



Universitatea Tehnica de Construcții București
Facultatea de Hidrotehnica
Specializarea: Automatică și Informatică Aplicată

LUCRARE DE LICENTA

Coordonator științific:



Absolvent:

Cernea Ion-Alexandru

București, 2022



Universitatea Tehnica de Construcții București
Facultatea de Hidrotehnica
Specializarea: Automatica și Informatica Aplicată

**Aplicație software de tip joc video bazat pe
îndeplinirea unor activități ținta într-un mediu 3D**

Coordonator științific:



Absolvent:

Cernea Ion-Alexandru

București, 2022

Cuprins

| | |
|--|-----------|
| INTRODUCERE..... | 1 |
| CAPITOLUL I. | 3 |
| TEHNOLOGII ȘI INSTRUMENTE SOFTWARE FOLOSITE ÎN IMPLEMENTAREA PROIECTULUI | 3 |
| 1.1 Unreal Engine | 3 |
| 1.1.1 Versiuni Unreal Engine..... | 3 |
| 1.1.2 Blueprints..... | 7 |
| 1.1.2.1 Tipuri de Blueprints | 8 |
| 1.1.2.2 Anatomia Blueprint-ului..... | 10 |
| 1.1.3 Plugin-uri | 11 |
| 1.2 Vizual Studio 2019..... | 12 |
| CAPITOLUL II..... | 14 |
| STUDIU DE CAZ. APLICAȚIE SOFTWARE DE TIP JOC VIDEO BAZAT PE INDEPLINIREA UNOR ACTIVITAȚI ȚINTĂ INTR-UN MEDIU 3D | 14 |
| 2.1. Studiu bibliografic..... | 14 |
| 2.1.1 Istoria și bazele unui joc video..... | 14 |
| 2.1.2 Arhitectura și componentele unui joc single-player | 15 |
| 2.1.3 Diagrame UML | 17 |
| 2.2. Prezentarea detaliata a aplicației..... | 20 |
| 2.2.1 Inspirația | 20 |
| 2.2.2 Primele etape ale aplicației | 21 |
| 2.2.2.1 Primele setări..... | 21 |
| 2.2.2.2 Realizarea nivelului | 25 |
| 2.2.2.3 Realizarea caracterului | 27 |
| 2.2.3 Sistem de scriptări folosite în joc..... | 32 |
| 2.2.3.1 Ridicarea obiectelor | 32 |
| 2.2.3.2 Interacțiunea cu mediul înconjurător | 34 |
| 2.2.3.3 Obiecte de colecționat | 35 |
| 2.2.3.4 Lumini care se activează cu ajutorul butoanelor | 37 |
| 2.2.3.5 Realizarea magneților | 39 |
| 2.2.3.6 Sequence..... | 41 |
| 2.2.3.7 Zona de activare a sunetului | 42 |
| 2.2.3.8 Plasarea unui obiect pe o anumita direcție..... | 43 |
| 2.2.3.9 Realizarea portalelor | 44 |
| 2.2.3.10 Mișcarea robotului..... | 46 |

| | |
|--|----|
| 2.2.3.11 Cadran | 48 |
| 2.2.3.12 Peretele laser..... | 48 |
| 2.2.3.12 Cutie fizica | 49 |
| 2.2.3.13 Platforma buton | 49 |
| 2.2.3.14 Platforma de teleport..... | 50 |
| 2.2.3.12 Placa de presiune..... | 50 |
| 2.2.4 Ecranul de meniu și pauza..... | 51 |
| 2.3. Scenariul de funcționare al aplicației | 52 |
| 2.4. Manual de instalare și utilizare..... | 53 |
| 2.3.1 Cerințe software și hardware | 53 |
| 2.3.2 Instalare și rulare..... | 54 |
| CONCLUZII, CONTRIBUTII SI PERSPECTIVE DE VIITOR | 55 |
| Concluzii..... | 55 |
| Contribuții personale | 55 |
| Perspective de viitor | 55 |
| BIBLIOGRAFIE..... | 56 |
| ANEXE..... | 58 |

INTRODUCERE

- **Motivația**

Jocurile video nu sunt doar pentru divertisment, ele oferă și locuri de muncă. Dezvoltarea acestor jocuri necesită cooperarea a numeroși dezvoltatori care se ocupă de orice, de la producție la programare. Aceștia produc jocuri care pot fi jucate pe o varietate de platforme precum console, computere și telefoane mobile.

Jocurile video sunt o activitate de a petrece timpul liber, care continuă să crească în popularitate. Această lucrare este o examinare a ceea ce motivează oamenii să creeze și să joace jocuri pe calculator. Când toate motivațiile unui individ de a juca jocuri video sunt pentru urmărirea "distracției", se spune că o motivație intrinsecă este motivația cea mai răspândită. Jocurile pot oferi un mediu sigur, în care înveți despre consecințele acțiunilor prin experiență. Jocurile pe calculator pot facilita dezvoltarea unor mecanisme de automonitorizare și de adaptare.

Scopul proiectului este de a oferi o metoda de instruire jucătorului prin realizarea unui joc 3D folosind un motor grafic. Scopul jocului este explorarea unei fabrici defecte cu scopul de a o reporni. Jocul va fi single-player, alcătuit dintr-un singur nivel. Player-ul (caracterul) va putea interacționa cu mediul înconjurător și va trebui să rezolve diverse probleme pentru a putea prograda. În cadrul aplicației se vor implementa diverse obiecte, obstacole și modul audio.

- **Obiectivul proiectului**

Principalul obiectiv al acestei lucrări a fost construirea unui mediu virtual interactiv pentru caracter și dezvoltarea cunoștințelor personale în cadrul realizării unui joc video în motorul grafic Unreal.

Ideea principală a jocului constă în deplasarea player-ului dintr-un punct de start, într-un punct final. Player-ul se deplasează și este ghidat cu ajutorul instrumentelor pentru a ajunge la sfârșitul jocului. Jocul se termina atunci când jucătorul a rezolvat problema finală.

Jocul va fi construit în aşa fel încât să poată fi folosit ușor de orice persoană capabilă de a lucra la un nivel mediu sau redus cu un computer. Nu sunt necesare cunoștințe tehnice

deosebite. Este necesar un echipament cu hardware avansat deoarece în joc s-au folosit texturi de calitate ridicata.

- **Structura lucrării**

Aceasta lucrare conține două capitole, anticipate printr-o introducere unde se vorbește despre motivația și obiectivul proiectului și sfârșește cu concluziile, contribuțiile și perspectivele de viitor. Aspectele teoretice despre tehnologiile și instrumentele folosite vor fi discutate în primul capitol. Capitolul doi conține o analiză a jocurilor video și a istoriei acestora, după care este prezentată aplicația aşa cum a fost concepută, realizată și funcțională. Documentația se încheie cu referințele web și anexele prezentate în bibliografie.

CAPITOLUL I.

TEHNOLOGII ȘI INSTRUMENTE SOFTWARE FOLOSITE ÎN IMPLEMENTAREA PROIECTULUI

Există mai multe software-uri gratuite de design 3D disponibile publicului, inclusiv Unreal Engine și Blender, care promovează comunitățile care se autoeducre, precum și modele și tutoriale 3D pe piață pentru începători.

1.1 Unreal Engine



Figura 1.1 Unreal Engine Logo

Unreal Engine a fost prezentat prima dată în 1998 de compania Epic Games ca un motor de joc 3D pentru jocurile de tip first person shooter. Datorită modularității și ritmului rapid de învățare, Unreal Engine a fost adoptat și de alte industrii cum ar fi cea a filmelor. Folosind limbajul de programare C++, acesta dispune un grad ridicat de portabilitate pentru mai multe platforme, atât desktop cat și mobile.

1.1.1 Versiuni Unreal Engine

Calculele grafice au fost gestionate de CPU în prima generație de Unreal Engine (1995), care se baza în întregime pe sistemul de operare al software-ului. Cu toate acestea, de-a lungul timpului, a fost posibil să se profite de capacitațile oferite de plăcile grafice special concepute pentru acest scop, concentrându-se pe API-ul Glide. Deși OpenGL și Direct3D au fost acceptate, au raportat o performanță mai slabă în comparație cu Glide, din cauza limitărilor lor la momentul respectiv în gestionarea texturii. Sweeney a criticat driverele OpenGL în special pentru calitatea lor slabă, numindu-le „foarte supărătoare” și codul de implementare ca fiind „teribile”, spre deosebire de suportul Direct3D mai simplu și mai bine conceput.

Printre caracteristicile sale s-au inclus capacitatele de a identifica coliziunile, iluminarea colorată și un tip restrictiv de filtrare a textului. Motorul include, de asemenea, un editor de nivel numit UnrealEd, care a susținut operațiuni constructive de geometrie solidă în timp real din 1996 și le permite cartografilor să schimbe aspectele cele mai mici ale nivelului.

Cu ajutorul simulatoarelor de „light blooms, fog volumes și composite skies”^[1], Unreal a crescut miza la ceea ce se pot aștepta jucătorii de la următoarele produse.



Figura 1.2 Joc realizat în motorul Unreal Engine (Unreal Tournament)

Generația Unreal Engine 2(2001) a înregistrat un progres vizibil în ceea ce privește randarea, precum și îmbunătățirea noilor instrumente. O varietate de funcții, cum ar fi un instrument de editare cinematografică, sisteme de particule, plug-in-uri de export pentru 3D Studio Max și Maya și un sistem de animație intrigant văzut pentru prima dată în versiunea pentru PlayStation 2 a Unreal Tournament, au fost încorporate în motor, făcând este capabil să creeze texturi care sunt aproape de 100 de ori mai detaliate decât cele găsite în predecesorul sau Unreal. În plus, interfața utilizatorului UnrealEd a fost rescrisă în C++ folosind un set de instrumente numite „wxWidgets”^[2], care, potrivit lui Sweeney^[3], este „cel mai bun lucru disponibil în prezent”. Pentru a rula simulări fizice, cum ar fi coliziunile jucătorilor și „ragdolls”^[4], Epic a folosit motorul de control fizic Karma^[5], un produs al studioului britanic Math Engine.

¹ Light blooms, fog volumes, composite skies sunt efecte de grafica computerizata

² wxWidgets este un set de instrumente pentru crearea interfeței grafice

³ Sweeney Tim este programator și fondatorul lui Epic Games

⁴ ragdolls este un tip de animație procedurală

⁵ Karma este un motor fizic de corp rigid



Figura 1.3 Exemplu de joc realizat în Unreal Engine 2 (Quake)

Noi funcții de „render”^[7], fizice, audio și instrumentale au fost caracteristicile principale în Unreal Engine 3 (2006). UE3 a fost realizat special pentru a profita de hardware-ul shader complet programabil. Toate calculele de iluminare și de umbre au fost efectuate pixel pe pixel, mai degrabă decât per-vertex^[6]. În ceea ce privește jocul, Unreal Engine 3 a oferit suport pentru o gamă dinamică înaltă de render, obiecte și medii distructibile perfecționate, „crowd simulation” și „soft-body”^[8], o compatibilitate iOS, integrare Steamworks, o soluție pentru iluminarea globală în timp real și 3D pentru a conduce grafica în timp real la un nou nivel.



Figura 1.4 Exemplu de evoluție a graficii pana la Unreal 3

⁶ per-vertex culoarea este calculata pentru fiecare vertex și apoi interpolata

⁷ render este procesul de generare a unei imagini dintr-un model 2D sau 3D

⁸ soft-body se concentrează pe simulările fizice realiste ale obiectelor deformabile

Principala caracteristică destinată pentru Unreal Engine 4 (2014) a fost iluminarea globală în timp real, folosind „voxel cone tracing”^[9], eliminând iluminarea pre-calculată. Cu toate acestea, această caracteristică, cunoscută sub numele de Sparse Voxel Octree Global Illumination (SVOGI) și demonstrată cu varianta de test denumita „Elemental”, a fost înlocuită cu un algoritm mai eficient din punct de vedere computațional din cauza problemelor de performanță. În plus, UE4 are un nou sistem de scriptare vizuală numit „Blueprints” (un înlocuitor pentru „Kismet”-ul generației precedente) care permite dezvoltarea rapidă a logicii jocului fără a fi nevoie de cod, rezultând un decalaj mai mic între artiștii tehnici, designerii de jocuri și programatorii.

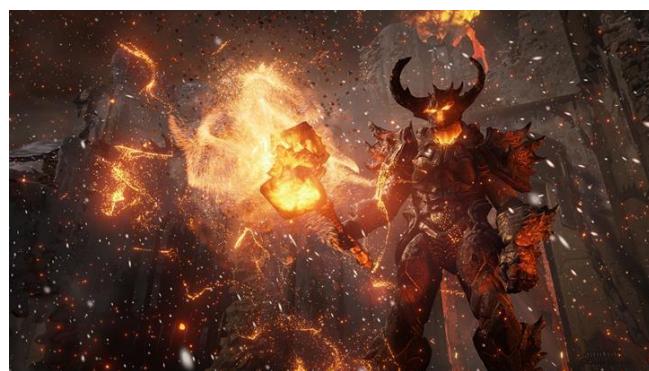


Figura 1.5 Reclama la Unreal Engine 4

Una dintre principalele caracteristici ale Unreal Engine 5 (2022) este „Nanite”, un motor care permite importul de conținut fotografic detaliat în jocuri cu ajutorul tehnologiei Quixel, care este cea mai mare biblioteca de fotogrammetrie, achiziționat de Epic anterior. Scopul lui Unreal Engine 5 a fost să le fie cât mai simplu posibil pentru dezvoltatori să creeze lumi complexe de joc fără a fi nevoie să petreacă cantități exagerate de timp pentru a dezvolta detalii de textură complicate. Nanite este capabil să importe aproape orice altă reprezentare existentă a unui obiect în trei dimensiuni, inclusiv modele ZBrush și CAD, permitând utilizarea lor și în calitate cinematografică.

Spre deosebire de ceea ce ar face în mod normal un artist, Nanite gestionează automat nivelul de detaliu (LOD) al acestor obiecte importând „target platform”^[10] și „draw-distance”^[11] corespunzător.

⁹ voxel cone tracing este un algoritm de urmărire a razelor

¹⁰ target platform face referință la o platformă de focus

¹¹ draw-distance este distanța maximă a obiectelor dintr-o scenă, cele trecute peste nu vor fi afișate

O altă componentă este „Lumen”, descrisă ca o „soluție de iluminare globală, complet dinamică, care reacționează instantaneu la schimbările de scenă și lumină”. Lumen elimină nevoia artiștilor și dezvoltatorilor de a crea o sursă de lumină strălucitoare pentru o anumită scenă, calculând schimbul dintre reflexiile de lumină și umbră, permitând surselor de lumină să se comporte aşa cum ar fi în timp real. O altă caracteristică nouă a lui Unreal Engine 5 se numește „Virtual Shadow Maps”, care este descrisă ca „o nouă metodă de creare a hărților de umbre utilizate pentru a oferi umbre adiacente, cu rezoluție mai mare, care funcționează de-o calitate cinematografică și o iluminare dinamică în lumi deschise”. Componentele suplimentare includ „Niagara” pentru dinamica fluidelor și a particulelor și „Chaos” pentru un motor fizic.



Figura 1.6 Exemplu de joc făcut în UE5 folosind componente Nanite și Lumen

1.1.2 Blueprints

Sistemul de scriptare vizuală „Blueprint” al Unreal Engine este folosit pentru scriptarea jocului și se bazează pe ideea de a folosi câteva interfețe de noduri pentru a crea elemente de joc în cadrul editorului Unreal.

