ZÁRÓDOLGOZAT

Rónay István Vajk

SZOFTVERFEJLESZTŐ

BAJAI SZC TÜRR ISTVÁN TECHNIKUM

SZOFTVERFEJLESZTŐ

ZÁRÓDOLGOZAT

CAVE ADVENTURE

Rónay István Vajk

2024

NYILATKOZAT A ZÁRÓDOLGOZATRÓL

Alulírott .....*Rónay István Vajk*..... (név) tanuló

kijelentem, hogy

.....*Cave Adventure*..... című záródolgozatomat (nyomtatott és elektronikus formában) a Bajai SZC Türr István Gazdasági Szakgimnáziumának pedagógusai és tanulói:

- felhasználhatják (pl. hivatkozás alapjául, olvasótermi használatra) későbbi munkájukhoz a szerzői jogok tiszteletben tartása mellett).
- nem használhatják fel (titoktartási nyilatkozat csatolása mellett).

Ugyanakkor kijelentem, hogy a záródolgozat ***saját munkám eredménye*.**

Baja, 2021. április 29.

.....
aláírás

Tartalomjegyzék

1. Bevezetés

Azért választottam ezt a témát, mert mindig is érdekelt a játékkészítés és kifejezetten szeretem a metroidvania[1] típusú játékmenetet. Sok ilyen fajta játék van így megvan ennek is a maga közönsége. A játék elkészítése során használt programok a Visual Studio, Unity, GIMP, GitHub, XAMPP.

1.1. A dolgozat célkitűzései

- A saját ötletem elkészítése egy metroidvania játékhoz
- Saját világ realizálása

1.2. A dolgozat felépítése

1. Téma választása
2. Játékvilág tervezése
3. Grafikai elemek
4. Játékmenet
5. Adatbázis
6. Felhasználói dokumentáció
7. Fejlesztői dokumentáció
8. Összegzés

[1] A metroidvania az akció-kalandjátékok és/vagy platformjátékok egy alműfaja, amely az irányított, nem lineáris és segédeszköz-függő felfedezésre és fejlődésre összpontosít.

2. Projekt tervezése

A Projekt kódja, a kész játék, a tervek és a tesztelés elérhetőek az alábbi linken: https://github.com/shuffleV/Zaro-Dolgozat

A játék stílusa és mechanikái már az elején megvoltak, így a pálya dizájnját, a játék grafikáit az ellenfelek viselkedését, az akadályokat, a megszerezhető tárgyak implementálását és a pontok gyűjtését és szerzési módját kellett megterveznem.

2.1. Játékvilág megtervezése

Viszonylag gyorsan kitaláltam, hogy a játék egy barlang rendszerben fog játszódni. Ennek a rendszernek a külön régiói külön tematikával, mérettel és nehézséggel rendelkeznek.

- A Kezdő Terület egy egyszerű kék köves, törmelékes terület. Ez a terület főleg a játék világába való bevezetésre szolgál. A terület kialakítása viszonylag egyszerű, hogy a játkos megszokja az irányítást és megtanuljon néhány visszatérő játékelemet, mint az ellenfelek, a ládák és a törhető falak.
- A második terület, amit terveztem az a Növény Terület. Ez egy kissé élénkebb növényekkel benőtt barlang.
- Már a tervezés elején tudtam, hogy szeretnék egy Kristály Területet csinálni így ez az ötlet elég gyorsan megszületett. Ez a terület egy kristály lelőhely ezért a köveket néha átszúrják a kialakult kristályok.
- Tudtam, hogy a Végső Területnek egy sötéteb hangulatot szeretnék csinálni így ez egy nagyon sötét, majdnem fekete kőzetből épül fel.
- Kellet egy átmenet a Növény Terület és a Végső Terület közé mivel nem szerettem volna, hogy az élénkebb növény borította barlangból egyből egy ilyen sötét környezetbe kerüljön a játékos így eszerint terveztem meg az Elhalt Növény Területet. Egy átvezetés a két terület között mindkettő elemeinek a felhasználásával. Egy sötétebb terület helyenként elhalt növényekkel.
- Az utolsó hely, amit még meg kellet terveznem az a map bal oldalán található terület volt. Hosszas gondolkodás után úgy döntöttem, hogy kontrasztnak a sok sötét területhez kéne egy világosabb és letisztultabb terület ezért egy viszonylag világosabb szürke kőzetből épül fel, aminek mesterséges kinézete van.

2.2.Grafikai elemek

A játékban minden grafikát vagy saját magamtól terveztem és készítettem vagy pedig egy létező asset-et módosítottam, hogy megfeleljen a világnak és az elhelyezésének. Minden terület a saját tematikájával rendelkezik, ezért mindegyikhez új grafikát kellett készítenem, ami tartalmazza az építő elemeket, a ládákat, az akadályokat és az ellenfeleket is.

