

**GÁBOR DÉNES FŐISKOLA**



**MÉRNÖKINFORMATIKUS**

**SZAKDOLGOZAT**

**JÁTÉK FEJÉLESZTÉS UNREAL ENGINE 5-BEN**

**Kiss Bence Baldwin  
ZZYZRX  
197/2020**

## **SZAKDOLGOZATI / ZÁRÓDOLGOZATI TÉMAVÁZLATI LAP**

**A lapot nem képként kell beilleszteni, a témaavázlat  
elfogadásakor feltöltött pdf-et kell összefűzni a  
szakdolgozat/záródolgozat többi részével.**

## **Konzultációs lap**

**A konzultációs lapot elektronikusan kell kitölteni.**

**A lapot nem képkét kell beilleszteni.**

**Amennyiben a konzultációk során kézzel került aláírásra,  
akkor bescannelhető a dokumentum, majd össze kell fűzni a  
szakdolgozat/záródolgozat többi részével.**

**Ha elektronikusan lett vezetve, akkor az aláírásokat is  
elektronikusan kell ráhelyezni, majd összefűzni a  
szakdolgozat/záródolgozat többi részével.**

## **Nyilatkozat**

a szakdolgozat eredetiségéről

Alulírott Kiss Bence **ZZYZRX** ezennel büntetőjogi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy a **JÁTÉK FEJÉLESZTÉS UNREAL ENGINE 5-BEN 197/2020** szakdolgozatom saját, önálló munkám eredménye; az abban hivatkozott nyomatott és elektronikus szakirodalom felhasználása a szerzői jogok és a szakdolgozatirás szabályainak megfelelően készült.

Kijelentem, hogy a plágium fogalmát megismertem és ahol mások eredményeit vagy gondolatait idéztem azt minden esetben azonosítható módon feltüntettem, a dolgozatban közölt ábrák és képek közlése mások szerzői jogait nem sértik.

Budapest, 2021. év ..... hó ..... nap

.....  
hallgató aláírása

## **Nyilatkozat**

a szakdolgozatról

Alulírott Kiss Bence **ZZYZRX** kijelentem, hogy **JÁTÉK FEJÉLESZTÉS UNREAL ENGINE 5-BEN 197/2020** szakdolgozatomat a Gábor Dénes Főiskola oktatói és hallgatói, szükség esetén más érdeklődők is felhasználhatják (pl. hivatkozás alapjául, olvasótermi használatra) későbbi munkájukhoz a szerzői jogok tiszteletben tartása mellett.

Budapest, 2021. év ..... hó ..... nap

.....  
hallgató aláírása

## **Kivonat**

Létrehoztam egy olyan játékot, amely használja a Steam-es mérföldköveket. A játék az új Unreal Engine 5-ben készült.

A céлом az volt, hogy létrehozzak egy prototípus játékot, létrehozzam a játékmenethez szükséges rendszereket. Létre akartam hozni egy olyan ellenfél osztályt, amit adattáblából lehet bővíteni. Létre akartam hozni olyan fegyver és tárgy osztályt, amelyből származtatást után csak egy vagy két függvényt kell felülírni új fegyver létrehozásához és a többi adatot adattáblából tölti be. Létre akartam hozni egy olyan ellenfelet, amely a főnöként funkcionál és legyőzésével a játékos nyerni tud, ennek a lénynek speciális támadásai lesznek.

A játék elérhető a következő hivatkozáson keresztül:

# Tartalomjegyzék

1.	Bevezetés .....	8
1.1.	A dolgozat célkitűzései .....	8
1.2.	A dolgozat felépítése .....	8
2.	Unreal Engine .....	9
2.1.	UnrealScript .....	9
2.2.	Blueprint.....	9
2.3.	Steam.....	9
2.3.1.	Steamworks .....	9
3.	The Witch's Pact .....	10
3.1.	Játékmenet.....	10
4.	Projekt előkészítése .....	11
5.	Játékos karakter létrehozása .....	12
5.1.	Karakter alapértékek .....	12
5.1.1.	Capsule_Component.....	12
5.1.2.	Spring_arm .....	13
5.1.3.	Capsule .....	13
5.1.4.	Widget .....	14
5.1.5.	Sprite.....	14
5.2.	Mozgás implementációja.....	14
5.2.1.	Set_anim_Direction(Left/right).....	15
5.3.	State Machine .....	15
5.3.1.	Update_anim.....	15
5.3.2.	Animáció értesítés kezelése.....	15
5.3.3.	Fire_event.....	16
5.4.	Event AnyDamage .....	16
5.5.	Event BeginPlay.....	16

5.5.1.	Trail .....	17
5.5.2.	Regen.....	18
5.6.	Heal().....	18
5.7.	Reset_To_Idle().....	18
5.8.	Add_Xp(Xp, New_Xp) .....	18
5.9.	Level_Up() .....	18
5.10.	Add_Extra_Xp(Xp).....	19
5.11.	Fix_field_Of_View().....	19
5.12.	Konstruktor.....	19
5.13.	Event Tick esemény .....	19
5.14.	Base_Player_Stats .....	19
6.	Master_ai .....	20
6.1.	Konstruktor.....	20
6.2.	Event_Begin_Play .....	20
6.2.1.	Event_tick_for_stuff.....	21
6.2.2.	Tick.....	22
6.3.	Event_Actor_Begin_overlap.....	22
6.3.1.	Apply_dmg.....	22
6.4.	Event_Actor_End_Overlap .....	22
6.5.	Event_Any_Damage .....	22
6.6.	Event_Destoryed .....	23
6.7.	Play_Anim() .....	23
6.8.	New_Location() .....	24
6.9.	Master_ai Alapértékek .....	24
6.9.1.	Box .....	24
6.9.2.	PaperFilpbook.....	25
6.10.	Mob_Types.....	25

6.11.	Ai_data .....	25
6.11.1.	Ai_stats Struktúra .....	26
6.12.	Ai_stats_DT Adattábla .....	26
7.	Ellenfél fajták.....	27
7.1.	Eye.....	27
7.2.	Goblin.....	28
7.3.	Skeleton .....	29
7.4.	Mushroom .....	30
7.5.	Necromancer .....	31
7.6.	King.....	32
7.7.	Hounds .....	33
8.	Tárgyak és Tapasztalat .....	34
8.1.	Enumerációk.....	34
8.1.1.	Bonus_stats_names.....	34
8.1.2.	Consumables_names .....	34
8.1.3.	Weapon_names.....	34
8.2.	Struktúrák .....	34
8.2.1.	Bonus_stats_struct .....	34
8.2.2.	Consumables_struct.....	34
8.2.3.	Item_anim_data_struct .....	35
8.2.4.	Ui_data_struct.....	35
8.2.5.	Weapon_stats_struct.....	35
8.2.6.	Weapon_data_struct .....	35
8.2.7.	Consumables_datatable .....	35
8.2.8.	Bonus_stats_datatable .....	35
8.2.9.	Weapon_datatable .....	35
8.3.	Master_consumable.....	42

8.3.1. Event_Actor_Begin_Overlap .....	42
8.3.2. Do_Stuff(Potency).....	42
8.3.3. Konstruktor.....	42
8.3.4. Xp .....	42
8.3.5. Souls .....	43
8.4. Fejlesztések .....	43
8.4.1. Event_Begin_Play() .....	43
8.4.2. Update_Stats() .....	43
8.4.3. Level_Up() .....	44
9. Fegyverek .....	45
9.1. Master_weapon .....	45
9.1.1. Event_Begin_Play .....	45
9.1.2. Do_It_Again.....	45
9.1.3. Fire.....	46
9.1.4. Spawn_Actor() .....	46
9.1.5. Level_Up() .....	46
9.1.6. Calculate_Cd().....	47
9.1.7. Calculate_Transform(Transform).....	47
9.1.8. Override_Transform(Transform) .....	47
9.1.9. Achievement.....	47
9.2. Energy_ball .....	47
9.2.1. Calculate_Transform() .....	47
9.2.2. Override_Transform() .....	48
9.3. Torch_item .....	48
9.3.1. Calculate_Transform() .....	48
9.4. Ice_ball .....	49
9.5. Demon_pact_item .....	49

9.6.	Boom_fire_item .....	49
9.6.1.	Override_Transform() .....	50
9.7.	Beam_item .....	50
9.7.1.	Event_Do_it_again() .....	50
10.	Lövedékek.....	51
10.1.	Master_projectal.....	51
10.1.1.	Konstruktor.....	52
10.1.2.	Event_Begin_Play .....	52
10.1.3.	Destroy .....	52
10.1.4.	Event_Actor_Begin_Overlap .....	52
10.2.	Energy_ball_projectal osztály .....	53
10.3.	Ice_ball_projectal osztály.....	53
10.4.	Torch_projectal osztály .....	53
10.5.	Beam osztály .....	53
10.5.1.	Konstruktor.....	53
10.5.2.	Event_Begin_Play .....	53
10.5.3.	Do_dmg .....	53
10.5.4.	Event_Destoryed .....	53
11.	Szintek .....	54
11.1.	Menu_map szint .....	54
11.1.1.	Event_Begin_Play .....	54
11.2.	Underworld.....	54
11.2.1.	Plane .....	54
12.	Boss.....	55
12.1.	Konstruktor.....	55
12.2.	Event_tick.....	56
12.3.	Event_Any_Damage .....	56

12.4. End .....	56
12.5. Basic_attack .....	56
12.6. Spell_1 .....	57
12.7. Boss_bh behavior tree .....	57
12.7.1. Attack Selector .....	57
12.7.2. Spells Selector .....	57
12.7.3. Basic_attack feladat.....	57
12.7.4. In_range_for_ba dekorátor .....	57
12.7.5. Move_to_player_offset feladat.....	58
12.7.6. Spell_1 .....	58
12.7.7. Cd dekorátor .....	58
13. Game_instance.....	59
13.1. Event_Init .....	59
13.2. Music_volume .....	59
13.3. Event_Shutdown .....	59
13.4. Game_over esemény .....	59
13.5. Play_music .....	59
13.5.1. Finsih .....	59
14. Játék módok .....	60
14.1. Menu_gamemode .....	60
14.2. InGame_gm .....	60
14.2.1. Event_On_Post_login.....	60
14.2.2. Tick.....	60
15. Material.....	63
15.1. player_material .....	63
16. Licenszek .....	64
16.1. Boszorkány animációk: .....	64

16.2. Skeleton , Mushroom, Goblin, Eye szörnyek animációi.....	64
16.3. King animációk .....	64
17. Összegzés.....	65
Irodalomjegyzék .....	67
Ábrajegyzék .....	69
Táblázatjegyzék .....	70
Mellékletek jegyzéke .....	<b>Error! Bookmark not defined.</b>
1. melléklet: Harci ideggázok hatásfoka.....	<b>Error! Bookmark not defined.</b>

# **1. Bevezetés**

A szakdolgozatom az egyik legjobban optimalizált valós idejű 3D játék fejlesztő szoftvert használja. Azért választottam ezt a témát, mert szabadidőmben szeretek számítógépes játékokkal játszani. Mindig is szerettem volna csinálni egy saját „Steam”-es publikált játékot, bár munkám egyelőre csak technikai jellegű a marketing és branding témaival jelenleg nem foglalkozom. Célom az, hogy készítsek egy prototípus verziót, melyet majd egy grafikus és marketinges segítségével lehet eladni.

## **1.1. A dolgozat célkitűzései**

- Steam-es funkciók használata
  - Steam Achievements
- Roguelike élmény létrehozása
- Tapasztalat szerzés az egyik legmodernebb játék fejlesztő szoftverrel

## **1.2. A dolgozat felépítése**

1. Bevezetés, célok, felépítés
2. Unreal Engine
3. Játék specifikáció
4. Játékos karakter létrehozása
5. Ellenfelek létrehozása
6. Ellenfelek fajtái
7. Tárgyak és tapasztalat létrehozása
8. Fegyverek létrehozása
9. Lövedékek létrehozása
10. Szintek létrehozása
11. Boss létrehozása
12. játék példány létrehozása
13. Játék mód létrehozása
14. Anyagok létrehozása
15. Összefoglalás

## **2. Unreal Engine**

Az Unreal Engine egy játék motor, amit az Epic Games fejlesztett ki. 1998-ban volt az első demójuk, egy lövöldözős játék, melynek az volt a neve, hogy „Unreal” először csak a PC (ezen belül is a játékokra volt tervezve a motor) volt a cél platform, de azóta a filmparban is jelentős szerepet játszott. Az Unreal Engine egy C++ írt szoftver a többi játék motorral szemben az Unreal Engine már a kezdetekben is használt vizuális scripting nyelvet a magasszintű játék funkciók létrehozásához. Unreal Engine 4 előtt ezt „UnrealScript”-nek hívták, majd ezt lecserélte Blueprint-re.

### **2.1. UnrealScript**

Egy objektum orientált nyelv volt, ami hasonlít a Java nyelvre, egyszeres öröklődés volt benne, és nem volt burkoló osztály a primitív típusokhoz. Az interfészket csak az Unreal Engine 3-tól támogatták, támogatja az operátor túlterhelést, de metódus túlterhelést még nem.

### **2.2. Blueprint**

A Blueprint egy vizuális scripting nyelv teljes játéklogikákat lehet gyorsan és hatékonyan leprogramozni egy csomópont alapú interfész segítségével. A motorban definiált objektum orientált osztályokat használja. A Blueprint rendszer rettentő hatékonnyá teszi a fejlesztést, mert azonkívül, hogy saját Blueprint osztályokat hozhatunk létre, ezt a rendszert nem csak programozok tudják használni. Könnyen tanulható (pl.: grafikus csapat is tudja használni).

### **2.3. Steam**

A Steam egy tartalom kezelő és továbbító rendszer, rengetek szoftvert árul a digitális áruházában („Steam Store”), nagyrést játékokat. Kényelmi funkciókat is tartalmaz. Saját DRM-je van, ami lehetővé teszi, hogy ne a géphez legyen kötve a megvett szoftver, hanem a felhasználói fiókhoz.

#### **2.3.1. Steamworks**

Steamworks vére hallgat a Steam saját API-a, ami tartalmazza az eszközöket a fejlesztők számára, hogy teljesen kihasználják a Steam kliens képességeit, de ezek csak lehetőségek nem kötelező velük élni.

### **3. The Witch's Pact**

Egy túlélős játék, ahol az a cél, hogy túl kell élni X ideig majd megölni a főnököt. A játék „Roguelike” elemeket is tartalmaz, lelkeket kell gyűjteni és azokból lehet majd fejlesztéseket venni a következő menetre legalábbis ez lenne a következő lépés a fejlesztésben.