Este folosit pentru a defini clase sau obiecte care sunt obiecte orientate pe obiecte (OO) în motor, aşa cum este cazul multor alte limbaje de scripting comune. Acest sistem este foarte puternic și flexibil, deoarece oferă designerilor posibilitatea de a utiliza întreaga gamă de concepte și instrumente care sunt de obicei disponibile numai programatorilor. În plus, disponibilitatea unui „Blueprint” specific în implementarea C++ a Unreal Engine permite programatorilor să creeze sisteme de bază care pot fi extinse la designeri.

În forma lor de bază, Blueprints sunt scripturi vizuale adăugate la joc. Prin conectarea nodurilor, evenimentelor, funcțiilor și variabilelor cu fire este posibil să se creeze elemente

complexe de joc. Blueprints funcționează prin utilizarea nodurilor grafice în diverse scopuri - construcția obiectelor, funcții individuale și evenimente generale de joc - care sunt specifice fiecărei instanțe a Blueprint-ului pentru a implementa caracterul și alte funcționalități.

1.1.2.1 Tipuri de Blueprints

Blueprint-ul este de mai multe tipuri în care fiecare are propria lui utilizare specifică, de la crearea de tipuri noi de evenimente, la nivel de scriptare, la definirea interfețe sau macro comenzi pentru a fi utilizate de către alte Blueprint-uri.

- „Blueprint class”, adesea prescurtată ca Blueprint, este un atu care permite creatorilor de conținut să adauge cu ușurință funcționalități pe lângă clasele de joc existente. Blueprint este creat vizual în interiorul lui Unreal Editor, în loc să tasteze cod și salvate ca active într-un pachet de conținut. Acestea definesc, în esență, o nouă clasă sau tip de actor, care pot fi apoi plasate în hărți ca instanțe care se comportă ca orice alt tip de actor.
- „Data-Only Blueprint” este o clasă Blueprint care conține numai codul (sub formă de grafice nod), variabile și componente moștenite de la părintele său. Acestea permit ca aceste proprietăți moștenite să fie optimizate și modificate, dar nu pot fi adăugate elemente noi. Acestea sunt, în esență, un înlocuitor pentru arhetipuri și pot fi folosite pentru a permite designerilor să optimizeze proprietățile sau să seteze elemente cu variații.
- „Level Blueprint” este un tip specializat de Blueprint care acționează ca un grafic de evenimente global la nivel. Fiecare nivel din proiect are propriul plan creat în mod implicit, care poate fi editat în editorul Unreal, cu toate acestea noi planuri de nivel nu pot fi create prin interfața editorului. Evenimentele referitoare la nivelul în ansamblu sau anumite instanțe ale actorilor din cadrul nivelului sunt utilizate pentru a declanșa secvențe de acțiuni sub formă de „Function Calls” sau „Flow Control”.
- „Blueprint Interface” este o colecție de una sau mai multe funcții ,numai nume, fără implementare, care pot fi adăugate la alte planuri. Orice Blueprint care are interfață adăugată este garantat să aibă aceste funcții. Funcțiile interfeței pot primi funcționalitate în fiecare dintre planuri. Acest lucru este, în esență, conceptul de o interfață care permite mai multe tipuri diferite de obiecte să fie partajate și să fie accesate printr-o interfață

comună. Mai simplu spus, interfețele Blueprint permit diferitelor planuri să partajeze și să trimită date unul altui.

- „Blueprint Macro Library” este un container care deține o colecție de macro comenzi sau grafice autonome care pot fi plasate ca noduri în alte blueprint-uri. Acestea pot economi timp, deoarece pot stoca secvențe de noduri utilizate în mod obișnuit, completate cu intrări și ieșiri atât pentru execuție, cât și pentru transferul de date. Macro comenzile sunt partajate între toate graficele care le fac referire. Aceasta înseamnă că bibliotecile de macro comenzi Blueprint nu trebuie să fie compilate. Cu toate acestea, modificările la o macro comandă se reflectă numai în graficele care fac referire la macro comanda respectivă atunci când planul care conține acele grafice este recompilat.
- „Blueprint Utility” (sau Blutility pe scurt) este un Blueprint doar pentru editor care poate fi utilizat pentru a efectua acțiuni de editor sau pentru a extinde funcționalitatea editorului. Acestea pot expune evenimente fără parametri ca butoane în interfață cu utilizatorul și au capacitatea de a executa orice funcții expuse la Blueprints și de a acționa asupra setului curent de actori selectați în viewport.

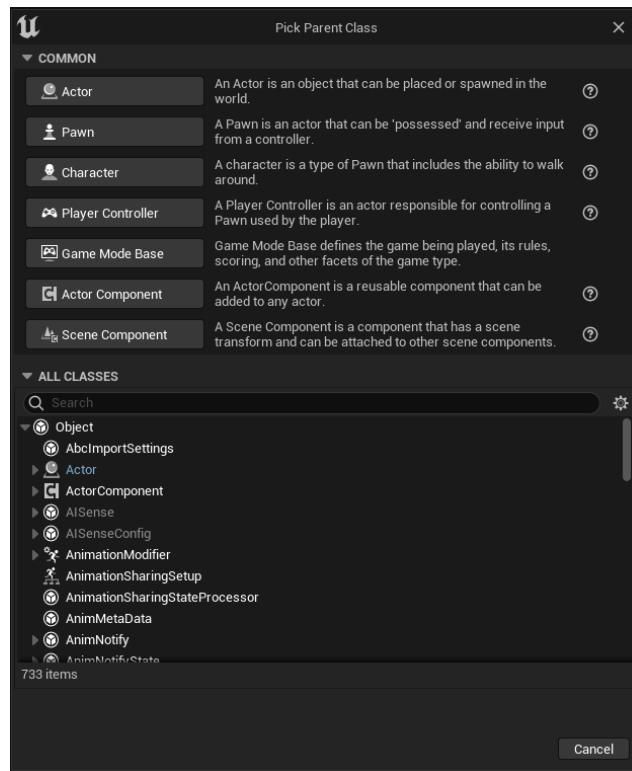


Figura 1.7 Meniu interfață Blueprint

1.1.2.2 Anatomia Blueprint-ului

Funcționalitatea Blueprint-ului este definită folosind diverse elemente, dintre care unele sunt prezente în mod implicit, în timp ce altele pot fi adăugate după necesitate. Acestea oferă capacitatea de a defini componente, de a efectua operațiuni de inițializare și configurare, de a răspunde la evenimente, de a organiza și modulariza operațiunile, de a defini proprietăți și altele.

Fereastra componentelor din editorul Blueprint vă permite să adăugați componente la Blueprint. Acest lucru oferă un mijloc de a adăuga coliziuni geometrice prin „CapsuleComponents”, „BoxComponents” sau „SphereComponents”, adăugând geometria randată sub formă de „StaticMeshComponents” sau „SkeletalMeshComponents”, controlând mișcarea folosind „MovementComponents” și aşa mai departe. Componentele adăugate în lista componentelor pot fi, de asemenea, atribuite variabilelor de instanță care oferă acces la ele în graficele acestui plan sau ale altor planuri.

- „Construction Script” execută urmând lista componente atunci când se creează o instanță a unei clase Blueprint. Aceasta conține un grafic nod care este executat permitând instanței Blueprint Class să efectueze operațiuni de inițializare precum setarea plaselor și materialelor.
- „Event Graph” în Blueprint conține un grafic nod care utilizează evenimente și apeluri de funcții pentru a efectua acțiuni ca răspuns la evenimentele de joc asociate cu Blueprint. Acest lucru este utilizat pentru a adăuga funcționalitate care este comună tuturor instanțelor. Aici sunt configurate interactivitatea și răspunsurile dinamice. De exemplu, un plan de lumină ar putea răspunde la un eveniment acționat prin oprirea componentei „LightComponent” și schimbarea materialului utilizat de „mesh-uri” 3D.
- „Funcțiile” sunt noduri grafice aparținând unui anumit Blueprint care pot fi executate sau apelate dintr-un alt grafic. Funcțiile au un singur punct de intrare desemnat de un nod cu numele funcției care conține un singur pin „exec” de ieșire. Când funcția este apelată dintr-un alt grafic, pinul exec de ieșire este activat determinând executarea rețelei conectate.
- „Variabilele” sunt proprietăți care dețin o valoare, fac referire la un obiect sau un actor din lume. Aceste proprietăți pot fi accesibile intern schiței care le conține sau pot fi accesibile extern, astfel încât valorile lor să poată fi modificate de proiectanții care lucrează cu instanțe ale Blueprint-ului plasate într-un nivel.

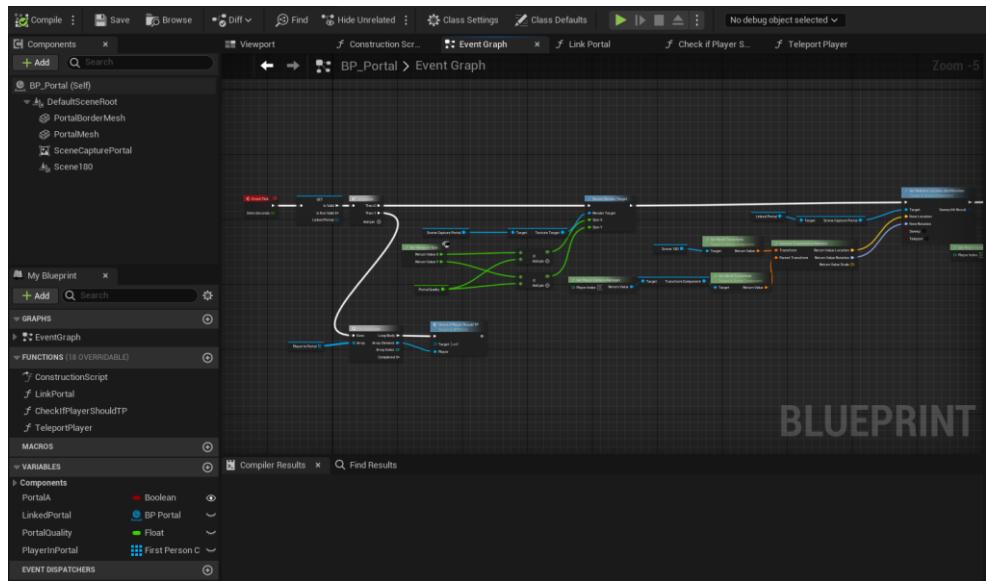


Figura 1.8 Exemplu Viewport Blueprint

1.1.3 Plugin-uri

Plugin-urile sunt colecții de date și cod pe care dezvoltatorii le pot activa sau dezactiva cu ușurință în editor pe bază de proiect. Plugin-urile pot adăuga funcționalitatea gameplay-ului runtime, pot modifica caracteristicile motorului încorporate sau pot adăuga altele noi, pot crea noi tipuri de fișiere și pot extinde capacitatele Editorului cu meniuri noi, comenzi din bara de instrumente și sub-moduri.

Pe lângă plugin-urile de baza s-au mai adăugat următoarele:

Async Loading Screen permite să configurați cu ușurință un sistem de „loading screen” în setările proiectului, unde acesta apare în timpul încărcării jocului.

Bridge, cu ajutorul acestuia ai acces la librăria „Megascans” unde puteți răsfoi colecții și adăuga „asset-uri” la proiectele Unreal Engine.

Flat Nodes este un plugin care nu are nici un impact asupra performanței și stabilității, ci adaugă un aspect vizual simplificat, mai plat, care face graficele mai clare și mai ușor de citit.

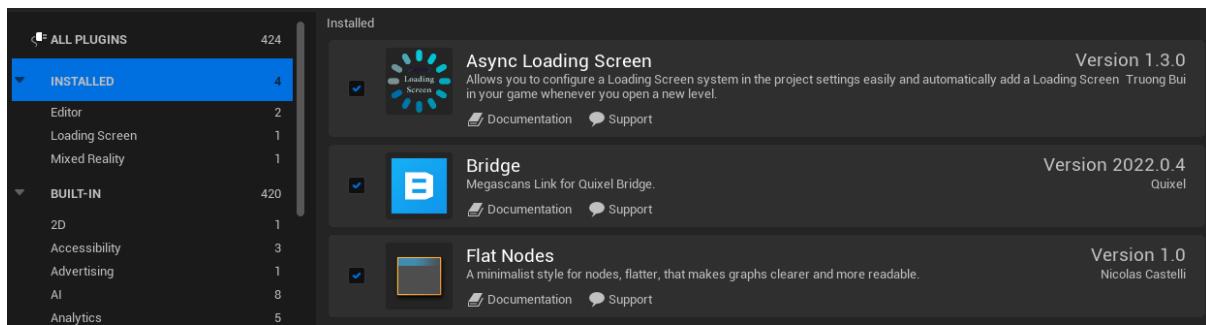


Figura 1.9 Plugin-uri folosite

1.2 Vizual Studio 2019



Figura 1.10 Logo Visual Studio

Microsoft Visual Studio este un mediu de dezvoltare integrat (IDE) de la Microsoft. Poate fi utilizat pentru a crea aplicații de consola și de interfață grafică pentru toate platformele acceptate de Microsoft Windows. Acesta oferă un editor, design, compilator și debugger pentru mai multe limbaje de programare cum ar fi: C, C#, C++, Basic, Web developer. De asemenea oferă și suport și pentru alte limbaje de scris precum: Python, Ruby, Rust, CSS și JavaScript.

Microsoft Visual Studio oferă caracteristici, inclusiv completarea automată a codului, crearea de fragmente de propoziție din cod folosind prescurtarea și formatarea codului. O altă caracteristică crucială este capacitatea de a depana codul, unde Visual Studio excellează și oferă utilizatorului o varietate de instrumente pentru a face procesul plăcut. Una dintre ele permite utilizatorului să mute liniile de cod ale programului, astfel încât să poată fi comparate una cu alta pentru analiză. O altă opțiune pune la dispoziția utilizatorului o listă de variabile inițializate și permite vizualizarea acestora pe parcursul execuției codului.

C++ este un limbaj de programare de uz general ca o extensie a limbajului de programare C sau "C cu clase". Limbajul s-a extins semnificativ de-a lungul timpului, iar C++ modern are acum caracteristici orientate pe obiecte, generice și funcționale, pe lângă facilitățile de manipulare a memoriei la nivel scăzut. Este aproape întotdeauna implementat ca un limbaj compilat și mulți furnizori oferă compilatoare C++, inclusiv Free Software Foundation, LLVM, Microsoft, Intel, Oracle și IBM, deci este disponibil pe multe platforme.

C++ a fost proiectat cu o orientare către programarea sistemelor și software încorporat, limitat de resurse și sisteme mari, cu performanță, eficiență și flexibilitate de utilizare ca evidențiază designul său. C++ a fost, de asemenea, considerat util în multe alte contexte, punctele forte fiind infrastructura software și aplicațiile limitate de resurse, inclusiv aplicațiile desktop, jocurile video, serverele (de exemplu, comerțul electronic, căutarea pe web sau bazele de date) și aplicațiile critice pentru performanță (de exemplu, comutatoare telefonice sau sonde spațiale).