1. ábra Saját grafika [1]

2.3.Játékmenet

A játékosnak el kell jutnia a végső területre és ott le kell győznie az ott tartózkodó fő ellenfelet. Ennek az eléréséhez a játékosnak szüksége lesz 3 új képesség megszerzésére. Ezek a képességek a dupla ugrás, a falról ugrás és a dash[2]-elés. az oda vezető területeken találhatóak meg, amiknek az átjárásához át kell jutnia az ott tartózkodó ellenfeleken és az ott elhelyezett csapdákon.

[2] Hirtelen, előre futás

2. ábra Játékmenet Plant Area [2]

3. ábra Játékmenet Dead Plant Area [3]

2.4.Adatbázis

A riv_zarodolgozat nevű adatbázis egy „highscore” nevű táblát tartalmaz. A táblában 3 oszlop található:

- **ID:** A játékos azonosítója, int típusú, Elsődleges kulcs, auto increment be van kapcsolva, maximum 11 karakter hosszú

- **User:** A játékos által beírt felhasználónév, varchar típusú, maximum 9 karakter hosszú
- **Highscore:** A játékos által elért legmagasabb pontszám, int típusú, maximum 11 karakter hosszú

A játék során ládákból és ellenfelek megölésekor a játékos változó mennyiségű pontokat kap és ezt az összpontszámot a játék végén láthatjuk összehasonlítva és rangsorolva a többi játékossal, akik végig vitték a játékot. Erre a célra használom az adatbázist. A játék befejezésekor megkéri a játékost, hogy adja meg a nevét és ennek beküldésekor felveszi az adatbázisba a megadott nevet és hozzá az elért pontszámot. Ha már szerepel az adatbázisban a játékos neve, akkor csak akkor tölti fel a pontszámot, ha az nagyobb az eddigi legnagyobb elért pontszámánál. Ez után a játék ki írja az elért pontszámot a képernyőre és alatta megjeleníti az adatbázisban szereplő 5 legnagyobb pontszámmal rendelkező játékos nevét és pontszámát.

3. Felhasználói dokumentáció

Rendszerkövetelmény, irányítás, fejlesztések, képességek, akadályok, ellenfelek, pontok

3.1. Rendszerkövetelmények

Operációs rendszer: Windows 7

Processzor: Intel Core 2 Duo E5200

Videokártya: NVIDIA GeForce 9800GTX

Memória: 4 GB

Tárhely:

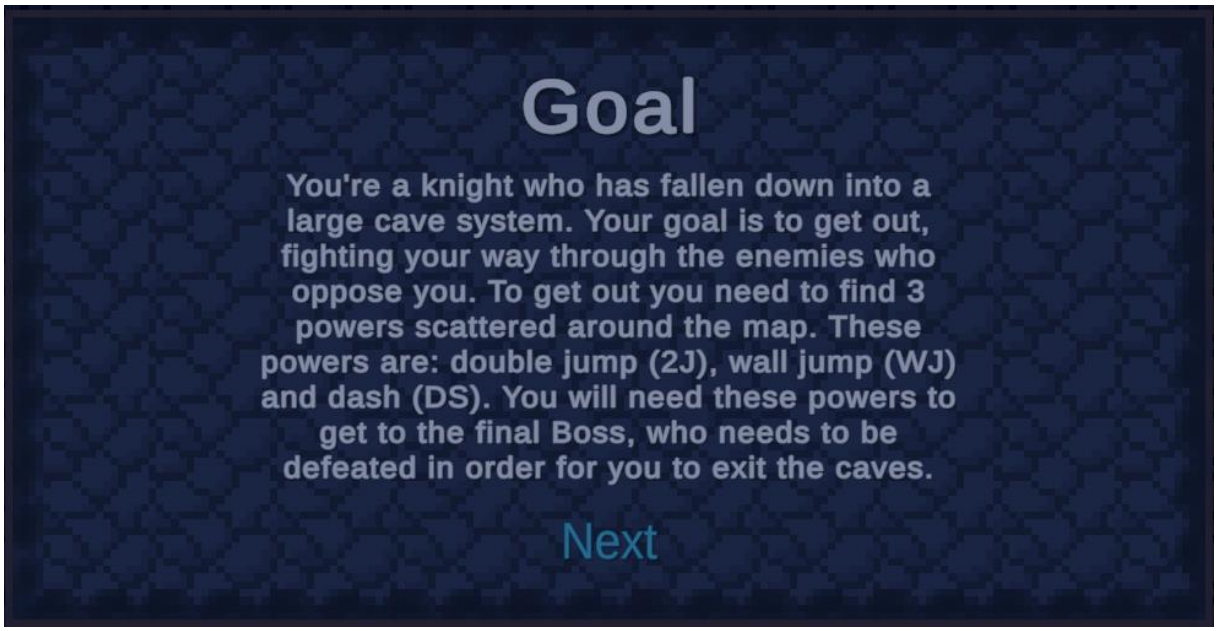
3.2. Fő Menü

A fő menüben két opciót választhatunk ki, a „Start Game”-et és a „Quit”-et. A „Start Game”-el tovább lépünk a bevezető képernyőhöz, a „Quit”-el pedig ki lépünk a programból.

