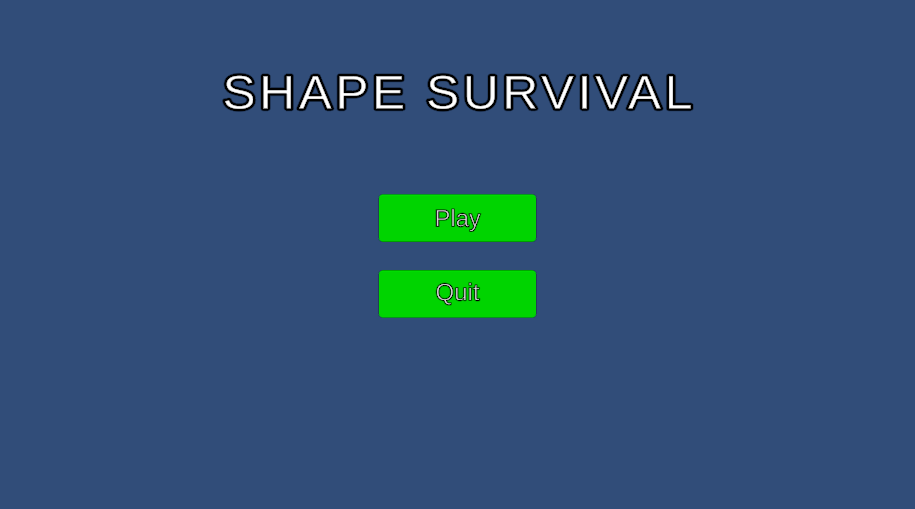
Shape Survival   
Dokumentáció

A Shape Survival egy kétdimenziós top-down survival shooter játék, amelyben a játékos célja, hogy lelője vagy kikerülje az ellenséges alakzatokat, amelyek a karakterhez érve megsebzik azt.

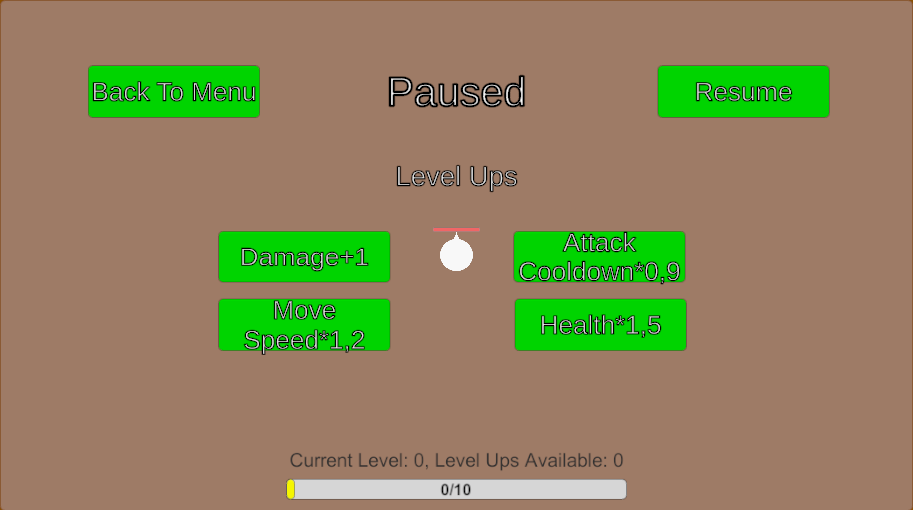
Jelenetek, Menük

A játékban található egy Fő menü, amely a játék indításakor jelenik meg. Itt a játék címe, valamint két gomb található. A „Play” gombra kattintva elindul a játék, a „Quit” gombra kattintva kilépünk a játékból.





Játék közben lehetőségünk van az „Esc” gomb megnyomásával megnyitni a „Pause Menu”-t, amely megállítja a játékot, itt tudjuk elkölteni a szintlépésért járó pontokat fejlesztésekre, továbbá lehetőségünk van kilépni a Fő menübe, vagy visszatérhetünk a játékhoz a „Resume” gomb lenyomásával, vagy az „Esc” gomb újbóli lenyomásával.