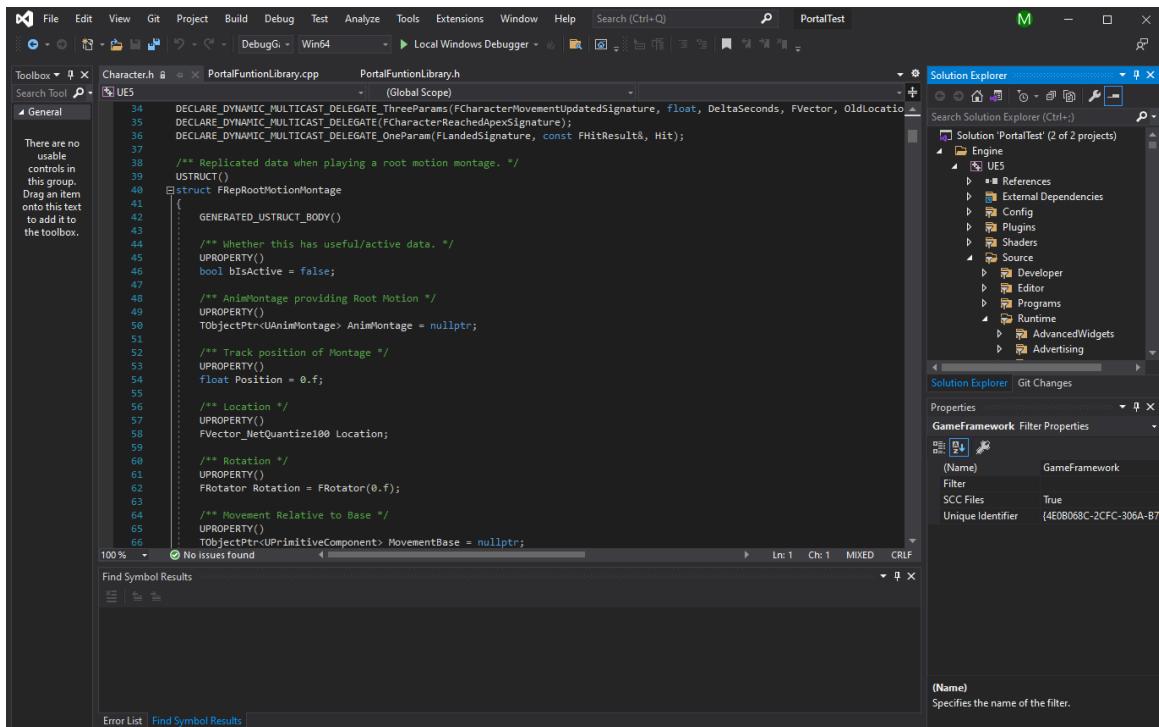


Figura 1.11 Interfața Visual Studio

CAPITOLUL II.

STUDIU DE CAZ. APLICAȚIE SOFTWARE DE TIP JOC VIDEO BAZAT PE INDEPLINIREA UNOR ACTIVITAȚI ȚINTĂ INTR-UN MEDIU 3D

2.1. Studiu bibliografic

2.1.1 Istoria și bazele unui joc video

Jocurile video sunt aplicații electronice dezvoltate pentru divertisment rulate pe dispozitive electronice precum computere, console, telefoane, mașini arcade și altele. În general jocurile prezintă un sistem de recompensare a jucătorului prin îndeplinirea unor obiective din joc.

Termenul „joc video” se referă la un joc în care ecranul calculatorului servește ca mediu de feedback principal, iar perifericele de intrare cum ar fi tastatura, mouse-ul sau controlerul, sunt folosite ca dispozitive de control.

Existența jocurilor video se datorează motoarelor grafice. Un sistem conceput pentru a crea și dezvolta un joc video este cunoscut sub numele de motor grafic. Oferă tehnologii de bază care facilitează operarea pe platforme diferite și eficientizează dezvoltarea acestora. Tehnologiile fundamentale includ un motor de redare 2D și 3D, detectarea coliziunilor, scripting, animație, scenografie și sunet. Primele jocuri și reguli au fost dezvoltate pe EDSAC sau osciloscop în anii 1950 și 1960. A. S. Douglas a dezvoltat o versiune a jocului X și O în 1952. Jocul „Tennis For Two” (sau ping-pong) a fost creat de William Higinbotham și a debutat în 1958.

Anii 1970-1980 au fost numiți „Epoca de aur a jocurilor Arcade” deoarece au apărut Jocurile Arcade care au căpătat o popularitate imensa datorită ușurinței utilizării. Primul joc arcade care funcționa pe bani a fost „Computer Space”. Industria jocurilor a început să crească atunci când compania „Atari” a lansat o ediție de „Pong” unde puteai să-l joci oriunde. Din cauza acestora tehnologia microprocesoarelor a avansat mult datorită cererii lor pe piața consolelor individuale.

Motorul grafic a apărut la mijlocul anilor 1990 și a avut o strânsă legătură cu jocurile 3D, în special cu cele „first person-shooter”. Ceea ce a ajutat la dezvoltarea motorului au fost jocurile „DOOM” și „Quake” unde din cauza popularității lor, ceilalți dezvoltatori au licențiat componentele de baza ale codului.

Multe jocuri se încadrează în mai multe categorii din cauza subiectivității genurilor. De exemplu seria de jocuri „The Elder Scrolls” are elemente de puzzle, aventura, Role Playing Games și acțiune. Ceea ce înseamnă ca mai multe jocuri din zilele noastre devin hibride combinând diferite caracteristici precum strategie, aventura, multiplayer, acțiune, sport, arcade, simulare, sociale, horror și altele.

2.1.2 Arhitectura și componentele unui joc single-player

Proiectarea jocurilor începe pe o schiță unde se vor nota diferite idei și concepte despre conținutul jocului, caracteristici, setări și poveste, cerințele programului, personalul și publicul țintă.

În timpul creării unui joc video sunt multe decizii de luat cu privire la design, de exemplu la limitările hardware și buget. Modificările acestea au un impact pozitiv sau negativ asupra resurselor disponibile.

- Un designer de joc este o persoană care scrie detalii amănunțite ale conceptelor lor pentru jocurile create. Pentru a asigura ca viziunea lui este îndeplinită, designerii colaborează îndeaproape cu programării și artiștii în timpul construcției unui joc. Sub îndrumarea unui designer principal, multe echipe de design se concentreză asupra diferitelor aspecte ale jocului. Unii designeri creează povesti și protagonisti, alții nivele și mecanici.
- Designerii mecanicilor de joc se concentreză pe aspectele esențiale ale gameplay-ului. Drept urmare, sarcinile se bazează pe tipul de joc video la care lucrează, unde se proiecteză și echilibrează diferite reguli ale jocului.
- Designerul de nivel poate construi peisaje fantastice sau realiste în jocuri. Acesta selectează setările pentru nivel și a misiunilor care se potrivesc cel mai bine genului de joc realizat pentru a-i atrage pe deplin pe jucători.
- Programatorii de jocuri video au o perspectivă diferita asupra unui joc fata de jucători sau alți membri de producție. Un joc video este compus dintr-o mulțime de linii de cod care îi spun computerului cum să gestioneze totul, de la regulile jocului până la grafica acestuia. Programatorii creează jocuri video de la zero, scriind instrucțiuni în programul de calculator, sub conducerea echipei principale de dezvoltare. Diferite limbi de codare sunt folosite de programatori. Programatorii aleg limbajul care răspunde cel mai bine cerințelor lor, deoarece fiecare are un set diferit de capabilități.

Exemple de design folosite într-un joc video:

- Designul lumii este crearea unei povesti de fundal, decor și temă pentru joc; de multe ori făcut de către designerul principal. Crearea lumii poate fi, de asemenea, crearea unui univers sau a unei hărți, precum și zone sau subiecte care sunt de interes și urmărite de jucător. Este o hartă la care se face referire pentru crearea a tot ceea ce arată unde este modelează direcția spre care se îndreaptă jocul.
- Designul sistemului este simularea adoptată a unui joc conceput pentru a interacționa sau a reacționa cu jucătorul prin crearea de reguli și modele matematice. "Experiența" pe care un jucător o are cu un joc este atribuită modului în care este proiectat sistemul jocului. Un sistem complex cu adâncime duce la o serie mai imprevizibilă de evenimente pentru a cufunda jucătorul în jocul video.
- Design de conținut se referă la dezvoltarea de personaje, puzzle-uri, misiuni și orice altă componentă a jocului funcțională și în conformitatea cu standardul viabil al produsului. În esență, conținutul este o complexitate adăugată unui produs pentru a ii crește valoarea pe piață.
- Unul dintre primele lucruri pe care oamenii le fac atunci când creează un joc video este dialogul, textul și scrierea poveștii. Conține o varietate de elemente diferite legate de proces. Elementele unui scenariu de joc includ actorie vocală, text, editare de imagini și muzică.
- Pentru a crea un univers de joc, nivelul de planificare utilizează domenii diverse. Pentru a atrage atenția unui jucător, sunt folosite elemente precum iluminarea, spațiul, limita, culoarea și contrastul. Aceste elemente pot fi apoi folosite de către un creator de joc fie pentru a conduce jucătorul într-o direcție specifică prin lumea jocului, fie pentru a-l induce în eroare.
- Proiectarea interfeței cu utilizatorul (UI) se concentrează pe crearea de interacțiuni utilizator-interfață, cum ar fi meniurile.
- Proiectarea interfeței este, de asemenea, o parte a proiectării mecanice. Determină căte informații primește jucătorul și modul în care designerul îl poate informa pe jucător despre lume.
- Designul audio implică crearea și încorporarea tuturor sunetelor, inclusiv muzica, efectele sonore și acțiunea vocală.

Nenumărate jocuri au elemente narrative care plasează evenimentele într-un context, făcând jocul mai puțin abstract și crescând varietatea. Jocurile sunt, în esență, un fel de naraciușe, după unii. În realitate, o poveste poate servi ca motiv pentru crearea unui joc sau poate fi încorporată într-un design ca set de mecanici de joc.

Gameplay-ul este o componentă interactivă a jocurilor video. Procesul de a juca un joc implică interacțiunea utilizatorului cu acesta și este de obicei în scopuri educaționale, divertisment sau recreative.

Un joc video „single-player” este un joc video în care se așteaptă informații de la un singur jucător pe tot parcursul sesiunii de joc. Cele mai multe jocuri moderne și jocuri arcade sunt concepute astfel încât acestea să poată fi jucat de către un singur jucător; deși multe dintre aceste jocuri au moduri care permit doi sau mai mulți jucători pentru a juca (nu neapărat simultan), foarte puține de fapt, necesită mai mult de un jucător pentru jocul care urmează să fie jucat. Seria Portal este un exemplu în acest sens.

2.1.3 Diagrame UML

Unified Modeling Language (abreviat UML) este un sistem standard pentru descrierea modelelor și specificațiilor software. Limbajul a fost conceput de Object Management Group (OMG), care este responsabil, printre altele, pentru standardul de mesagerie CORBA. UML a fost creată ca bază pentru reprezentarea unor programe complicate orientate pe obiecte, fundația fiind structurarea claselor și a instanțelor (nume și obiecte) ale acestora. UML este utilizat și în afara lumii datorită eficienței și clarității sale în reprezentarea elementelor abstracte. Ca urmare, există aplicații legate de UML pentru managementul de proiect, proiectarea proceselor de afaceri și așa mai departe.

Diagrama cazurilor de utilizare - reprezintă un model inițial conceptual al unui sistem în procesul de proiectare și exploatare. Esența acestei diagrame constă în faptul că sistemul proiectat se reprezintă ca o colecție de entități și actori care colaborează printr-un sistem de relații de dependență, asociere sau generalizare cu sistemul și cu ajutorul anumitor cazuri de utilizare. Diagrama cuprinde părțile componente ale motorului grafic, cum ar fi nivelul, UI și arhitectura jocului.

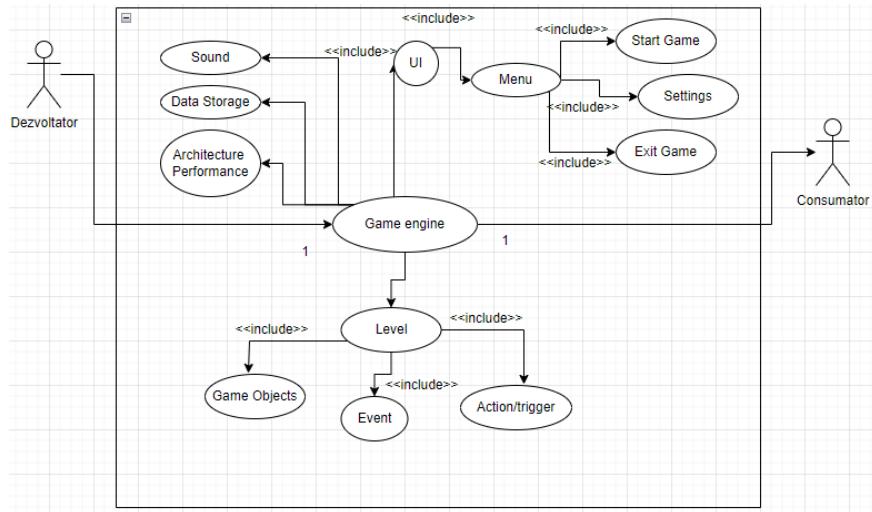


Figura 2.1 Diagrama cazurilor de utilizare

Diagrama de clase constituie punctul de plecare a altor tipuri de diagrame, este construit pentru a descrie structura sistemului, cum ar fi stările obiectelor sau colaborările dintre obiecte.

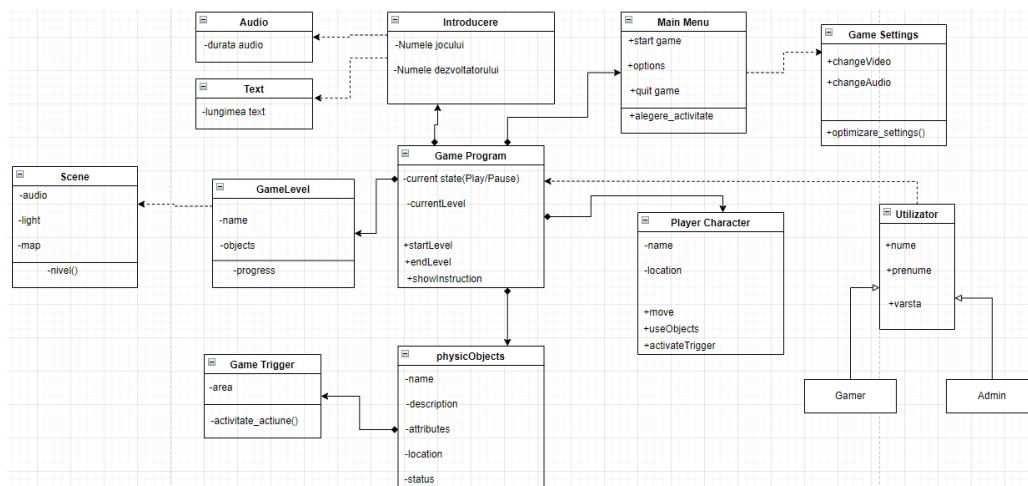


Figura 2.2 Diagrama de clase

Diagrama de secvență pune accentul pe aspectul pe ordonarea în timp a acțiunilor, a stărilor obiectelor, evenimentelor și organizarea acestora. Prezintă interacțiunea jucătorului cu aplicația și a caracterului cu diferite obiecte și evenimente.

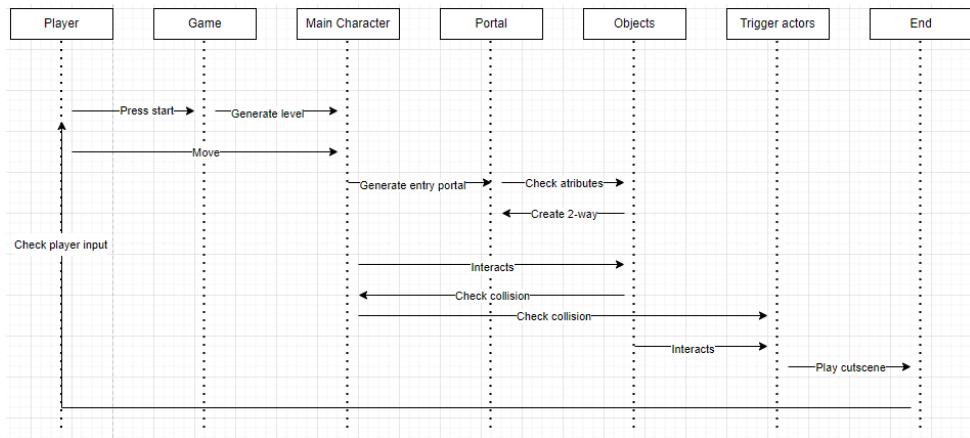


Figura 2.3 Diagrama de secvență

Diagrama de colaborare pune accentul pe organizarea structurală a obiectelor care participă la interacțiune. Ilustrează mai bine ramificări complexe, iterări și comportament concurrent.