4. ábra Főmenü [4]

3.3. Bevezető képernyő

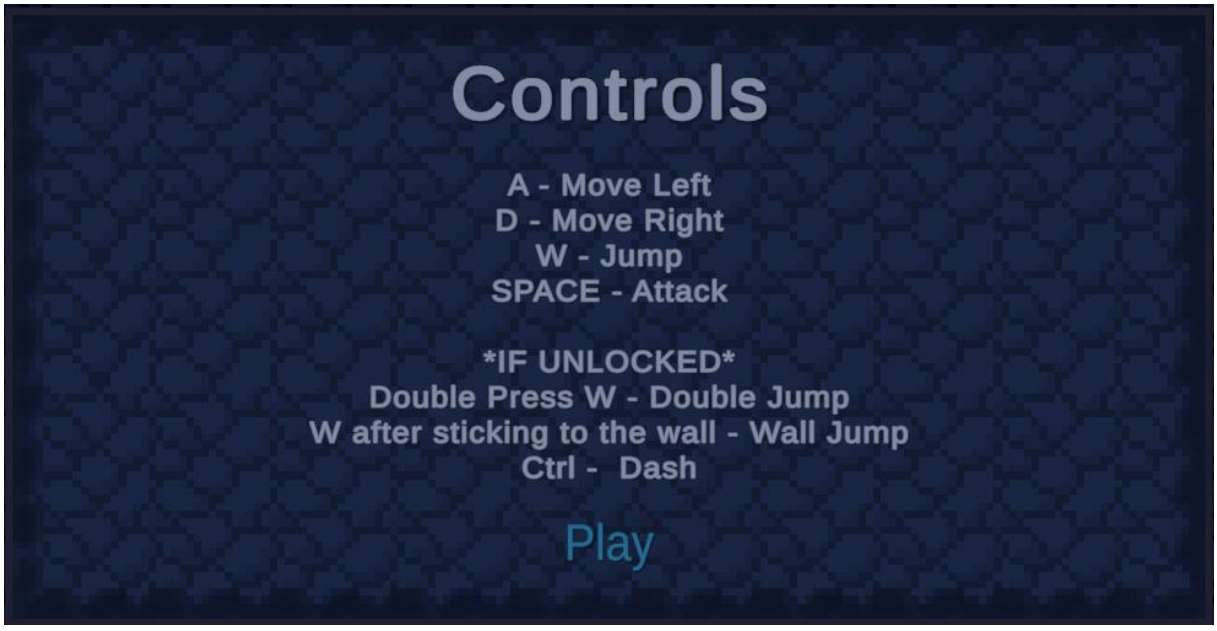
Itt láthatjuk, hogy mi a játéknak a célja, és hogy mit kell tennünk annak eléréséhez. A „Next” gombbal tovább lépünk az irányítások képernyőre.



5. ábra Bevezető képernyő [5]

3.4.Irányítások képernyő

Itt találhatóak a játék irányításai. A „Play” gombbal elindítjuk a játékot.



6. ábra Irányítások képernyő [6]

3.5.Irányítás

3.5.1. Alap irányítások

- A – Balra
- D – Jobbra
- W – Ugrás
- SPACE – Támadás
- Esc – Pause Menu

3.5.2. Megszerzendő képességek irányítása

- LCtrl – Dash
- Dupla W – Dupla ugrás
- Falon W – Fal ugrás

3.6.Játékos statisztikái

- Alap életerő: 100
- Alap sebzés: 25

3.7.Megszerezhető játékos fejlesztések

- Dupla ugrás – Megszerezhető az Elhalt Növény Területen
- Fal ugrás – Megszerezhető a Világos területen
- Dash-elés – Megszerezhető a Kristály Területen
- 3 db Sebzés fejlesztés
- 5 db Életerő fejlesztés

3.7.1. Statisztika fejlesztések

- Sebzés fejlesztés: +5 Sebzés, Mini Boss-ok legyőzése után kapjuk
- Életerő fejlesztés: + 20 Életerő, A map-on külön területeken szerezhetőek meg

3.8.Ládák

A ládák külön helyeken elhelyezkedő, összetörhető tárgyak. Egy ütéssel össze lehet törni és pontokat ad cserébe. A pontok mennyisége területtől függően változik.

- **Kezdő terület láda: 500 pont**
- **Növény terület láda: 1000 pont**
- **Halott növény terület láda: 1250 pont**
- **Világos terület láda: 1250 pont**
- **Kristály terület láda: 1750 pont**
- **Végső terület láda: 2500 pont**

3.9.Pontok

A játék alatt ládákból és ellenfelek legyőzéséből tudunk pontokat szerezni, amiket a játék a képernyő bal alsó sarkában számol. A pontok mennyisége függ a területtől, attól hogy ládából vagy ellenfélből van, és hogy milyen fajta ellenfelet győzünk le. Ezek a pontok alapján számolja ki a játék a végén megjelenő rangsort.