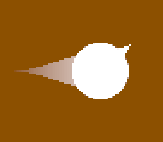


Amennyiben a karakterünk élete eléri a nullát, a játéknak vége, és megjelenik a „Death Screen”, amelyről visszamehetünk a Fő menüre, vagy újból elkezdhetjük a játékot.



A játékos karaktere

A karakterünk képe alább látható.

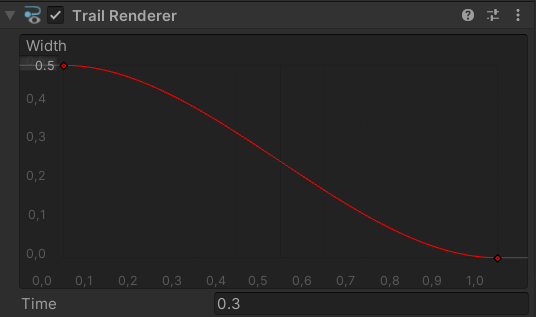


Mozgás közben, annak irányának jelzése érdekében csíkot húz maga után, amit Trail Renderer használatával hoztam létre.

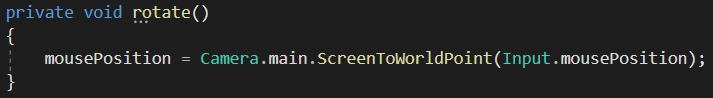
A Trail Renderer egy olyan vizuális effektus a Unity játékmotorban, amely egy objektum mozgása során húzott vonalat vagy csíkot jelenít meg. Be lehet állítani a kezdő és a végső széleségét, az anyagát, ami lehet textúra vagy szín, továbbá azt, hogy mennyi ideig jelenjen meg a csík.

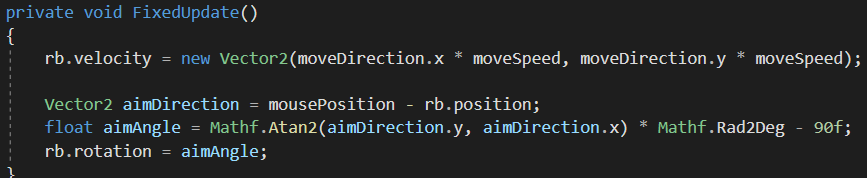
A kamera játék közben a karaktert követi. Ezt a Cinemachine nevű kiegészítővel valósítottam meg.

A Cinemachine egy olyan rendszer a Unity játékmotorban, amely segít a kamera mozgatásának és kezelésének automatizálásában. Célja, hogy könnyen kezelhetővé tegye a bonyolult kamera mozgásokat és beállításokat anélkül, hogy szükség lenne sok kódra vagy manuális beállításra.



A célzást a játékos irányíthatja például egérrel, melynek az irányát a karakter „csőre” is jelez. A célzás irányát az alábbi kóddal határozzuk meg:





Az rb.position a Rigidbody2D, avagy a karakterünk pozícióját adja meg a kétdimenziós térben.

Az egér bal gombjának megnyomásával a karakter lőni fog, amennyiben a lövés visszaszámlálója elérte a nullát, avagy elég idő telt el az előző lövés óta. Ugyan ezen gomb nyomva tartásával a lövés automatikusan megtörténik, amit letelt a kellő idő az előző lövés óta.

Amennyiben a lövedék nem talál el ellenfelet, akkor egy idő után magától megsemmisül, hogy ne terhelje le az erőforrást.

A lövedék szintén Trail Rendererrel lett ellátva vizuális célzattal.



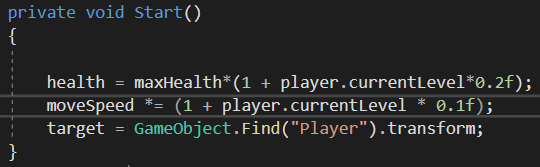
Ellenségek

A játékban jelenleg három típusú ellenség található.

|  |  |  |
| --- | --- | --- |
| Az alap ellenség a kék kör, amelynek nincsenek kimagasló értékei az életet és sebességét tekintve, viszont gyakran spawnlolnak. | A zöld háromszög a gyors ellenség, amely cserébe kevés élettel rendelkezik.  Ez a típusú ellenség közepes gyakorisággal spawnol. | A piros négyzet sok élettel rendelkezik, viszont lassú.  Ez a típusú ellenség spawnol a legritkábban. |
|  |  |  |
| Kezdeti értékek: | | |
| Életerő: 3  Mozgási sebesség: 2  Spawn idő: 3 másodperc | Életerő: 1  Mozgási sebesség: 3  Spawn idő: 5 másodperc | Életerő: 4  Mozgási sebesség: 1  Spawn idő: 10 másodperc |

Eleinte közeliek ezek az értékek, de a szintlépésnél lévő értéknövekedés miatt egyre feltűnőbb a különbség az ellenfelek között, továbbá a spawnolási idő is gyorsan csökken.