#### **3.1.Játékmenet**

A játékos egy boszorkány karaktert irányít a billentyűzet segítségével, és fejlesztéseket, új támadásokat szerez a játék folyamán. Miden szintlépésnél eldöntheti a játékos, hogy milyen fejlesztést szeretne. A cél, hogy túléje 20 percig majd megölje a főnököt, aztán a játéknak vége, vagy ha nem sikerül ezt teljesíteni akkor veszít a játékos.

## 4. Projekt előkészítése

Létrehoztam egy Blueprint projektet, majd bekapcsoltam az „Online SubSystem” plugin-t.

A defaultEngine.ini-ben definiáltam a NetDriver-t:

```
[/Script/Engine.GameEngine]  
+NetDriverDefinitions=(DefName="GameNetDriver",DriverClassName="OnlineSubsystemSteam.  
SteamNetDriver",DriverClassNameFallback="OnlineSubsystemUtils.IpNetDriver")
```

DefName: az egyedi neve a net driver definíiónak

```
[/Script/OnlineSubsystemSteam.SteamNetDriver]  
NetConnectionClassName="OnlineSubsystemSteam.SteamNetConnection"
```

DriverClassName: ez az elsődleges driver osztály neve

DriverClassNameFallback: ez annak az osztálynak a neve, ami a tartalék, ha az elsődleges driver nem megfelelően töltene be

Beállítottam, hogy az „Online Subsystem Steam” használja:

```
[OnlineSubsystem]  
DefaultPlatformService=Steam
```

OnlineSubsystemSteam modul konfigurálás:

```
[OnlineSubsystemSteam]  
bEnabled=true  
SteamDevAppId=1929430
```

TD\_Wave.build.cs -ben hozzáadtam a projekthez a steam modult:

```
DynamicallyLoadedModuleNames.Add("OnlineSubsystemSteam");
```

Engine-Inputs: létrehoztam a karakter mozgásért felelős bemeneti definíciókat, amelyek fogják a karakter 2D-ben mozgatni:

MoveForward:

- W lesz az előre ezért neki 1-es értéket adok
- S lesz a hátrafelé ezért ő az előrének a -1 szerese, azaz -1

MoveRight:

- D lesz a jobbra neki az értéke 1
- A lesz a balra ez pedig a jobbra -1 szerese

## 5. Játékos karakter létrehozása

Létrehoztam a master\_player Blueprint osztályt a Paper Character-ből származtattam, ami tartalmazott egy Capsule Component-st, egy Sprite komponenst, egy Arrow Component-et, és egy Character Movement komponenst, mivel jelenleg egy játszható karakter van, és későbbiekben nem lesz funkcionálisban eltérés a karakterek között, ezért nem származtatással fogom megcsinálni a karaktereket, hanem adattáblából fogom betölteni a master\_player osztályba az adatokat.

### 5.1. Karakter alapértékek

Az alábbi értékeket változattam meg a játékos karakternél:

#### 5.1.1. Capsule\_Component

Capsule\_Half\_Height: 9.0

Capsule\_Radius: 9.0

A kapszula méretét állítom a karakter méretéhez ez a kapszula lesz felelős a, hogy karakter ne essen át a talajon és az ellenfelek eltolásáért.

Enable\_Gravity: False

Generate\_Overlap\_Event: False

Nem szeretném, hogy ez a komponens átfedést tudjon képezni.

Collision\_Presets: Custom

Collison\_Enabled: Collison\_Enabled(Query and Physics)

Object\_type: player

Az ütközési beállítások a következők:

	Ignore	Overlap	Block	
<b>Collision Responses</b> ⓘ	—	—	—	
Trace Responses	—	—	—	
Visibility	✓	—	—	↔
Camera	✓	—	—	↔
<b>Object Responses</b>				
WorldStatic	—	—	✓	
WorldDynamic	—	—	✓	
Pawn	✓	—	—	↔
PhysicsBody	✓	—	—	↔
Vehicle	✓	—	—	↔
Destructible	✓	—	—	↔
items	—	✓	—	↔
ai	—	—	✓	↔
player	✓	—	—	↔
spells	—	✓	—	

1. ábra játékoskarakter Capsule\_Component ütközési beállításai

### 5.1.2. Spring\_arm

Target\_arm\_length: 0

Target\_Offset: 0,0,580

Így a kamera a karakter felett lesz 580 egységnyire

### 5.1.3. Capsule

Ez a kapszula lesz felelős a sebzés regisztrációért, és tárgy felvételéért.

Capsule\_Half\_Height: 15

Capsule\_Radius: 7

Ekkor lesz az a terület, ahol ütközést tudok detektálni az ellenfelekkel

Can\_Character\_step\_Up\_On: False

nem szeretném, hogy másik karakter ráléphessen.

Multi\_Body\_Overlap: True

egyszerre több ellenfél is hozzáérhet a karakterhez

Object\_type: player

	Ignore	Overlap	Block	
Collision Responses (?)	-	-	-	
Trace Responses				
Visibility	✓			↔
Camera	✓			↔
Object Responses				
WorldStatic	✓			↔
WorldDynamic	✓			↔
Pawn	✓			↔
PhysicsBody	✓			↔
Vehicle	✓			↔
Destructible	✓			↔
items		✓		↔
ai			✓	↔
player	✓			↔
spells		✓		

2. ábra játékoskarakter Capsule ütközési beállításai

#### **5.1.4. Widget**

Ez lesz az életerő csík a karakter alatt.

Widget\_Class: hp

#### **5.1.5. Sprite**

Source\_Flipbook: witch\_idle

### **5.2. Mozgás implementációja**

A MoveForward eseménynek az Axis értékét megvizsgálom, ha nagyobb, mint nulla akkor előre kell elmozdítani a karaktert, ha kisebb akkor hátrafelé. Az Add\_Movement\_Inupt() függvénynek négy bemenetiértéke van nekem ebből kettő kell a World\_Direction, ami megadja, hogy milyen irányba akarom a karaktert mozgatni, és a Scale\_Vaule, ami megadja, hogy milyen mértékben akarom elmozdítani a karaktert, ez a MoveForward Axis értéke lesz, World Direction-t ki lehet számolni a Get\_Control\_Rotation fügvény Z visszatérésiértékével a Get\_Forward\_Vector-t meghívom, és ez lesz a World Direction értéke.

A MoveRight eseménynek az Axis értékét megvizsgálom, ha nagyobb, mint nulla akkor jobbra kell elmozdítani a karaktert, ha kisebb akkor balra. Az Add\_Movement\_Inupt() függvénynek négy bemenetiértéke van nekem ebből kettő kell a World\_Direction, ami megadja, hogy milyen irányba akarom a karaktert mozgatni, és a Scale\_Vaule, ami megadja, hogy milyen mértékben akarom elmozdítani a karaktert, ez a MoveRight Axis értéke lesz, World Direction-t ki lehet számolni a Get\_Control\_Rotation\_fügvény Z visszatérésiértékével a Get\_Right\_Vector-t meghívnom, és ez lesz a World Direction értéke. Mozgás után meghívom a Set\_Anim\_direction(Left/Right) függvényt, hogy a karakter animáció a megfelelő irányba nézzen (jobbra vagy balra).

### 5.2.1. Set\_anim\_Direction(Left/right)

Megvizsgálom, hogy a bemeneti érték egy-e ha igen meghívom a Set\_Relative\_Rotation(New\_Rotation,Spirte) függvényt ahol a New\_Rotation -90,0,90 ha nem akkor a New\_rotation 90,0,270

## 5.3.State Machine

Sajnos az Unreal Engine 5 nem támogatja a nem 3D-s animáció State Machine-t, ezért nekem kell létrehoznom egyet, ehhez létrehoztam egy state\_enum-ot, amiben eltárolom a lehetséges állapotokat majd létrehozok egy anim\_struct-ot, ami egy struktúra, és ebben lesznek a Flipbook referenciák, a loop nevű Boolean és a későbbiekre tekintettel egy Integer (notifyFrame), amivel majd implementálni tudom az Animation Notification-t. Létrehoztam egy adattáblát az anim\_struct alapján witch\_anim\_data névvel majd feltöltöttem adattal:

	Row Na	flipbook	loop	notifyFrame
1	idle	PaperFlipbook'/Game/player/witch/idle/witch_idle.witch_idle'	True	-1
2	attack	PaperFlipbook'/Game/player/witch/attack/attack.attack'	False	3
3	hit	PaperFlipbook'/Game/player/witch/hit/witch_hit.witch_hit'	False	-1
4	run	PaperFlipbook'/Game/player/witch/run/run.run'	True	-1

notifyFrame alapértelmezett értéke -1-et adtam meg, mert ha nem szeretném, hogy aktiválja az eseményt akkor vagy egy hatalmas számot kell megadni vagy egy negatívat.

Kibővítettem a mozgás logikát úgy, hogy mozgásnál állítsa a state nevű enum-ot run-ra majd meghívja az update\_anim függvényt.

### 5.3.1. Update\_anim

Az update\_anim() függvény meghívja a Get\_Data\_Table\_Row() függvényt, a két bementiértéke: witch\_anim\_data és a state, a visszakapot adatsort elemire bontom: Flipbook, Loop, Notify Frame majd beállítom ezeket az értékeket Set Flipbook(Sprite,Flipbook), Set Looping(Sprite,Loop) és eljátszom a Flipbook-ot a Play(Sprite) függvénnnyel.

### 5.3.2. Animáció értesítés kezelése

Az Update\_anim függvényben megkapott Notify\_Frame értéket kell megvizsgálnom, ha nagyobb mint -1 akkor az animáció tartalmaz animáció értesítést, ha

a Branch(Notify\_Frame) igazzal tér vissza akkor létrehozok egy időzítőt, ami a Get\_world\_Delta\_Seconds() visszatéri értékével megegyező időközönként hívja meg a anim\_notify eseményt majd ezt a Timer\_Handler változóban eltárolom. A anim\_notify eseménynek meg kell vizsgálnia hogy a Get\_PlaybackPosition in Fames(Sprite) függvény nagyobb-e a Notify\_Frame értékénél, ha igen akkor beléphetünk a Do\_Once kapuba, amit majd az új időzítő létrehozásánál visszaállítok, majd meghívom a fire\_event eseményt és a Clear\_and\_Invalidate\_Timer\_by\_Handle(Timer Handler) függvényt.

### **5.3.3. Fire\_event**

A fire\_event esemény egy switch on State\_enum-ot hív meg, más animációknál más eseménynek kell megtörténnie.

## **5.4. Event AnyDamage**

Az Event AnyDamage esemény akkor fog lefutni, ha a karakter nullánál nagyobb sebzést kap. Amint a karakter sebzést kap át kell állítani a State-et hit-re azaz a karaktart éppen sebzik majd meg kell hívni a Update\_anim() függvényt, de előtte meg kell vizsgálni, hogy éppen hit animációt játszunk, amit a Get\_flipbook(Sprite) visszatérési értéke ad meg, ha az nem egyezik meg a witch\_hit-tel, akkor meg hívom a Update\_anim() függvényt, majd a sebzés érékét csökkentem a karakter armor (páncél) érékével, ha a ez az érték kisebb lenne mint 1 akkor 1-re állítom be az érékét majd azt kivonom a current\_hp változóból, ha az eredmény  $\leq 0$  akkor veszített a játékos. Megállítom a játékot, majd meghívom a Game\_over eseményt-et.

## **5.5. Event BeginPlay**

Az Event BeginPlay az első esemény a konstruktur után. Én két időzítőt hozok itt létre az egyik regenerációért lesz felelős (regen esemény) a másik meg az animációk alaphelyzetbe állításáért (trail esemény) és a játékos karakter utáni effektért felelős.

### 5.5.1. Trail

A karakter után elhalványuló körvonalak jelennie meg, hogy a mozgást szemléltesse.



3. ábra játékos körvonalak

Az effektet Niagara-val készítettem, ami az Unreal Engine-be épített VFX készítő rendszer.

Létrehoztam egy NiagaraEmitter-t player\_VFX névvel, ami a Fontain Emitter sablont használja. Töröltem a Spawn\_Rate, Shape\_Location, Add\_Velocity, Gravity\_Force, Drag, Solve\_Forces\_and\_Velocity komponenseket.

Hozzáadtam a Spawn\_Burst\_Instantaneous, Initial\_mesh Orientation, Sub\_UVAnimation komponenseket.

Módosítottam a Loop\_Behavior-t a Emitter\_State-ben „Once”-ra. A Sub\_UVAnimation-ban módosítottam az End\_Frame-et 7-ra, mert 8 képkocka hosszú az animáció és a nulladik képkockánál kezdődik, majd beállítottam a Sprite\_Renderer-ben a Material-t a player\_material-ra.

Megvizsgálom, hogy a karakter éppen milyen állapotban van, ha State.idle=False akkor meghívom a Spawn\_System\_at\_Location() függvényt. A System\_Template-nek a player\_VFX\_System-et állítottam és a Location bemeneti értéke az a Get\_Actor\_Location() + 0,0,-3 vektor eredménye lesz, így a létrehozott „Trail” effekt az a karakter alatt lesz és nem fogja eltakarni azt sem az ellenfeleket.

Megvizsgálom, hogy a karaktert vissza kell-e álltani „idle” helyzetbe ezt a Reset\_To\_Idle() visszatérési értéke fogja megadni, ha igen akkor a State-et módosítom „idle”-re és meghívom az Update\_Anim() függvényt.

### **5.5.2. Regen**

Ez az esemény meghívja a Heal(Heal) Függvényt a Heal nevű bemeneti paraméterrel, amit azt adja meg, hogy mennyit fog a karakter gyógyulni. A Heal értékét a Base\_Stats.Regen értéke adja meg.

### **5.6.Heal()**

Megvizsgálom, hogy a bejövő érték plusz a Current\_Hp az nagyobb-e a Base\_Stats.hp értékénél, ez azért fontos, hogy a játékosnak ne lehessen több élete, mint a maximum, de ha mégis több lenne az összeadás után akkor visszaállítom a Base\_Stats.hp-ra a Current\_Hp-t

### **5.7.Reset\_To\_Idle()**

Két esetben kell igazzal visszatérnie ennek a függvénynek: ha State.run= True és a Vector\_Lenght(Get\_Velocity()) értéke nulla vagy a Is\_Playing(Sprite)-nek a tagadása és a Vector\_Lenght(Get\_Velocity()) értéke nulla

### **5.8.Add\_Xp(Xp, New\_Xp)**

Létrehoztam egy Current\_xp nevű Real változót, ami a tapasztalat pontokat fogja tárolni. A current\_xp értékét úgy kapom meg hogy az újonnan kapott tapasztalatot megszorzom a (Base\_Stats.Xp+1) kifejezéssel majd hozzáadoma Current\_xp-hez, vagy ha nem újonnan kapott tapasztalat akkor szorzó 1 lesz. A Current\_xp nagyobb, mint a Xp\_needed (ez egy Real változó, amiben a következő szinthez szükséges összes tapasztalat lesz eltárolva), ha igen akkor meghívom a Level\_Up() függvényt.