3.10. Ellenfelek

Minden területen különböznek az ellenfelek, változnak a statisztikáik és a kinézetük.

3.10.1. Kezdő Terület

- Kezdő ellenfél: Lassabb, mint a játékos, 100 életereje van, 5-öt sebez, 250 pontot ad, ha legyőzzük

3.10.2. Növény Terület

- Sima növény ellenfelek: Lassabb, mint a játékos, 100 életereje van,10-et sebez, 500 pontot ad, ha legyőzzük
- Növény Mini Boss: Jelentősen lassabb, mint a játékos, 300 életereje van, 20-at sebez, 10000 pontot ad és sebzésfejlesztést, ha legyőzzük

3.10.3. Elhalt Növény Terület

- Sima elhalt növény ellenfelek: Lassabb, mint a játékos, 150 életereje van, 15-öt sebez, 750 pontot ad, ha legyőzzük
- Erős elhalt növény ellenfelek: Lassabb, mint a sima ellenfelek, 175 életereje van, 20-at sebez, 900 pontot ad, ha legyőzzük

- Elhalt növény Mini Boss: Jelentősen lassabb, mint a játékos, 350 életereje van, 30-at sebez, pontokat ad, ha legyőzzük, 15000 pontot ad és sebzésfejlesztést, ha legyőzzük

3.10.4. Világos Terület

- Sima világos ellenfelek: Lassabb, mint a játékos, 200 életereje van, 20-at sebez 750 pontot ad, ha legyőzzük
- Erős világos ellenfelek: Lassabb, mint a sima ellenfelek, 250 életereje van, 25-öt sebez, 900 pontot ad, ha legyőzzük
- Világos Mini Boss: Jelentősen lassabb, mint a játékos, 350 életereje van, 30-at sebez, pontokat ad, ha legyőzzük, 15000 pontot ad és sebzésfejlesztést, ha legyőzzük

3.10.5. Kristály Terület

- Sima kristály ellenfelek: Lassabb, mint a játékos, 200 életereje van, 15-öt sebez, 1000 pontot ad, ha legyőzzük
- Erős kristály ellenfelek: Lassabb, mint a sima ellenfelek, 250 életereje van, 20-at sebez, 1250 pontot ad, ha legyőzzük

3.10.6. Végső Terület

- Sima végső ellenfelek: Lassabb, mint a játékos, 250 életereje van, 30-at sebez, 2250
- Erős sötét ellenfelek: Lassabb, mint a sima ellenfelek, 300 életereje van, 35-et sebez, 2750 pontot ad, ha legyőzzük
- Fő Boss: Lassabb, mint a sima ellenfelek, 100 életereje van, 45-et sebez, pontokat ad, ha legyőzzük, 50000 pontot ad, ha legyőzzük

3.11. Akadályok

- Platformok: A rájuk való ugrással lehet tovább jutni
- Tüskék: Át kell őket ugrani, ha hozzáér a játékos, meghal
- Törhető falak: Ezek a falak elállják az utat, de egy ütéssel el tudja törni a játékos, hogy mögé juthasson

3.12. Játék vége

Az „End Screen” a játék befejezésekor, vagyis a főellenség legyőzésekor jelenik meg. Ezen a képernyőn láthatjuk az elért pontszámot és beírhatjuk a nevünket a felhasználónév mezőbe, amit a „Save Highscore” gomb megnyomásával az elért pontszámmal együtt feltölthetünk az adatbázisba, vagy létező felhasználónév esetén frissíthetjük a meglévő pontszámunkat amennyiben az alacsonyabb az új pontszámtól. A felhasználónévnek meg kell felelnie a leírt szabályoknak miszerint a felhasználónév legalább 3 és legfeljebb 9 karakter hosszú lehet, és nem tartalmazhat szóközt és/vagy speciális karaktereket. Ezeket a szabályokat a képernyő bal oldalán meg tudjuk tekinteni, ha a felhasználónév mező alatt elhelyezkedő „Username Rules” feliratú kapcsolót bekapcsoljuk.

7. ábra Játék vége [7]

Miután megnyomtuk a „Save Highscore” gombot, tovább fog irányítani a Leaderboard nevű képernyőre.

3.13. Ranglista

Ezen az oldalon tudjuk megtekinteni az 5 legjobb elért pontszámot és a játékosok nevét, akik elérték azokat.

8. ábra Ranglista [8]

A „Quit’ gombra kattintáskor a program bezárul.

4. Fejlesztői dokumentáció

A fejlesztéshez használt környezet és programok, valamint a fejlesztés alatt létrehozott és felhasznált scriptek, asset-ek és az elkészítés segítésére használt videók kijegyzése.

4.1. Fejlesztő környezet

Operációs rendszer: Windows 10 Pro

Rendszer típus: 64 bites operációs rendszer, x64-alapú processzor

Processzor: AMD Ryzen 5 2600X Six-Core Processor 3.60 GHz

Videokártya: NVIDIA GeForce GTX 1660 Ti

Memória: 16 GB

4.2. Fejlesztéshez használt programok

Visual Studio[3] és hozzá a Game development with Unity: A játék kódjának megírásához használt program. Ennek a programnak a segítségével írtam meg a játék működéséhez szükséges scripteket.