Az értékeket az alábbi kód alapján kapják meg az ellenfelek, amikor létrejönnek.

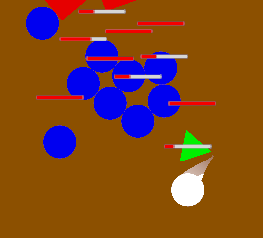


Az ellenségeket és a lövedékeket az eredeti elkészülte után „prefab”-ként mentettem el, hogy spawnerek használatával egyszerűen tudjam többször is felhasználni őket.

Unity-ben a "prefab" egy előre definiált objektum, amely tartalmaz minden szükséges komponenst, tulajdonságot és beállítást ahhoz, hogy az objektum egy példánya könnyen létrejöhessen a játékban. A "prefab" tehát egy sablon, amelyet előre elkészítünk, majd később többször is felhasználhatjuk az alkalmazásban.

A spawnerek úgy vannak beállítva, hogy a játékostól random irányban és pozícióban hozza létre az ellenséges karaktereket. A pozíciót megadó random generátornak van minimum és maximum értéke is, hogy az ellenségek ne tudjanak a játékos karakterre spawnolni, de ne legyenek túl messze se tőle.

Az ellenségek és a játékos karaktere is rendelkezi életcsíkkal, amely dinamikusan változik attól függően, hogy mennyi élete van.



Mindegyik karakter rendelkezik Rigidbody2D-vel és Collider2D-vel is. Utóbbi az érintkezések detektálására szolgál, például amikor eltalálunk egy ellenfelet a lövedékünkkel, vagy amikor az ellenség eléri a játékost. Előbbi az érintkezésekre adott reakció fizikai részét befolyásolja, például amikor meglövünk egy ellenfelet, akkor egy kicsit hátrébb löki.

Szintlépés

Amennyiben a játékos elég ellenséget semmisít meg, lehetősége van szintlépésre a „Pause Menu”-ben. Minden következő szinthez több és több ellenséget kell elpusztítani.

A következő szinthez szükséges ellenségek elpusztításának számát a képernyő alján egy dinamikusan változó sáv mutatja vizuálisan és számban kiírt formában is. Az első szám a következő szinthez eddig megszerzett tapasztalati pontokat mutatja, a második pedig azt, hogy mennyi kell összesen.

Szintlépésnél nem csak a mi karakterünk erősödik, hanem az ellenségek is. Gyakrabban fognak spawnolni, több életük lesz, gyorsabban haladnak a játékos felé.

A szintlépésért kapott pontokból 4 fejlesztés közül választhatunk. Növelhetjük a sebzést, a lövések gyakoriságát, a mozgási sebességet vagy az életerőt.

A sebzés minden fejlesztésnél 1-gyel növekszik, az alap értéke kezdésnél 1.

