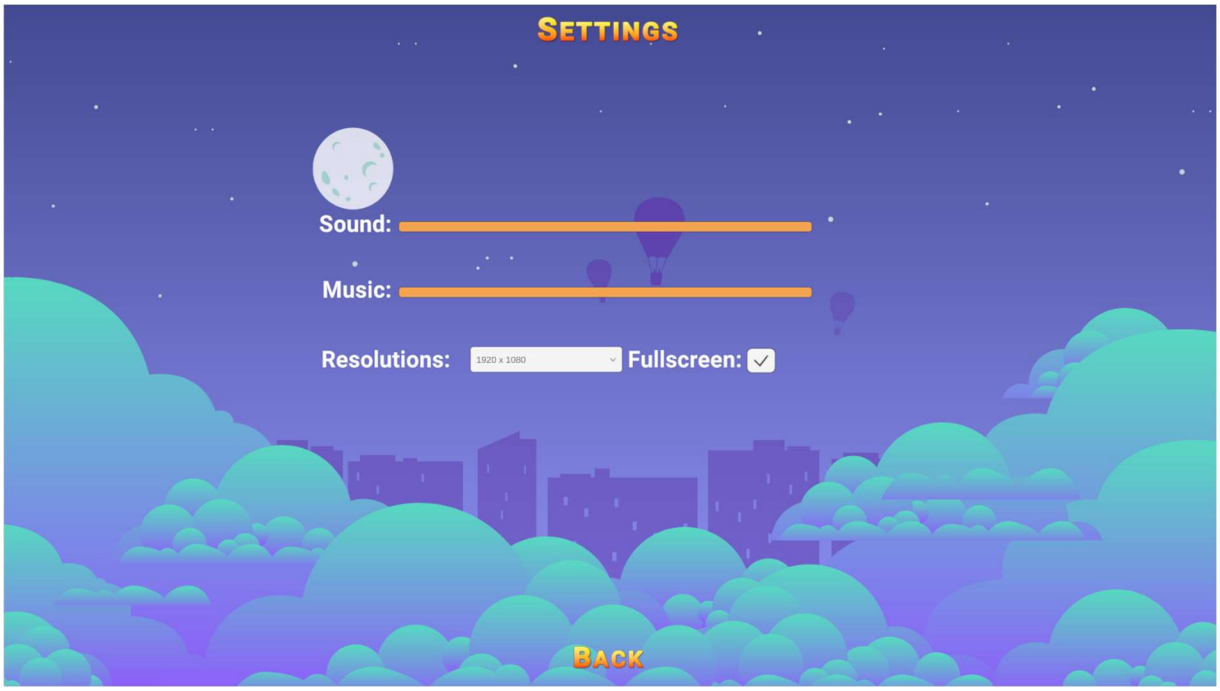
A játék indítása után a Regisztráció és Bejelentkezésnél találjuk magunkat. Itt regisztráció után be lehet jelentkezni és elkezdeni a játékot.

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.