- **Verzió:** Visual Studio Community 2019 16.11.31
- **Programozási nyelv:** C#

XAMPP[4]: Az adatbázis futtatásához használt program. Ennek a segítségével hoztam létre a játékhoz használt adatbázist.

- **Verzió:** XAMPP Control Panel v3.3.0

GIMP[5]: A játék grafikáinak létrehozásához és szerkesztéséhez használt program. Ebben a programban hoztam létre vagy szerkesztettem, módosítottam a játék által használt grafikai elemeket.

- **Verzió:** GIMP 2.10.36

Unity[6] Hub és Editor: A Unity Hub-ból tudjuk elérni a különböző verziójú editor-okat és a projektjeinket. A játék elkészítéséhez használt játék motor. Itt készül el maga a játék. Ebben a programban készülnek el a jelenetek, az objektumok, a játék szerkezete, a felépítése, valamint

[3] https://visualstudio.microsoft.com/
[4] https://www.apachefriends.org/hu/index.html
[5] https://www.gimp.org/
[6] https://unity.com/

itt kerülnek felhasználásra a többi program segítségével létrehozott elemek, mint a script-ek és az assetek, amikből felépül a játék.

- **Verzió:**
    - **Unity Hub:** Unity Hub 3.7.0
    - **Unity Editor:** Unity 2022.3.13f1

GitHub Desktop[7]**:** Verziókezelésre használt program. A játék verzióit nyomon követi így vissza lehet nézni a változtatásokat és vissza is lehet tölteni egy régebbi verziót változtatás vagy akár hiba javítás szempontjából is.

4.3. Adatbázis

Az adatbázis tárolja a játék végén beírt játékosnevet és a hozzá tartozó legmagasabb elért pontszámot. A játék végén kiírja a benne tárolt 5 legmagasabb pontszámot elérő játékost a pontszámával együtt.

Az adatbázis beüzemeléséhez le kell tölteni az XAMPP nevű alkalmazát, futtatni, elindítani az Apache és MySQL modulokat és beimportálni a játékhoz tartozó riv_zarodolgozat nevű sql kiterjesztésű fájlt.

4.4. Script-ek

Itt vannak a játék működéséhez szükséges scriptek. A script-ek külön funkciókat látnak, el a játékon belül azonban vannak olyan script-ek, amelyeknek együttes működésével megy végbe egy feladat, más script-ek pedig a saját feladatuk végrehajtásához felhasználnak egy másik script-ben lévő metódust.

4.4.1. PlayerMovement.cs

Ebben a script-ben van benne a játékos irányításának a kódja, az ugrás és a jobbra, balra való mozgás, valamint a megszerezhető képességek,a dupla ugrás, a falról ugrás és a dash-elés irányítása

```
horizontal = Input.GetAxisRaw("Horizontal");
        if (!isWallJumping)
        {
            physics.velocity = new Vector2(horizontal * 7f, physics.velocity.y);
        }


        if (havedoublejump)
```

[7] https://desktop.github.com/

```
        {
            if (IsOnGround() && !Input.GetButton("Jump"))
            {
                doublejump = false;
            }

            if (Input.GetButtonDown("Jump"))
            {
                if (IsOnGround() || doublejump)
                {
                    physics.velocity = new Vector2(physics.velocity.x, 16f);

                    doublejump = !doublejump;
                }
            }

            if (Input.GetButtonUp("Jump") && physics.velocity.y > 0f)
            {
                physics.velocity = new Vector2(physics.velocity.x, physics.velocity.y
* 0.5f);
            }

            if (havewalljump)
            {
                WallSlide();
                WallJump();

                if (!isWallJumping)
                {
                    Flip();
                }
            }

            Flip();
        }
```

4.4.2. PlayerLife.cs

Ez a script kezeli a játékos aktuális életerejét, maximum életerejét, sebzését, a sebzés indikátorját, a gyógyulást, maximum életerő fejlesztést és a játékos halált.

```
private IEnumerator DamageIndicator(Color color)
    {
        GetComponent<SpriteRenderer>().color = color;
        yield return new WaitForSeconds(0.15f);
        GetComponent<SpriteRenderer>().color = Color.white;
    }

public void Damage(int amount)
    {
        if (amount < 0)
        {
            throw new System.ArgumentOutOfRangeException("Nem lehet negatív sebzés");
        }
        this.health -= amount;
        healthbar.SetHealth(health);
        StartCoroutine(DamageIndicator(Color.clear));

        if (health <= 0)
        {
```

```
            PlayerDeath();
        }
    }

    public void Heal(int amount)
    {
        if (amount < 0)
        {
            throw new System.ArgumentOutOfRangeException("Nem lehet negatív
gyógyítás");
        }

        if (health + amount > MAX_HEALTH)
        {
            this.health = MAX_HEALTH;
        }
        else
        {
            this.health += amount;
        }
    }


    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("HP"))
        {
            MAX_HEALTH += 20;
            healthbar.SetMaxHealth(MAX_HEALTH);
            health = MAX_HEALTH;
        }
        else if (collision.gameObject.CompareTag("Spikes"))
        {
            PlayerDeath();
        }
    }
```

