Debrecen 2022.04

ZÁRÓDOLGOZAT

Jakab Dániel

Fejlesztői dokumentáció

A választott programozási nyelv

A **Unity** egy videójáték-motor, amelyet a Unity Technologies fejleszt. A Unityban háromdimenziós, illetve kétdimenziós játékokat lehet létrehozni. Többek között nagy előnye, hogy a szoftver képes nagyméretű adatbázisokat kezelni, kihasználni a kölcsönhatások és animációk képességeit. Biztosítani Továbbá használható geometriai eszközcsomagok továbbítására, illetve viselkedési elemek hozzáadására egyes objektumokhoz. Ezek mellett a játékmotor folyamatosan megőrzi a végleges változat megjelenítését.

Egyéb szoftver

000webhost egy külső szerver, ahol az adatbázist bárhol bármikor elérhetjük.

Unityban lévő kód nyelv

C#

A 90-es években a **Microsoft** a saját Java keretkörnyezetét a saját operációsrendszer-specifikus függvényeivel és szolgáltatásaival bővítette ki, amire viszont nem volt engedélyük Sun Microsystemstől. A Microsoft-féle Javára fejlesztett alkalmazások nem lettek volna futtathatók más rendszereken, ami ellenkezik a Java platform-függetlenségre vonatkozó alapelvével.
A Sun végül beperelte a Microsoftot, és ezt követően a Microsoft a Java eltávolítására kényszerült a Windows rendszerekből. A Microsoft egy saját keretrendszer fejlesztésébe kezdett. Ez lett a .NET és ehhez a keretrendszerhez adták ki programnyelvként a C# első verzióját.

PHP

A **PHP** egy általános szerveroldali szkript nyelv dinamikus weblapok készítésére. Az első szkript nyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a webszerver PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre. Rasmus Lerdorf 1995-ben találta ki a nyelvet.
Ma a The PHP Group tartja fenn és fejleszti. A PHP szabad szoftver, de licence nem csereszabatos a GNU licenccel, mivel megkötéseket tartalmaz a PHP név használatára.

A záródolgozat témája

A Zelda nevezetű játék tovább gondolása. Felülnézeti szerep játék (rpg/role playing game) ahol kisebb ellenfelekel kell megküzdeni és minél jobb fegyvereket/páncélt kell szerezni, Amiket lehet hordani és használni. A cél, hogy minél több ellenséget kell megölni halál nélkül. Halál esetén a játék az elejéről kezdődik.

Regisztráció és bejelentkezés

A regisztrációt és a bejelentkezés a PHP nyelv segítségével oldottam meg. Először is egy c# kódban hivatkozni kell a PHP-ra és a beviteli mezőkre, ahol az adatokat kérjük be (felhasználó név és a jelszó).

Regisztráció

```
using UnityEngine.UI;
using UnityEngine.Networking;

public class registration : MonoBehaviour
{
    public InputField nameField;
    public InputField passwordField;

    public Button submitButton;
    public void CallRegister()
    {
        StartCoroutine(Register());
    }
    IEnumerator Register()
    {
        WWWForm form = new WWWForm();
        form.AddField("name", nameField.text);
        form.AddField("password", passwordField.text);
        UnityWebRequest request = UnityWebRequest.Post("https://mrblz.000webhostapp.com/register.php", form);
        yield return request.SendWebRequest();
```

1.A regisztráció kódja

A UnityEngine.Networking segítségével lehet a webhosthoz csatlakozni.

A nameField és a passwordField egy beviteli mező, ahol bekérjük a felhasználót és a jelszót.

Az adatbázishoz csatlakozást PHP-ban oldottam meg.

A bejelentkezésnél pedig az adatbázisban tárolt felhasználó nevet és a jelszót ellenőrzi, amit bejelentkezés oldalon megadott a felhasználó és ha egyezik akkor sikeresen bejelentkezett.