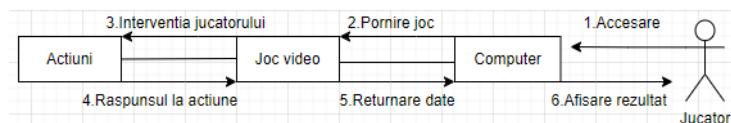


Figura 2.4 Diagrama de colaborare

Diagrama de activități este folosită pentru a modela dinamica unui proces sau a unei operații și scoate în evidență controlul execuției de la o activitate la alta.

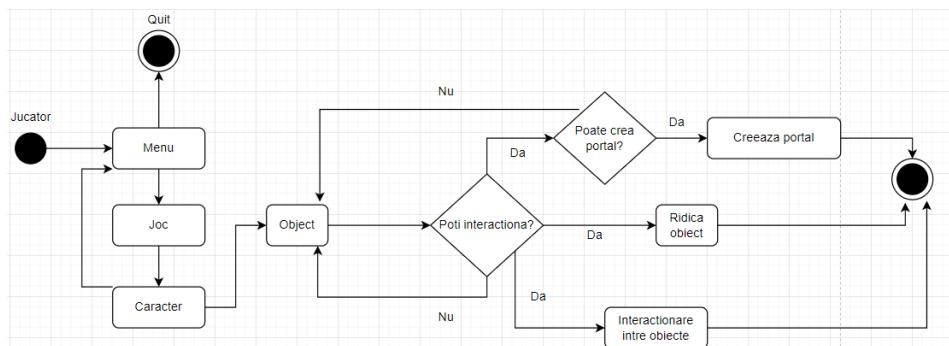


Figura 2.5 Diagrama de activități

Diagrama de stare este folosită pentru a modela comportamentul mai multor obiecte. Diagrama de stări specifică o secvență de stare prin care trece un obiect ca răspuns la evenimente.

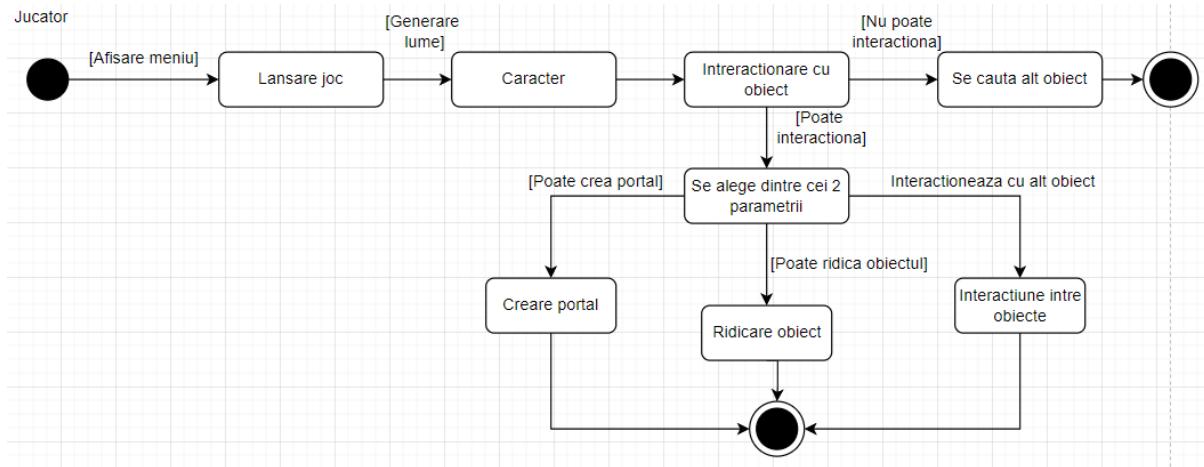


Figura 2.6 Diagrama de stare

2.2. Prezentarea detaliata a aplicației

2.2.1 Inspirația



Figura 2.7 Logoul jocului Portal

Inspirația principala a aplicației a fost jocul „Portal”, o serie de două jocuri, Portal și Portal 2, dezvoltată de „Valve”. Personajul principal, Chell, este forțată să se supună unor serii de teste în cadrul Aperture Science Enrichment Center de către o inteligență artificială rău intenționat, GLaDOS, care controlează facilitatea.

Ambele jocuri „Portal” au loc în centrul imaginariu "Aperture Science Enrichment Center" fondată de Cave Johnson. Cercetările sale au ajutat la descoperirea tehnologiei portalului și, în curând, au devenit un concurent direct cu Black Mesa Research Facility (Half-Life) pentru finanțare guvernamentală. Johnson a achiziționat drepturile asupra unei mine de sare dezafectate din Peninsula Superioară Michigan, unde au început să construiască un set labirintic de birouri, laboratoare, facilități și camere de testare. În acest timp, Johnson a devenit

otrăvit de le expunerea la praful lunii, o componentă cheie a vopselei necesare pentru tehnologia portalului.

Cele mai multe dintre teste implică utilizarea "Aperture Science Handheld Portal Device", "arma portal" care creează o gaura de vierme între două suprafete plane. Caracterul jucătorului sau obiectele din lumea jocului se pot deplasa prin portaluri, conservându-și în același timp impulsul. Acest lucru permite utilizarea unor manevre complexe pentru a traversa goluri largi sau pentru a efectua alte realizări pentru a ajunge la ieșire pentru fiecare cameră de testare. O serie de mecanici, cum ar fi lasere sau mașinării există pentru a ajuta sau a împiedica jucătorul să ajungă la ieșire.

2.2.2 Primele etape ale aplicației

Aceasta lucrare are ca obiectiv crearea unei aplicații software de tip joc video bazat pe îndeplinirea unor activități ținta într-un mediu 3D. Implementarea acestuia a fost posibila cu ajutorul tehnologiilor și conceptelor prezentate anterior. Astfel, are la baza motorul Unreal Engine 5 prin intermediul căruia sunt expuse utilizatorului aspecte vizuale, audio și input-uri.

2.2.2.1 Primele setări

Prima dată când pornim motorul grafic Unreal ni se cere să alegem ce fel de proiect dorim să începem. În cazul de fata, alegem „Games” și „Blank”, unde se deschide un proiect gol.

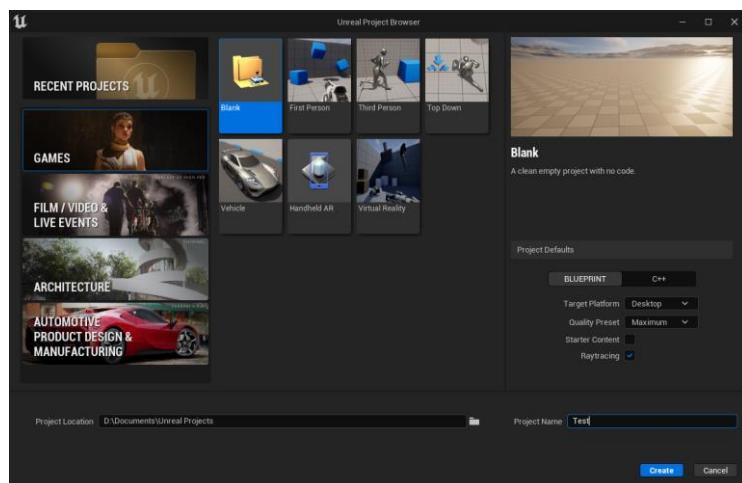


Figura 2.8 Interfața de pornire a motorului grafic

Când deschizi Unreal Engine 5, prima dată apare Editorul de nivel, ca în figura de mai jos.

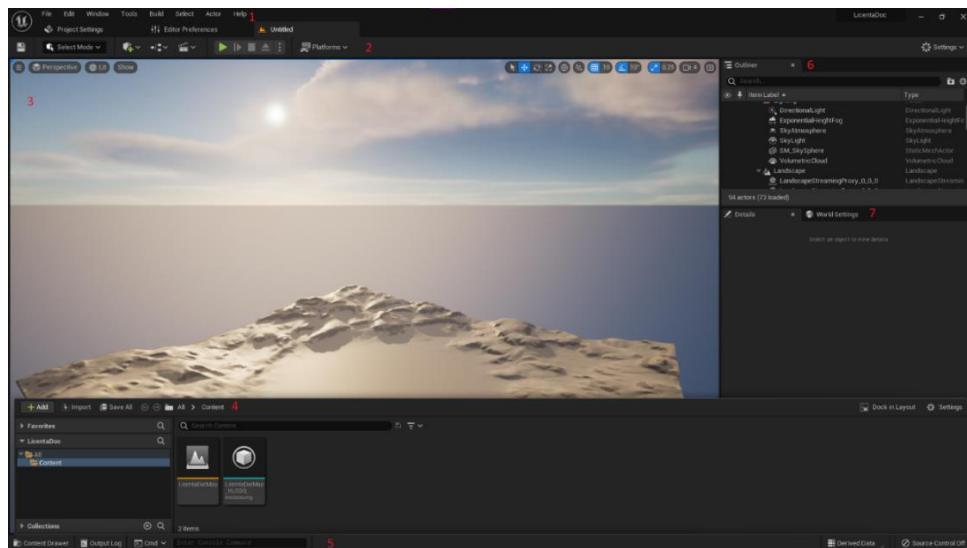


Figura 2.9 Interfața principală a motorului grafic

- 1) Bara de meniu, fiecare editor din Unreal Engine are o bară de meniu care se află în partea stânga sus a ferestrei editorului respectiv.

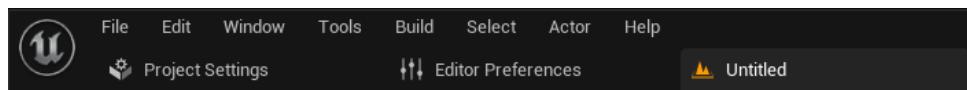


Figura 2.10 Bara de meniu

- 2) Bara de instrumente sus conține comenzi rapide la unele dintre cele mai utilizate instrumente și comenzi în Unreal Editor.



Figura 2.11 Bara de instrumente sus

- 3) „Level viewport” afișează conținutul nivelului care este deschis în prezent. Când deschideți un proiect în Unreal Engine, nivelul implicit al proiectului se deschide. Aici puteți vizualiza și edita conținutul nivelului activ, indiferent dacă este vorba despre un mediu de joc, o aplicație de cinema sau altceva.

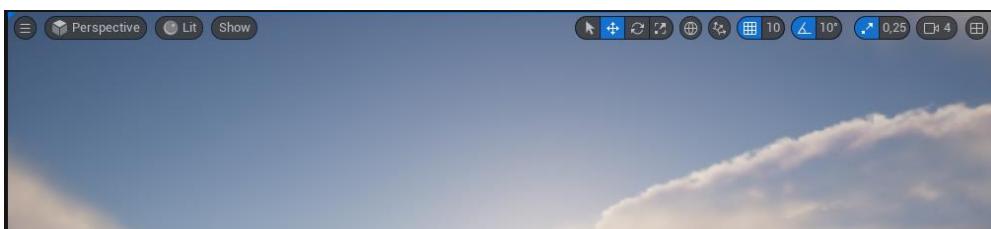


Figura 2.12 Nivelul Viewport

- 4) Bara de conținut este o fereastră de explorer de fișiere care afișează toate activele, blueprint-urile și alte fișiere conținute în proiect unde putem naviga printre acestea.

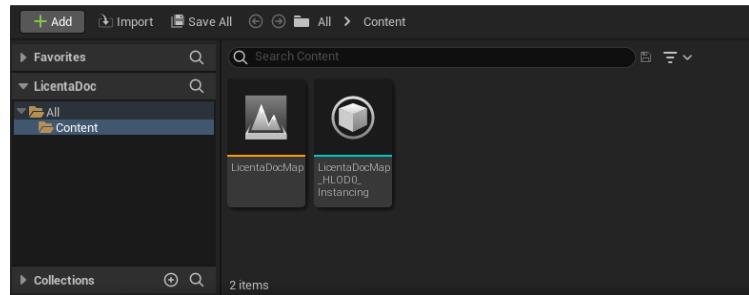


Figura 2.13 Bara de conținut

- 5) Bara de instrumente jos conține scurtături către consola de comandă, jurnal de ieșire, date derivate și sursa de control.



Figura 2.14 Bara de instrumente jos

- 6) „Outliner-ul” afișează o vizualizare ierarhică a întregului conținut din nivel.

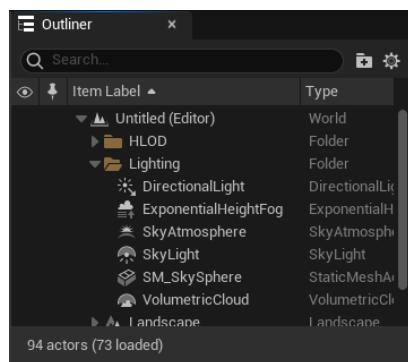


Figura 2.15 Oferă informații cu privire la obiectele adăugate în nivel

- 7) Panoul de detalii, atunci când selectam un actor în Level Viewport, panoul Detalii va afișa setările și proprietățile obiectului selectat.

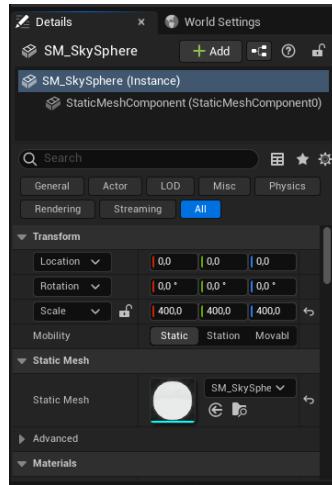


Figura 2.16. Panoul de detalii

Pentru a profita maxim de capacitatele calculatorului personal și de noile îmbunătățiri ale motorului grafic a necesar schimbarea unumitor setări. Prima dată intram în secțiunea de „Project Setting”, care se află în tab-ul „Edit”, unde putem observa diverse setări. În stânga căutam setarea „Rendering” care se află în partea de „Engine”. În „Rendering” selectăm la „Global Illumination”, „Dynamic Global Illumination Method” metoda „Lumen”, la fel procedăm și pentru „Reflections”, „Reflection Method”. Aceasta setare deblochează efectul de „Lumen”, un sistem de iluminare și reflexii globale complet dinamic. Activăm și suportul „Hardware și Software Ray Tracing” pentru a oferi o accelerare al algoritmilor de generare imagini.