4.4.3. PlayerAttack.cs

A PlayerAttack script felel a játékos támadásának aktiválásának lekezelésével. A szóköz lenyomására meghívja az Attack metódust. Az Attack metódus igazra állítja az attack változót és aktiválja az attackArea-át. Ha az attack változó igaz akkor a beállított idő után deaktiválja.

```
void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            Attack();
        }

        if (attack)
        {
            timer += Time.deltaTime;

            if (timer >= timeToAttack)
            {
                timer = 0;
                attack = false;
                attackArea.SetActive(attack);
```

```
            }
        }
    }

    private void Attack()
    {
        attack = true;
        attackArea.SetActive(attack);
    }
```

4.4.4. AttackArea.cs

Ez a script a második fele a játékos támadásának végrehajtásában. Az AttackArea script végzi el az ellenfelek sebzését valamint a ládák és a törhető falak törését.

```
private void OnTriggerEnter2D(Collider2D collider)
    {
        if (collider.GetComponent<EnemyHealth>() !=null)
        {
            EnemyHealth health = collider.GetComponent<EnemyHealth>();
            health.Damage(damage);
        }
        else if (collider.GetComponent<BreakableWall>() !=null)
        {
            BreakableWall IsBroken = collider.GetComponent<BreakableWall>();
            IsBroken.WallBreak();
        }
        else if (collider.GetComponent<BreakableWall>() !=null)
        {
            BreakableWall IsBroken = collider.GetComponent<BreakableWall>();
            IsBroken.WallBreak();
        }

    }
```

4.4.5. EnemyHeatlh.cs

Az EnemyHealth kezeli le az ellenfelek életerejét, a kapott sebzését, halálát és a sebzés indikátorját. Felveszi az előrre megadott életerőt, és ha sebzést kap, akkor kivonja belőle. Ha az életerő eléri a 0-át, akkor az ellenfél meghal.

```
public void SetHealth(int health)
    {
        this.health = health;
    }

    public void Damage(int amount)
    {
        if (amount < 0)
        {
            throw new System.ArgumentOutOfRangeException("Nem lehet negatív sebzés ");
        }
        this.health -= amount;
        StartCoroutine(DamageIndicator(Color.clear));

        if (health <= 0)
        {
            Death();
        }
    }
```

4.4.6. Enemy.cs

Az Enemy script felelős az ellenfelek viselkedéséért, támadásáért és mozgásáért.

```
void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        SetEnemyValues();
    }

void Update()
    {
        Attack();
    }

private void SetEnemyValues()
    {
        GetComponent<EnemyHealth>().SetHealth(data.hp);
        damage = data.damage;
        speed = data.speed;
    }

private void Attack()
    {
        if (Mathf.Abs(player.transform.position.x - transform.position.x)<10 &&
Mathf.Abs(player.transform.position.y - transform.position.y) < 3.5f)
        {
            transform.position = Vector2.MoveTowards(transform.position,
player.transform.position, speed * Time.deltaTime);
        }
        if (Mathf.Abs(player.transform.position.x - transform.position.x) < 2 &&
Mathf.Abs(player.transform.position.y - transform.position.y) < 2)
```

```
        {
            if (Time.time > cooldown)
            {
                cooldown = Time.time + cdtime;
                player.GetComponent<PlayerLife>().Damage(damage);
            }
        }
    }
```

4.4.7. EndUI.cs

Az EndUI script-ben történnek meg a játék befejezésekor megjelenő képernyővel kapcsolatos események. Ez a script több dologért is felelős valamint kapcsolatot teremt az adatbázissal.

```
string connStr =
"server=localhost;user=RIV;database=riv_zarodolgozat;port=3306;password=Admin123";
```

Indításkor a játékban elért pontszámot kiírja a képernyőre. Ez után lekérdezi az adatbázisból az eddig rögzített neveket és a hozzájuk tartozó pontszámokat, amelyeket ez után rögzít egy listában.

```
FinalScoreText.text = $"Elért Pontszám: {score}";
        Debug.Log(score);

        MySqlConnection conn = new MySqlConnection(connStr);
        try
        {
            conn.Open();
            string sql = "SELECT `Username`, `Highscore` FROM `highscore` WHERE 1";
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            MySqlDataReader rdr = cmd.ExecuteReader();

            while (rdr.Read())
            {
                username.Add(rdr[0].ToString() + " " + rdr[1].ToString());
            }
        }
        catch (System.Exception ex)
        {
            Debug.Log(ex.ToString());
        }
```

