5/16/2022 8:21:49 PM

Compare Results

Old File:		New File:
ADP8B4B.tmp	versus	**szd.pdf**
11 pages (475 KB)		**76 pages (922 KB)**
5/16/2022 8:19:29 PM		5/16/2022 8:06:34 PM

Total Changes

467

Content

34 Replacements
245 Insertions
14 Deletions

Styling and Annotations

7 Styling
167 Annotations

Go to First Change (page 1)

GÁBOR DÉNES FŐISKOLA

MÉRNÖKINFORMATIKUS

SZAKDOLGOZAT

JÁTÉK FEJLESZTÉS UNREAL ENGINE 5-BEN

Kiss Bence Baldvin
ZZYZRX
197/2020

SZAKDOLGOZATI / ZÁRÓDOLGOZATI TÉMAVÁZLATI LAP

A lapot nem képkét kell beilleszteni, a témavázlat 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 nyomtatott és elektronikus szakirodalom felhasználása a szerzői jogok és a szakdolgozatírás 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 célom 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 főnőként funkcionál és legyőzésével a játékos nyerni tud, ennek a lénynek speciális támadásai lesznek.

A játék elérthető a következő hivatkozáson keresztül:

Tartalomjegyzék

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 témával 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 Achivements
- 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 filmiparban 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 lecserélték 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 interfészeket 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 lépes 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 percing 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 definíciónak

```
[/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 funkcionalitásban 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 változtattam 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 beálítások a következők:

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 felvételért.

Capsule_Half_Height: 15

Capsule_Radius: 7

Ekkor lesz az a terület, ahol ütközét 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

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 MoveFoward 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 MoveFoward 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 bontóm: 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) függvénnyel.

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 <=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 konstruktor 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 jelennek 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, Inital_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 módosítóm „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_contoller(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) függvénnyel.