### **5.9.Level\_Up()**

Egy Level nevű Integer-ben eltárolom a karakter szintjét. Inkrementálom a Level változót majd típuskényszerítés az ingame\_hud-ba a Get\_HUD(Get\_player\_controller(0)) függvény visszériérékével és meghívom Level\_Up\_Hud-ot.

## **5.10. Add\_Extra\_Xp(Xp)**

Kivonom a Current\_xp-t a Xp\_needed-ből és az eredményt eltárolom a Xp\_left Real változóban, mert ez a függvény szint lépés után fog csak lefutni, és itt tárolom el az extra tapasztalatot és állítom vissza a Current\_xp-t. Az Xp\_left értéke ha nagyobb mint nulla, akkor amikor a karakter szintet lépet még több tapasztalata volt mint amennyi kellett volna a szintlépéshez, ezért ezt a többletet most vissza adom az Add\_xp(Xp\_left, True) függvénnel.

## **5.11. Fix\_field\_Of\_View()**

Különböző elbontású monitorokon különböző méretű lenne a látószög ezért létrehoztam ezt a függvényt, amit egységesíti a látószöget minden monitoron. A Get\_Viewpoer\_size(Get\_player\_Controller(0)) X és Y kimenetét elosztom egymással (X/Y) majd megszorzom 35-tel ez lesz a Camera.Field\_Of\_View értéke.

## **5.12. Konstruktor**

Get\_Data\_Table\_Row(player\_datatable,witch) függvényt hívja meg a konstruktor majd elmenti a függvény kimeneti értékét a Base\_Stats nevű változóba, majd a Base\_Stats.Hp értékét beállítja a Current\_Hp értékének. A Base\_Stats.Speed értéke lesz a Character\_Movement.Max\_Walking\_Speed értéke.

## **5.13. Event Tick esemény**

Minden egyes képkockanál meghívom a Fix\_Field\_Of\_View() függvényt, mert lehet hogy a játékos módosítja a játékkablak méretét, és akkor is a fix látószöget kell beállítani.

## **5.14. Base\_Player\_Stats**

A Base\_Player\_Stats struktúra tartalmazza azt az adat modellt, amely leírja a játékos összes lehetséges attribútumát: életrejét, sebességét, páncélját, regenerációs képességét, sebzését, támadásainak területi méretét, támadásainak hosszát, lövedékszámát, újratöltési képességét, szerencséjét, tapasztalatát, és begyűjtött lelkek számát

## **6. Master\_ai**

Létrehoztam egy „master\_ai” nevű Blueprint Pawn osztályt, ami tartalmaz egy Box Collision komponenst és egy PapperFlipbook komponenst Ez az osztály lesz felelős az ellenfelekért kivéve a „Boss”-ért. Az ellenfelek egymást tudják majd tolni szóval kell majd fizikájuknak lennie, nem lesznek támadás animációk, csak ha a játékos túl közel kerül hozzájuk akkor fog megsebződni. Az ellenfelek csak kinézetben és attribútumokban fognak eltérni, így nem lesz szükség felülírásra, mert egy adattáblából fogom ezeket az adatokat betölteni.

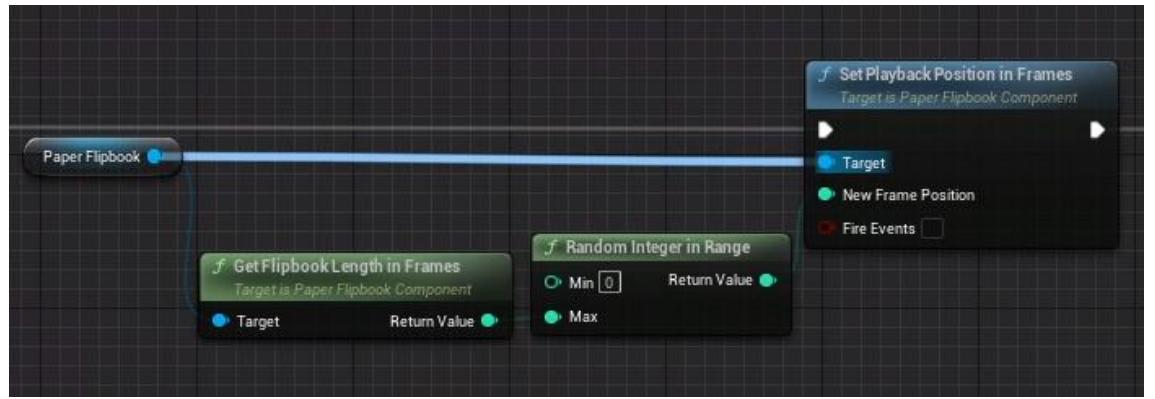
### **6.1.Konstruktor**

Mivel egy osztály van, és csak adat különbségek vannak, ezért, hogy könnyebb dolgom legyen a játéklogika megírásánál már most implementáltam a címkéket így egy egyszerű beépített függvény is vissza fogja adni, hogy hány darab van az egyes ellenfelekből.

Betöltöm az adatokat a Get\_Data\_Table(ai\_stats\_DT) függvény segítségével és eltárolom a Data változóban, majd beállítom a Hp Real változóban a lény életpontját, ami t a Data.Stats.Hp-ból kapok meg. Beállítom a Current\_Hp a Hp értékére. Mivel minden lénynek más mérete van ezért mindig máshol lenne a középpontja a karakterek így ezt egyesével kell ellensúlyozni ezt az Data.Stats.Offset-ből kapom meg és mivel minden lény külön méretű ezért a „Box”-ot is növelnem vagy csökkentenem kell ezt a Data.Stats.Szie-ból kapom meg. Beállítottam a Set\_Flipbook(Paper\_Flipbook, Data.Stats.Run) függvény segítségével a futás animációját majd ezt elindítottam a Play(Paper\_Flipbook) függvénnel

### **6.2.Event\_Begin\_Play**

Mivel egyszerre több mint egy ellenfelet fogok megjeleníteni, szerintem nagyon furcsa lenne, ha minden a pl.: száz lény egyszerre lépne vagy legalábbis ugyanazt az animációt játszaná le ugyan akkor ezért hozzáadtam egy kis véletlenszerűséget a Set\_Playback\_Position\_in\_Frames(Paper\_Flipbook,Random\_Integer\_in\_Range(0,Get\_Flipbook\_Length\_in\_Frames(Paper\_Flipbook))) függvény segítségével, ami így néz ki Blueprint-ben:



4. ábra Blueprint változat

Létrehozok egy időzítőt, ami a mozgásért, az animáció visszaállításáért és a karakter irányáért lesz feleős. Az esemény, amit meghív az időzítő az a Event\_tick\_for\_stuff névre hallgat. Létrehozok egy időzítő, ami a tick nevű eseményt fogja meghívni, ami azt fogja figyelni, hogy a lény kikerült-e a játékos képernyőéről, ha igen akkor majd az ellenkező oldalon meg fog jelenni.

#### 6.2.1. Event\_tick\_for\_stuff

Megvizsgálom, hogy a lény ütközik-e a játékos karakterével, ha nem akkor meghívom az Add\_Impulse() függvényt, aminek az Impulse bemenetét a következő módon fogom kiszámítani: tudnom kell milyen irányban van a játékos karakter ezt a következő módon fogom megállapítani:

```
Find_Look_at_Rotation(Get_Actor_Location(self),Get_Actor_Location()
Get_player_character(0))) majd ennek a vektornak a hosszát megszorzom a
Data.Stats.Speed-el ami a lény sebességét fogja visszaadni, és a z értéket nullával helyettesítem, így a mozgás mindenkorán két dimenziós lesz.
```

Megvizsgálom, hogy a Paper\_Flipbook éppen játszik-e animációt, ha igen akkor megnézem, hogy a lény a játékos karakterfelé néz-e, ha igen akkor nem csinálok semmit.

A Paper\_Flipbook éppen nem játszik le animációt, akkor meghívom a Play\_Anim(Data.Run,true) függvényt ami eljátssza a lényhez tartozó futás animációt.

A lény nem a játékos felé néz akkor a Set\_Relative\_Rotation(Paper\_Flipbook,Rotation) függvény segítségével megfordítom hogy a helyes irányba nézzen.

### **6.2.2. Tick**

Megvizsgálom, hogy a lény kikerült volna a képernyőről ezt a Convert\_World\_Location\_To\_Screen\_Location(Get\_Player\_Controller(0), Get\_Actor\_Location(self)) függvény segítségével csinálom ha a lény nincs a képernyőn akkor meghívom a Set\_Actor\_Location(self,New\_Location()) függvényt.

## **6.3. Event\_Actor\_Begin\_overlap**

Megvizsgálom, hogy a dolog, amivel ütközött a lény annak van-e „player” címkeje, ha igen akkor meghívom az Apply\_Dmg eseményt, majd létrehozok egy időzítőt, ami 0.25 másodpercenként ismétlődik és eltárolom egy Damage\_Timer\_Ref-ben. Fontos meg hívni az Apply\_Dmg eseményt az időzítő előtt, mert az időzítő csak 0.25 másodperc múlva hívta volna meg először és addigra lehet, hogy a játékos már nem ütközik a lénnel, ezért nekem az kell, hogy abban a pillanatban amikor a játékos hozzáér az ellenfélhez megsebződjön és minél tovább hozzáér annál több sebzést kapjon.

### **6.3.1. Apply\_dmg**

Meghívom az Apply\_Damage(target, DataStats.Dmg, self) függvényt, itt sebzi meg lény a játékos karaktert és minden 0.25 újra meg fogja sebezni(amig a játékos nem lesz elég távol a lénytől)

## **6.4. Event\_Actor\_End\_Overlap**

Megvizsgálom, hogy a dolog amivel már nem ütközik a lény annak van-e player címkeje ha igen akkor meghívom a Clear\_and\_Invalidate\_Timer\_by\_Handle(Damage\_Timer\_Ref) függvényt, mert csak a játékosnak van „player” címkeje ezért ha valaki már nem ütközik a lénnel és volt „player” címkeje akkor az csak a játékos lehetett.

## **6.5. Event\_Any\_Damage**

Eltárolom a sebzés értékét egy Damage nevű Real változóban, majd elmozdítom a lényt az Add\_Impulse(Boksz, Impulse) függvény segítségével, ahol az impulzus értéke, úgy számítható ki, hogy: Get\_Forward\_Vector(Find\_Look\_at\_Rotation(Get\_actor\_Location(self), Get\_Actor\_Location(Damage\_Causer))) vektor visszatérésiértékét megszorzom -5000-rel, hogy hátrafelé mozduljon el, majd a Z értéket nullával helyettesítem és ez lesz impulzus értéke.

Meghívom a Play\_Anim(Data.Take\_Hit,False) függvényt, majd a Current\_Hp-ból kivonom a Damage értékét és megvizsgálom az eredményt ha az kisebb mint nulla akkor

meghívom az Play\_Anim(Data.Death,False) hogy a halál animáció lejátszódjon majd megállítom a karaktert a Clear\_and\_Invalidate\_Timer\_by\_Handle(Impulse\_Force) függénnyel (ehhez a időzítőhöz volt a mozgás kötve) Kikapcsolom a fizikát ennél a lénynél a Set\_Simulate\_Physics(Box,False) függvény segítségével, majd kikapcsolom az ütközést is a Set\_Collision\_Enabled(Box, No\_Collision) függvény segítségével. Elrejtem a karaktert így a játékos nem láthatja de még nem is lesz törölve egyből, ezt a Set\_Actor\_Hidden\_In\_Game(self,True) függvény segítségével csináltam.

Meghívom a Cache\_Achievements(Get\_Player\_Controller(0)) függvényt hogy betöltsem a mérföldköveket, majd lekérdezem a lényhez specifikus mérföldkő progresszió állapotát a Get\_Cached\_Achievement\_Progress( Get\_Player\_controller(0), Data.Achievement) függvény segítségével. A „Progress” értéket növelem egyel majd meghívom a Write\_Achievement\_Progress(Get\_Player\_Controller(0), Data.Achievement, (inkrementált progresszió érték)). Sikerességtől függetlenül meghívom a Destory\_Actor(self) függvényt, hogy töröljem a lényt a pályáról.

## 6.6. Event\_Destoryed

Ez az esemény akkor fog lefutni, ha a lény meghalt, és itt kell kezelnem a tárgy és tapasztalatot is.

Meghívom a Spawn\_Actor\_From\_Class(Xp,Transform,Data.Xp\_Data) függvényt a transzform bementi értke a követkőképpen számítom:

Get\_Actor\_location(self) visszaadja a helyet, de nekem kell még irány és méret is, amik a következők: irány: 90,0,270 méret:1,1,1

## 6.7. Play\_Anim()

Ennek a függvénynek a feladata, hogy beállítsa az animációt és elindítsa azt. Van két bementi értéke az egyik a Flipbook benne van egy hivatkozás az új Flipbook-ra a másik Looping névre hallgat és ez egy Boolean ebben az az érték van eltárolva, hogy az animációnak kell-e ismétlődni.

Meghívom a Set\_Flipbook(PaperFlipbook,Flipbook) függvényt hogy beállítsam az új Flipbook-ot, majd a Set\_Looping(PaperFlipbook,Looping)függénnyel beállítom az ismétlőést és végül elindítom az animációt a Play(PaperFlipbook) függénnyel.

## **6.8.New\_Location()**

Ez a függvény adja majd vissza, hogy hova kell helyezi a lényt miután kikerült a képernyőről. Nekem úgy kell eltolnom a lényt, hogy az pl.: a képernyő baloldalán volt utoljára és a jobb oldalon jelenjen meg ezt úgy lehet megcsinálni kivonom a játékos helyzetéből a lény helyzetét, majd ezt megszorzom 0.9-del (hogy a képernyőn belül jelenjen meg és ne kerüljön egy végtelen ciklusba, ahol jobbról balra lenne folyamatosan mozgatva) majd ezt az értéket hozzáadom a játékos helyzetéhez és ez lesz az a helyvektor, amit keresek.

## **6.9.Master\_ai Alapértékek**

### **6.9.1. Box**

Box\_Extent: ezt dinamikusan módosítom az adattáblából betöltött adattal

Apply\_impulse\_on\_Damage: False

Nekem nem kell ez a funkció, mert megírtam a sajátomat az Event\_any\_Damage eseményben

Simulate\_Physics: True

Ez azért fontos, mert a karakter mozgása a fizika alapú és anélkül nem működne.

