Liceul Teoretic „Arany János”
Arany János Elméleti Líceum

Szatmari Alex-Levente

Lucrare de atestat la informatică
Informatika szakdolgozat

2023

Liceul Teoretic „Arany János”
Arany János Elméleti Líceum

2D Platformer
C#

Készítette:
Szatmari Alex-Levente

Irányítótanár:
Nagy Ildikó

2023

Tartalomjegyzék

Bemutatás

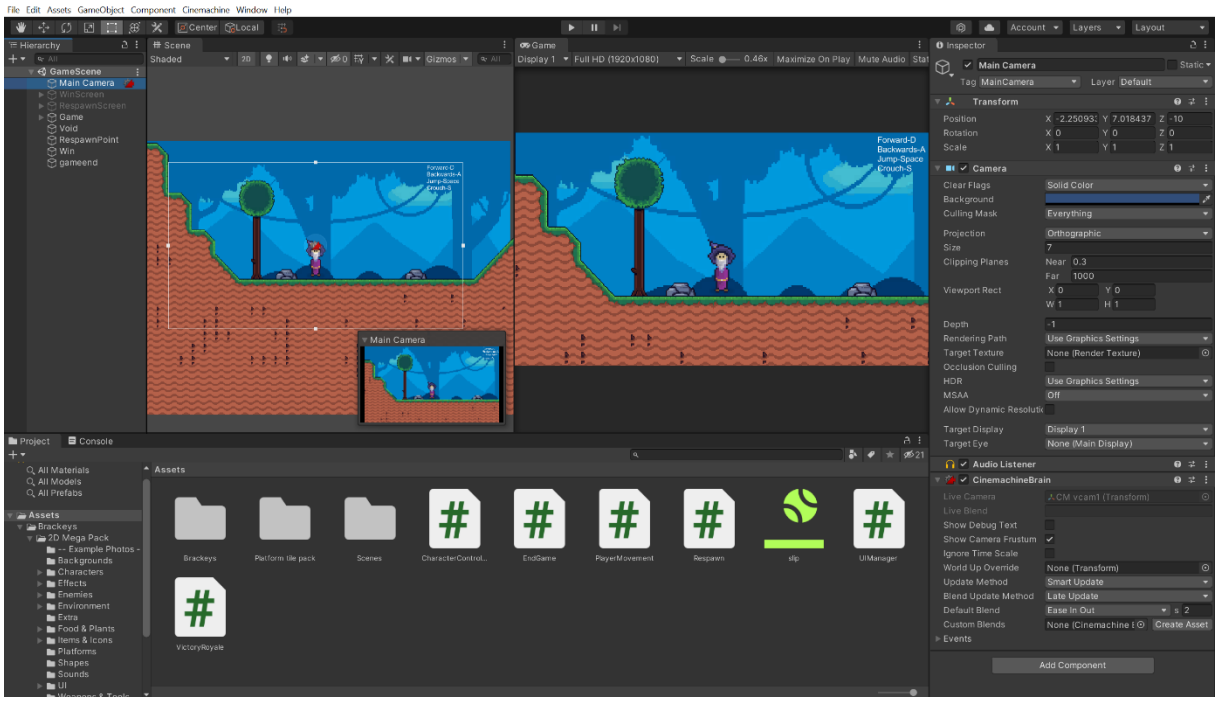
A C# programozási nyelvnek jelentős szerepe van a videójátékok fejlesztésében és az egyik leghasználtabb ezen téren.

Majdnem az összes 2D és egyes esetekben 3D játékok ennek segitségével készülnek, mivel a nyelvet egyszerű megtanulni és nagyon jól bánik az adatok és felhasználói bevitelek feldolgozásával.

A Videójátékok fejlesztéséhez több “engine” létezik, magyarul olyan alkalmazások amik segitenek a programozási részt összhangba hozni a 2D-s elemekkel és objektumokkal. Ezek közül a legismertebbek a Unity és az Unreal Engine, amelyek segitségével készülnek napjainkban is hiresebb cimek mint a Fortnite, Tomb Raider.