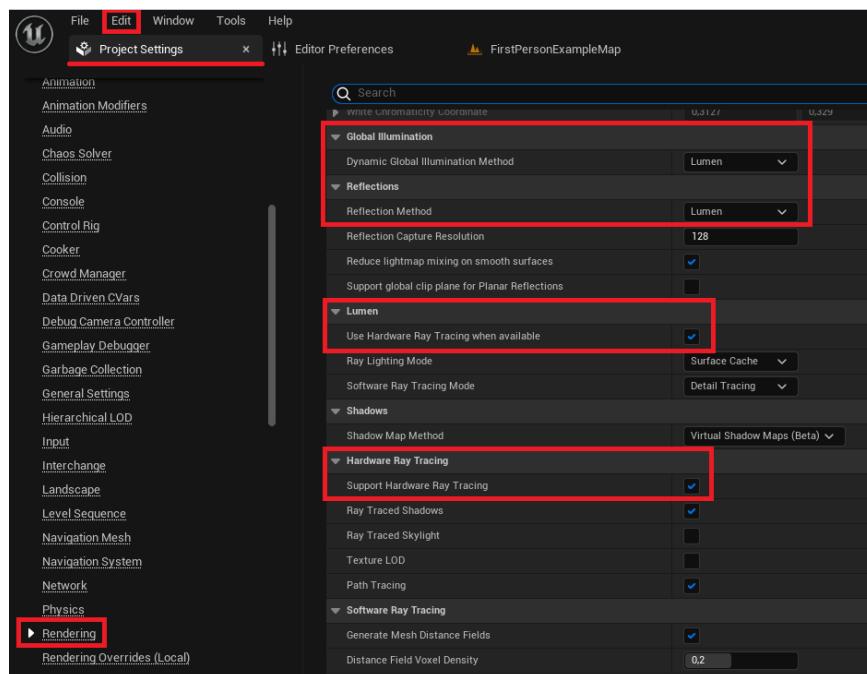


Figura 2.17 Primele setări pentru Editor

2.2.2.2 Realizarea nivelului

Pentru a avea diferite efecte ale cerului se vor adăuga următoarele efecte vizuale:

- Atmospheric Fog adaugă un efect de ceata când privești în depărtare
- ExponentialHeightFog controlează densitatea, direcția și înălțimea la ceata
- DirectionalLight simulează lumina care este emisă de la o sursă invizibilă.
- SkyLight captează părțile îndepărtate ale nivelului și aplică lumină scenei. Astă înseamnă că aspectul cerului și iluminarea sau reflexiile sale se vor potrivi, chiar dacă vine din atmosferă sau nori stratificați pe vârful unui munte.

Pentru a realiza o hartă se creează un teren drept, unde se adaugă și modifica diferite atribuții cu ajutorul modurilor „Landscape” și „Foliage”

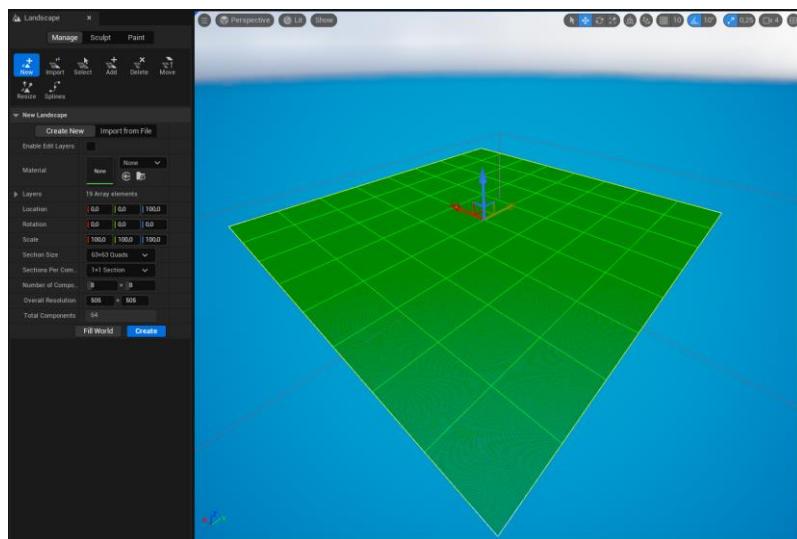


Figura 2.18 Se previzualizează suprafața hărții

La „Landscape”, „Manage” avem modulul de creare a unei parți de harta unde avem diferite setări pentru scara, locație, componente și rezoluții.

Modulul „Sculpt” implică utilizarea unei varietăți de instrumente care modifică harta. Aceste instrumente variază de la instrumentul simplu Sculpt care pictează valorile înălțimii folosind o scară de pensulă și rezistență, la multe alte instrumente care utilizează algoritmi complecși. Fiecare instrument are un set de proprietăți care determină modul în care instrumentul afectează peisajul. La fel se procedează și în cazul „Paint” unde selectăm stratul țintă.

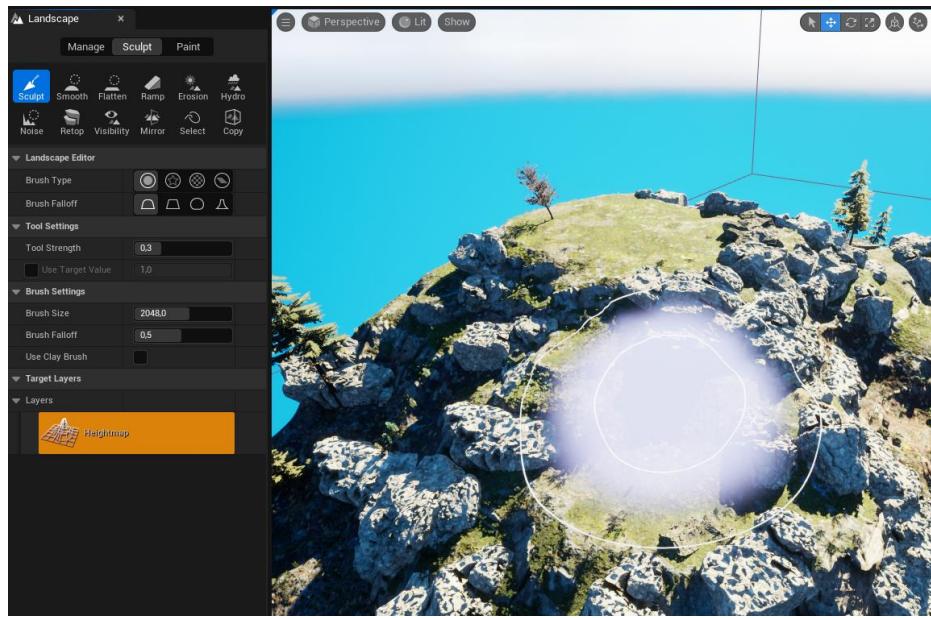


Figura 2.19. Modul landscape cu "Brush" tool.

În modul „Foliage” în acest caz pentru a nu pune manual diferite tipuri de copaci, pietre și iarba folosim un „Procedural Foliage Spawner” care va plasa automat obiectele alese. Îl plasam în mijloc și îl dimensionam cat să cuprindă marginile hartii. În tool-ul Foliage Spawner avem opțiunea Foliage Type, cu aceasta putem să selectăm toate obiectele ce dorim să fie adăugate pe harta. Obiectele precum plantele, copacii au fost împrumutate de la STF3D.

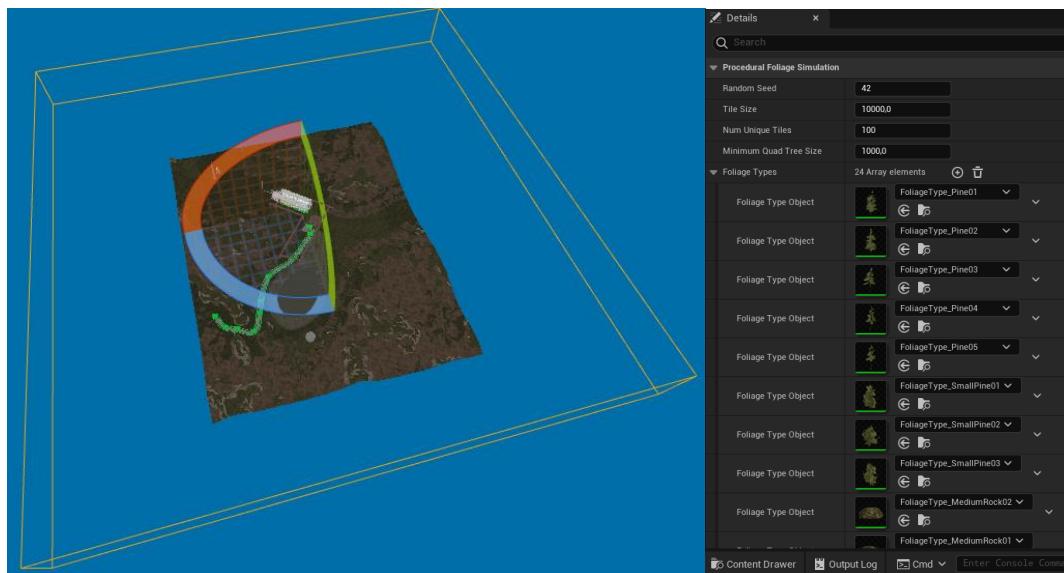


Figura 2.20 Modul Foliage privit de sus cu tipurile de obiecte alese

Cu ajutorul modului „Select” și a „Content Drawer” în care poți să navighezi printre fișierele tale ajuta la simplificarea plasării și a editării diferitelor elemente în nivel.

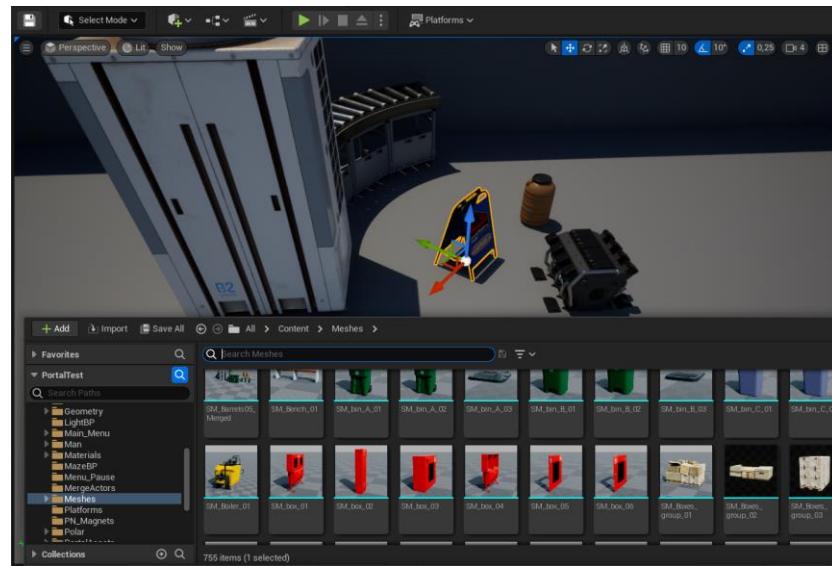


Figura 2.21 Exemplu de detaliere a nivelului

2.2.2.3 Realizarea caracterului

Pentru a avea un caracter care poate să se miște și să interacționeze cu lumea, va trebui să realizam input-uri care primește informațiile de la tastatura. În „Project Settings”, la „Engine”, „Input”, avem „Action Mappings” și „Axis Mappings” unde putem numele și tasta atribuită acestei acțiuni.

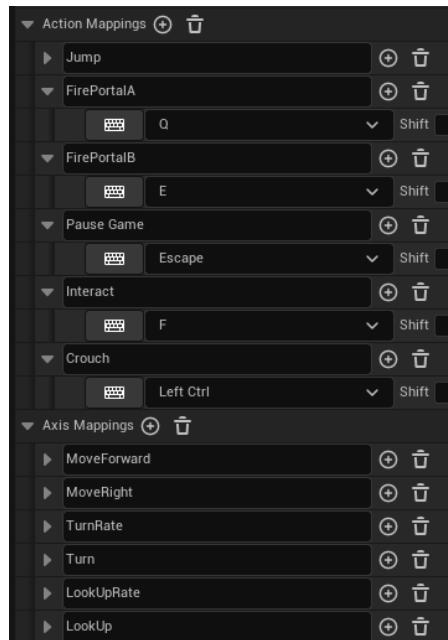


Figura 2.22 Intrările de la tastură utilizate

Caracterul va începe cu evenimente care sunt declanșate de la „PlayerController” și va folosi scriptarea pentru a controla caracterul. În cazul unui joc la persoana întâi, acesta este

adesea doar o pereche de brațe plutitoare, noi putem adăuga un personaj complet modificând în interiorul capsulei component mesh-ul caracterului, unde am pus personajul cu denumirea „SK_Man_Full_04” și la osul denumit „head” am lipit „FPCamera” pentru a avea o perspectivă apropiată nivelului lui. Camera a fost mutată mai în fata cu 22 de unități pe Y pentru a nu capta și fata în cazul mișcării caracterului.

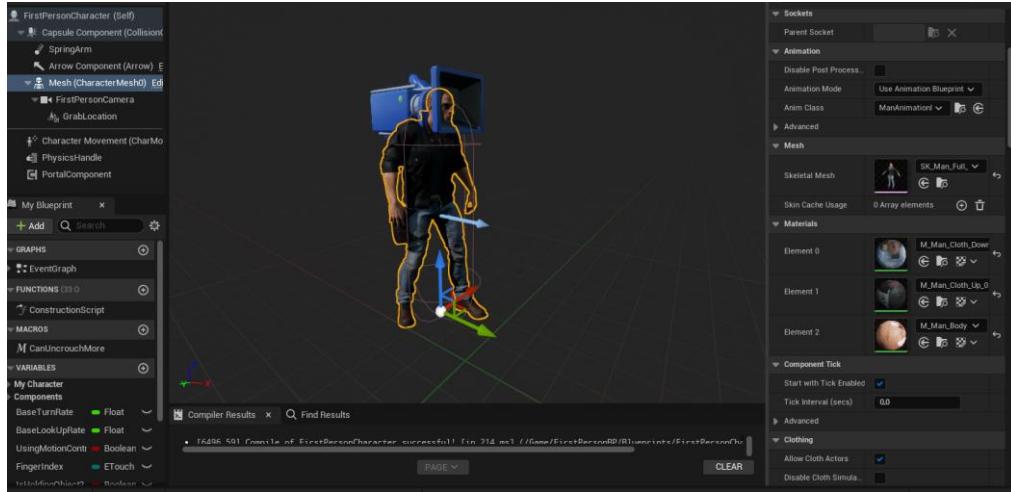


Figura 2.23 Blueprint-ul caracterului

Se creează un Event care citește de la tastatura un buton specific (în cazul de fata, la InputAxis MoveForward avem tastele W și S) și se executa comanda de mișcare în timpul apăsării butonului. Pentru ca personajul să meargă în direcția dorita se alege componentul camerei și se conectează la „Get Forward Vector”, după împreună cu „Get Actor Forward Vector” ce vor fi extrase de funcția Select unde urmează să se încheie în vectorul „World Direction” din Add Movement Input.

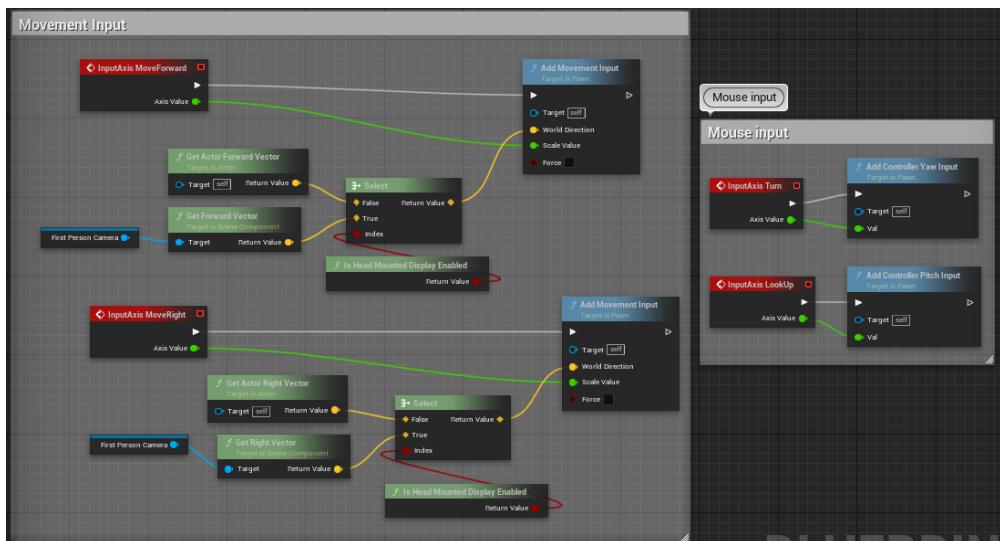


Figura 2.24 Prezentarea scriptului pentru mișcarea în spațiu al caracterului.