Bekapcsoltam a karakter súlyát: Mass(kg):20

Linear\_Damping:45

Súrlódás növeli így nem fog olyan messze elcsúszni a karakter.

Should\_Update\_physics\_Volume: True

Mozgásnál frissíti a fizikai teret.

Can\_Character\_Step\_Up\_On: False

Nem szeretném, hogy karakterek rá tudjanak lépni.

Collison\_Enabled: Collison\_Enabled(Query and Physics)

Object\_Type: ai

	Ignore	Overlap	Block	
<b>Collision Responses</b> ⓘ	—	—	—	
<b>Trace Responses</b>				
Visibility	—	✓	—	↔
Camera	—	✓	—	↔
<b>Object Responses</b>				
WorldStatic	✓	—	—	↔
WorldDynamic	✓	—	—	↔
Pawn	✓	—	—	↔
PhysicsBody	✓	—	—	↔
Vehicle	✓	—	—	↔
Destructible	✓	—	—	↔
items	✓	—	—	↔
ai	—	—	✓	↔
player	—	—	✓	↔
spells	—	✓	—	↔

5. ábra master\_ai ütközési beállításai

Always\_Create\_Physics\_State: True

Ennek a karakternek muszáj, hogy minden legyen fizikai állapota.

Multi\_Body\_Overlap: True

Egyszerre akár több objektummal is ütköznie kell tudnia.

### 6.9.2. PaperFilpbook

Can\_Character\_Step\_Up\_On: False

Colision\_Presets: Block\_All

## 6.10. Mob\_Types

A Mob\_types az egy enumeráció, ami tárolja a lehetséges lények megnevezését: „Eye”, „Goblin”, „Mushroom”, „Skeleton”, „Necromancer”, „King”, „Hounds”

## 6.11. Ai\_data

Az Ai\_data az egy olyan struktúra, amely meghatározza az ellenfelekhez tartozó adatok modelljét tartalmazza az ai\_stats nevű struktúrát és ezen kívül még három Flipbook hivatkozást a futás animációhoz, a megsérülés animációhoz, és a halál animációhoz, és a halál után megjelenítendő tárgy megnevezését, és a mérföldkő nevét is.

### **6.11.1. Ai\_stats Struktúra**

Az ai\_stats tartalmazza az ellenfelekhez tartozó számbeli adatok modelljét: életerő (hp), sebzés (dmg), sebesség (speed), ütközési doboz mérete (size), középre igazításhoz szükséges eltolási vektor (offset).

### **6.12. Ai\_stats\_DT Adattábla**

Az ai\_stats\_DT nevű adat tábla tartalmazza az adatokat az ellenfelekhez. A sorazonosító megegyezik a Mod\_types soraival. Itt lehet módosítani az egyes értékeket, vagy ha új lényt akarok hozzáadni azt is itt kell megtennem (a Mod\_types enumerációban a megnevezést meg kell tennem előtte)

## 7. Ellenfél fajták

Az ellenfelek egy adat táblából vannak betöltve, ezt az adat táblát akár lehetne egy szerveren is tárolni és akkor azonnal lehetne módosítani az értékeit.

### 7.1. Eye

Az Eye egy szörny. Két szárny és egy nagy szembolyóból áll. Normál sebességgel közlekedő karakter, ami az egyik legkönnyebben elpusztítható, halál után tapasztalatot hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	4.0
Dmg	1.0
Speed	250.0
Size	8.0, 13.0, 4.0
Offset	4.0, 3.0, 0.0
Run	eye_flight
Take_hit	Eye_Take_Hit1
Death	Eye_Death1
Xp_data	lesser_xp
achievement	bat

I. táblázat az Eye szörnyeteg adatai

## 7.2. Goblin

A goblin egy szörny, ami egy zöld manóra hasonlító teremtmény és kés van a kezében. Normál sebességgel közlekedő karakter, ami az moderált mennyiségű életerővel rendelkezik, de az egyik leggyengébb karakter, halál után tapasztalatot hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	22.0
Dmg	3.0
Speed	250.0
Size	16.0, 10.0, 2.0
Offset	10.0, 3.0, 0.0
Run	goblin_run
Take_hit	goblin_hit
Death	goblin_death
Xp_data	lesser_xp
achievement	goblin

2. táblázat az *Goblin* szörnyeteg adatai

### 7.3. Skeleton

A skeleton egy szörny, ami egy csontokból álló karakter és egy kard és egy pajzs van a kezében. Normál sebességgel közlekedő karakter, ami az moderált mennyiségű életerővel rendelkezik, de az egyik leggyengébb karakter, halál után tapasztalatot hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	35.0
Dmg	4.0
Speed	200.0
Size	22.0, 10.0, 2.0
Offset	0.0, 4.0, 0.0
Run	skeleton_run
Take_hit	skeleton_hit
Death	skeleton_death
Xp_data	lesser_xp
achievement	skeleton

3. táblázat az Skeleton szörnyeteg adatai

## 7.4. Mushroom

Ez a szörny egy ember méretű és emberi vonásokat rendelkező gombaszerű karakter. Normálnál lassabb sebességgel közlekedő karakter, ami az moderált mennyiségű életerővel rendelkezik, moderáltan erős karakter, halál után tapasztalatot hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	300.0
Dmg	10.0
Speed	150.0
Size	18.0, 8.0, 2.0
Offset	7.0, 0.0, 0.0
Run	mushroom_run
Take_hit	mushroom_Take_Hit
Death	mushroom_death
Xp_data	xp
achievement	mushroom

4. táblázat az *Mushroom* szörnyeteg adatai

## 7.5.Necromancer

Ez a szörny egy ember méretű és emberi vonásokat rendelkező lebegő karakter egy bottal a kezében. Normális sebességgel közlekedő karakter, ami az jelentős mennyiségű életerővel rendelkezik, erős karakternek számít, halál után tapasztalatot hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	666.0
Dmg	12.0
Speed	300.0
Size	20.0, 10.0, 2.0
Offset	33.0, 0.0, 0.0
Run	Necromancer_walk
Take_hit	Necromancer_hit
Death	Necromancer_death
Xp_data	significant_xp
achievement	necromancer

5. táblázat az Necromancer szörnyeteg adatai

## 7.6. King

Ez a szörny egy ember és egy királyra hasonlít. A fején egy arany korona található, nyakában pedig arany ékszerek. Normálnál egy kicsit nagyobb sebességgel közlekedő karakter, ami az jelentős mennyiségű életerővel rendelkezik, erős karakternek számít, halál után tapasztalatot hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	333.0
Dmg	15.0
Speed	300.0
Size	20.0, 8.0, 2.0
Offset	-9.0, 0.0, 0.0
Run	spr_KingWalk_strip_no_bkg1
Take_hit	spr_Kinghit_strip_no_bkg2
Death	spr_KingDeath_strip_no_bkg1
Xp_data	significant_xp
achievement	None

6. táblázat az King szörnyeteg adatai

## 7.7. Hounds

Ez a szörny egy pokoli változata a kutyáknak. A hounds egy speciális szörny, amelyt csak a Boss tud lehívni. Nagyon nagy sebességgel közlekedő karakter, ami az jelentős mennyiséggű életerővel rendelkezik, erős karakternek számít, halál után soul-t (lelket) hagy maga után, amit a játékos fel tud venni. A következők a beállításai:

Hp	100.0
Dmg	12.0
Speed	400.0
Size	10.0, 20.0, 1.0
Offset	3.0, 0.0, 0.0
Run	hell-hound-walk1
Take_hit	None
Death	None
Xp_data	soul
achievement	None

7. táblázat az Hounds szörnyeteg adatai

## **8. Tárgyak és Tapasztalat**

### **8.1. Enumerációk**

#### **8.1.1. Bonus\_stats\_names**

A bonus\_stats\_names enumeráció tartalmazza a játékoskarakterhez tartozó fejlesztések nevét: hp, hp\_regen, armor, bonus\_dmg, duration, projectal\_count, cd, luck, xp, soul, aoe, speed

#### **8.1.2. Consumables\_names**

A consumables\_names enumeráció tartalmazza az olyan tárgyakat, amelyeket a játékos csak egyszer tud interakcióba lépni pl.: tapasztalat, ha a játékos felszedi a tapasztalatot akkor az eltűnik a pályáról, így ugyanazzal a tapasztalattal nem tud interakcióba lépni még egyszer. Az enumeráció a következő megnevezéseket tartalmazza: soul, lesser\_xp, xp, significant\_xp.

#### **8.1.3. Weapon\_names**

A weapon\_names enumeráció tartalmazza a játékos számára elérhető fegyvezek megnevezését: energy\_ball, boom\_fire\_item, ice\_ball\_item, double\_shot, torch, beam

## **8.2. Struktúrák**

### **8.2.1. Bonus\_stats\_struct**

A bonus\_stats\_struct névre hallgató struktúra definiálja az adatmodellt a játékoskarakterhez tartalmazza az osztály megnevezését (ami Master\_Bonus\_Stats típusú), tartalmazza a fejlesztéshez tartozó adatokat (ami Base\_Player\_Stats típusú) és tartalmazza a felhasználói felület létrehozásához szükséges adatokat (ami Ui\_Data\_Struct típusú)

### **8.2.2. Consumables\_struct**

A consumables\_struct struktúra tartalmazza a földről felvehető egyszer használatos tárgyak adatik. Tartalmaz egy Paper\_Flipbook hivatkozást ez lesz az az animáció, amit a tárgy fog lejátszani, amíg a földön van. Tartalmaz egy potenciál értéket, és egy Sound\_Base hivatkozást is, ami azt a hangot fogja tárolni, amit tárgy felvétel után szeretnék lejátszani.

### **8.2.3. Item\_anim\_data\_struct**

Az item\_anim\_data\_struct struktúra fogja megadni azt az adatmodellt, ami alapján én a fegyverek animációját létrehozom. Tartalmazza az animációhoz szükséges Paper\_Flipbook hivatkozást egy Boolean változót, hogy az animációnak ismétlődni kell-e egy méret vektort, és egy Sound\_Cue hivatkozást, hogy hangot tudjak adni a fegyvereknek.

### **8.2.4. Ui\_data\_struct**

Az ui\_data\_struct struktúra fogja megadni azt az adatmodellt, ami alapján meg tudom az egyes tárgyakat jeleníteni a felhasználói felületen, ennek az adatmodellnek tartalmaznia kell a tárgy nevét és egy ikont a kinézetéről.

### **8.2.5. Weapon\_stats\_struct**

A weapon\_stats\_struct struktúra fogja megadni azt az adatmodellt, ami tartalmazza a fegyver működéséhez vagy fejlesztéséhez szükséges adatokat tartalmaznia kell a következő adatokat: sebzés, sebesség, méret, lövedékszám, újratöltési idő, áttörés, időtartam.

### **8.2.6. Weapon\_data\_struct**

A weapon\_Data\_struct struktúra fogja megadni azt az adatmodellt, ami majd egy fegyver összes adatát határozza meg. Tartalmaz egy Item\_anim\_data tömböt arra az esetre, ha a fegyvernek több animációja van, tartalmazza a fegyver osztály nevét, tartalmazza a fegyver lövedékének az osztály nevét, tartalmazza a fegyverhez tartozó adatok tömbjét, ami Weapon\_stats típusú ( minden szint egy elem ebben a tömbben), tartalmazza a felhasználói felület létrehozásához szükséges adatokat, ami UI\_Data\_Struct típusú és tartalmazza a mérföldkő megnevezését is.

### **8.2.7. Consumables\_datatable**

A consumables\_datatable táblában tárolom a tapasztalat és a lélek adatait.

### **8.2.8. Bonus\_stats\_datatable**

A bonus\_stats\_datatable táblában tartalmazza a fejlesztések adatait.

### **8.2.9. Weapon\_datatable**

A weapon\_datatable tábla tartalmazza a fegyverek adatait.

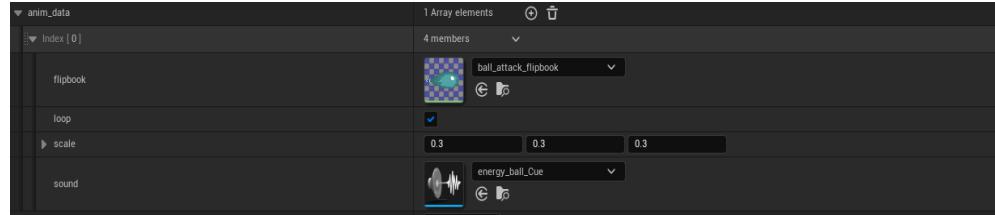
### *energy\_ball*

Az *energy\_ball* a következő adatokat fogja betölteni:

item_class	energy_ball
spawnable	energy_ball_projectal
achievement_name	scepter

*8. táblázat energy\_ball adatai 1*

Az *anim\_data* tömb egy elemet tartalmaz:



A *weapon\_stats\_new* egy három elemet tartalmazó tömb:

Index[0]	érték	Index[1]	érték	Index[2]	érték
Damage	10.0	Damage	10.0	Damage	10.0
Speed	100.0	Speed	10.0	Speed	10.0
AoE_size	0.0	AoE_size	0.0	AoE_size	0.0
Projectal_count	2	Projectal_count	0	Projectal_count	1
Cd	2.0	Cd	0.0	Cd	0.0
Penetration	1	Penetration	0.0	Penetration	3
Duration	10.0	Duration	0.0	Duration	0.0

*9. táblázat energy\_ball adatai 2*

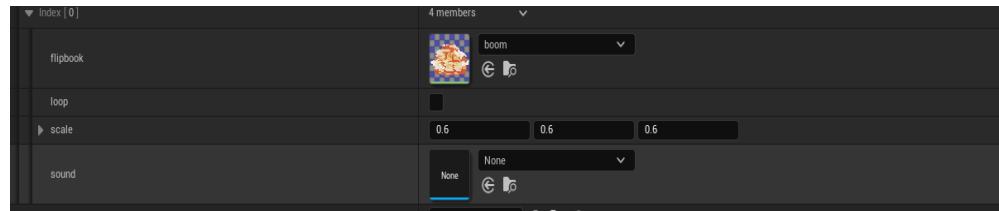
### *boom\_fire\_item*

Az *boom\_fire\_item* a következő adatokat fogja betölteni:

item_class	boom_fire_item
spawnable	boom_fire_projectal
achievement_name	boomball

10. táblázat *boom\_fire\_item* adatai 1

Az *anim\_data* tömb egy elemet tartalmaz:



A *weapon\_stats\_new* egy három elemet tartalmazó tömb:

Index[0]	érték	Index[1]	érték	Index[2]	érték
Damage	10.0	Damage	20.0	Damage	20.0
Speed	0.0	Speed	0.0	Speed	0.0
AoE_size	0.0	AoE_size	0.0	AoE_size	1.0
Projectal_count	2	Projectal_count	0	Projectal_count	0
Cd	2.0	Cd	0.0	Cd	0.0
Penetration	0	Penetration	0	Penetration	0
Duration	4.0	Duration	0.0	Duration	0.0

11. táblázat *energy\_ball* adatai 2

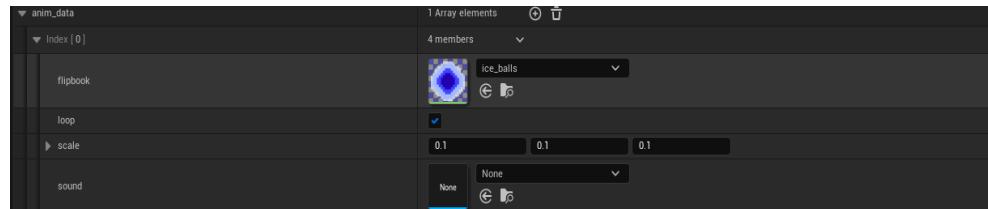
### *ice\_ball\_item*

Az *ice\_ball\_item* a következő adatokat fogja betölteni:

item_class	ice_ball_item
spawnable	ice_ball_projectal
achievement_name	iceball

12. táblázat *ice\_ball\_item* adatai 1

Az *anim\_data* tömb egy elemet tartalmaz:



A *weapon\_stats\_new* egy három elemet tartalmazó tömb:

Index[0]	érték	Index[1]	érték	Index[2]	érték
Damage	5.0	Damage	10.0	Damage	30.0
Speed	150.0	Speed	0.0	Speed	20.0
AoE_size	0.0	AoE_size	0.0	AoE_size	1.0
Projectal_count	3	Projectal_count	2	Projectal_count	3
Cd	2.0	Cd	0.0	Cd	1.0
Penetration	1	Penetration	0	Penetration	1
Duration	10.0	Duration	0.0	Duration	0.0

13. táblázat *ice\_ball\_item* adatai 2

### *double\_shot*

Az double\_shot a következő adatokat fogja betölteni:

item_class	demon_pect_item
spawnable	none
achievement_name	demonpact

14. táblázat *double\_shot* adatai 1

Az anim\_data tömb nem tartalmaz elemet.

A weapon\_stats\_new egy három elemet tartalmazó tömb:

Index[0]	érték	Index[1]	érték	Index[2]	érték
Damage	0.0	Damage	0.0	Damage	0.0
Speed	0.0	Speed	0.0	Speed	0.0
AoE_size	0.0	AoE_size	0.0	AoE_size	1.0
Projectal_count	1	Projectal_count	0	Projectal_count	0
Cd	5.0	Cd	1.0	Cd	1.0
Penetration	0	Penetration	0	Penetration	0
Duration	0.0	Duration	0.0	Duration	0.0

15. táblázat *double\_shot* adatai 2

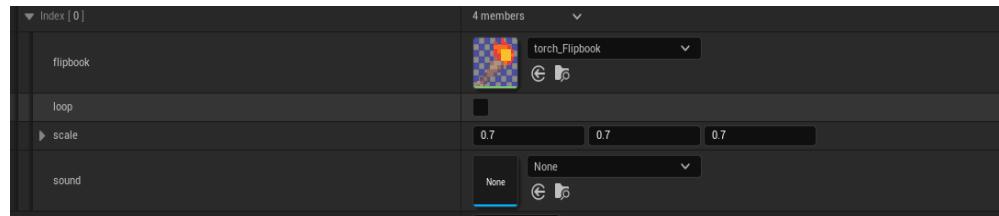
### *torch*

Az torch a következő adatokat fogja betölteni:

item_class	torch_item
spawnable	torch_projectal
achievement_name	torch

*16. táblázat torch adatai 1*

Az anim\_data tömb egy elemet tartalmaz:



A weapon\_stats\_new egy három elemet tartalmazó tömb:

Index[0]	érték	Index[1]	érték	Index[2]	érték
Damage	20.0	Damage	10.0	Damage	30.0
Speed	80.0	Speed	20.0	Speed	0.0
AoE_size	1.0	AoE_size	1.0	AoE_size	0.0
Projectal_count	1	Projectal_count	1	Projectal_count	0
Cd	4.0	Cd	0.0	Cd	1.0
Penetration	0	Penetration	0	Penetration	2
Duration	10.0	Duration	0.0	Duration	0.0

*17. táblázat torch adatai 2*

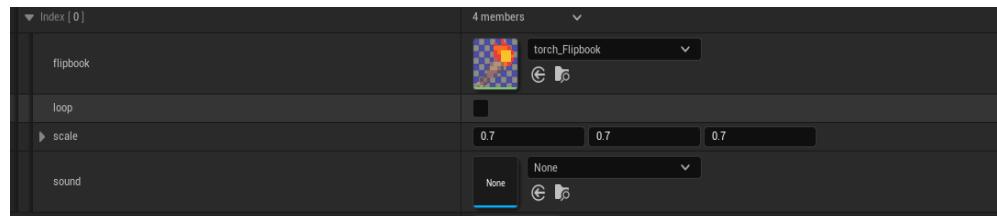
### *beam*

Az beam a következő adatokat fogja betölteni:

item_class	beam_item
spawnable	Beam
achievement_name	torch

18. táblázat beam adati 1

Az anim\_data tömb egy elemet tartalmaz:



A weapon\_stats\_new egy három elemet tartalmazó tömb:

Index[0]	érték	Index[1]	érték	Index[2]	érték
Damage	2.0	Damage	10.0	Damage	10.0
Speed	40.0	Speed	10.0	Speed	10.0
AoE_size	0.0	AoE_size	1.0	AoE_size	0.0
Projectal_count	1	Projectal_count	0	Projectal_count	0
Cd	0.25	Cd	0.0	Cd	0.0
Penetration	0	Penetration	0	Penetration	0
Duration	10.0	Duration	0.0	Duration	0.0

19. táblázat beam adatai 2

## **8.3.Master\_consumable**

A master\_consumable egy Paper\_Flipbook\_actor osztályból származtatott osztály, amit kiegészítettem egy Sphere\_Collision komponenssel, aminek az lesz a feladat, hogy a játékoskarakterrel való ütközést regisztrálja.

### **8.3.1. Event\_Actor\_Begin\_Overlap**

Megvizsgálom, hogy a objektum, amivel ütközött a tárgy annak van-e játékos címkeje, ha igen akkor meghívom a do\_stuff(Out\_Row.Potency) függvényt, majd meghívom a Play\_Sound\_2D(Out\_Row.Sound,3,1,0) függvényt és utána törlöm a tárgyat.

### **8.3.2. Do\_Stuff(Potency)**

A Do\_Stuff függvény az egy üres függvény, amit majd később kell felülírni.

### **8.3.3. Konstruktor**

A Get\_Data\_Table\_Row(consumables\_datatable,item\_name) függvény segítségével kiválasztom a consumables\_datatable táblából a megfelelő adatsot és eltárolom az Out\_Row változóban meghívom a Set\_Flipbook(Rander\_Component, Out\_Row.Sprite) függvényt, hogy beállítsam a az animációt, majd meghívom a Set\_Looping(Render\_Component, true) függvényt, hogy beállítsam az animációt ismétlődésre, és végezetül meghívom a Play(Render\_Component) függvényt, hogy eljátsszam az animációt.

### **8.3.4. Xp**

Az xp a master\_consumable osztályból van származtatva. Felülírtam a do\_stuff() függvényét, és konstruktorát. Ez az egy osztály lesz felelős a tapasztalat pontok megjelenítésért, háromféle tapasztalat pont van, de funkciójukat tekintve ez az egy osztály képes betölteni mindegyik szerepét.

#### *Konstruktor*

A létrehozok egy új name változót, és átadom annak az értékét a Item\_Name változónak majd megívom a szülői konstruktort.

### *do\_stuff()*

Master\_player típuskényszerítést hajtok végre a Get\_Player\_Character(0) függvény visszatéríti értékén, majd meghívom az Add\_Xp(Master\_player, Potency, False) függvényt.

### **8.3.5. Souls**

A souls nevű Blueprint osztály a master\_consumable osztályból van származtatva, ahol felülírtam a do\_stuff() függvényt. Ez az osztály felelős a lelkek megjelenítéséért.

### *do\_stuff()*

Master\_player típuskényszerítést hajtok végre a Get\_Player\_Character(0) függvény visszatéríti értékén, majd Master\_player.Base\_Stats.souls értékét megnövelem eggyel, és megszorzom a Potency értékével majd ezt az értéket felelő kerekítem és hozzáadom a lelkek mennyiségehez.

## **8.4.Fejlesztések**

A játékosnak többfajta fejlesztése lehet egyszerre, de maximum hét darab lehet. Létrehoztam egy master\_bonus\_stats nevű osztályt, amit a Actor\_Component osztályból származtattam. Az összes fejlesztést, meg lehetett volna oldani csak egy master\_bonus\_stats osztállyal, csak az nehézségeket okozott volna a későbbiekbén. Annak ellenére, hogy nincs funkcionális különbség az osztályok között (csak adat beli eltérés van) nekem egyszerűbb itt származtatni és minden egyes fejlesztésnek külön osztályt létrehozni, mert akkor nem kell foglalkoznom a címkézés problémájával.

### **8.4.1. Event\_Begin\_Play()**

A Get\_Owner(self) függvény visszatéríti értékén master\_player típuskényszerítést hajtok végre, majd eltárolom a AS\_Master\_Player változóban. Meghívom a Get\_Data\_Table\_Row(bonus\_stats\_datatable,name) függvényt és eltárolom a kapott adatsort a Data nevű változóban, majd a tárgy szintjét nullára állítom és meghívom a Update\_Stats() függvényt.

### **8.4.2. Update\_Stats()**

Frissíttem a tárgy adatait a következő módon: lekérdezem a játékos Base\_stats változó értékét majd hozzáadom a Data.Stats[Level] értékét, és az összeget eltárolom a játékos Base\_Stats változójában.

### **8.4.3. Level\_Up()**

Ez a függvény felelős a játékos fejlesztő tárgyának a szintlépéseiért.

A függvény nagyon egyszerű működik inkrementálom a Level változót majd meghívom a Update\_Stats() függvényt. Fejlesztési típusok:

*bonus\_aoe*

Ez a fejlesztés lesz felelős a játékos támadás méretének növeléséért.

*bonus\_armor*

Ez a fejlesztés lesz felelős a játékos páncél értékének növeléséért.

*bonus\_cd*

Ez a fejlesztés lesz felelős a játékos újratöltési idejének csökkentéséért.

*bonus\_damage*

Ez a fejlesztés lesz felelős a játékos sebzésének növeléséért.

*bonus\_duration*

Ez a fejlesztés lesz felelős a játékos támadás idejének növeléséért.

*bonus\_heath*

Ez a fejlesztés lesz felelős a játékos életrejének növeléséért.

*bonus\_luck*

Ez a fejlesztés lesz felelős a játékos szerencséjének növeléséért.

*bonus\_projectal*

Ez a fejlesztés lesz felelős a játékos lövedék számának növeléséért.

*bonus\_regen*

Ez a fejlesztés lesz felelős a játékos regenerációs képességének növeléséért.

*bonus\_souls*

Ez a fejlesztés lesz felelős a játékos által gyűjtött lelkek számának növeléséért.

*bonus\_speed*

Ez a fejlesztés lesz felelős a játékos sebességének növeléséért.

*bonus\_xp*

Ez a fejlesztés lesz felelős a játékos által gyűjtött tapasztalat növeléséért.

## **9. Fegyverek**

A játékosnak különféle fegyverei lehetnek egyszerre maximum hét darab fegyvere lehet (jelenleg csak 6 féle fegyver van létrehozva). A fegyverek automatikusan lőnek. Egy Olyan master\_weapon osztályt hoztam létre, ami automatikusan fogja betölteni a fegyver adatait, de ezen felül még származtatni is kellett, mert a fegyverek működése teljesen eltér egymástól.

### **9.1.Master\_weapon**

A master\_weapon osztályt a Actor\_Component osztályból származtattam, és a fegyvereket majd ebből az osztályból fogom létrehozni.

#### **9.1.1. Event\_Begin\_Play**

Betöltöm a fegyver adatait a Get\_Data\_Table\_Row(weapon\_datatable, Item\_Name) függvény segítségével, majd eltárolom a kapott sort a Weapon\_Data változóban, a fegyver szintjét nullára állítom a Level változón keresztül, majd elmentem a Current\_Stat nevű változóba a Weapon\_Data.Weapon\_Stats\_New[Level] értékét ez adja vissza a nullás szintű fegyver adatait. Eltárolom a játékos hivatkozását egy változóban, ennek az értékét úgy kapom meg hogy master\_player típuskényszerítést végzek a Get\_Player\_Chacter(0) függvény kimenetiértékén, végezetül meghívom a lövés, eseményt aminek a Do\_It\_Again nevet adtam.

#### **9.1.2. Do\_It\_Again**

a Do\_It\_Again esemény indítja el azt az eseménysort, ami a lövéséhez szükséges, amivel a játékos nem tudja irányítani, hogy a fegyver mikor és hova lő, ezért ezt automatizálni kellett, így létrehoztam egy ciklust, amit a Do\_It\_Again esemény indít el és önmaga meghívása lesz az utolsó lépése.

Létrehozok egy időzítőt, ami a Calculate\_Cd() idő elteltével végrehajtja a hozzárendelt fire eseményt (Sajnos a Set\_Timer\_by\_Event(fire, Calculate\_cd(), False) függvényel létrehozott időzítő intervallumát nem lehet dinamikusan módosítani, ezért kell a Do\_It\_Again eseménynek önmagát megírnia).

### **9.1.3. Fire**

Lekérdezem a lövedékszámot ez a játékos Base\_Stats.Projectal-ban van eltárolva és ehhez hozzáadom a fegyver lövedékszámát, ennek a két számnak az össze fogja megadni, hogy hány lövedéket kell kilőnie a fegyvernek, ezt az értéket eltárolom a Projectal változóban.

Eltárolom a Calaculate\_Transform() visszatérésiértékét egy változóban, aminek az a neve, hogy: Transform\_For\_Projectal, majd létrehozok egy új időzítőt a Set\_Timer\_by\_Function\_Name(self, spawn\_actor, Time, True) függvény segítségével, ahol a Time 0.1-del van elosztva a Projectal változó értékével, majd eltárolom a időzítő hivatkozását a Spawn\_Timer változóban (azért osztom 0.1 a lövedékszámmal, hogy a „golyók” ne egyszerre hagyják el a játékos karaktert).