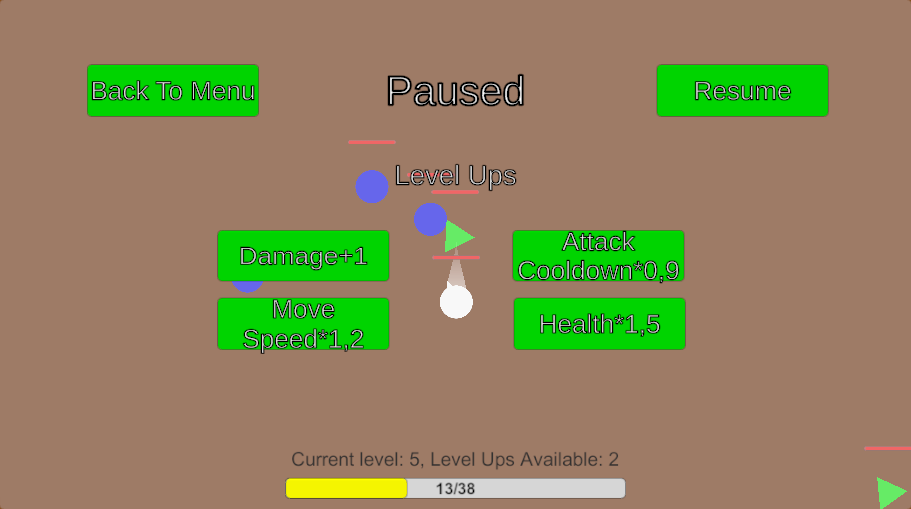
Lőni alapesetben fél másodpercenként tudunk, fejlesztésenként ezt beszorozzuk 0,9-cel.

A mozgási sebességünk kezdéskor 5-ös, fejlesztésenként ezt 1,2-szeresére emeljük.

Életerőből 5-öt kapunk az elején, ezt fejlesztésenként a másfélszeresére emeljük.

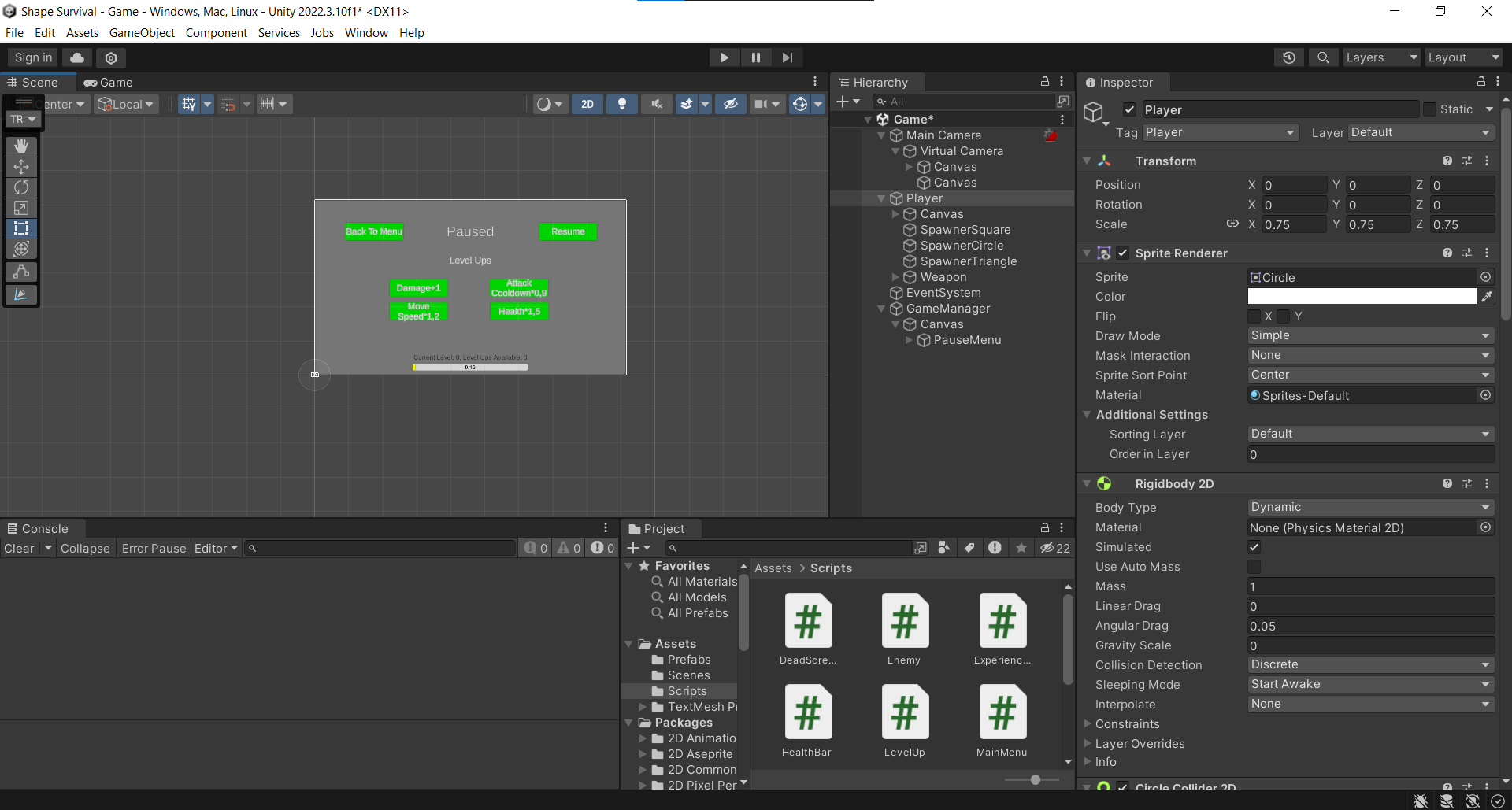
Minden szorzással kapott új, nem kerek számot felfelé kerekítünk.