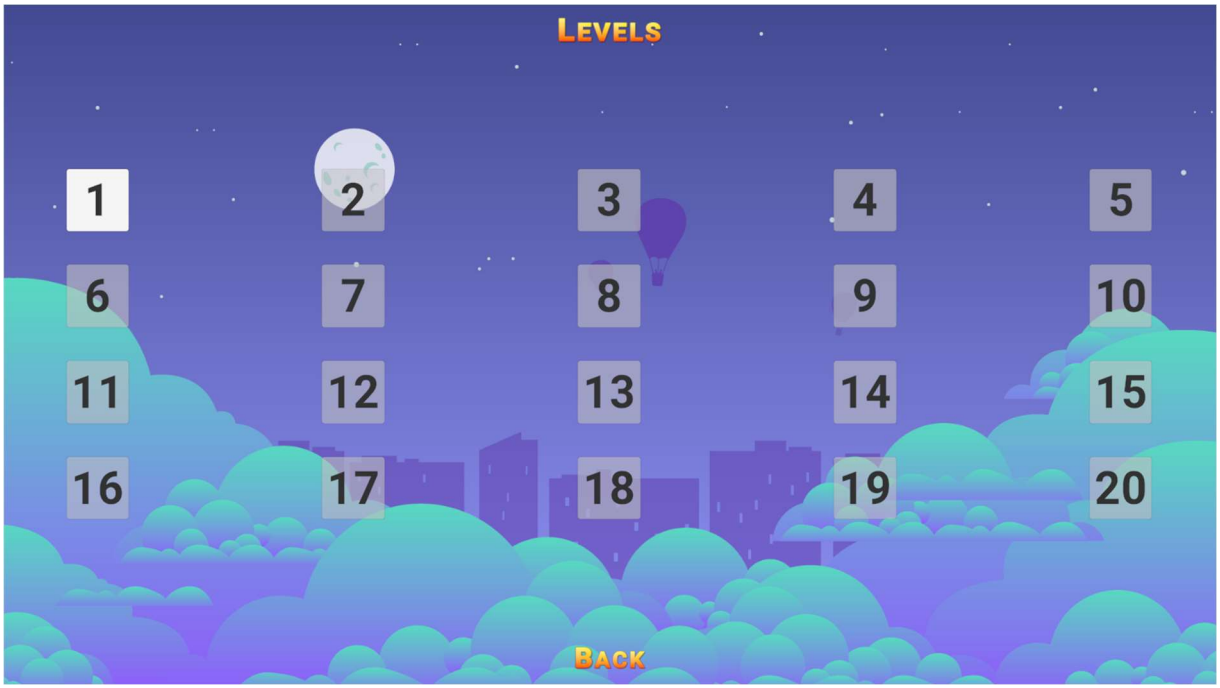
Teljes képernyős mód és ablakos mód között válthatunk a Fullscreen feliratú jelölőnégyzet segítségével. Ha bejelöljük akkor a játék teljes képernyős módra vált, ha megszüntetjük a bejelölést, akkor ablakos módra.

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.

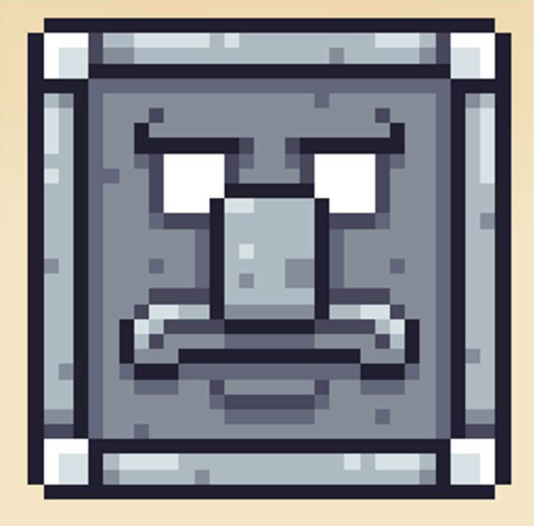
2.6. A játékmenet

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.

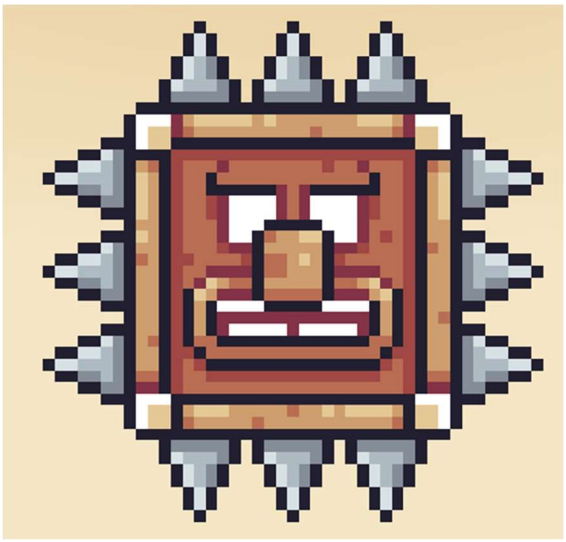
2.6.2. Kőfej



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



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

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 először 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. C# Programozási nyelv