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 játékablak 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: életerejét, 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 au 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.

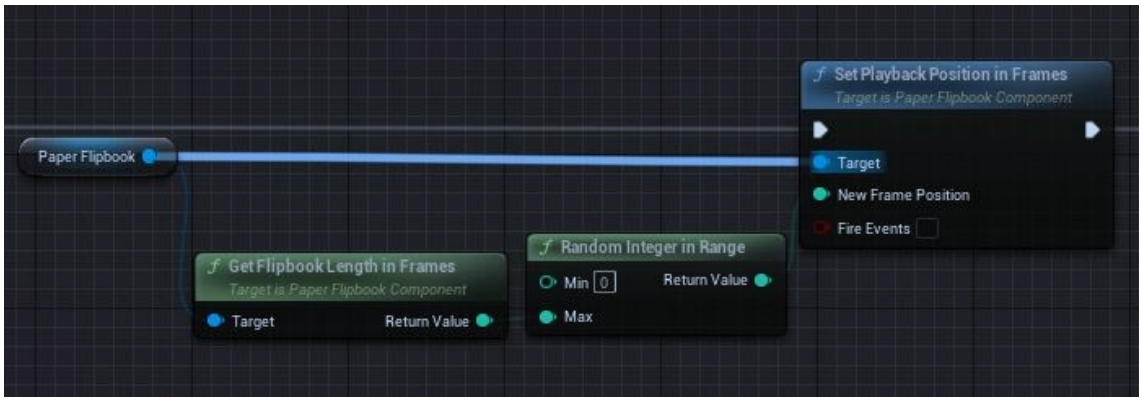
6.1.Konstruktor

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 karakternek 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) függvénnyel

6.2.Event_Begin_Play

Mivel egyszerre több mint egy ellenfelet fogok megjeleníteni, szerintem nagyon furcsa lenne, ha mind 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_Lenght_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 felelős. 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 mindig 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 játékosfelé 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” címkéje, 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 lénnyel, 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, Data.Stats.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 címkéje ha igen akkor meghívom a Clear_and_Invalidate_Timer_by_Handle(Damage_Timer_Ref) függvényt, mert csak a játékosnak van „player” címkéje ezért ha valaki már nem ütközik a lénnyel és volt „player” címkéje 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) függvénnyel (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_Collison_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_Contoller(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 tapasztalatót is.

Meghívom a Spawn_Actor_From_Class(Xp,Transform,Data.Xp_Data) függvényt a transzform bementi értke a következőképpen 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 feladat, 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 ismétlődnie.

Meghívom a Set_Flipbook(PaperFlipbook,Flipbook) függvényt hogy beállítsam az új Flipbook-ot, majd a Set_Looping(PaperFlipbook,Looping)függvénnyel beállítom az ismétlődést és végül elindítom az animációt a Play(PaperFlipbook) függvénnyel.

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

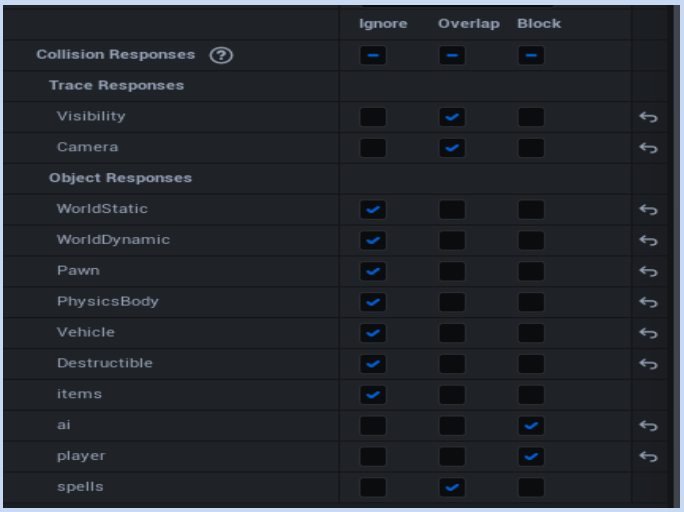
Mozgásnál frissíti a fizikai teret.

Can_Character_Step_Up_On: False

Nem szeretném, hogy karakterek rá tudjanak lépni.

Collison_Enabled: Collison_Enabled(Query and Physics)

Object_Type: ai



5. ábra master_ai ütközési beállításai

Always_Create_Physics_State: True

Ennek a karakternek muszáj, hogy mindig 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

1. 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 Skeleteon 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 mennyiségű 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 lepni 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 ismétlődnie 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_datatabletábla tartalmazza a fejlesztések adatait.