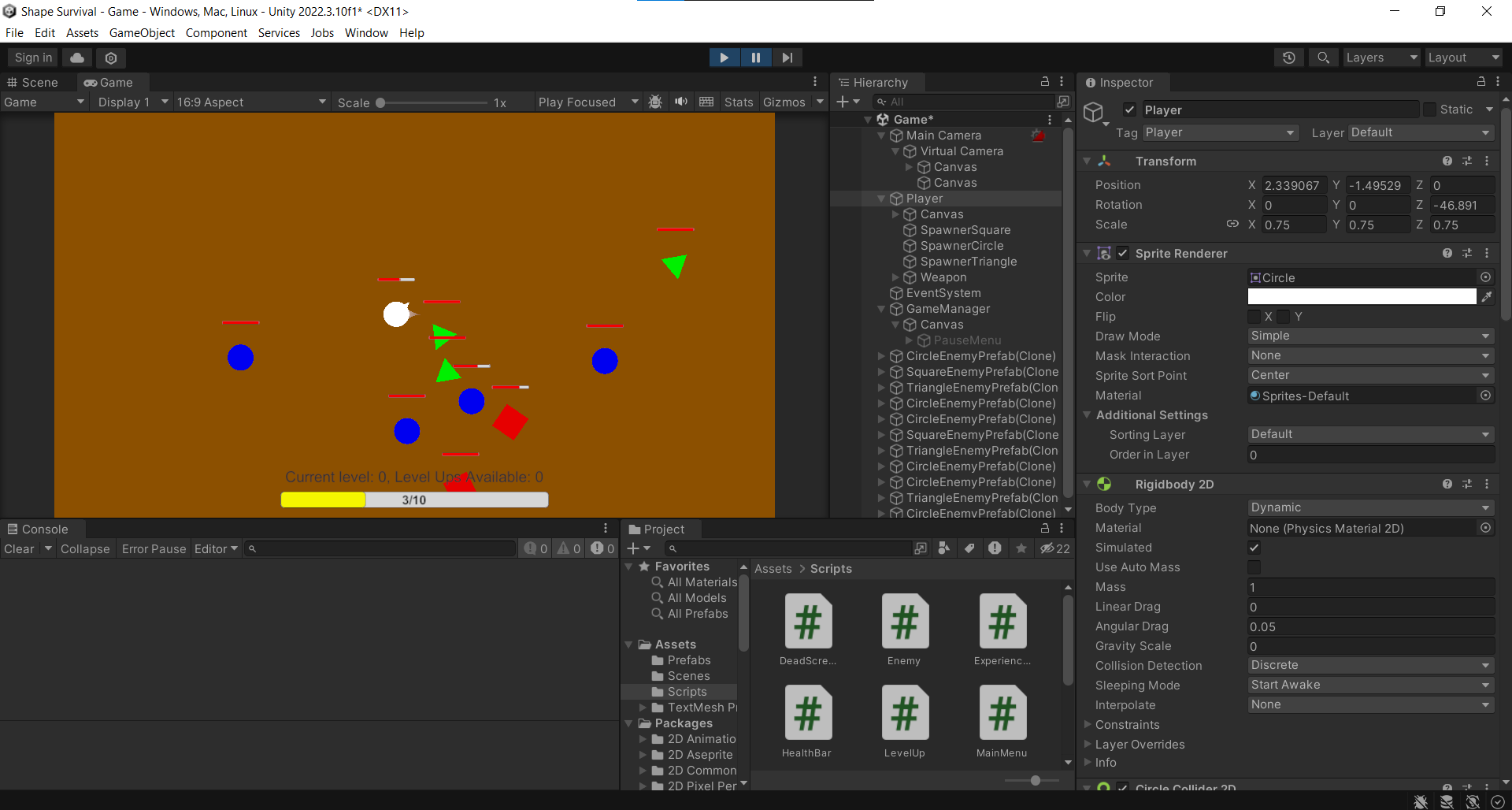
A karakterünk élete minden szintlépésnél vissza töltődik a maximális értékre.