Pentru a activa „jump” și „crouch” la „Character Movement” în tab-ul „Components”, unde în dreapta la „Details” selectam „Movement Capabilities” ca „True”. În editorul event formam acțiunea acestora.

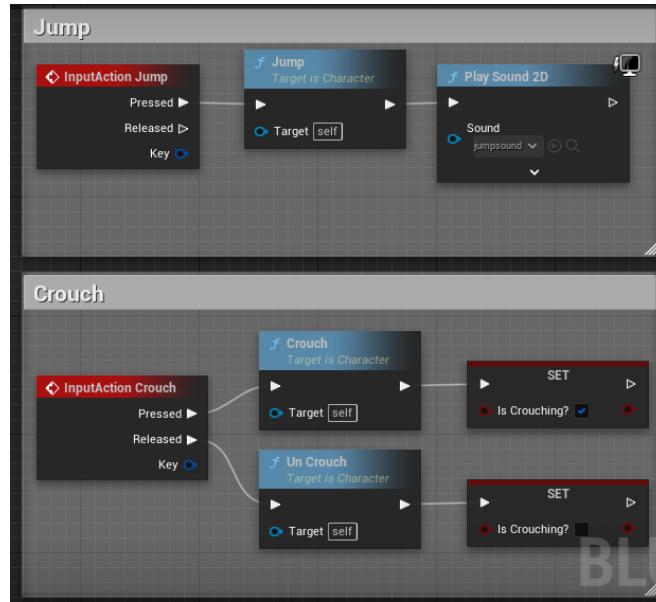


Figura 2.25 Comanda pentru săritura și mers ghemuit

Pentru ca personajul nostru să aibă o mișcare cat mai naturală vom implementa o animație de 8 cai de mișcare direcțională, săritura și mersul ghemuit.

Intr-un folder oarecare descarcam animațiile propuse și inițializăm un „Blend Space” pe care îl denumim ManAnimationBP.

Pentru mișările 8WayDirection se creează un alt Blend Space numit „8WayBS” separat unde folosim toate cele 8 animatii după viteza și directia miscarii. Am setat miscarea ca fiind de 600 de unitati și directia de 180 grade. Directia se află de la stanga cu valoarea -180 la dreapta cu +180 și viteza de la mijloc jos la 0 pana la valoarea 600 în sus.

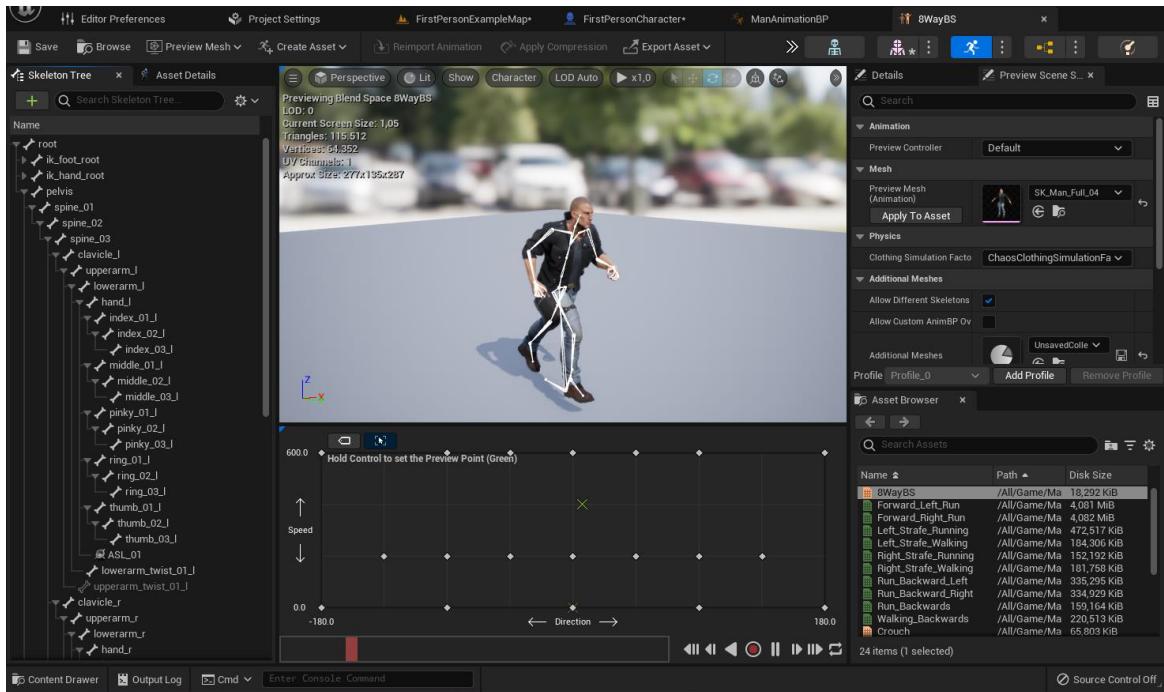


Figura 2.26 Viewportul animației

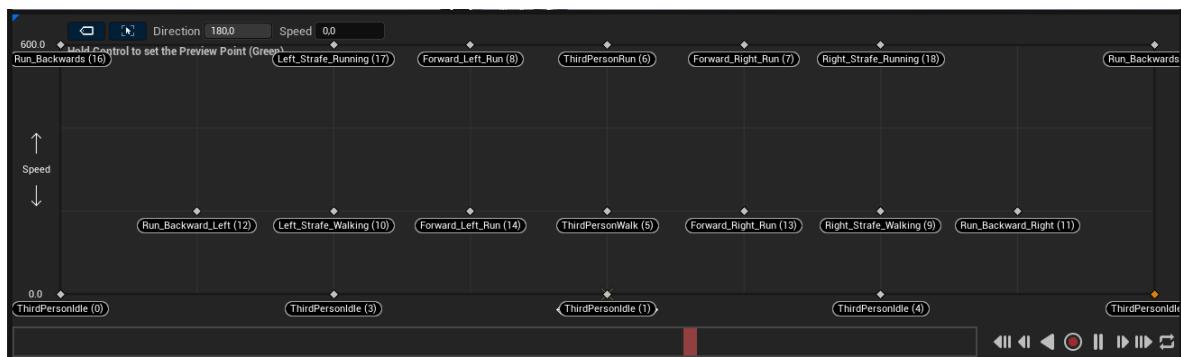


Figura 2.27 Punctele de animații setate pe direcția și viteza mișcării

Ca aceste animații să fie redate va trebui să mergem în ManAnimationBP la graficul animației, locomotion, Idle/Walk/Run ca să putem seta variabilele de viteza și direcție.

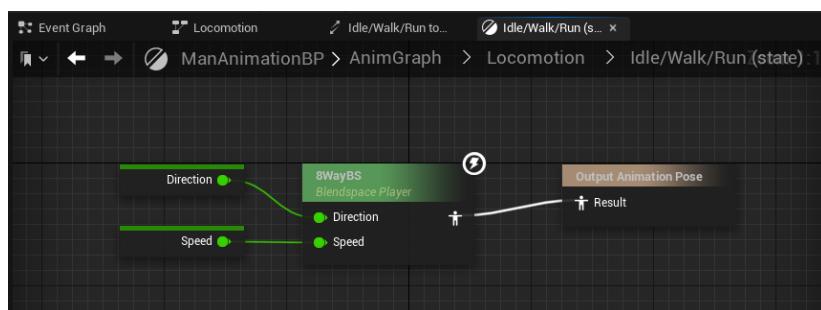


Figura 2.28 Setarea intrării valorilor în output-ul animației

La fel procedăm și la animațiile Crouch și Jump, unde setăm variabile booleane pentru a verifica dacă acțiunea a început sau a luat sfârșit.

În partea de jos se calculează dacă jucătorul a acționat butonul de „jump” sau „crouch” și în acel moment se setează valoarea variabilelor „Is Jumping” sau „Is Crouching” ca adevărat. Se calculează traiectoria și rotația caracterului pentru a calcula direcția în care merge acesta. În momentul în care locomoția detectează aceste intrări se va acționa animația atribuită.

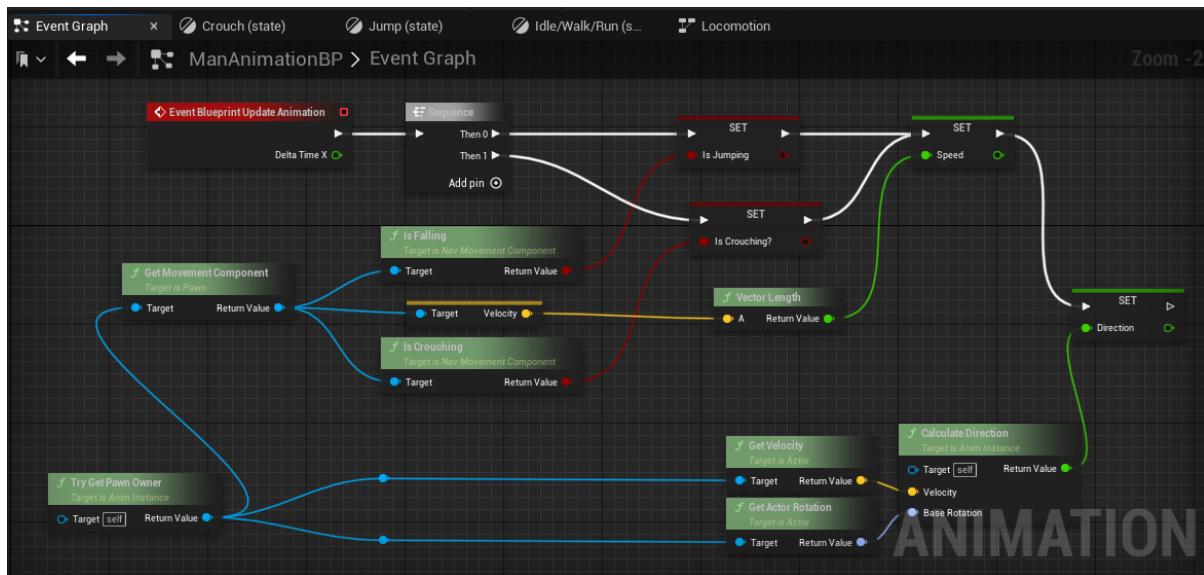


Figura 2.29 Funcția de detectare a mișcării

Intr-un final adăugăm un sistem dinamic de pași, pentru a reda un sunet specific când caracterul nostru merge pe o anumita suprafață. Se realizează un semnal sonor pe care îl putem utiliza ca un material fizic. Acest material este folosit pentru a defini răspunsul unui obiect fizic atunci când interacționează dinamic cu lumea. Se setează 2 notificări de sunet în animație, pentru ca sunetul să se activeze atunci când caracterul păsește cu piciorul pe o suprafață delimitată. În imaginea de mai jos este prezentat logica blueprintului de analiza a sunetului. Aceasta analiza redă un sunet predefinit în funcție de poziția caracterului din interiorul jocului. De exemplu, dacă caracterul se află deasupra unei platforme de metal și este în mișcare, blueprintul va folosi algoritmul de calcul pentru a determina sunetul necesar și va fi redat către jucător ca un sunet de pas pe un material solid din metal.

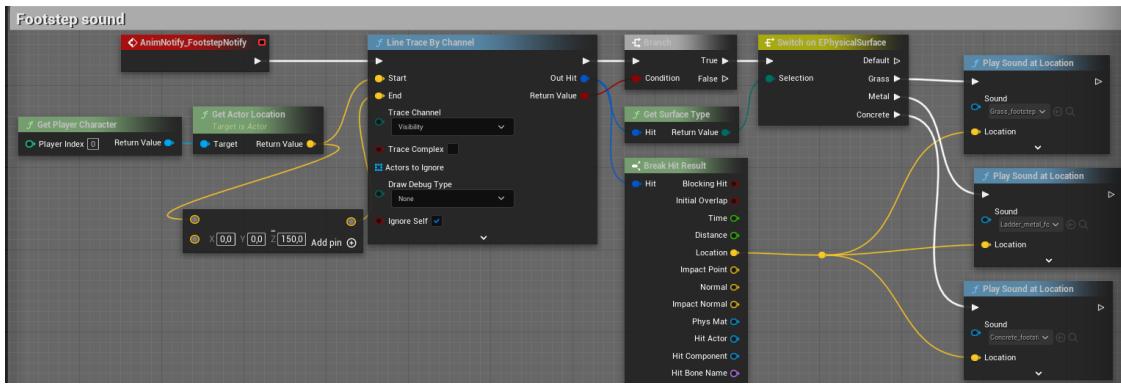


Figura 2.30 Redare sunet pentru fiecare pas

2.2.3 Sistem de scriptări folosite în joc

2.2.3.1 Ridicarea obiectelor

Ca personajul nostru să poată ridica obiecte specifice cu ajutorul unui buton realizam o confecțiune între evenul butonului și targetul caracterului. Se creează o variabilă „Is Holding Object?” care este falsă și se va schimba la activarea butonului.

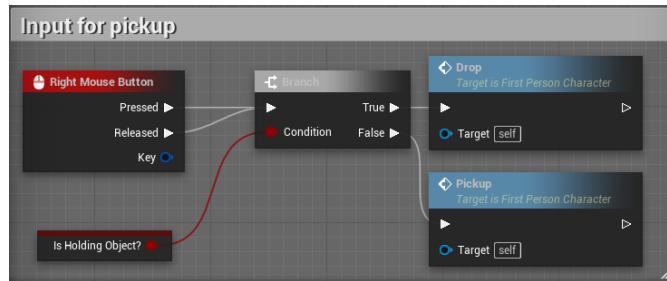


Figura 2.31 Activarea scriptului la buton

În prima parte se creează distanță de prindere care este la 500 de unități în față și centrul ecranului și regula de a tine anumite obiecte în „mână”. „Line trace for object” este folosit ca un depanator ca să verificam dacă actualul obiect este „atins”.

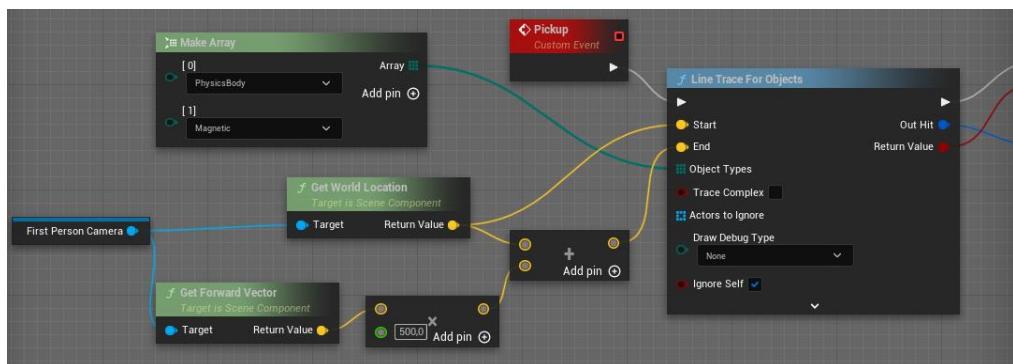


Figura 2.32 Scriptul pentru funcția de ridicare (1).