A php részében először is csatlakozni kell az adatbázishoz utána ellenőrzi, hogy sikeres-e a csatlakozás. Utána a c#-ből bekérd felhasználónevet és a jelszót és egy változóba tároljuk és egy parancs sorral feltöltjük az adatokat.

```
$namecheckquery = "SELECT felhasznalonev FROM felhasznalo WHERE felhasznalonev='" .$username . "';";
```

Utána ellenőrzi, hogy a megadott felhasználó név nem foglalt-e. A jelszót pedig titkosítjuk, avagy crypteljük a salt és a hash paranccsal majd a titkosított jelszót is eltároljuk az adatbázisban. A végén ellenőrzi, hogy sikeresen belekerült-e az adatbázisba.

Bejelentkezés

Mint a regisztrációnál itt is azzal kell kezdeni, hogy az adatbázishoz kell csatlakozni És itt is ellenőrizni kell a csatlakozást. Aztán megint a c#-ből bekért felhasználó nevet és jelszót bekérjük. A következőbben annyi a különbség a regisztrációnál, hogy a regisztrációban felvettük az adatot, de itt lekérjük.
Ezután ellenőrzi, hogy bejelentkezésnél érvényes felhasználó nevet add meg a játékos. A jelszónál decrypteli a cryptel jelszót és utána ellenőrzi, hogy érvényes jelszót adott-e meg a felhasználó.

A Játék

Játékos Mozgása

Először egy a karakter mozgását oldottam meg. Az volt az első, hogy fel, le, ballra és tudjon menni. A megoldás az lett, hogy az X és az Y tengelyen változik a helyzete, ha lenyomjuk a mozgáshoz a gombokat W= előre, S=hátra, D=balra, A=jobbra.

A pálya

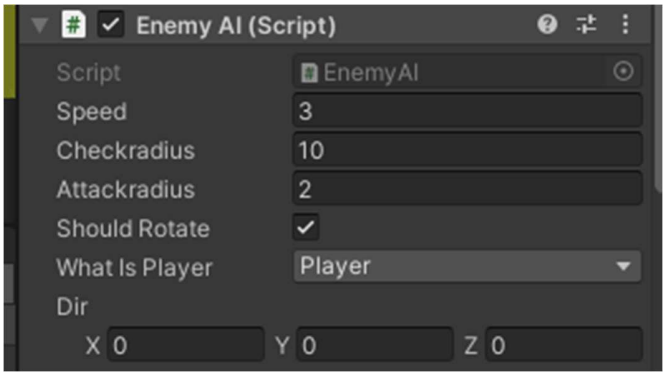
Következőnek a játék pályáját csináltam meg. Mivel egy felül nézetes játék ezért az úgy is kell kinéznie a pályának. Először is egy gridet helyeztem le, ami fel kockázta a játékot. Erre került egy layer, amin látható a fű és az út. A következő layeren vannak az épületek tárgyak és egyebek.

Azért volt fontos külön layerekre tenni ezeket mivel a falaknak és a többi tárgynak kell lennie egy Collider, ami segít, hogy ne menjünk át a falakon és csak neki menjünk, avagy állítson meg. A harmadik Layer hasonít a másodikra csak annyiban más, hogy itt csak a vizet tárolom, ami segít, hogy a játékos ne hagyja el a megadott pályát.

Ellenfelek

AI

Az ellenségek az a célja, hogy megölje a játékost. Meg lehet neki az Editorban adni a sebességét, a támadási erejét, a látás körét, és a támadási körét. Míg a játékos nincs a közelében addig egyhelyben van és alszik, de ha a látás körében van akkor elkezd felé menni amíg el nem éri a támadási körét és csak akkor támad.



2. Ellenfél sebessége és sugarai.

A What is Playerben be lehet állitani annak a Gameobjectnek a tagjét, amit követnie kell.

Ellenfelek élete

Az életnél külön meg kell nézni a teljes élet szintet és az aktuálist. Az elején egyenlíteni kell őket, hogy ugyan annyi legyen.

```
public float health;
3 references
public float  curHealth;
0 references
private void Start()
{
    curHealth = health;
}
```
3.az életerő

Ezt azért kell mivel a Health változó nem változhat így a curhealth mutatja ki, hogy aktuálisan mennyi élete van egy karakternek. Ha az ellenség meghal akkor nagyon kis eséllyel eldobhat egy tárgyat és eltűnik a gameobject.

Ellenfelek halála

Ellenfelek halála után egy megadott tárgyat kidob, amit a játékos felvehet. A tárgyakat, amiket kidob a szkripten belül meglehet határozni, hogy mennyi eséllyel és mennyit dobjon.