8.2.9. Weapon_datatable

A weapon_datatable tábla tartalmazza a fegyverek adatait.

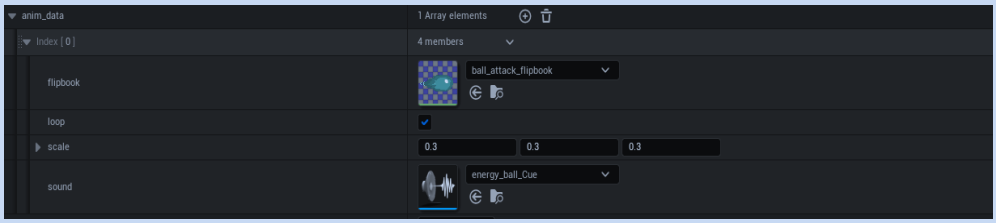
energy_ball

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. táblázatenergy_ball adatai 2

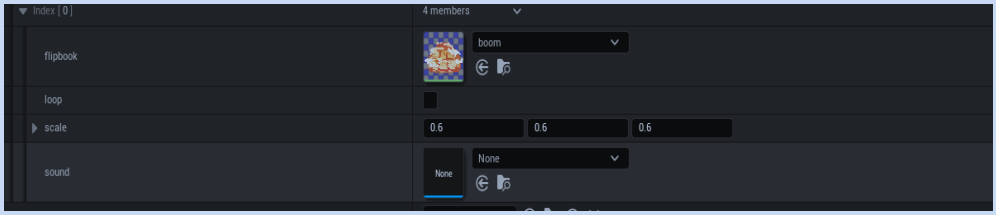
boom_fire_item

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. táblázatenergy_ball adatai 2

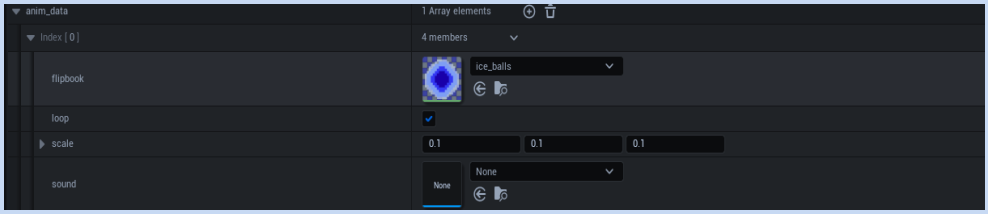
ice_ball_item

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

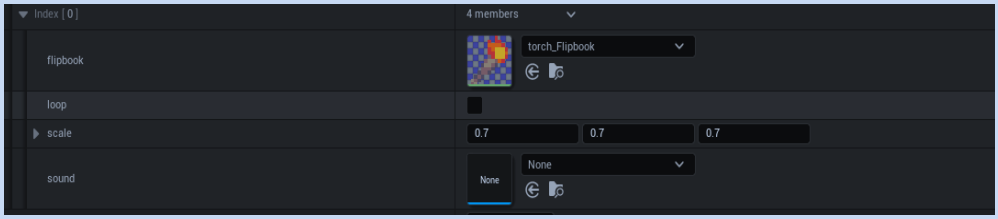
torch

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

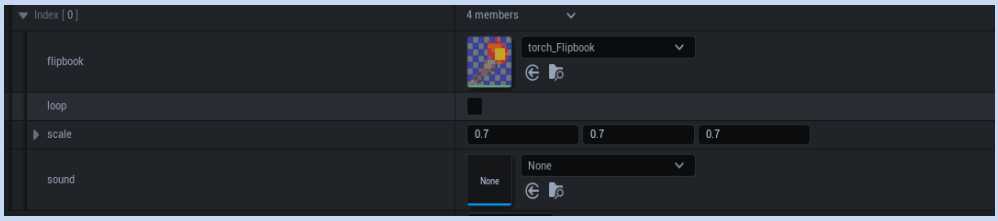
beam

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 címkéje, 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 visszatérítési 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 megjelenítésért.

do_stuff()

Master_player típuskényszerítést hajtok végre a Get_Player_Character(0) függvény visszatérítési értékén, majd Master_player.Base_Stats.souls értékét megnövelem eggyel, és megszorzom a Potency értékével majd ezt az értéket felelé kerekítem és hozzáadom a lelkek mennyiségéhez.

8.4.Fejlesztések

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

8.4.1. Event_Begin_Play()

A Get_Owner(self) függvény visszatérési 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()

Frissítem 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 szintlépéséért.

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.

donus_duration

Ez a fejlesztés lesz felelős a játékos támadás idejének növeléséért.

bonus_heath

Ez a fejlesztés lesz felelős a játékos életerejének 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_Chracter(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) függvénnyel létrehozott időzítő intervallumát nem lehet dinamikusan módosítani, ezért kell a Do_It_Again eseménynek önmagát megívnia).

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 tettem 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 összeszorzom 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(„randoom ellenfél”)) függvény segítségével kiszámolom hogy milyen irányba kell kilőni a lövedéket, majd a Make_Transform(Get_Actor_Locaition(Player_Ref), 90, 0, Float, 1, 1, 1) függvénnyel 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 mínusz 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(„randoom ellenfél”)) függvény segítségével kiszámolom hogy milyen irányba kell kilőni a lövedéket, majd a Make_Transform(Get_Actor_Locaition(Player_Ref), 90, 0, Float, 1, 1, 1) függvénnyel 