### **9.1.4. Spawn\_Actor()**

A Spawn\_Actor() függvény felelős a lövedék létrehozásáért, a Do\_It\_Again esemény újra meghívásáért és önmaga leállításáért.

Mivel ez a függvény többször is le fog futni egy lövési fázisban, ezért használok benne egy Do\_Once kaput és a kapu utána hívom meg a Do\_It\_Again eseményt, ha ezt nem így tettek volna akkor csak egy lövedéket lőne ki a játékos.

Megvizsgálom, hogy a Projectal változónak mi az értéke, ha kisebb vagy nulla akkor meghívom a Clear\_and\_Invalidate\_Timer\_by\_Handle(Spawn\_Timer) függvényt, ha nagyobb az érték mint nulla, akkor dekrementálom az értékét és utána meghívom a SpawnActor() függvényt, hogy létrehozzam a lövedéket. A SpawnActor() függvény bemeneti értékei: Weapon\_Data.Spawnable az osztály, a transzformáció a Override\_Transform( Transform\_For\_Projectal) függvényből jön, az Anim\_Data a Weapon\_Data.Anim\_Data, a játékos hivatkozás a Player\_Ref változóból kaptam meg.

Utoljára meg meghívtam a Play\_Sound\_2D(Weapon\_Data.Anim\_Data[0].Sound 1,1,0, True) függvényt és ezzel hangot adtam a lövedéknek.

### **9.1.5. Level\_Up()**

Megvizsgálom, hogy a Level értéke kisebb-e, mint a fegyverhez tartozó Weapon\_Data.Weapon\_Stats\_New hossza, ha igen akkor inkrementálom majd megvizsgálom, hogy a Level értéke megegyezik a Weapon\_Data.Weapon\_Stats\_New hosszával, ha igen akkor meghívom az Achievement eseményt (mert az azt jelenti, hogy

a játékos elérte a maximális szintjét a fegyvernek). Frissítem a fegyver adatait a Weapon\_Data.Weapon\_Stats\_New[Level] értékeit hozzáadom a Current\_Stat értékeihez majd elmentem őket a Current\_Stat változóba.

#### **9.1.6. Calculate\_Cd()**

A függvénynek egy Float visszatérésiértéke van, amit úgy számolok ki, hogy összeszorozom a játékos (Base\_Stats.Cd+1) értéket a Current\_Stats.Cd értékével.

#### **9.1.7. Calculate\_Transform(Transform)**

A Calculate\_Transform(Transform) függvény feladata, hogy kiszámítsa milyen irányba legyen a lövedék kilőve.

#### **9.1.8. Override\_Transform(Transform)**

Az Override\_Transform(Transform) függvényt olyan esetekre hoztam létre, amikor támadásnál a lövedékeknek nem egy irányba kell menniük pl.: szeretném, hogy a karakter jobboldalán jelenjenek meg a lövedékek és képernyő jobboldala felé haladjanak, de 10 fokos szögeltéréssel az előző lövedékhez képest.

#### **9.1.9. Achievement**

Ebben az eseményben kezelem azt az eseményt, amikor a játékos eléri az egyik fegyverrel a maximális szintet.

Meghívom a Cache\_Achivements(Get\_Player\_Controller(0)) függvényt, majd betöltöm a mérföldkő állapotát a Get\_Cached\_Achievement\_Progress(Get\_Player\_Controller(0), Weapon\_Data.Achievement\_Name) függvény segítségével, majd megvizsgálom, hogy a mérföldkő állapota nem egy-e, ha nem egy akkor beállítom az értékét egyre és ezzel a játékos megkapja a mérföldkövet.

### **9.2.Energy\_ball**

Ez egy fegyver osztály a master\_weapon osztályból származtatva. Implementálom a Override\_Transform(Transform) függvényt és a Calculate\_Transform() függvényt.

#### **9.2.1. Calculate\_Transform()**

Kiválasztok véletlenszerűen egy ellenfelet és a játékos alapján létrehozok egy Transform kimenetiértéket.

A `get_All_Actors_with_Tag(“ai”)` függvény segítségével létrehozok egy tömböt, amiben csak az ellenfelekre mutató objektum hivatkozásai vannak. Kiválasztok véletlenszerűen egy elemet a tömbből és a `Find_Look_at_Rotation(Get_Actor_Location(Player_Ref),Get_Actor_Location(“random ellenfél”))` függvény segítségével kiszámolom hogy milyen irányba kell kilőni a lövedéket, majd a `Make_Transform(Get_Actor_Location(Player_Ref), 90, 0, Float, 1, 1, 1)` függvénnnyel létrehozom a kimenetiérték Transform visszatérési értékét. Ha nincsenek ellenfelek a pályán akkor a Float értéke egy véletlenszerű szám nulla és háromszázhatvan között.

### **9.2.2. Override\_Transform()**

A bemeneti Transform értéket (amit a `Calculate_Transform()` függvény fog létrehozni) elemeire bontom, majd Location és Scale értéket a változtatás nélkül visszaadom. A Rotation-t értékeire bontom, majd kiválasztom a Z értéket és hozzáadok egy véletlenszerű számot minusz húsz és plusz húsz között, majd ezzel az összeggel létrehozok egy új Rotator értéket a `Make_Rotator(X, Y, Z)` függvény segítségével.

## **9.3.Torch\_item**

Ez egy fegyver osztály a master\_weapon osztályból származtatva. Implementálom a `Override_Transform(Transform)` függvényt.

### **9.3.1. Calculate\_Transform()**

Kiválasztok véletlenszerűen egy ellenfelet és a játékos alapján létrehozok egy Transform kimenetiértéket.

A `get_All_Actors_with_Tag(“ai”)` függvény segítségével létrehozok egy tömböt, amiben csak az ellenfelekre mutató objektum hivatkozások vannak. Kiválasztok véletlenszerűen egyet a tömbből és a `Find_Look_at_Rotation(Get_Actor_Location(Player_Ref),Get_Actor_Location(“random ellenfél”))` függvény segítségével kiszámolom hogy milyen irányba kell kilőni a lövedéket, majd a `Make_Transform(Get_Actor_Location(Player_Ref), 90, 0, Float, 1, 1, 1)` függvénnnyel kimenetiértéke lesz a Transform visszatérési érték. Ha nincsenek ellenfelek a pályán akkor a Float értéke egy véletlenszerű szám nulla és háromszázhatvan között.

## **9.4.Ice\_ball**

Ez egy fegyver osztály, amelyet a master\_weapon osztályból származtattam. Implementálom a Override\_Transform(Transform) függvényt.

Meg keresem a legközelebbi ellenfelet, amihez meghívom a Get\_All\_Actors\_with\_Tag(„ai”) függvényt és a visszakapót tömbön egy ciklus végig megyek és megállítom a Get\_Distance\_To(Array\_Element, Player\_Ref) függvénnnyel a távolságot és eltárolom az értéket egy Distance nevű változóban, ha a Distance változó értéke nagyobb mint a Get\_Distance\_To(Array\_Element, Player\_Ref) függvénnnyel kapott visszatérésiértéknél akkor eltárolom a tömb elemét egy változóban, amit Target-nek hívok. Find\_Look\_at\_Rotation(Get\_Actor\_Location( Player\_Ref), Get\_Actor\_Location(Target)) függvény segítségével kiszámolom hogy milyen irányba kell kilőni a lövedéket, majd a Make\_Transform(Get\_Actor\_Locaiton(Player\_Ref), 90, 0, Float, 1, 1, 1) függvénnnyel kimenetiértéke lesz a Transform visszatérési érték. Ha nincsenek ellenfelek a pályán akkor a Float értéke egy véletlenszerű szám minusz száznyolcvan és pozitív száznyolcvan között.

## **9.5.Demon\_pact\_item**

A Demon\_pact\_item az egy speciális fegyver, ami az egyik másik fegyvert lövésre kényszeríti.

Felülírom a Event\_Do\_It\_Again eseményt létrehozok egy időzítő a Set\_Timer\_by\_Event( Calculate\_Cd(),Event\_Fire,False) függvénnnyel. Listázom a játékos fegyvereit a Get\_Components\_By\_Class(Get\_Player\_Character(0),Master\_Weapon) a függvény visszatérési értékét eltárolom a, majd ebből a listából kivonom a Get\_Component\_by\_Class(Get\_Player\_Character(0),Demon\_Pact\_item) visszatérésiértékét, hogy a saját Fire eseményét ne hívja meg. Megvizsgálom a Weapon\_List tömb hosszát, ha nagyobb, mint nulla akkor a Weapon\_List egyik véletlenszerűen kiválasztott elemét és meghívom a Fire eseményét, majd meghívom a Do\_it\_Again eseményt.

## **9.6.Boom\_fire\_item**

A boom\_fire\_item nevű fegyver a képernyőn lévő ellenfelek egyikét fogja „felrobantani”.

### **9.6.1. Override\_Transform()**

Meghatározom, hogy melyik ellenfelek vanna a képernyőn, ehhez egy Multi\_Box\_Trace\_For\_Object() függvényt használok, aminek a következők a bemenetiértékei:

Start: Get\_Player\_Actor\_Location(Player\_Ref)

End: Get\_Player\_Actor\_Location(Player\_Ref) visszatérésiértékéhez hozzáadok egy vektort(0,0,1) értékkel

Hald\_Size: a képernyő méretéből tudom kiszámolni: elosztom négygyel az X és Y értéket, majd egy vektort csinálok a kapott érékekből, ami X/4,Y/4,0 értékű lesz

Orientation: 0,0,90

Object\_Types: létrehoztam egy tömböt, aminek egy eleme van: „ai” (a Mutli Box Trace csak tömböt fogad el bementként Object\_Type-ra)

Trace\_Complex: True

A visszakapott halmazból kiválasztottam egyet, majd lekérdeztem a helyzetét és létrehoztam abból egy transzformációt a Make\_Transform(Location, 90,0,270,1,1,1) függvény segítségével.

## **9.7.Beam\_item**

A beam\_item fegyver létrehoz egy lézercsík szerű sugarat, ami követi a játékos karaktert, de csak az ellenfeleket sebzi meg. Felülírtam az Event\_do\_it\_again() eseményt.

### **9.7.1. Event\_Do\_it\_again()**

Létrehozom a „lövedéket” a Spawn\_Actor( Weapon\_Data.Spawnable, Get\_Actor\_Location(Player\_Ref), 90, 0, 270, 1 , 1 ,1 , Weapon\_Data.Anim\_Data, Current\_Stats, Player\_Ref) függvény segítségével.

## 10. Lövedékek

A fegyvereknek különféle lövedéke van, ezért létrehoztam egy master\_projectal osztályt, amiből származtatomban majd a lövedékeket. A master\_projectal az Actor osztályból származtattam.

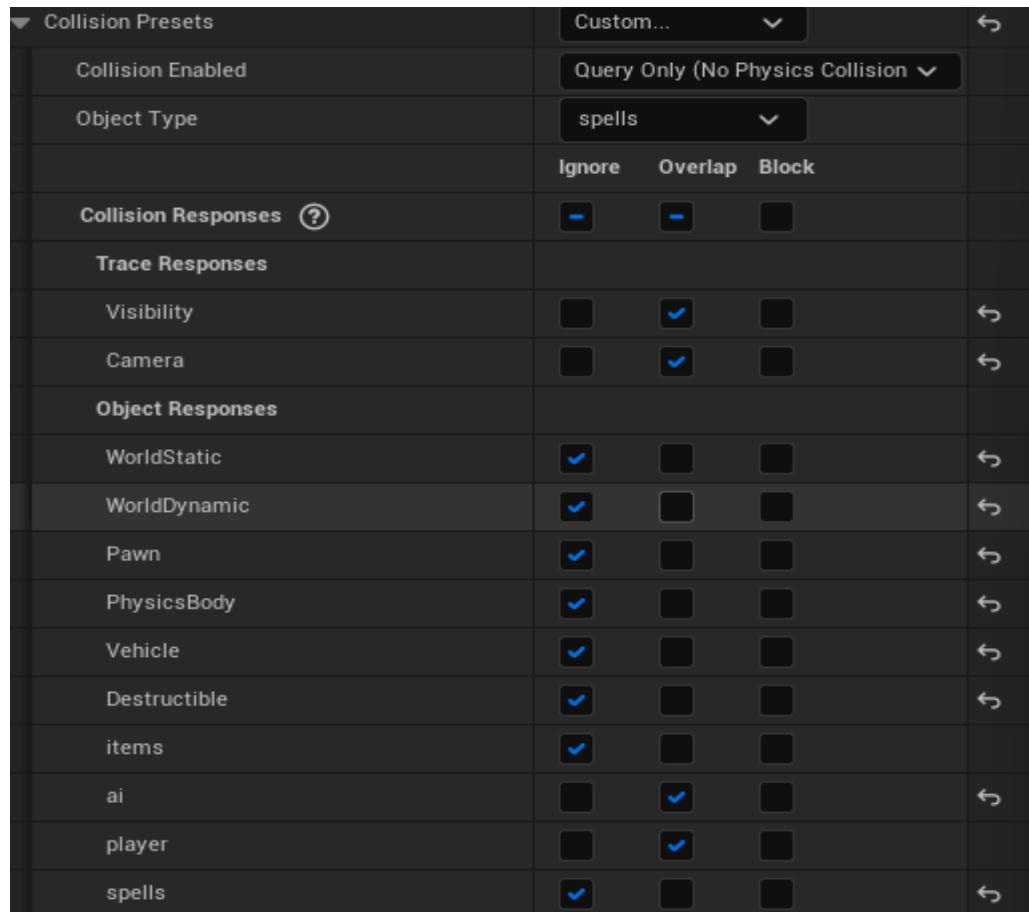
### 10.1. Master\_projectal

Ebben az osztályban létrehoztam a sebzéskezeléshez szükséges eseményeket és az ütközéskezeléshez szükséges eseményeket. Hozzáadtam egy ProjectMovement komponenst az osztályhoz, ennek segítségével tud majd a lövedék mozogni. Hozzáadtam egy Render\_Component nevű Paper\_Flipbook komponenst ezt fogja majd az animációkat lejátszani. Ütközés érzékeléséről a Sphere komponens lesz a felelős.