Ellenfél mozgása

```
public float speed;
public float checkradius;
public float attackradius;

public bool shouldRotate;

public LayerMask whatIsPlayer;

private Transform target;
private Rigidbody2D rb;
private Animator anim;
private Vector2 movement;
public Vector3 dir;

private bool isInChaseRange;
private bool isInAttackRange;
```

4.A játékos adatainak bekérése

Először is meg kell adni az ellenfél sebeségét majd a sugarat, amiben érzékeli a játékost utána az a sugarat, amiben támad/megáll. Meg kell adni a playernek a layerét, ami kisegíti az ellenfelet, hogy merre van a játékos majd kövesse azt.
A mozgás animációját az animatorban meg van adva, hogy mikor melyik irányba forduljon amikor a tengelyen változik a mozgása.

Ellenfél támadása

Meg keresi a játékos tag-jét és ha a támadásnak a collider-jében van akkor rá támad egyszer. A sebzés számát a statokban megadott értékét adja meg.

A játékos élete

Nagyából ugyan az, mint az ellenfeleké, hogy meglehet adni, hogy mennyi életük legyen csak itt a felvett páncélok és fegyverek hatnak az életnek a számaira.

Játékos támadása

Ha lenyomják azt a gombot, amivel támad a játékos akkor elindul az animáció a támadáshoz. Az animátorban meg van adva, hogy amikor elindul a támadás animáció.

Healthbar

5.healtbar

Két image kell, ami az egyik egy slider, ami a szám csökkenésével fogyatkozik. Lekéri a statok scriptből a maximális és az aktuális életerőt. És ahogy változik az aktuális életerő úgy változik a slider képei.

```
    void Update()
    {
        if(slider.value <=slider.minValue)
        {
            fillImage.enabled=false;
        }
        if(slider.value > slider.minValue && !fillImage.enabled)
        {
            fillImage.enabled=true;
        }
        float fillValue= playerHealth.curHealth / playerHealth.health;
        slider.value = fillValue;
    }
}
```
6.a healthbar kódja

Itemek

Először meg kell adni az itemek típusait amikel később meghatározhatjuk, hogy melyik helyre tehetjük őket (például: kardot a kard helyére).

Következőnek a képeségeit lehet beállítani, ami növeli az életpontokat és a támadást. A createAssetmenu segítségével létre lehet hozni egy item file, amit szabadon lehet módosítani.

7.Az assetmenu kód

Ki lehet választani, hogy melyik menübe és hogy milyen alap neve legyen.

Meg kell adni az item-nek a sprite-ját (kinézettét) ami a játékban meg is jelenik. Meg lehet adni, hogy egy item halmozható-e így, ha felvesszük, akkor nem külön-külön teszi bele a táskába, hanem egy helyre és kiírja, mennyi van.
Meglehet adni neki egy nevet és egy kis leírást róla. Csinálni kell Az item-nek egy ID változót, ami abban segít, hogy meghatározza az item-eket az inventory-ban.

Item Adatbázis

Mint az itemeknél itt is createassetmenu-vel lehet készíteni az adatbázist, amiben, eltároljuk az itemeknek az ID-ját.

Inventory Adatbázis

Itt tárolja el a felszedett itemeket és tárolja el az item adatbázis alapján. Erre hivatkozik a táska, ami megjeleníti az itemet.

Item felvétele

A játékoson és az itemeken van egy-egy collider, ami segíti elő az itemeket felvevését.

A játékos akkor tudja felvenni az itemet, ha a két collider-jük össze ütközik.

Össze ütközésnél az item felkerül az Inventory adatbázisba és az item fizikai formája törlődik.

Táska (inventory)

Az Inventory automatán generálja le a slot-okat, amikben eltároljuk az itemeket.

Slot

Egy előre elkészített gombok, amik megjelenítik az item sprite-ját. A gombra rákatintván elveszi onnan az itemet majd át lehet helyezni máshova vagy a játékosra. Ha az inventory és az equipment közzé tesszük akkor az item törlődik.