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) függvénnyel 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) függvénnyel kapott visszatérésiértéknél akkor eltárolom a tömb elemet 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_Locaition(Player_Ref), 90, 0, Float, 1, 1, 1) függvénnyel kimenetiértéke lesz a Transform visszatérési érték. Ha nincsenek ellenfelek a pályán akkor a Float értéke egy véletlenszerű szám mínusz 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) függvénnyel. 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 néggyel az X és Y értéket, majd egy vektort csinálok a kapott étékekből, 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_againt() 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 származtatom majd a lövedékeket. A master_projectal az Actor osztályból származtattam.

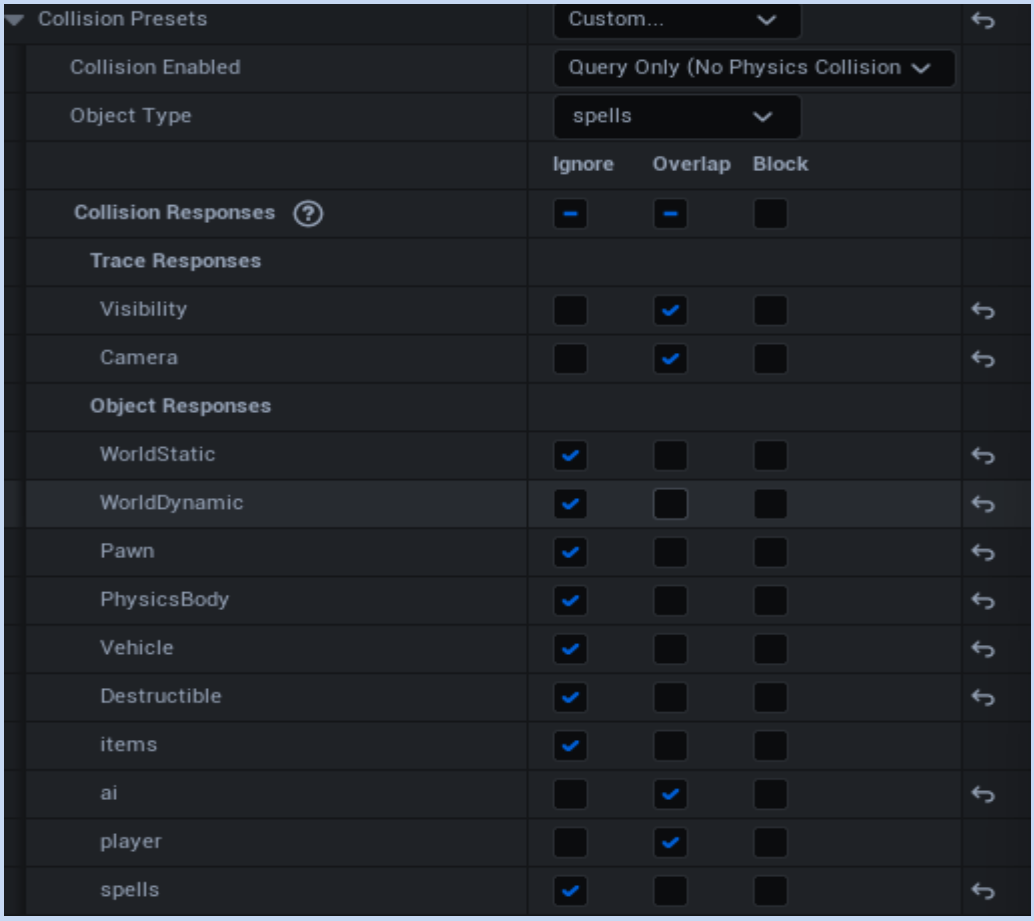
10.1. Master_projectal

Ebben az osztályban létrehoztam a sebzéskezeléshez szükséges eseményeket és az ütközéskezeléshez szükséges eseményeket. Hozzáadtam egy PorjectalMovement 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 érzékelésért a Sphere komponens lesz a felelős.

Sphere komponens ütközési beállításai:

Can_Character_Step_Up_On: No

Collison_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) függvénnyel beállítottam, hogy az animációnak ismétlődnie kell-e. A Set_Actor_Scale_3D(self, Anim_Data[0].Scale*(1 + Weapon_Data.aoe)) függvénnyel beállítottam a lövedék új méretét. Beállítottam a lövedék kezdő sebességét és maximum sebességét a Weapon_Stats.Speed értékére. A Play(Render_component) függvénnyel 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) függvénnyel, a Time értékét a következő módon számolom ki: a játékos Base.Stats.Duration értékét megszorzom a (Weapon_Stats.Duration + eggyel). Az időzítő feladta az, hogy meghívja a destory eseményt.

10.1.3. Destroy

Ez az esemény felelős a lövedék megsemmisítésért.

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) függvénnyel, ha egy lövedék el van rejtve 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 eseményét majd ezután törlöm.

10.1.4. Event_Actor_Begin_Overlap

Megvizsgálom, hogy az objektum amivel ütközött a lövedék, annak van-e „ai” címkéje, 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) megszorzom 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) függvénnyel 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” címkéje 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

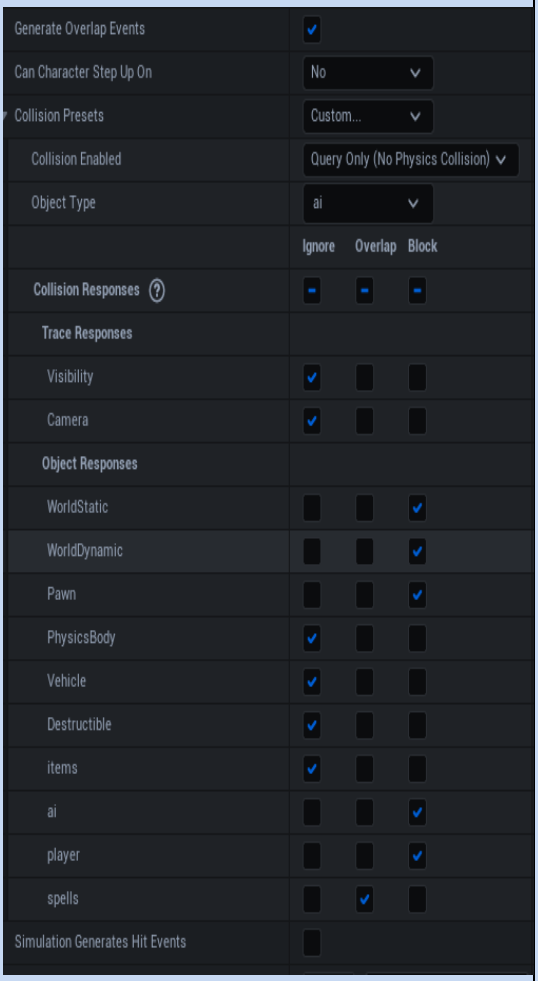
20. táblázat Plane objektum adatai

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 osztályból származtattam. Hozzáadtam egy Box_Collision komponenst ütközés, egy PaperFlipbook-ot, egy FloatingPawnMovement komponenst és egy BehaviorTree komponenst.