Az EndScene képernyőn található mező kódja is itt található, amely először lerögzíti, a beírt nevet majd ellenőrzi, hogy megvan-e minimum 3 betű, maximum 9 betű-e és van-e benne szóköz vagy speciális karakter. Ha a nem felel meg a beírható hossznak vagy tartalmazza a tiltott karaktereket, akkor igazra állítja a hibaüzenet aktív állapotát. Ha viszont megfelel a követelményeknek, akkor viszont hamisra állítja.

```
bool specialchar = false;
        text = user;
        Debug.Log(text);
```

```
        string specialChar = @"Speciális karakterek";
        foreach (var item in specialChar)
        {
            if (text.Contains(item))
                specialchar = true;
        }
        if (specialchar||text==""||text.Length<3||text.Length>9)
        {
            UserError.SetActive(true);
        }
        else
        {
            UserError.SetActive(false);
        }
```

Itt található az eredmény adatbázisba való rögzítéséért felelős kód is. Ez a kódrész először ellenőrzi, hogy aktív-e a felhasználónév helytelen beírásakor megjelenő hibaüzenet. Ha aktív, akkor nem megy végbe semmi, ha viszont nem aktív, akkor végbe megy a rögzítési folyamat. A kód ellenőrzi, hogy a beírt név szerepel-e az adatbázisból kapott nevek listájában, ha szerepel, akkor megnézi, hogy a játékmenet alatt szerzett végső pontszám nagyobb-e a már adatbázisban szereplő pontszámánál és egy változóba beírja, hogy már létező a felhasználónév. Amennyiben a jelenlegi pontszám kisebb a már rögzítettnél, a rögzítés nem megy végbe, viszont ha nagyobb, akkor felülírja a régit mivel az adatbázisban csak a játékos által elért legnagyobb pontszám szerepel.

```
foreach (var item in username)
            {
                if (item.Split(' ')[0]==text)
                {
                    if (score>Convert.ToInt32(item.Split(' ')[1]))
                    {
                        try
                        {
                            conn.Open();
                            string sql = $"UPDATE `highscore` SET `Highscore`={score}
WHERE `Username`='{text}'";
                            MySqlCommand cmd = new MySqlCommand(sql, conn);
                            cmd.ExecuteNonQuery();
                        }
                        catch (System.Exception ex)
                        {
                            Debug.Log(ex.ToString());
                        }
                    }
                    vane = true;
                }
            }
```

Miután végig ment ezen a kódrészen, megnézi, hogy hamis-e a változó, ami azt jelzi, hogy az előző kódrészben talált-e egyezést a nevek listájában. Amennyiben hamis, akkor új adatként rögzíti a játékosnevet és a pontszámát, ha hamis, akkor ezt a lépést kihagyja és folytatja

a kód futtatását. Ezek után már csak az van hátra, hogy elindítja a harmadik jelenetet, a Leaderboard-ot.

```
if (vane==false)
            {
                try
                {
                    conn.Open();
                    string sql = $"INSERT INTO `highscore` (`Username`, `Highscore`)
VALUES ('{text}','{score}')";
                    MySqlCommand cmd = new MySqlCommand(sql, conn);
                    cmd.ExecuteNonQuery();
                }
                catch (System.Exception ex)
                {
                    Debug.Log(ex.ToString());
                }
            }
            SceneManager.LoadScene(3);
```

4.5.Assetek

4.5.1. Pixel Adventure 1[8]

Ezeket az asset-eket főleg ideiglenes grafikaként használtam, de a grafikák nagy része az ebben található fájlok átalakításával vagy teljes helyettesítésével jöttek létre.

[8] https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360

4.6.Program megírásában segítő videók

- Coding in Flow nevű csatorna Build a 2D Platformer in Unity | Unity Beginner Tutorial[9] nevű lejátszás lista első pár videója segített elkezdeni és megtanulni a 2 dimenziós játékok készítését.
- Modding by Kaupenjoe nevű csatorna Unity BEGINNER Tutorials[10] nevű lejátszás lista videói szintén segítettek bizonyos játék elemek megértésében és használati módjában.
- bendux nevű csatorna által készített videók, név szerint How To Double Jump In Unity[11], How To Wall Slide & Wall Jump In Unity[12] és How To Dash In Unity[13] segítettek a megszerezhető képességek kódjában

[9] https://www.youtube.com/playlist?list=PLrnPJCHvNZuCVTz6lvhR81nnaf1a-b67U
[10] https://www.youtube.com/playlist?list=PLKGarocXCE1H7pJk6k_CSwS359mtt3MVI
[11] https://www.youtube.com/watch?v=RdhgngSUco0
[12] https://www.youtube.com/watch?v=O6VX6Ro7EtA
[13] https://www.youtube.com/watch?v=2kFGmuPHiA0