A program C# (ejtsd szísárp) programnyelvet használja az Unity játékmotor szkript komponenseihez. A C# egy modern, objektum-orientált, típus-alapú programnyelv. A C# nyelv lehetővé teszi a fejlesztőknek, hogy fejlett, biztonságos alkalmazásokat készítsenek, amelyek .NET-ben futnak le. A C# a C programnyelvi családból származik, rokonnyelvei a C, C++, Java és a JavaScript. Mivel a C# egy objektum-orientált programnyelv, a nyelv felépítése ezeken az elveken alapszik, emiatt egy gyakran alkalmazott nyelv szoftverkomponensek készítésére is. Megjelenése óta a C# új munkafolyamatokkal és programtervezési gyakorlatokkal állt elő. A fejlesztő adja meg az adattípusokat és a tulajdonságukat.[9]

3.6. Backend

A backend a háttérben futó folyamatokkal foglalkozik, pl. szerveroldali programozással, űrlapon beküldött adatok feldolgozásával, statisztika készítéssel. A háttérben automatikusan lefut az adott programrész, anélkül, hogy a felhasználó beavatkozna. A játék rendelkezik egy felhasználói regisztrációs és bejelentkezés felülettel, ami 3 különböző programnyelvet is alkalmaz: C#, ezt a játékbeli, helyileg tárolt adatok begyűjtésére, a bekért adatok megfelelő formátumának hitelesítésére, illetve fogadására használt. Az adatok tárolásáért a https://www.000webhost.com/ által üzemeltetett és fenntartott távoli szerverről elérhető MySQL adatbázis felelős. Az Unity játékmotor karakterisztikája miatt itt szükséges implementálni egy közvetítőt az adatbázisnak, amit a https://www.000webhost.com/ szolgáltatásainak köszönhetően fenntartott PHP weboldal formájában van elkészítve. Az alkalmazáshoz a https://www.000webhost.com/ ingyenes csomagja van igénybe véve, ami 1 weboldalt, 300MB tárhelyet, 3GB havi adatforgalmat, 1 MySQL adatbázis, Cloudflare védelmet és 99%-os üzemidőt ígér.

3.6.1 MySQL adatbázis

A MySQL adatbázis a játékból érkező adatok fogadója és a lekért feladója itt van tárolva a felhasználónév, felhasználói azonosító, titkosított jelszó, a játékban elért szint és statisztikái. A játék innen lekérdezve jeleníti meg az helyileg nem tárolt adatokat.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	stats	Browse Structure Search Insert Empty Drop	38	InnoDB	utf8_unicode_ci	16.0 KiB	-
☐	username	Browse Structure Search Insert Empty Drop	14	InnoDB	utf8_unicode_ci	32.0 KiB	-
	2 tables	Sum	52	InnoDB	utf8_unicode_ci	48.0 KiB	0 B

3.1. ábra. MySQL adatbázis

A játék MySQL adatbázis struktúrája 2 adattáblából áll:

- username: Itt kerülnek tárolásra a felhasználó szükséges adatai, ami hitelesítéséhez elengedhetetlen.
- stats: A játékban különböző statisztikák kerülnek rögzítésre, ezeket itt szerepelnek, és a játék innen lekérdezve jeleníti majd meg őket.

username tábla

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	Username	varchar(16)	utf8_unicode_ci		No	*None*		
3	hash	varchar(100)	utf8_unicode_ci		No	*None*		
4	salt	varchar(50)	utf8_unicode_ci		No	*None*		
5	ReachedLevel	int(2)			No	*None*		

3.2. ábra. username tábla

Az username táblában a felhasználói hitelesítésre való adatok vannak eltárolva, a játék bejelentkezési felületén keresztül írható és olvasható az adatbázis. A regisztrációs felület használatakor egy sikeres regisztrációs folyamat esetén itt kerül rögzítésre az adatmezőkbe az adatok, és a bejelentkezéshez lekért adatok innen vannak lekérdezve hitelesítés céljából. Az username adattábla további négy adatból áll:

- **ID:** Egy elsődleges kulcs, a felhasználó egyedi azonosítója, nem szerepelhet ugyanaz az adat kétszer. Az azonosító egy 1-10 számjegyű egész szám lehet. A regisztrációs felületen ezt nem lehet megadni, a program automatikusan kiosztja a felhasználónak az azonosítószámot, és ez nem változhat, illetve nem átadható másik fióknak, és a felhasználói fiókhoz kötött fog maradni. Az azonosítókiosztás a felhasználónév, hash, salt és a ReachedLevel rögzítése után kiválasztja a létező legnagyobb számú azonosítót, és ezt inkrementálja eggyel, biztosítva, hogy teljesen egyedi maradjon az adat.
- **Username:** A felhasználónév ebbe a mezőbe van tárolva, a regisztrációs felületen történő sikeres regisztráció esetén a felhasználó által megválasztott név bekerül az

adatbázisba. Ez az adat nem ismétlődhet, ezt a szabályt a PHP-ban és phpMyAdminba ellenőrzi, megelőzve az esetleges duplikálásokat.

- **hash:** Egy kriptográfiai hash függvénnyel a felhasználó által megadott jelszót egy megadott hosszúságú van leképezve. Ennek a végterméke a hash.
- **salt:** Minden jelszóhoz generálódik egy egyedi salt, ez azt előzi meg, hogy ismételt jelszavaknál ne legyen ugyanaz a hash és salt kombináció.
- **ReachedLevel:** Az elért level (szint) amit a felhasználó elért.

stats tábla

#	Name	Type	Collation	Attributes	Null	Default
1	**Username_ID**	int(11)			No	*None*
2	**Level**	varchar(255)	utf8_unicode_ci		No	*None*
3	**Time**	varchar(255)	utf8_unicode_ci		No	*None*
4	**DeathCounter**	int(11)			No	*None*

3.3. ábra. stats tábla

A stats (statisztikák) tábla a játék végén elért teljesítmény összessége, tartalma minden játék végén frissül.

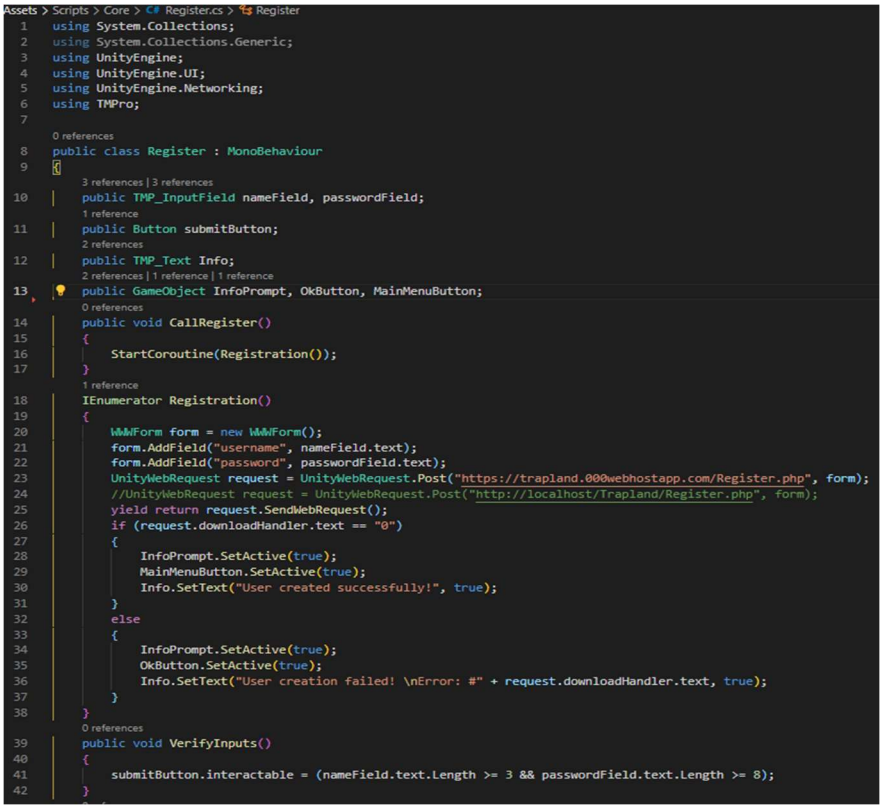
Username_ID: Ez a username táblában lévő ID-re való idegen kulcs hivatkozás. Ennek segítségével képes a program a megfelelő felhasználóhoz beírni a rögzítendő adatot.

Level: A pályaszint neve, ahol sikeresen végig ment a játékos.