Ütközési beálltások:

Box	PaperFlibook
	

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) függvénnyel.

Megvizsgálom, hogy éppen a lény lejátszik-e animációt ezt a Is_Playing(Paper_Flipbook) függvénnyel 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 játékost.

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() függvénnyel 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_Recceive_Execute_AI eseményt típuskényszerítettem Controlled_Pawn változó értékét, majd meghívom a Stop_Movement_immediately(Floating_Pawn_Movement) függvényt, meghívom a Basic_Attack() függvényt. Hozzákötöm 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) függvénnyel 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) függvénnyel, 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) függvénnyel, 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) függvénnyel.

13.4. Game_over esemény

Megállítom a játékot a Set_Game_Paused(True) függvénnyel, 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 rakok 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_Contoller_Ref, in_Pawn) függvénnyel.

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 címkéje, 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 betölteve.

15.1. Menu_hud

15.1.1. Event_Begin_play

Létrehozok egy Widget elemett 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) függvénnyel.

15.2. ingame_hud

15.2.1. Event_Begin_Play

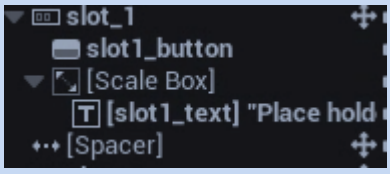
Létrehozok egy Widget elemett a Create_Widget(Souls_Xp, Player_Owner) függvény segítségével, majd hozzáadom a nézőponthoz a Add_to_Viewport(Return_Value) függvénnyel.

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) függvénnyel, 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 megjelenítésért. A következő elemekből építettem fel. ezt a Widget-et:



7. ábra UI slot szerkezete

Mivel maximum négy fejlesztést jelenítek 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 betöltöm az adatait. Meghívom a bonus_Stats(self, Item_Name_not_Found, index) függvényt és betöltöm 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 licensszel 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 felhasználkészített

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 amig a játékos ütközik vele.

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

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

Létrehoztam egy Boss ellenfelet, aminek olyan Behavior tree rendszert késztettem, 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. februér 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

Táblázatjegyzék