Pause menu

8.pause menu

Egy canvas bejön amikor az escape gombot nyomot, amin van három gomb, ami három opciónak felel meg. Egy gomb, ami visszadob a játékra egy másik, ami visszadob a fő menübe és az utolsó, ami kilép a játékból.

Death scene

Amikor a játékosnak az életereje eléri a nullát akkor bejön ez a canvas, ami kijelzi, hogy a játékos meghalt. Két opció van, egy újra éledés és egy játékból kilépés.

Tooltip

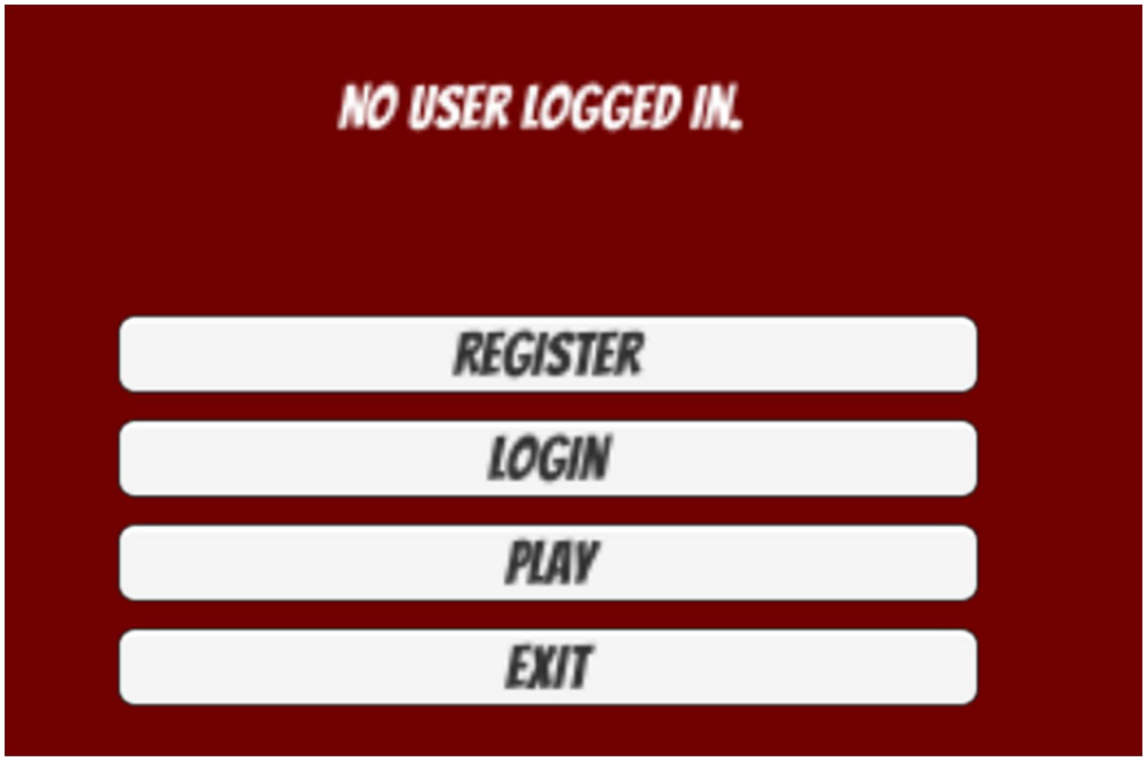
A Tooltip egy kisegíti a játékost sok mindenben és kiírja, hogy mi mire való. Bármelyik gameobjectre rá lehet tenni a scriptet és lehet adni neki egy címet és egy tartalmat, amit kiír, ha az egeret rá visszük a gameobjectre.

Felhasználói dokumentáció

A játék elindítása

A játék elindításhoz be kell illeszteni a lemezt a lemez meghajóba. A lemez behelyezése után a lemezen lévő built mappát kell megtalálni.
A mappában található a játék Marblez.exe néven.

Főmenü



9.fő menü

Játék elindításánál első dolog, amit lehet látni a main menüt, ahol lehet regisztrálni, avagy fiókot lehet készíteni a login fülnél a regisztrált fiókkal lehet belépni a fiókba.
Alatta lévő gombra kattintva el lehet érni a játékot, amibe be lehet lépni fiók nélkül viszont nem lehet fiók nélkül elmenteni a játékot.
Végül van esély arra a főmenüben, hogy teljesen bezárja a játékot.