Time: Az idő mialatt sikerült végig haladni a pályán.

DeathCounter: Az összes szám ahányszor sikerült meghalnia a karakternek a pálya végig vitele közbe.

3.6.2. PHP Backend



3.4. ábra. Register.cs

Ahhoz, hogy az Unity játékmotor létesíteni tudjon kapcsolatot a távoli MySQL adatbázissal, szükséges az UnityEngine.Networking osztály használata, és egy ehhez tartozó külső weboldal, ahonnan a PHP kód által végrehatott MySQL lekérdezéseken keresztül tud az alkalmazás küldeni és fogadni adatot.

A PHP egy webszerveren futó szkripteket támogató programnyelv, amelyeket általában dinamikus weboldalakhoz van alkalmazva, a programhoz a MySQL bővítményei vannak felhasználva.

A PHP fájlok, mint a MySQL adatbázis, a https://www.000webhost.com/ webszolgáltató ingyenes PHP és MySQL weboldal szolgáltatását használja.

A PHP backend 4 külön PHP állományból áll.

1. Register.php

Trapland > Register.php > ...

```php
<?php
$con = mysqli_connect('localhost','id18623828_root','L=M\7-dJvYwbe|dv','id18623828_trapland');
//check that connection happened
if(mysqli_connect_errno())
{
   echo "1: Connection failed";
   exit();
}
$username = $_POST["username"];
$password = $_POST["password"];
//Felhasználónév ellenőrzése
$namecheckquery = "SELECT Username FROM username WHERE Username='" .$username . "';";
$namecheck=mysqli_query($con, $namecheckquery) or die("2: Name check query failed.");
if (mysqli_num_rows ($namecheck)> 0)
{
   echo "3: Name already exists";
   exit();
}
//Adat hozzáadása
$salt="\$5\$rounds=5000\$" . "sajtoskenyer" . $username . "\$";
$hash= crypt($password,$salt);
$insertuserquery ="INSERT INTO username (Username, hash, salt, ReachedLevel) VALUES('" . $username . "', '" . $hash . "', '" . $salt . "', 1);";
mysqli_query($con, $insertuserquery)or die ("4: Insert Player query failed");
echo ("0");
?>
```

3.5. ábra. Register.php

A regisztrációs felületen a felhasználó által megadott adatok alapján létrehoz egy felhasználói fiókot. Felveszi a kapcsolatot az adatbázissal, és a játék Register.cs szkriptjéből POST metódussal érkező adatokat kivizsgálja a PHP kód megadott kritériumok alapján:

A PHP oldal megpróbál kapcsolatot teremteni az MySQL adatbázissal:

- Amennyiben ez nem sikerül, mint például a távoli szerver szolgáltatásának átmeneti kimaradása, vagy a felhasználói részről leállt vagy korlátozott internetkapcsolat esetén, akkor a regisztráció nem lehetséges, továbbá nem lehetséges a regisztráció funkció használata a probléma meg nem oldásáig.

A PHP kód ellenőrzi, hogy létezik-e már a felhasználó által kiválasztott felhasználónév:

- A felhasználónévnek egyedinek kell lennie. Ha már foglalt, akkor a felhasználónak egy másik nevet kell választania, ami még szabad, de erről a játék szól egy üzenetbe.

Ha ezeken a folyamatokon mind a program és felhasználó sikeresen végig haladt, akkor a program a MySQL adatbázis username táblájába beilleszti a felhasználónevet, a hashelt jelszót és a saltot, és foglal egy eddig nem használt ID-t.

Sikeres regisztráció esetén a program a ’ReachedLevel’ mező értéket beállítja egyre.

Sikeres regisztráció esetén az oldal a 0 kódot írja ki és a játékba kiírja, hogy sikeres regisztráció. Hiba esetén az oldal az adott hibakódot írja ki az üzenettel együtt.