5. Összegzés

Ezzel a játékkal az volt a célom, hogy létrehozhassak egy általam kitalált interaktív világot. A játék készítése közben kezdtem látni, hogy kezd megvalósulni ez a világ és minél többet adtam hozzá annál jobban kezdett kibontakozni. Emellett programozás téren elég sok új dolgot tanultam a játék készítés folyamatáról, mint például a Tilemap készítése és a különböző objektumok közötti interakciók létrehozása. Időnként frusztráló, de mégis érdekes volt megtapasztalni, hogy néha az egyszerűnek tűnő dolgokat nem is olyan könnyű elkészíteni, mint ahogy azt hinné az ember, de ennek a fordítottja is igaz. Előfordult az is hogy egy nagyobb, nehezebbnek tűnő feladatról kiderül, hogy nem is annyira bonyolult az elvégzése, mint amire először számítottam. A grafikák elkészítése során meglepődtem, hogy habár egyszerű 16bit-es grafikáról van szó mégis egész jóra sikeredtek. A tervek alapján egész egyszerűen sikerült elkészíteni az ellenfeleket és a fő ellenfeleket, a ládákat is meglepően egyszerű volt megvalósítani akárcsak a törhető falakat. A játék végén megjelenő képernyő szintén nem volt nagy kihívás, habár akadt egy két dolog, amivel meggyűlt a bajom, mint például a különböző objektumok közötti interakció és az adatbázis adatai használatának a realizálása, de szerencsére végül sikerült megoldani ezeket és végül jobb is lett, mint amire számítottam.

A tervek nagy része sikeresen elkészült ugyanakkor akadt 3 dolog, aminek az implementálása nem ment végbe. Az első hogy nem minden területen van fő ellenség, ennek oka, hogy készültek területek, amelyeknek a kialakításába nem illettek volna bele, emellett rá jöttem, hogy viszonylag repetitív lett volna, ha minden terület végén ugyan az a fajta kihívás várna a játékosra. A második nem megvalósult terv az a gyógyító italok. Ezek főként azért nem kerültek bele a játékba, mert a checkpoint-ok az elérésükkor teljesen feltöltik a játékos életerejét és viszonylag gyorsan vissza lehet futni ezekre a helyekre. Emellett a kezdő területen kívül minden területen található egy maximum életerő fejlesztés, ami szintén teljesen feltölti a játékos életerejét és úgy gondoltam, hogy ezek mellé még több gyógyító tárgy feleslegessé vált volna. Végül az utolsó dolog, aminek az implementálása nem ment vége az a Sötét terület. Ennek a területnek a készítése a fejlesztési folyamat nagy részében még tervben volt, de a map készítése közben rá kellett jönnöm, hogy ahogy alakult a végső kinézet, egyre kevésbe illett bele az összképbe így hát úgy döntöttem, hogy végül nem lesz benne a kész verzióban.

Összességében úgy gondolom, hogy sokat tanultam a játékkészítési folyamatról és az ez által szerzett tapasztalatok miatt, ha adott az alkalom, nagyobb önbizalommal és jobb rálátással tudok majd belekezdeni egy újabb projektbe.

Fejlesztési lehetőségek:

- Több fajta ellenfél
- Új területek, amik tovább építik a világot
- Újabb megszerezhető képességek
- Játékon belüli bolt, ahol el lehet költeni a pontokat különböző képességekre és fejlesztésekre

Irodalomjegyzék

Programok

MICROSOFT VISUAL STUDIO:

https://visualstudio.microsoft.com/ (Megtekintés ideje: 2024.04.29.)

XAMPP:

https://www.apachefriends.org/hu/index.html (Megtekintés ideje: 2024.04.29.)

GIMP:

https://www.gimp.org/ (Megtekintés ideje: 2024.04.29.)

UNITY:

https://unity.com/ (Megtekintés ideje: 2024.04.29.)

GITHUB:

https://desktop.github.com/ (Megtekintés ideje: 2024.04.29.)

Assetek

Pixel ADVENTURE 1

https://assetstore.unity.com/packages/2d/characters/pixel-adventure-1-155360
(Megtekintés ideje: 2024.04.29.)

Videók

BUILD A 2D PLATFORMER IN UNITY | UNITY BEGINNER TUTORIAL

FELTÖLTŐ: CODING IN FLOW

https://www.youtube.com/playlist?list=PLrnPJCHvNZuCVTz6lvhR81nnaf1a-b67U
(Megtekintés ideje: 2024.04.29.)

UNITY BEGINNER TUTORIALS

FELTÖLTŐ: MODDING BY KAUPENJOE

https://www.youtube.com/playlist?list=PLKGarocXCE1H7pJk6k_CSwS359mtt3MVI
(Megtekintés ideje: 2024.04.29.)

HOW TO DOUBLE JUMP IN UNITY

FELTŐLTŐ: BENDUX

https://www.youtube.com/watch?v=RdhgngSUco0 (Megtekintés ideje: 2024.04.29.)

HOW TO WALL SLIDE & WALL JUMP IN UNITY

FELTÖLTŐ: BENDUX

https://www.youtube.com/watch?v=O6VX6Ro7EtA (Megtekintés ideje: 2024.04.29.)

HOW TO DASH IN UNITY

FELTÖLTŐ: BENDUX

https://www.youtube.com/watch?v=2kFGmuPHiA0 (Megtekintés ideje: 2024.04.29.)

Ábrajegyzék