Bejelentkezés és regisztráció

Regisztráció

A register gombra kattintva feldobja a regisztrációs oldalt. Itt is vissza lehet lépni a fő menübe vagy teljesen ki a játékból.

Meg kell egy felhasználót, ami a játékos neve lesz és egy jelszót, amivel be lehet lépni majd a bejelentkezésnél. Fontos, hogy olyan jelszót kell megadni, amit nem felejt el a felhasználó mert később nem lehet megváltoztatni.

Bejelentkezés

A login gombra rákattintása után bejön a bejelentkezés. Itt, mint a regisztrációnál itt is ki lehet lépni a fő menübe és a játéból is teljesen.

Felhasználó névvel és a jelszó megadásával be lehet lépni. Ha a felhasználónév vagy a jelszó rosszul lett megadva vagy nem létezik akkor sikertelen lesz a belépés.

A játék

A bejelentkezés után elindíthatjuk a játékot, amiben ki írja a játékos felhasználó nevét és az életerejét a bal alsó sarokba. A játék közepén a karakter található, amit a játékos irányit.

10. A játék

Egyből mellette itemek található, amihez, ha közel sétál a játékos akkor a karakter felveszi és az inventoryban, avagy a táskában eltárol.

Mozgás ás egyebek

W-a karakter felfele mozog

A-a karakter balra mozog

S-a karakter jobbra mozog

D-a karakter lefele mozog

Esc-a játék megállítása

Space-az inventory elmentése

Enter-a mentet inventory betöltése

Ball click (mouse 1) -karakter támadása

E- inventory, avagy a táska megnyitása

E-equipment megnyitása

Inventory/equipment

A felvett itemeket eltárolja az inventoryban és ott szabadon ellehet rendezni és bárhova tenni a táskán belül. Több fajta dolgot lehet felvenni például: sisak, páncélt, cipőt, kardot, ételt és pajzsot.
Az equipment megnyitása után lehet látni, hogy melyik tárgy hova lehet tenni.

11. equipment

Mindegyik helyre egy fajta dolgot lehet csak tenni kivéve, ahol a pajzs van mert oda lehet egy másik fegyvert is tenni. Mindenhova egyszerre csak egy item fér kivéve az ételhez mivel az az egyetlen, ami egymásra rakódik.

Azok az itemek, amiket a játékos felvett magára segíti a játékost sok mindenben. Például több élet ereje lesz a páncéltól, A fegyverek több sebzést adnak.

A pályán szét vannak szórva az itemek és meg kell őket találni.

Ellenfelek

A pálya egyes részein léteznek ellenfelek, akik folyamatosan követnek és támadnak. A cél, hogy vissza kell támadni mielőtt megölik a játékost.

Halál esetén a játék újra indul.

Az ellenfélnek egy nagyon kevés esélye van, hogy dob valamilyen itemet, amit később lehet használni.

Karakter életereje és támadási ereje

Játékosnak alapból 10 életereje van és 2 támadási ereje. Az itemek hozzá adnak egy bizonyos összeget és így segítenek egyszerűbben legyőzni az ellenfeleket.

A játékos életerejét a játék jobb felső sarkában lehet látni.

Ha a játékos meghal akkor feljön egy ablak, ahol újra lehet kezdeni vagy kilépni.

Összefoglalás

Az előre eltervezett és meghatározott funkciókat pár kivételével sikerült megvalósítani. A játékos statjának kivételével a játék jól működik. A legnagyobb segítségem az interneten talált tutorialok voltak. 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.

Továbbifejlesztési lehetőség

Egy pályát bármeddig lehet bővíteni és szerkeszteni.

Új ellenfeleket létrehozni.

Új tárgyakat.

Fejlettebb harc rendszer.

Fejlettebb AI rendszer.

Többjátékos mód.

Forrás és segítségek

https://opengameart.org/content/zelda-like-tilesets-and-sprites

https://www.youtube.com/playlist?list=PLJWSdH2kAe_Ij7d7ZFR2NIW8QCJE74CyT

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

Tartalomjegyzék

Ábrák Jegyzéke