2. Login.php

Trapland > Login.php > ...

```php
<?php
   $con = mysqli_connect('localhost','id18623828_root','L=M\7-dJvYwbe|dv','id18623828_trapland');
    //check that connection happened
    if (mysqli_connect_errno()) {
        echo "1: Connection failed"; //error code #1 = connection failed
        exit();
    }
    $username = $_POST["username"];
    $password = $_POST["password"];
    //check if name exists
    $namecheckquery = "SELECT Username, hash, salt, ReachedLevel FROM username WHERE Username='" . $username . "';";
    $namecheck = mysqli_query($con, $namecheckquery) or die("2: Name check query failed"); //error code #2 - name check query failed
   if (mysqli_num_rows($namecheck) != 1) {
       echo "5: Either no user with name, or more than one"; //error code #5 - number of names matching != 1
       exit();
   }
    //get login info from query
    $existinginfo = mysqli_fetch_assoc($namecheck);
    $hash = $existinginfo["hash"];
    $salt = $existinginfo["salt"];
    $loginhash = crypt($password, $salt);
   if ($hash != $loginhash) {
       echo "6: Incorrect password"; //error code #6 password does not hash to match table
       exit();
   }
    echo "0\t" . $existinginfo["ReachedLevel"];
    ?>
```

3.6. ábra. Login.php

A program bejelentkezési felületen a felhasználó a helyes bejelentkezési adatok megadása után a PHP kód felveszi a kapcsolatot a MySQL adatbázissal, és a POST metódust alkalmazva fogadja a program beviteli adatait a POST metódussal, majd egyeztet az adatbázis felhasználói adataival.

A PHP oldal megpróbál kapcsolatot teremteni az MySQL adatbázissal:

- Amennyiben ez nem sikerül, mint például a távoli szerver szolgáltatásának átmeneti kimaradása, vagy a felhasználói részről leállt vagy korlátozott internetkapcsolat esetén, akkor a regisztráció nem lehetséges, továbbá nem lehetséges a regisztráció funkció használata a probléma meg nem oldásáig.

A PHP kód lekérdezi a megadott felhasználónévvel megegyező adatsorokat a MySQL adatbázisból. Ez több céllal is szolgál:

- Ellenőrzi, hogy szigorúan csak 1 adatsort ad vissza a lekérdezés, ha nincs eredmény vagy több adatsor van, akkor a bejelentkezés nem sikerülhet, mivel vagy nem létezik ilyen felhasználónév, vagy több ugyanolyan felhasználónév létezik.

Ha csak 1 felhasználó létezik, akkor a felhasználónevet, hashet és a saltot egy asszociatív tömbbe helyezi.

A beviteli mezőben megadott jelszót a PHP kód hasheli, és egyezteti a salt-al. Amennyiben ez megegyezik az adatbázisban szereplő hashel, a bejelentkezés sikeres.

Sikeres regisztráció esetén az oldal a 0 kódot írja ki és a játékba kiírja, hogy sikeres regisztráció. Hiba esetén az oldal az adott hibakódot írja ki az üzenettel együtt.

3. Savedata.php

Trapland > Savedata.php > ...