Jelentése a mindennapokban

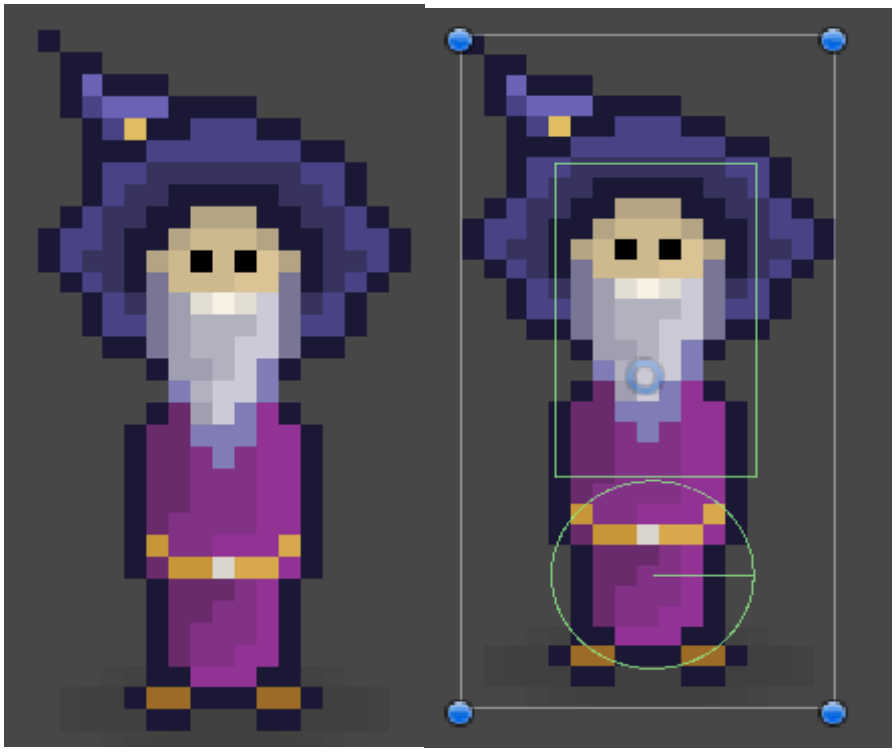
Az egyik legnagyobb ipar a videójátékipar ami rengeteg pénzt hoz be a fejlesztőknek és egyes esetekben a fejlesztőcég országát is jutalmazza. Rengeteg munkalehetőséget nyit meg a kódolók előtt és egy kellemes időtöltést biztosit.



(Demonstráció a projektről Unityben)

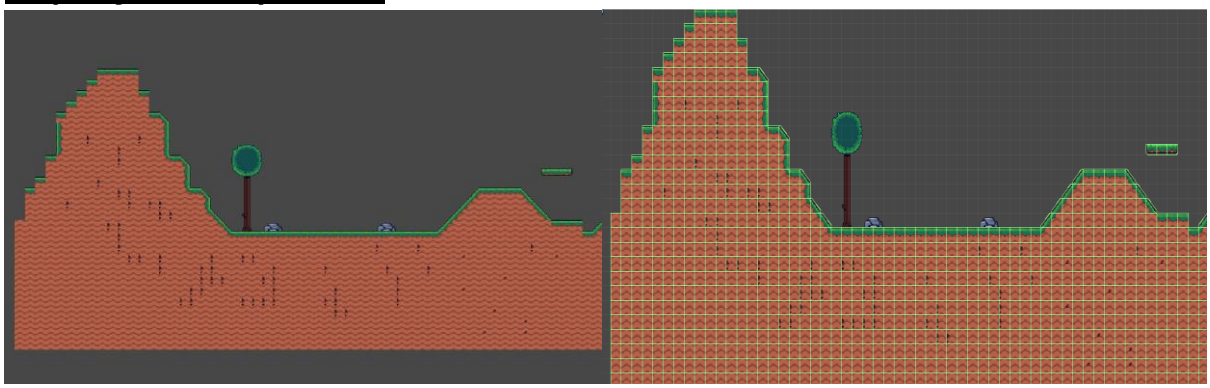
Felépitése

A karakter felépitése



A karakter egy 2D ben ábrázolt modell, egy varázsló. Amint látható két részre van felosztva az ütközője(Objektum amivel határokat adunk meg). A felső rész egy téglalap, hogy meghatározza a karakter méretét és formáját. Az also része egy kör, ami azért szükséges, hogy a mozgás zökkenőmentes legyen még a nem sima felületeken is.