Fejlesztői környezetek

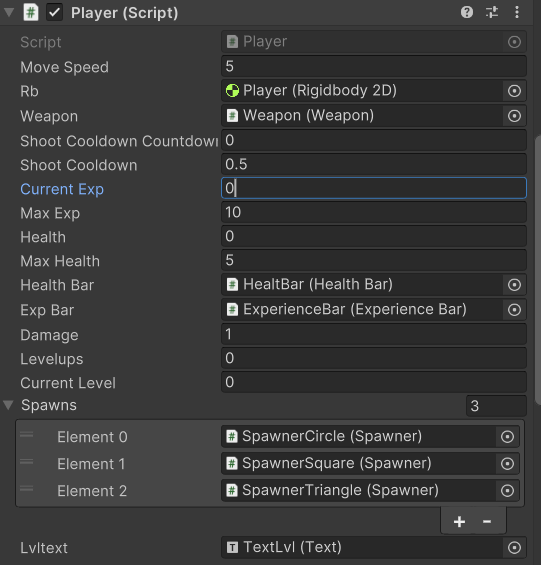
A játék fejlesztéséhez Unity-t használtam.





A Unity egy cross-platform játékmotor és fejlesztői környezet, melyet a Unity Technologies fejlesztett ki. Elsőként 2005 júniusában jelentették be és adták ki, mint egy Mac OS X játékmotor. Azóta fokozatosan bővült a támogatása, hogy többféle asztali, mobil, konzol és virtuális valóság platformot is kezeljen. Különösen népszerű az iOS és Android mobiljáték-fejlesztés területén, könnyen használhatónak számít kezdő fejlesztők számára, és kedvelt a független fejlesztők által is.

A motort használhatjuk háromdimenziós (3D) és kétdimenziós (2D) játékok, valamint interaktív szimulációk és egyéb élmények létrehozására. A motor már elterjedt más területeken is a videojátékokon kívül, mint például a filmipar, az autóipar, az építészet és a mérnöki tervezés.



A Unityben létrehozhatunk különböző komponenseket, például gombokat, jeleneteket vagy spriteokat. Utóbbiakat a játékos karakter és az ellenfelek létrehozására használtam. Továbbá itt adhatunk meg alap értékeket, komponenseket és szkripteket rendelhetünk egymáshoz.

A szkriptek írásához a Microsoft Visual Studio-t használtam

A Microsoft Visual Studio egy integrált fejlesztői környezet (IDE), amelyet a Microsoft fejlesztett ki, és széles körben használják szoftverfejlesztéshez, beleértve a játékfejlesztést is. Először 1997-ben adott ki a Microsoft, és azóta is folyamatosan fejlesztik és adnak ki új verziókat.

A Visual Studio Community Edition ingyenesen elérhető, és teljes funkciókészlettel rendelkezik, amelyet egyéni fejlesztők, oktatási intézmények és kisvállalkozások használhatnak.

A Visual Studio 36 különböző programozási nyelvet támogatés lehetővé teszi, hogy a kódszerkesztő és hibakereső támogasson (különböző mértékben) szinte bármely programozási nyelvet, feltéve, hogy a nyelv-specifikus szolgáltatás rendelkezésre áll. Beépített nyelvek közé tartozik a C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML és CSS. Támogatás áll rendelkezésre más nyelvekhez is, mint például a Python, Ruby, Node.js, és M, amelyek kiegészítők segítségével érhetők el.

Ehhez a projekthez a szkriptek megírására a C# programozási nyelvet használtam.