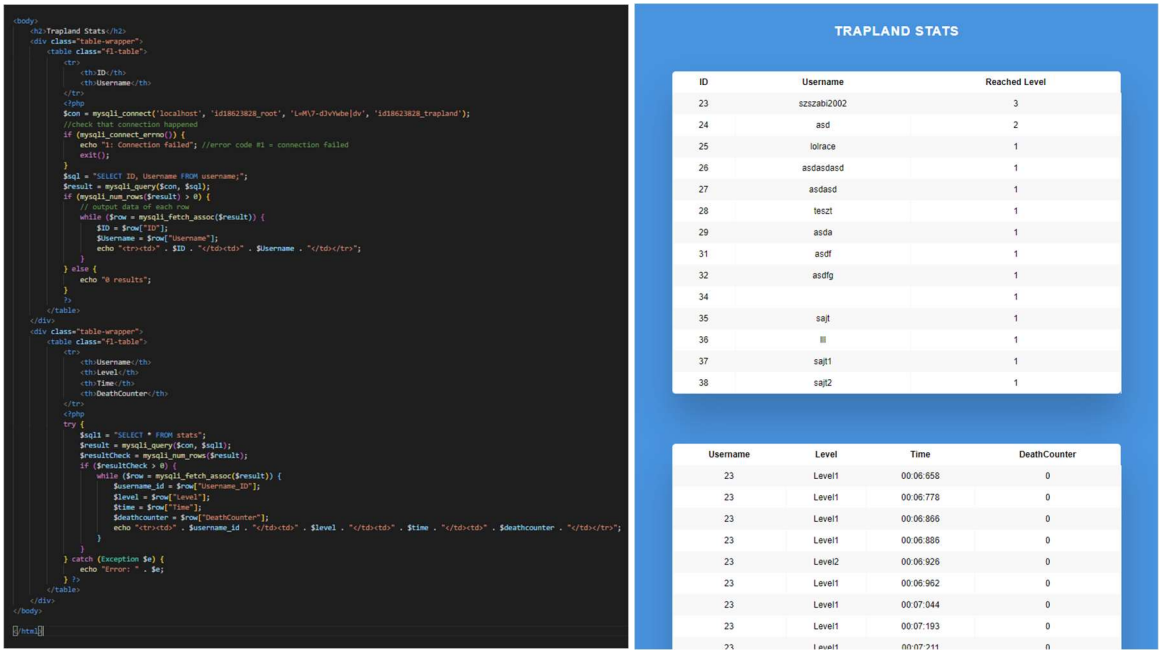
```php
<?php
$con = mysqli_connect('localhost', 'id18623828_root', 'L=M\7-dJvYwbe|dv', 'id18623828_trapland');
if (mysqli_connect_errno()) {
    echo "1: Connection failed"; //error code #1 = connection failed
    exit();
}
$username = $_POST["usernamePost"];
$level = $_POST["LevelPost"];
$time = $_POST["TimePost"];
$deathcounter = $_POST["DeathCounterPost"];
$reachedlevel = $_POST["ReachedLevelPost"];
$useridquery = "SELECT ID FROM username WHERE Username='" . $username ."';";
$userid = mysqli_query($con, $useridquery)->fetch_object()->ID or die("105: Fetching userID failed");
$namecheckquery = "SELECT Username FROM username WHERE Username='" . $username . "';";
$namecheck = mysqli_query($con, $namecheckquery) or die("2: Name check query failed."); //error code #2 - name check
if (mysqli_num_rows($namecheck) != 1) {
    echo "5: Either no user with name, or more than one"; //error code #5 - number of names matching != 1
    exit();
}
$updatequery = "UPDATE username SET ReachedLevel='" . $reachedlevel . "' WHERE Username='" . $username . "';";
mysqli_query($con, $updatequery) or die("7: Update query failed"); //error code #7 - update query failed
$insertquery = "INSERT INTO stats (Username_ID, Level, Time, DeathCounter) VALUES('" . $userid . "', '" . $level . "', '" . $time . "', '" . $deathcounter . "');";
mysqli_query($con, $insertquery) or die("9: Update query failed"); //error code #9 - update query failed
echo "0";
?>
```

3.7. ábra. Savedata.php

Miután hozzá érűnk a zászlóhoz menti az új adatokat az adatbázisba. A sikeres kapcsolatellenőrzés után a PHP oldal rögzíti a ’stats’ táblába Username_ID, Level, Time, DeathCounter-t és frissíti a ’username’ táblába lévő ReachedLevel mező értékét azzal, amit a játékba sikerült elérni.

Sikeres regisztráció esetén az oldal a 0 kódot írja ki és a játékba kiírja, hogy sikeres regisztráció. Hiba esetén az oldal az adott hibakódot írja ki az üzenettel együtt.

4. index.php



3.8. ábra. index.php

Összes statisztikai adat kiírásáért felelős

Sikeres regisztráció esetén az oldal a 0 kódot írja ki és a játékba kiírja, hogy sikeres regisztráció. Hiba esetén az oldal az adott hibakódot írja ki az üzenettel együtt.

3.7. Játékmenet

Bejelentkezés után a menübe kiválaszthatjuk melyik pályát akarjuk végig játszani azok közül, ami fel van oldva. A kiválasztott pálya betöltése után a cél, hogy a leggyorsabb módon eljutni a célba.

3.8. Tesztelés

3.8.1 Tesztelési terv

A fejlesztés során a tesztelés manuálisan történt. Unity-n belül a Play gomb megnyomásával tesztelhető. A játékot nem szükséges build-elni, gyorsan és egyszerűen kipróbálható a működése az IDE-n belül.