În partea a doua se verifica masa și tipul obiectului, să nu fie mai grea de 500 de unități și se pornește coliziunea acestuia cu caracterul.

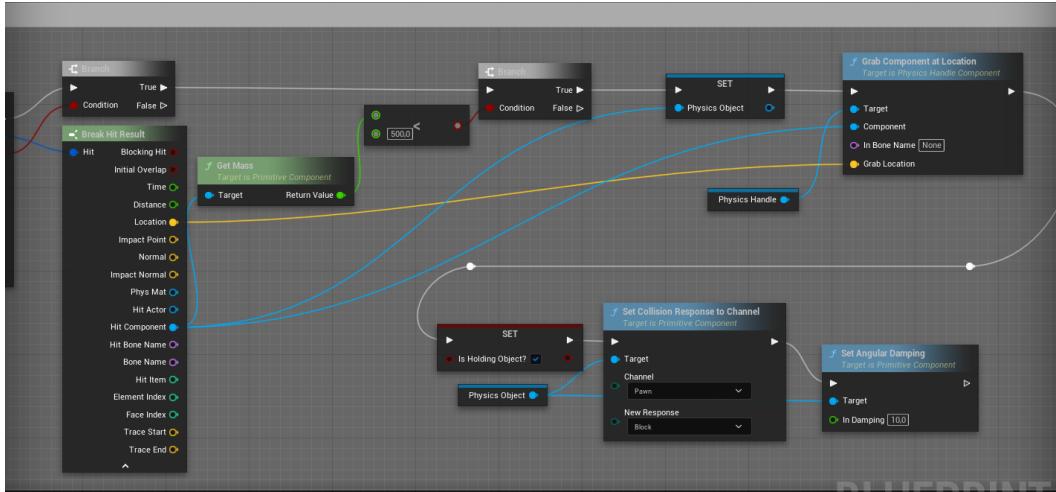


Figura 2.33 Scriptul pentru funcția de ridicare (2).

Partea a treia a regulii, se creează un Event Tick care pentru fiecare „frame per second” schimba poziția obiectului ca să nu rămână în urma.

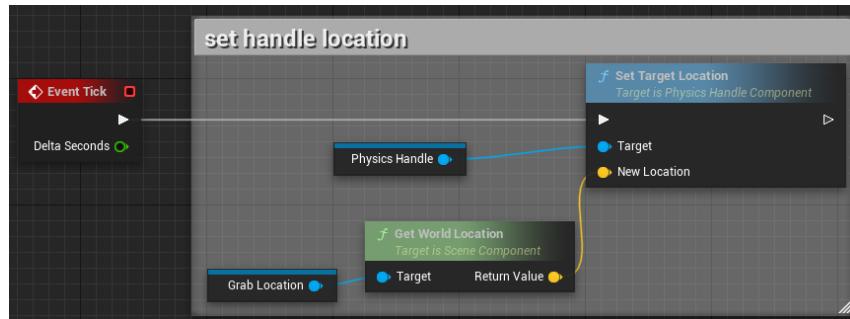


Figura 2.34 Poziția obiectului în timp real

Ultima parte reprezintă revenirea obiectului la setarea lui inițială și păstrarea momentului în timpul aruncării.

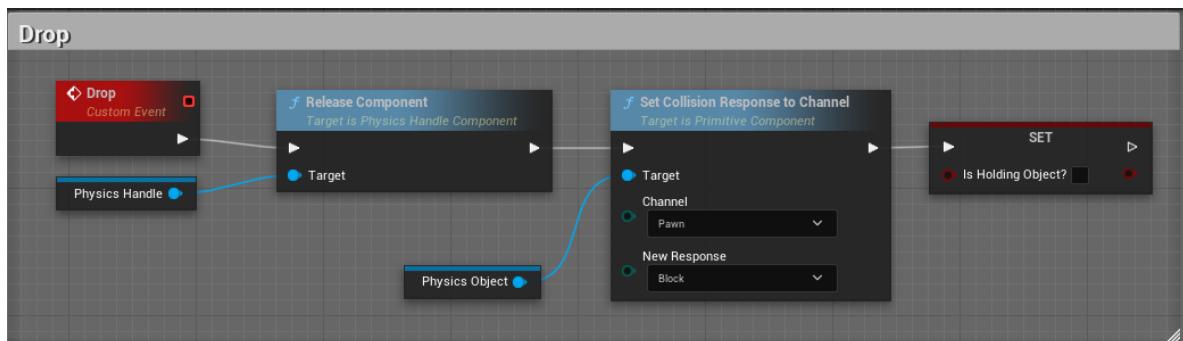


Figura 2.35 În timpul eliberării obiectul revine la starea să inițială "Block"

2.2.3.2 Interacțiunea cu mediul înconjurător

Interacțiunea personajului cu diferite obiecte se realizează prin utilizarea următoarei funcții: se setează input-ul butonului care este tasta „F” și se implementează într-o repetare a funcției care verifică dacă target-ul este actorul propus.

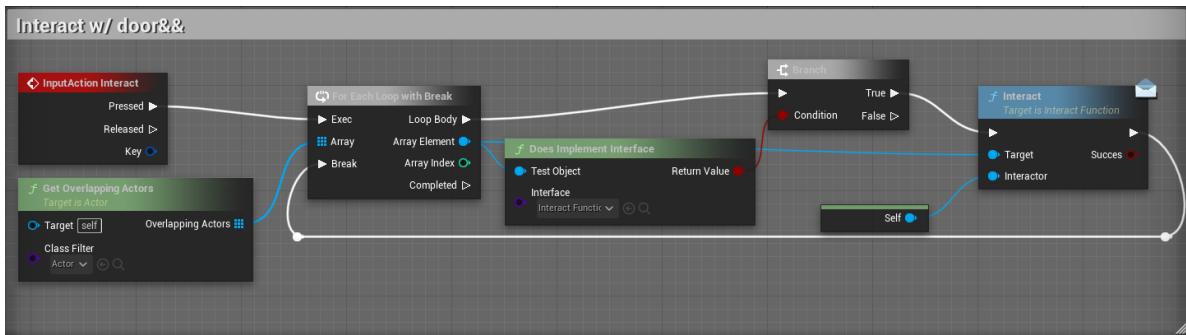


Figura 2.36 Scriptul pentru interacțiunea cu anumite obiecte

Aceasta comandă este folosită pentru a detecta dacă personajul se află în contact cu „collision box” în care ii permite interacțiunea cu obiectul respectiv. Se repeta și la alte blueprint-uri.

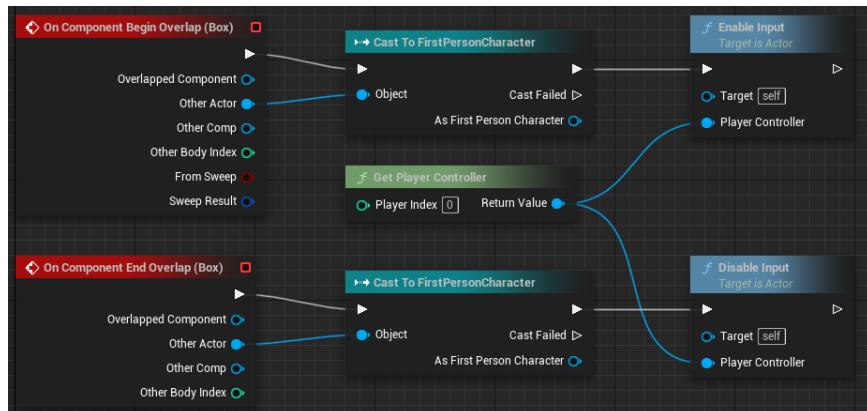


Figura 2.37 Contact cu caracterul jocului

Pentru a crea o simplă ușă care se deschide la interacțiunea jucătorului, este nevoie de un blueprint unde adăugăm obiectul și o cutie de coliziune ca în figura următoare.

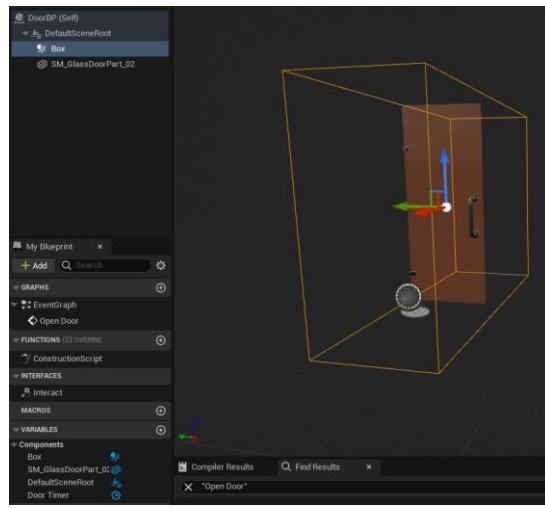


Figura 2.38 Plasarea obiectelor în viewport

În „Event Graph” se utilizează un eveniment personalizat numit „Open Door” care se leagă de un nod care alternează în acest caz animația ușii. În Door Timer setam o animație a ușii în care se deschide timp de o secundă, la 90 de grade de poziția inițială. Se încheie cu nodul „Set Relative Rotation” care are ca target obiectul și rotirea în axul Z final.

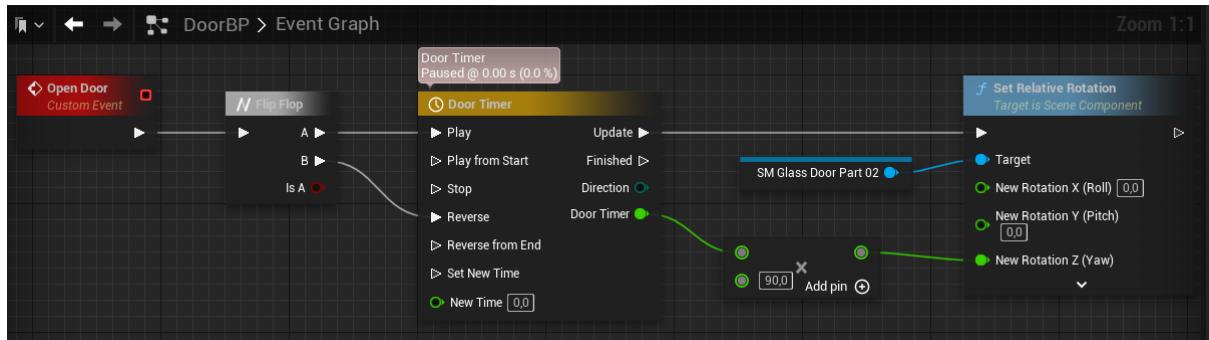


Figura 2.39 Activarea animației de deschidere a ușii

2.2.3.3 Obiecte de colecționat

Pentru „CollectibleBP” se procedează în viewport ca la blueprintul anterior, un obiect și o cutie de coliziune. În fereastra event aceasta funcție adaugă în „viewport” un widget clasa de obiecte, unde calculează cate obiecte mai sunt de colecționat. La găsirea unui obiect, după interacțiune, acesta dispără.

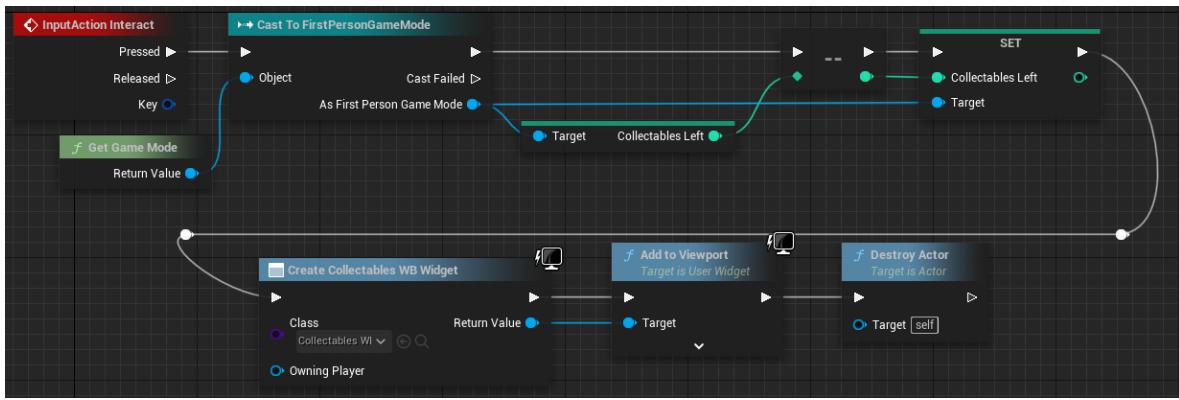


Figura 2.40 Plasarea obiectelor în nivel

Se creează clasa widget unde în „Designer” realizam un text: „X/Y Collected”, unde în modul „Graph” realizam o funcție „Get Text 0” unde se calculează cate obiecte sunt în lume și cate au fost colecționate.

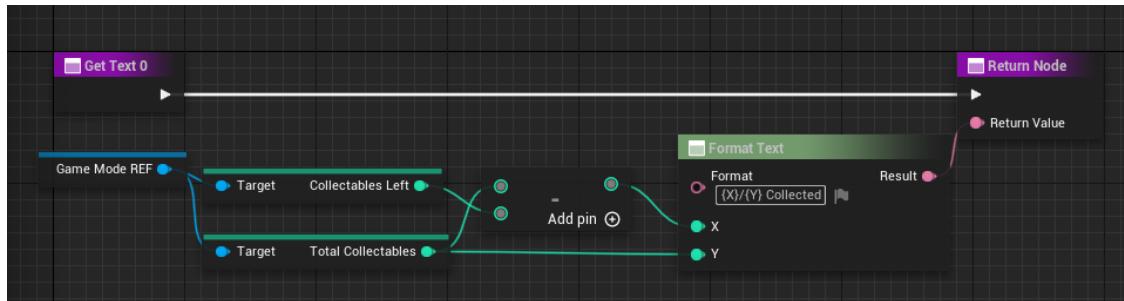


Figura 2.41 Calcularea obiectelor găsite și ramase în nivel

În aceasta parte când un obiect este selecționat, va apărea și dispărea un text animat care prezintă numărul obiectelor și cate au rămas, în momentul în care ai colecționat toate obiectele se va reda un sunet aleatoriu.

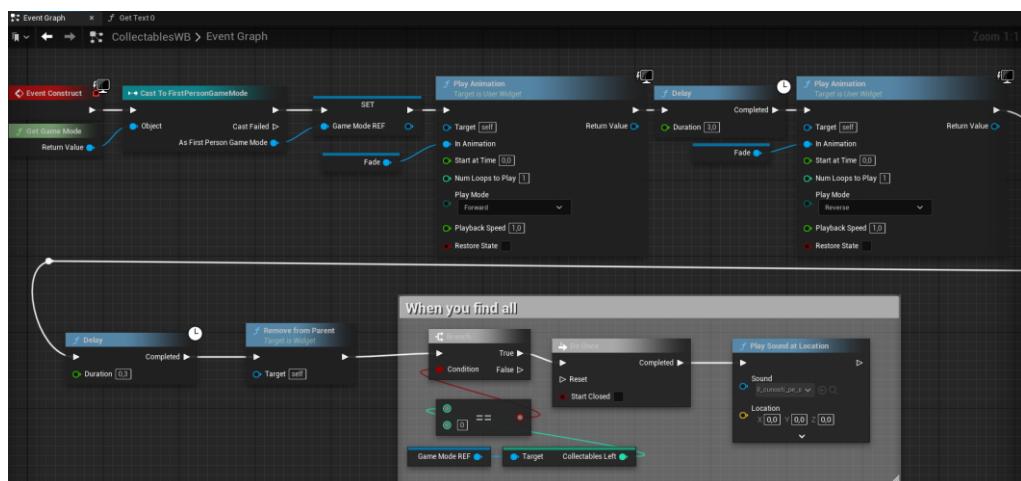


Figura 2.42 Animația textului în momentul găsirii obiectului

2.2.3.4 Lumini care se activează cu ajutorul butoanelor

Pentru a activa luminile cu ajutorul butoanelor, prima data cream un „BlueprintInterface” cu funcția de „ToggleLight” ca să existe o comunicare intre cele doua blueprint-uri.