Sphere komponens ütközési beállításai:

Can\_Character\_Step\_Up\_On: No

Collision\_Presets: Custom



6. ábralövedék ütközési beállításai

### **10.1.1. Konstruktor**

Az Anim\_Data[0] eleméből, beállítom az adatokat Set\_Flipbook(Render\_Component, Anim\_Data[0].Flipbook) függvény segítségével beállítottam az animációt a lövedéknek. A Set\_Looping(Rander\_Component, Anim\_Data[0].Loop) függvénnnyel beállítottam, hogy az animációnak ismétlődni kell-e. A Set\_Actor\_Scale\_3D(self, Anim\_Data[0].Scale\*(1 + Weapon\_Data.aoe)) függvénnnyel beállítottam a lövedék új méretét. Beállítottam a lövedék kezdő sebességét és maximum sebességét a Weapon\_Stats.Speed értékére. A Play(Render\_component) függvénnnyel elindítottam az animáció lejátszását, majd eltároltam a Penetration változóban, hogy a lövedék hány ellenfelen tud keresztül menni.

### **10.1.2. Event\_Begin\_Play**

Létrehozok egy időzítőt Set\_Timer\_by\_Event(destory, Time, False) függvénnnyel, a Time értékét a következő módon számolom ki: a játékos Base.Stats.Duration értékét megszorozom a (Weapon\_Stats.Duration + eggyel). Az időzítő feladta az, hogy meghívja a destory eseményt.

### **10.1.3. Destroy**

Ez az esemény felelős a lövedék megsemmisítéséről.

Meghívom a Set\_Actor\_Hidden\_In\_Game(self, True) függvényt és elrejtem a játékos elől a lövedéket majd 0.001 másodperccel később elpusztítom a Destory(self) függvénnnyel, ha egy lövedék el van rejte akkor nem tud interakcióba lépni más objektumokkal, ha a Desotry\_Actor(self) függvényt akkor hívom meg amikor még interakcióban van a lövedék egy másik objektummal akkor hibaüzeneteket dobna a játék (nem akadna el), ezért várok 0.001 másodpercet, hogy végrehajtsa az összes függőben lévő eseményt majd ezután törlöm.

### **10.1.4. Event\_Actor\_Begin\_Overlap**

Megvizsgálom, hogy az objektum amivel ütközött a lövedék, annak van-e „ai” címkeje, ha igen akkor dekrementálom a Penetration változót és meghívom az Apply\_Damage(Other\_Actor, Base\_Damage, self) függvényt, ahol a Base\_Damage a következő módon számítom ki: a játékos (Base\_Stats.Damage + egyet) megszorozom a Weapon\_Stat.Damage értékével. Megvizsgálom a Penetration értékét, ha kisebb vagy nulla akkor meghívom a Destory eseményt.

## **10.2. Energy\_ball\_projectal osztály**

Az energy\_ball\_projectal osztályban nem hajtottam végre módosítást a master\_projectal osztályból való származtatás után.

## **10.3. Ice\_ball\_projectal osztály**

Az ice\_ball\_projectal osztályban nem hajtottam végre módosítást a master\_projectal osztályból való származtatás után.

## **10.4. Torch\_projectal osztály**

Az torch\_projectal osztályban nem hajtottam végre módosítást a master\_projectal osztályból való származtatás után.

## **10.5. Beam osztály**

Az Beam osztályt a master\_projectal osztályból származtattam kiegészítettem a konstruktor és az Event\_Begin\_Play eseményt.

### **10.5.1. Konstruktor**

Set\_Relative\_Rotation(Render\_Component, 0, 90, 0) függvénytel felforgattam az Y tengelyen a lövedéket, majd bekapcsoltam a követés funkcióját, célpontnak megadtam a Player\_Ref változóban tárolt hivatkozást, sebeségnek megadtam a játékos (Base\_Stats.Speed + egy) szorozva a Weapon\_Stat.Speed értékével.

### **10.5.2. Event\_Begin\_Play**

Létrehoztam egy időzítőt a Set\_Timer\_by\_Event(do\_dmg, 0.25, True) függvény segítségével, ennek az időzítőnek lesz a feladat, hogy minden 0.25 másodpercben megsebezze az összes ellenfelet aki éppen érintkezik a lövedékkel.

### **10.5.3. Do\_dmg**

Egy ciklussal végig megyek az összes Objektumon, ami éppen érintkezik a lövedékkel, majd megvizsgálom melyiknek van „ai” címkeje és azoknál meghívom a Apply\_Damage() függvényt.

### **10.5.4. Event\_Destoryed**

Létrehozok egy új példányt a lövedékből a Beam\_item Do\_it\_Again eseményének meghívásával.

## **11. Szintek**

### **11.1. Menu\_map szint**

Létrehoztam egy szintet csak a menünek. A szint Blueprint-ben létrehoztam az Event\_Begin\_Play eseményt és Módosítottam a GameMode\_Override-ot menu\_gamemode-ra.

#### **11.1.1. Event\_Begin\_Play**

Game\_instance típuskényszerítést hajtok végre a Get\_Game\_Instance() visszatérési értékén, majd meghívom a Play\_Music(As\_Game\_Instance), így a zene csak egyszer fog elindulni hiába kerülök vissza a menu\_map szintre.

## **11.2. Underworld**

Létrehoztam egy szintet a játékban, ahol a játékos játszani tud majd. Módosítottam a GameMode\_Override-ot inGame\_gm-re, létrehoztam egy platformot (Plane), amin a játékos és az ellenfelek lesznek. Létrehoztam egy Post\_Process\_Volume Blueprint-et.

#### **11.2.1. Plane**

A következő beállításokat módosítottam a Plane-en:

Location	0, 0, -37.0
Scale	100.0, 100.0, 1.0
Static_Mesh	Plane
Materials	floor_1_mat

*20. táblázat Plane objektum adatai*

## 12. Boss

A „Boss” egy olyan egy speciális ellenfél, aminek több élete van erősebbet üt, és vannak képességei.

Létrehoztam egy demon\_lord nevű osztályt. amit a Pawn osztáyból származtattam. Hozzáadtam egy Box\_Collision komponenst ütközés, egy PaperFlipbook-ot, egy FloatingPawnMovement komponenst és egy BehaviorTree komponenst.

Ütközési beállítások:

Box		PaperFlibook		
Generate Overlap Events	<input checked="" type="checkbox"/>	Generate Overlap Events	<input checked="" type="checkbox"/>	
Can Character Step Up On	No	Can Character Step Up On	No	
Collision Presets	Custom...	Collision Presets	Custom...	
Collision Enabled	Query Only (No Physics Collision)	Collision Enabled	Collision Enabled (Query and Physic	
Object Type	ai	Object Type	ai	
	Ignore   Overlap   Block		Ignore   Overlap   Block	
Collision Responses ⓘ	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Collision Responses ⓘ	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	
Trace Responses		Trace Responses		
Visibility	<input checked="" type="checkbox"/>	Visibility	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Camera	<input checked="" type="checkbox"/>	Camera	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Object Responses		Object Responses		
WorldStatic	<input type="checkbox"/>	WorldStatic	<input type="checkbox"/>	<input checked="" type="checkbox"/>
WorldDynamic	<input type="checkbox"/>	WorldDynamic	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Pawn	<input type="checkbox"/>	Pawn	<input type="checkbox"/>	<input checked="" type="checkbox"/>
PhysicsBody	<input checked="" type="checkbox"/>	PhysicsBody	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Vehicle	<input checked="" type="checkbox"/>	Vehicle	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Destructible	<input checked="" type="checkbox"/>	Destructible	<input type="checkbox"/>	<input checked="" type="checkbox"/>
items	<input checked="" type="checkbox"/>	items	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ai	<input type="checkbox"/>	ai	<input type="checkbox"/>	<input checked="" type="checkbox"/>
player	<input type="checkbox"/>	player	<input type="checkbox"/>	<input checked="" type="checkbox"/>
spells	<input type="checkbox"/>	spells	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Simulation Generates Hit Events	<input type="checkbox"/>	Simulation Generates Hit Events	<input checked="" type="checkbox"/>	

21. táblázat Boss ütközési adatai

### 12.1. Konstruktor

A Current\_hp változót beállítom tízezerre.

## **12.2. Event\_tick**

Megvizsgálom, hogy a játékos jobb vagy bal oldalán van a Boss-nak ezt a Find\_Look\_at\_Rotation(Get\_Actor\_Location(self), Get\_Actor\_Location( Get\_Player\_Character( 0))) függvény segítségével teszem meg, majd beállítom a relatív forgásszöget a Set\_Relavtive\_Rotation(PaperFlipbook) függvénnel.

Megvizsgálom, hogy éppen a lény lejátszik-e animációt ezt a Is\_Playing(Paper\_Flipbook) függvénnel teszem meg, ha nem játszik le éppen animációt akkor beállítom neki a demon\_walk animációt a Set\_Flipbook(Paper\_Flipbook, Demon\_walk) függvény segítségével.

Megvizsgálom, hogy éppen milyen animációt játszik a lény, ha az „demon\_cleave” akkor megvizsgálom, hogy hol tart az animációban, ha még csak a 0,1 másodperc telt el az animációból akkor a Do\_once kaput visszaállítóm alapállapotba, ha 0.7 másodperc telt már el akkor tovább lépek a Do\_Once kapuba és meghívom a Box\_Overlap\_Actor( Box\_Pos, 100,100,1) függvényt a Box\_Pos értékét úgy számolom ki hogy hozzáadok 100 Get\_Acotr\_Location(self) X kértéhez vagy elveszek 100-at attól függően hogy a játékos a lény jobb vagy bal oldalán van. Egy ciklussal végig megyek a fedésbe került objektumokon és meghívom az Apply\_Damage(Array\_element, 30.0, self) függvényt, ami megsebzi a játékosat.

## **12.3. Event\_Any\_Damage**

Current\_hp-ból kivonom a kapott sebzést és így kapom meg a Current\_hp. Megvizsgálom, hogy a Current\_hp kisebb, mint nulla akkor lejátszom a halál animációját a lénynek és hozzákapcsolom az animáció végéhez az end eseményt.

## **12.4. End**

A game\_instace típuskényszerítettem a Get\_Game\_Instance() függvény visszatérési értékét, és meghívom a Game\_Over eseményt.

## **12.5. Basic\_attack**

Lejátszom a demon\_cleave animációt a Set\_Flipbook(Paper\_Flipbook demon\_cleave), Set\_Looping(Paper\_Flipbook, False), Play(Paper\_Flipbook) függvények segítségével.

## **12.6. Spell\_1**

Létrehozok lényeket a Spawn\_Actor() függvényvel véletlenszerűen a képernyőn.

## **12.7. Boss\_bh behavior tree**

Létrehoztam egy behavior tree a Boss-hoz boss\_bh néven, és létrehoztam bb néven egy Blackboard-ot, amiben három kulcsot tárolok: SelfActor, player\_location, Spell\_1\_next\_time\_useable.

Létrehoztam egy Sequence csomópontot, amiből két csomópontba léphet a lény: attack, spells.

### **12.7.1. Attack Selector**

Első opció: basic\_attack feladat, in\_range\_for\_ba díszítővel van ellátva

Második opció: move\_to\_player\_offset feladat

Az attack selector sikeres lesz, ha az egyik feladat sikeresen végrehajtódik.

### **12.7.2. Spells Selector**

Első opció: spell\_1 feladat, cd díszítővel van ellátva

Második opció: move\_to\_player\_offset feladat

Az spells selector sikeres lesz, ha egyik feladata sikeresen végrehajtódik.

### **12.7.3. Basic\_attack feladat**

Létrehoztam az Event\_Receive\_Execute\_AI eseményt típuskényszerítettem Controlled\_Pawn változó értékét, majd meghívom a Stop\_Movement\_immediately(Floating\_Pawn\_Movement) függvényt, meghívom a Basic\_Attack() függvényt. Hozzáötöm a támadás animációhoz a done eseményt.

Done esemény meghívja a Finish\_Execute(self,True) függvényt és ezzel sikeresen kilépek a feladatból.

### **12.7.4. In\_range\_for\_ba dekorátor**

Box\_Trace\_For\_Object(Get\_Actor\_Location(Controlled\_Pawn)+(5,0,0,), Get\_Actor\_Location(Controlled\_Pawn)+(5,0,1,), player, True, Controlled\_Pawn, True) függvényel létrehozok egy lekérdezést a lény körül, ami megadja hogy a lény meg tudja-e ütni a játékos karakterét.

#### **12.7.5. Move\_to\_player\_offset feladat**

Megvizsgálom, hogy a lény éppen milyen animációt játszik le, ha nem demon\_cleave animációt játszik le akkor meghívom a Move\_to\_Location( Get\_Actor\_location(Get\_Player\_Character(0))+ offset, 1.0, True, True) függvényt és lény a játékos mellé fog mozgni. Az eltolást úgy számolom ki, hogy megvizsgálom melyik oldalon van a játékos a lénynek az X tengelyen vagy mínusz negyven vagy plusz negyven az Y tengelyen plusz száz vagy mínusz száz lesz az eltolás mértéke. Mozgás után meghívom a Finish\_Execute(self, True) függvényt és sikeresen kilépek a feladatból.

#### **12.7.6. Spell\_1**

Meghívom a demon\_lord Spell\_1() függvényét majd sikeresen kilépek a Finish\_Execute(self, True) függvénnyel.

#### **12.7.7. Cd dekorátor**

Megvizsgálom, hogy a Blackboard kulcs értéke nagyobb-e, mint a Get\_Game\_Time\_in\_Seconds() visszatérésiértéke, ha igen akkor hozzáadok tíz másodperce és eltárolom a Blackboard-ban és igazzal térek vissza, ha nem akkor hamissal térek vissza a függvényből.

## **13. Game\_instance**

### **13.1. Event\_Init**

Megvizsgálom, hogy létezik a játék mentés a Does\_Save\_Game\_Exist(Slot\_Name, 0) függvénytel, ha létezik akkor betöltöm és elmentem a Save\_Object\_Ref változóba, ha nem létezik akkor létrehozom a Create\_Save\_Game\_Object(Save\_Object\_Class) és elmentem a Save\_Game\_to\_Slot(Save\_Game\_Object, Slot\_Name) függvénytel, majd betöltöm és elmentem a Save\_Object\_Ref változóba.

### **13.2. Music\_volume**

Betöltöm a hanggal kapcsolatos adatokat és beállítom őket a Set\_Sound\_Mix\_Class\_Override(sound\_mix, master, volume, 1, 0, True), Set\_Sound\_Mix\_Class\_Override(sound\_mix, music, volume, 1, 0, True), Set\_Sound\_Mix\_Class\_Override(sound\_mix, SFX, volume, 1, 0, True). Majd meghívom a Push\_Sound\_Mix\_Modifier(sound\_mix) függvényt.

### **13.3. Event\_Shutdown**

Elmentem a játékot a Save\_Game\_toSlot(Save\_Object\_Ref, Slot\_Name) függvénytel.