3.8.2. Tesztelt funkciók

Főmenü: Regisztráció, Bejelentkezés, Kilépés

Várt eredmény: A regisztráció és bejelentkezés csak a megadott feltételek mellett sikerül. A kilépés gomb sikeresen kilépteti a felhasználót és bezárja a játékot. A beléptetett felhasználó nevét kiíratja az alkalmazás, ha sikeres volt a bejelentkezés.

Kapott eredmény: A regisztráció és bejelentkezés sikeres az adott feltételek mellett, ezek hiányában a folyamatot nem lehet végrehajtani. Kilépéskor az alkalmazás bezárja magát. A belépett felhasználó neve helyesen megjelenik.

Főmenü: Pálya választó, Beállítások és Kijelentkeztetés

Várt eredmény: A kijelentkezés gomb sikeresen kilépteti a felhasználót, és a nem bejelentkezett státuszba kerül vissza az alkalmazás. A hangbeállítások mentve vannak minden alkalommal amikor a csúszka el van engedve, más ablakra fókuszáláskor. A pályaválasztó menübe csak annyi pálya van feloldva amennyit a felhasználó tényleg elért.

Kapott eredmény: Kijelentkezéskor az alkalmazás visszaáll az indításkori állapotba. A hangbeállítások mentve vannak sikeresen a csúszka elengedése után és az alkalmazásról levett fókusz esetén is. A pályaválasztón csak megadott pálya elérhető.

Játék: Információs panel megjelenítése

Várt eredmény: Az információs panelen megjelenik az idő és a halál számláló.

Kapott eredmény: Az információs panelen megfelelően megjelenik az idő és a halál számláló.

Játék: Szünet menü, pálya végi menü

Várt eredmény: Az „ESC” billentyű leütésével lehet bekapcsolni a szünet menüt, bekapcsolva a játékbeli elemek nem kattinthatók. A játék végén az összegzés helyesen megjelenik pontos adatokkal. A szünet és pálya végi menü gombjai megfelelően működnek.

Kapott eredmény: Az „ESC” billentyű leütésével lehet bekapcsolni a szünet menüt, a játékbeli elemek nem kattinthatók szünet alatt. A játék végén az összegzés helyesen megjelenik pontos adatokkal. A szünet és pálya végi menü gombjai megfelelően működnek.

4. Fejezet

Összefoglaló

A Trapland célja, hogy kihívásokkal teli platformer játék legyen. A záródolgozat feladata az egy egyjátékos kétdimenziós videójáték, amely egy MySQL adatbázishoz csatlakozik. Az előre eltervezett és a témabejelentőben meghatározott funkciókat sikerült megvalósítani. A technikumon szerzett tudásomat felhasználtam a fejlesztésbe, emellett sok tapasztalattal gazdagodtam játékfejlesztés területén, illetve Unity és C# ismereteimet is bővítettem. Szívesen foglalkoznék ezután is játékfejlesztéssel, hiszen egy játék készítése izgalmas és kreatív munka, de egyben kihívás is, ezért sok tapasztalat szerezhető vele. Egy játékot mindig lehet tovább fejleszteni, bővíteni, finomítani. Sok erre irányuló ötlet merült fel bennem a fejlesztés során, azonban ezekre idő hiányában nekem kevés lehetőségem volt, de szívesen megvalósítanék még sok funkciót a jövőben, vagy szívesen látnám, ha valaki kedvet kapna hozzá és továbbfejlesztené ezt a játékot.

3.4.1. Továbbfejlesztési lehetőségek

- Több pályát kéne készíteni.
- Játékon belüli globális toplista.
- Karakter mozgásának finomítása.
- Több és változatosabb csapdák.
- Fejlesztett biztonság.
- További kényelmi funkciók, finomítások a kezelőfelületen.
- Elfelejtett jelszó funkció.
- Bejelentkezett adat módosítás.
- Több háttér zene hozzáadása.

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

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

[4] https://en.wikipedia.org/wiki/Unity_(game_engine)

[5] https://techterms.com/definition/user_interface

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

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

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

[9] https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Ábrák Jegyzéke