A pálya felépitése



A pálya felépitése egy egszerű képek összessége. Rendelkezésünkre áll egy panel (Tilemap) ami segitségével kiválasztjuk a bizonyos “kockát” amit leszeretnénk rakni. A Unit engine magától leképez rá egy ütközőt , amit a zöld vonalak indikálnak, ez biztositja hogy a karakter nem esik le a semmibe.

Logikai háttér

Mozgásrendszer

A mozgásrendszer eléggé complex egy ilyen egszerű játk esetén is. Minden bevitt billentyűlenyomást felismer és átalakitja mozgási paraméterekbe. Ugyebár számitásba kell venni a gyorsulást, sebességet és gravitációt is. Szerencsére a programozási felület megenegedi, hogy direktbe a lenyomott gombokra rakjunk be cselekményeket és ez jóval megkönnyíti a logikai részét.

```
public class PlayerMovement : MonoBehaviour
{
    public CharacterController2D controller;
    public float runspeed = 40f;
    float horizontalMove = 0f;
    bool jump = false;
    bool crouch = false;

    // Update is called once per frame
    void Update()
    {
        horizontalMove = Input.GetAxisRaw("Horizontal") * runspeed;
        if (Input.GetButtonDown("Jump"))
            jump = true;
        if (Input.GetButtonDown("Crouch"))
        {
            crouch = true;
        }
        else if (Input.GetButtonUp("Crouch"))
        {
            crouch = false;
        }

    }
    void FixedUpdate()
    {
        controller.Move(horizontalMove * Time.fixedDeltaTime, crouch, jump);
        jump = false;


    }
}
```

runspeed = konstans sebessége a karakternek mikor mozog

bool jump/crouch = az ugrás és gugolás érzékelése igaz hamis feltétellel

horizontalMove = Mozgás a sikban, ha lenyomásra kerülnek a Horizontal csoportba lévő gombok(A,D, jobb-bal nyíl).

Input.GetButtonDown = Ha lenyomásra kerül a gomb

Input.GetButtonUp = Ha felengedik a gombot

Controller.Move(x, y, z) – egy külső alprogram ami a karakter elemeit összeszinkronizálva változtatja a poziciokat a következők függvényében: mozgás a síkban, gugolás érteke (i/h) és ugrás értéke(i/h);

Újraéledés

Egy videójátékban létfontosságú az újrajátszás lehetősége így ha meghal a karakter van rá lehetőség hogy újraéledjünk. A következő kód segítségével ez lehetséges is.

```
public class Respawn : MonoBehaviour
{
    public void RestartGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
    public bool alive = true;
    [SerializeField] private Transform player;
    [SerializeField] private Transform respawnPoint;
    public GameObject gameOverMenu;
    public GameObject gameOverlay;
    void OnTriggerEnter2D(Collider2D other)
    {
        gameOverMenu.gameObject.SetActive(true);
        player.transform.position = respawnPoint.transform.position;
    }


}
```

SceneManager = A jeleneteket rendezi, megnézi melyik jelenet aktiv és melyiket tölti be.

Transform player, átalakitja a játékos pozicióit számokba és kódba.

Transform respawnPoint, átalakitja a pontot amelyhez visszakell rakja a karakter újraéledéskor.

gameOverMenu, piros képernyő, a játékos meghalt.

A player pozicióját átváltja az újjáéledési pontra igy visszakerül az elejére.

Működése

A játék egy rövid “platformer”, vagyis kétdimenziós elemeket tartalmaz, célja eljutni A pontból B pontba anélkül, hogy meghaljunk. A fentebb említett rendszerek segítségével ez zökkenőmentesen kivitelezhető és egy dinamikus játékélményt nyújt.

A mozgásért felelős gombok a következők:A és D – mozgás az x tengelyen, Space – mozgás az y tengelyen avagy ugrás és S – gugolás. A megfelelő mozgásokat hasznaálva és kikerülve az akadályokat amik megjelennek a pályán eljutunk a győzelmi képernyőhöz , viszont ellenkező esetben a játék végetér és újra kell kezdeni az elejétől.

Rövid demonstrációs videó a játékról: https://www.youtube.com/watch?v=HZUfPqx4Zpo

Forrás

Az elkészitéshez használt weboldalak:

https://github.com/

https://stackoverflow.com/

https://www.youtube.com/c/Brackeys

https://assetstore.unity.com/

https://unity.com/