### **13.4. Game\_over esemény**

Megállítom a játékot a Set\_Game\_Paused(True) függvénytel, majd létrehozom a Game\_Over widget-et és hozzáadom a nézőponthoz.

### **13.5. Play\_music**

Megvizsgálom, hogy a Music\_Ref érvényes-e, ha nem akkor létrehozok egy hangot a Create\_Sound\_2D(random\_music\_ref, 1, 1, 0, True, True) függvény segítségével és eltárolom a Music\_Ref változóba majd az elejétől kezdve lejátszom, az On\_Audio\_Finished eseményhez kötöm a Finish eseményt és frissítem a hang beállításokat a Music\_Volume eseménnyel.

#### **13.5.1. Finsih**

Null éréket rakk a Music\_Ref változóba és meghívom a Play\_Music eseményt.

## 14. Játék módok

### 14.1. Menu\_gamemode

Létrehozok egy Game\_Mode\_Base osztályból származtatott Blueprint osztályt, aminek Menu\_gamemode lesz a neve módosítom a következő beállításokat:

Player_Controller_Class	menu_playercontroller
HUD_Class	menu_hud
Default_Pawn_Class	None

22. táblázat menu\_gamemode módosított elemei

### 14.2. InGame\_gm

Létrehozok egy Game\_Mode\_Base osztályból származtatott Blueprint osztályt, aminek inGame\_gm lesz a neve módosítom a következő beállításokat:

Player_Controller_Class	Player_controller_inGame
HUD_Class	ingame_hud
Default_Pawn_Class	None

23. táblázat ingame\_gm módosított elemei

#### 14.2.1. Event\_On\_Post\_login

Eltárolom a játékos kontrollert egy változóba, majd létrehozom a játékos karakterét, majd ezt a játékos megszállja a Possess(Player\_Controller\_Ref, in\_Pawn) függvényel.

Létrehozok két időzítőt az egyiket 1200 másodpercre állítom és az húsz perc múlva fogja a spawn\_boss eseményt meghívni, ami létrehozza a Boss-t. A másik 0.1 másodpercenként meghívja a tick eseményt.

#### 14.2.2. Tick

Betöltöm ais\_to\_spawn adattáblából azt a sort, ami megegyezik a játékidő osztva hatvannal felel kerekítve. Egy For\_Each ciklussal végig megyek az adatsoron majd megnézem, hogy hány ellenfélnek van a sorban szereplő címkeje, ha kevesebbnek van, mint amennyit meghatároztam az adattáblában akkor létrehozok annyit, hogy kiegészítsem a megadott mennyiségre.

## **15. Felhasználói felület**

Létrehoztam két HUB osztályt az egyiket a Menu\_map nevű szinthez lesz ez a Menu\_hud, a másik az ingame\_hud névre hallgat és az underworld szinten lesz betölve.

### **15.1. Menu\_hud**

#### **15.1.1. Event\_Begin\_play**

Létrehozok egy Widget elemet a Create\_Widget(Menu, Player\_Owner) függvény segítségével, majd eltárolom a Main\_Menu\_Ref változóban és hozzáadom a nézőponthoz az ADD\_to\_Viewport(Main\_Menu\_Ref) függvénnnyel.

### **15.2. ingame\_hud**

#### **15.2.1. Event\_Begin\_Play**

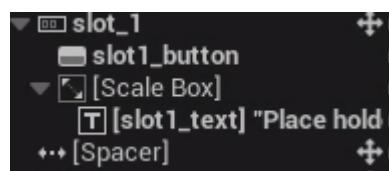
Létrehozok egy Widget elemet a Create\_Widget(Souls\_Xp, Player\_Owner) függvény segítségével, majd hozzáadom a nézőponthoz a Add\_to\_Viewport(Return\_Value) függvénnnyel.

#### **15.2.2. Level\_up\_hud**

Ez az esemény akkor fog meghívásra kerülni, amikor a játékos szintet lép. Megállítom a játékot a Set\_Game\_Paused(True) függvénnnyel, majd létrehozom a LevelUp felhasználói felületet és hozzáadom a nézőponthoz.

### **15.3. level\_up**

Ez a felhasználói felület lesz felelős a játékos által kiválasztható fejlesztések megjelenítéséről. A következő elemekből építettem fel. ezt a Widget-et:



*7. ábra UI slot szerkezete*

Mivel maximum négy fejlesztést jeleníték meg a játékosnak egyszerre négyszer ismétlem az előbbi UI felépítést (slot1, slot2, slot3, slot4)

### **15.3.1. Event\_Construct**

Megvizsgálom, hogy hány fegyvere van a játékosnak és ezek közül hány van maximum szinten (Weapon\_that\_are\_max\_level) és hány nem (Weapon\_that\_are\_not\_max\_level) megvizsgálom, hogy Weapon\_that\_are\_max\_level tömbnek a hosszúsága nagyobb vagy egyenlő héttel, ha igen akkor a játékosnak már nem kell több fegyver fejlesztést mutatom, mert elérte a maximumot. Lekérdezem az összes fegyvert és kivonom belőle Weapon\_that\_are\_max\_level majd az eredményt eltárolom a Possible\_weapons-ben.

Megvizsgálom, hogy hány fejlesztése van a játékosnak és ezek közül hány van maximum szinten (Stats\_that\_are\_max\_level) és hány nem (stats\_that\_are\_not\_max\_level) megvizsgálom, hogy stats\_that\_are\_max\_level tömbnek a hosszúsága nagyobb vagy egyenlő héttel, ha igen akkor a játékosnak már nem kell több fejlesztést mutatom, mert elérte a maximumot. Lekérdezem az összes fejlesztést és kivonom belőle stat\_that\_are\_max\_level majd az eredményt eltárolom a stats\_weapons-ban.

Összecsatolom a lehetséges fejlesztéseket a lehetséges fegyverekkel és létrehozom a Possible\_Stats\_and\_weapons tömböt, és meghívom a Set\_Options függvényt, ami kirajzolja a lehetséges fejlesztési opciókat.

Létrehozok három tömböt: Slot\_Array, Button\_Array, Text\_array

létrehozok egy for ciklust nullától háromig majd, elmentem az indexet az index változóba. Meghívom a Weapon(self, index) függvényt amiben véletlenszerűen választok a kombinált fegyver és fejlesztés tömbből ha megtalálom a fegyver adattáblában akkor betöltem az adatait. Meghívom a bonus\_Stats(self, Item\_Name\_not\_Found, index) függvényt és betöltem a fejlesztés adatait

Megvizsgálom, hogy a kombinált fegyver és fejlesztés tömb hossza nagyobb vagy egyenlő (index+1) kifejezéssel, ha igen akkor a select értéke „Visible” ha hamis akkor „Hidden”, majd meghívom a Set\_Visibility(Get(slot\_array(index)),select) függvényt és beállítom, hogy látható legyen a slot\_array[index] eleme azaz van-e elég fejlesztés, hogy index számú slot-ot kelljen megjeleníteni.

## **16. Material**

### **16.1. player\_material**

Létrehoztam egy material-t, módosítottam a Blend\_mode-ot „Translucent”-re, hozzáadtam egy „TextureSampleParameterSubUV” elneveztem Flipbook-nak, majd beállítottam a Flipbook értékét „B\_witch\_run(texture)”

## **17. Licenszek**

Az Unreal Engine-be építette funkciókkal összeillesztettem a letöltött képeket Sprite-okba.

### **17.1. Boszorkány animációk:**

A <https://9e0.itch.io/witches-pack> oldalról töltöttem le a képeket. A 9E0 nevű felhasználó által készítette.

### **17.2. Skeleton , Mushroom, Goblin, Eye szörnyek animációi**

A <https://luizmelo.itch.io/monsters-creatures-fantasy> oldalról töltöttem le a png-fájlokat, amiket LuizMelo nevű felhasználó készített. A fájlok CC0-ás típusú licenszel tette fel az internetre.

### **17.3. King animációk**

A <https://oco.itch.io/medieval-fantasy-character-pack-3> oldalról töltöttem le a képeket, amiket az OcO nevű felhasználó készített

## 18. Összegzés

Létrehoztam egy olyan játékot, amit mint demó verzióként lehetne már használni. Implementáltam a játékmenethez szükséges alapvető funkciókat, amikre lehet a későbbiekben építeni. A karakterekhez szükséges animációkat én állítottam össze Sprite-okba, de nem én készítettem a képfájlokat.

Létrehoztam egy játékos karaktert, amit a játékos W, A, S, D-vel tud irányítani. A jövőben, ha kiszeretném bővíteni játszható karakterek számát akkor, nagyon kicsi módosítással létre tudok hozni egy új karaktert, mert úgy implementáltam a master\_player osztályt, hogy adattáblákból töltse be a játékos karaktert, így majdnem elég csak egy új sort létrehoznom az adattáblában.

Létrehoztam egy adattáblából bővíthető master\_ai osztályt, ennek az osztálynak az a feladat, hogy létrehozzon különféle ellenfeleket, és hogy azok az ellenfelek a játékos felé mozogjanak. Amikor a játékost eléri az egyik ellenfél akkor az megsebzi a játékost és addig fog sebzést okozni amíg a játékos ütközik vele.

Létrehoztam egy fegyverkezelő osztályt ebből az osztályból származtatom majd a játékos fegyvereit ezek a fegyverek automatikusan céloznak és lönek. Egy fegyverből a játékosnál egyszerre csak egy lehet. A fegyverek adatait egy adattáblából töltöm be. A fegyverek lövedékét egy master\_projectal nevű osztályból származtatatom a fegyver átadja a lövedéknek a megfelelős adatokat (animáció, sebzés stb.).

Létrehoztam egy felvehető tárgyat (pl.: tapasztalat) ezeket egy osztályból származtatóm, ha a jövőben szeretném létrehozni tejesen különböző eseményeket megvalósító tárgyat azt is lehetővé teszi a mostani osztály implementáció.

Létrehoztam egy Boss ellenfelet, aminek olyan Behavior tree rendszert készítettem, hogy akárhány új képességgel lehetne bővíteni. Létrehoztam új feladatokat és dekorátorokat.

Létrehoztam a játékmenetéhez szükséges adatokat, szabályzó rendszereket és mentési objektumot.

Szerintem elértem a célkitűzéseimet, mint minden szoftverfejlesztési projektnél, így a játékfejlesztésnél is lehetne valamit jobban vagy optimálisabban csinálni a fejlesztés sosem áll meg amíg van fejlesztő, így persze ezt a projektet is lehetett volna máshogy megvalósítani, de szerintem ezek a funkciókat, amiket én létrehoztam az alapkövei egy

játéknak amire lehet építeni pl.: egy játékban található bolt implementálása, ahol a játékos elköltheti a lelkeket, amiket összegyűjtőt ez a funkció nincs implementálva de az alapjai igen: játékmentési objektum és a souls tárgy vagy statisztikák pl.: hogy melyik fegyver mennyit sebzett stb.

## Irodalomjegyzék

- [1] "Steam Keys (Steamworks Documentation)," Valve, [Online]. Available: <https://partner.steamgames.com/doc/features/keys>. [Accessed 12 március 2022.].
- [2] "Stats and Achievements (Steamworks Documentation)," Valve, [Online]. Available: <https://partner.steamgames.com/doc/features/achievements>. [Accessed 8. március 2022.].
- [3] "Behavior Tree in Unreal Engine - Quick Start Guide | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/behavior-tree-in-unreal-engine---quick-start-guide/>. [Accessed 20. február 2022.].
- [4] "Behavior Tree in Unreal Engine - Overview | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/behavior-tree-in-unreal-engine---overview/>. [Accessed 20. február 2022.].
- [5] "Behavior Tree in Unreal Engine - User Guide | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/behavior-tree-in-unreal-engine---user-guide/>. [Accessed 20. február 2022.].
- [6] "Behavior Tree Node Reference in Unreal Engine | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/behavior-tree-node-reference-in-unreal-engine/>. [Accessed 21. február 2022.].
- [7] "Import Sprites in Unreal Engine | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/import-sprites-in-unreal-engine/>. [Accessed 11. február 2022.].
- [8] "Paper 2D Flipbooks in Unreal Engine | Unreal Engine Documentation," Epic Games, [Online]. Available:

- <https://docs.unrealengine.com/5.0/en-US/paper-2d-flipbooks-in-unreal-engine/>. [Accessed 11. február 2022.].
- [9] "Paper 2D Sprites in Unreal Engine | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/paper-2d-sprites-in-unreal-engine/>. [Accessed 11. február 2022.].
- [10] "Paper 2D Tile Sets and Tile Maps in Unreal Engine | Unreal Engine Documentation," Epic Games, [Online]. Available: <https://docs.unrealengine.com/5.0/en-US/paper-2d-tile-sets-and-tile-maps-in-unreal-engine/>. [Accessed 11. február 2022.].
- [11] "Steam DRM (Steamworks Documentation)," Valve, [Online]. Available: <https://partner.steamgames.com/doc/features/drm>. [Accessed 10. március 2022.].

## **Ábrajegyzék**

1. ábra játékoskarakter Capsule_Component ütközési beállításai.....	12
2. ábra játékoskarakter Capsule ütközési beállításai .....	13
3. ábra játékos körvonalak .....	17
4. ábra Blueprint változat .....	21
5. ábra master_ai ütközési beállításai .....	25
6. ábralövedék ütközési beállításai .....	51
7. ábra UI slot szerkezete .....	61

## Táblázatjegyzék

1. táblázat az Eye szörnyeteg adatai .....	27
2. táblázat az Goblin szörnyeteg adatai .....	28
3. táblázat az Skeleton szörnyeteg adatai .....	29
4. táblázat az Mushroom szörnyeteg adatai.....	30
5. táblázat az Necromancer szörnyeteg adatai.....	31
6. táblázat az King szörnyeteg adatai .....	32
7. táblázat az Hounds szörnyeteg adatai.....	33
8. táblázat energy_ball adatai 1 .....	36
9. táblázatenergy_ball adatai 2 .....	36
10. táblázat boom_fire_item adatai 1 .....	37
11. táblázatenergy_ball adatai 2 .....	37
12. táblázat ice_ball_item adatai 1 .....	38
13. táblázat ice_ball_item adatai 2 .....	38
14. táblázat double_shot adatai 1 .....	39
15. táblázat double_shot adatai 2 .....	39
16. táblázat torch adatai 1 .....	40
17. táblázat torch adatai 2 .....	40
18. táblázat beam adati 1 .....	41
19. táblázat beam adatai 2 .....	41
20. táblázat Plane objektum adatai .....	54
21. táblázat Boss ütközési adatai .....	55
22. táblázat menu_gamemode módosított elemei .....	60
23. táblázat ingame_gm módosított elemei .....	60