În viewportul „RemoteLightBP” se va adăuga o componenta a luminii numita „SpotLight”. Se creează o variabilă Integer denumita „Light Index” folosita ca să ii atribuie o comunicare intre mai multe lumini cu diferite butoane. O alta variabilă booleană este „IsOn?” folosita pentru ca lumina să fie deja stinsă sau aprinsă, în nivel, la alegerea editorului.

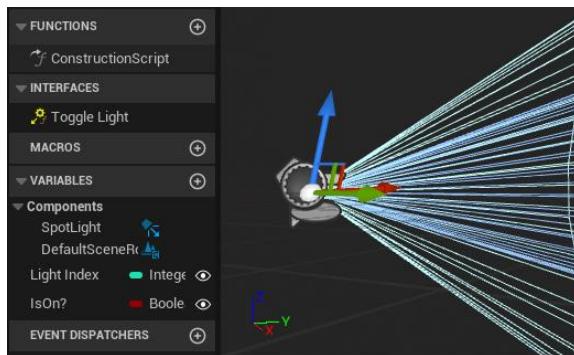


Figura 2.43 Viewportul de creare a lumini

Pe parte de „Construction Script” se setează construcția automată a luminii și vizibilitatea acestuia.

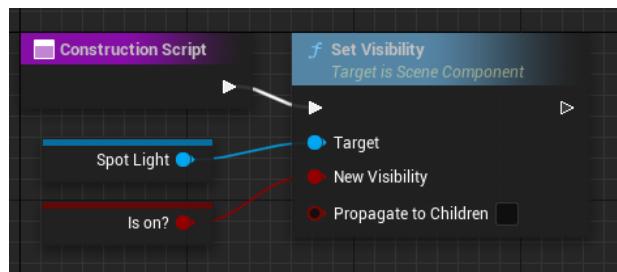


Figura 2.44 Script pentru setarea luminii în nivel

Funcția prezentată mai jos se implementează la evenul „Toggle Light”, în acest caz este apelata funcția „Set Visibility”, valoarea inițială fiind inversată.

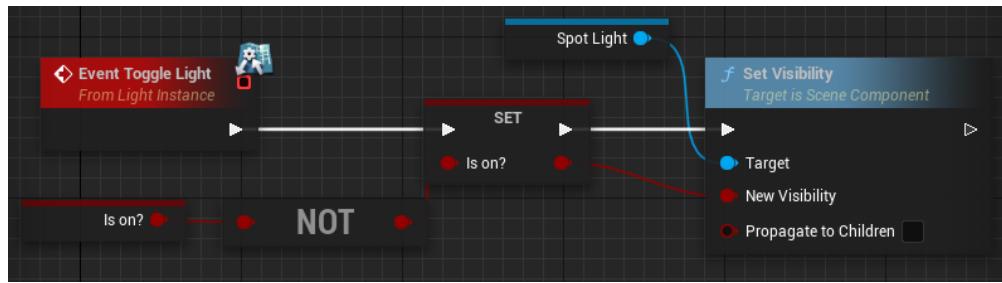


Figura 2.45 Script pentru inversarea valorii inițiale

Blueprint-ul butonului este asemănător cu cel al ușii, avem un obiect și o cutie de coliziune. Aceasta linie obține toate luminile din interiorul jocului și le adăuga intr-un array al butonului.

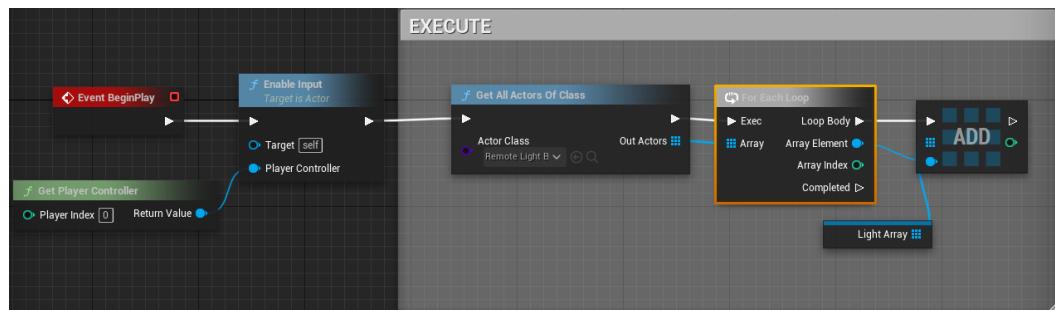


Figura 2.46 Inregistreaza luminile din nivel intr-un array

În stânga avem contactul cu caracterul jocului prezentat mai sus. La interacțiunea cu butonul se va reda un sunet și se vor activa luminile țintă ale acestuia.

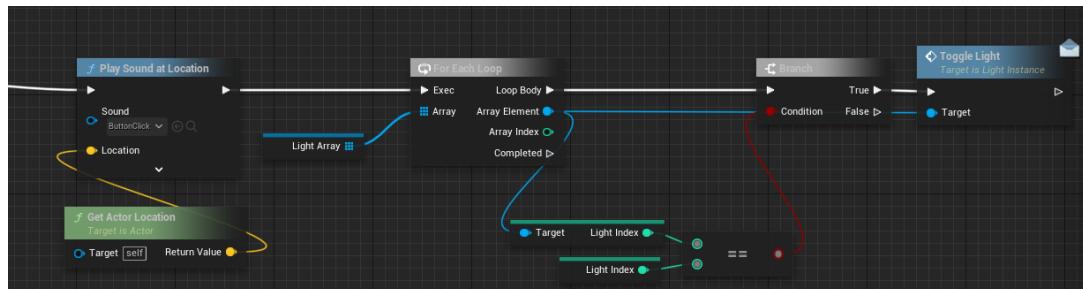


Figura 2.47 Funcția de redare sunet și activarea luminilor

În imaginea de mai jos s-a pus un buton și 2 becuri care au indexul comun butonului. Un bec este oprit din setare și celălalt este activat de buton. Daca se apasă încă o dată butonul se inversează valorile și becul din stânga va deveni cel aprins.

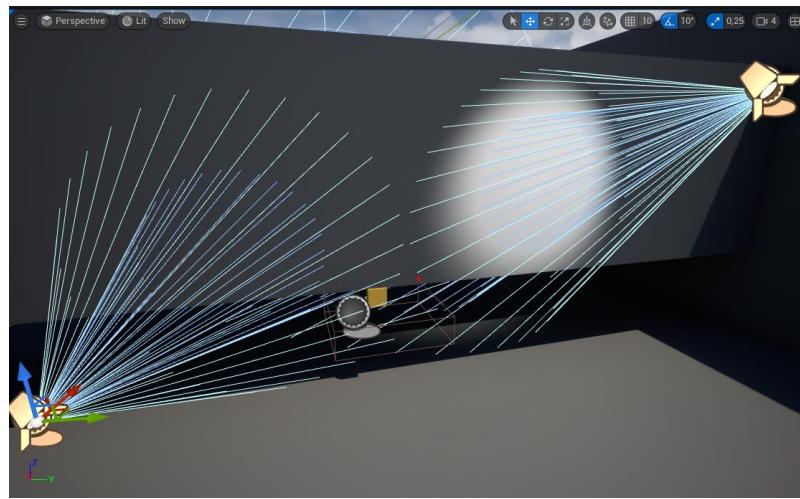


Figura 2.48 Rularea blueprint-ului

2.2.3.5 Realizarea magnetilor

Pentru a realiza un obiect care atrage și respinge obiecte după culoarea materialului, folosim un blueprint în care avem un obiect tip magnetic și o sferă de coliziune. Sferă de coliziune va fi folosita pentru a detecta ce obiect și material intra în contact cu aceasta și verifică dacă este o forță de atracție sau respingere.

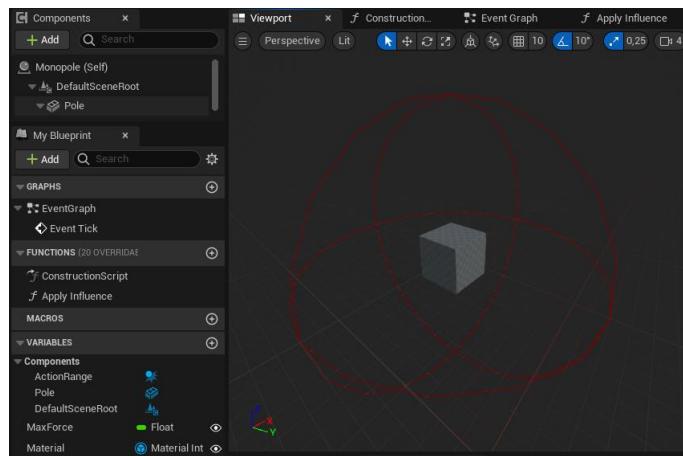


Figura 2.49 Viewportul Magnetului

Acest construction script este folosit pentru a seta culoarea materialul și polul respectiv la funcția materialului.

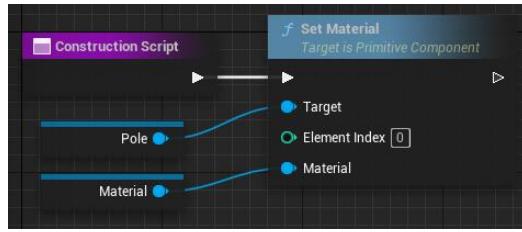


Figura 2.50 Setarea pentru functia materialului

Aplicam o funcție numita „Apply Influence” care prezinta următoarele parametrii: un pol magnetic, raza de acțiune și raza maxima de atracție a sferei.

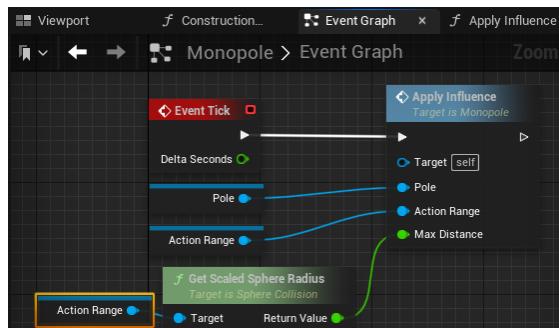


Figura 2.51 Functia Apply Influence

În funcția „Apply Influence” se creează patru variabile și anume: „L_Pole”, „L_ActionRange”, „L_ForeginPole” și „L_MaxDistance”. Se setează intrările pentru pol, distanța și raza ca să verifice în fiecare loop daca obiectul contactat este static sau exista o forță de atracție.

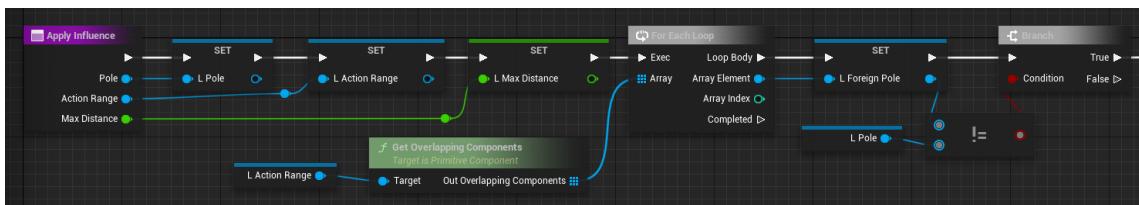


Figura 2.52 Verificare contact material

În chenarul de jos se calculează raza sferei și forța de atracție a obiectului, cu cat creste în volum implicit creste și puterea. Se calculează unde este localizat centrul obiectului folosit pentru a se roti necontrolat. Al doilea chenar reprezintă funcția de respingere a obiectului, unde se inversează forța de atracție. Daca materialul este diferit se activează forța de atracție al obiectului, daca este la fel, forța de respingere.

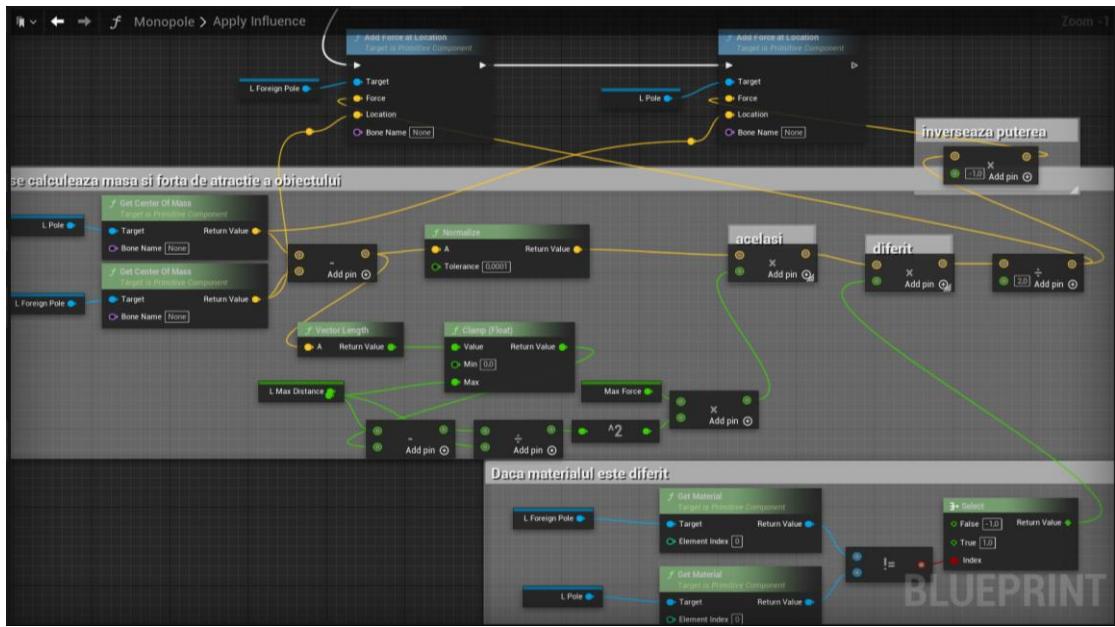


Figura 2.53 Calculare raza și forța magnetului

2.2.3.6 Sequence

„Sequencer” oferă posibilitatea de a crea cinematică în joc prin intermediul editorului său specializat multi-track. Prin crearea de secvențe de nivel, adăugarea de piese, și crearea de cadre cheie putem manipula diverse obiecte, caractere și camere.

Prima data se creează instanța „MyLevelSequence” și aducem câteva modificări asupra acestuia. Ne asigurăm ca înregistrarea se va face la 30 de cadre pe secunda și timpul editorului setat în secunde pentru a ne fi mai ușor să observăm în ce timp se întâmplă acțiunea.

Cu ajutorul butonului „Track”, situat în stânga editorului, putem adăuga actori, sunete și diferite camere. În prima secțiune, „Camera Cuts”, se afișează diferite secvențe statice de cinematic. În „Audio” putem selecta anumite sunete să fie redate la diferite intervale de timp. Cu ajutorul lui „Cine Camera Actor” putem să alegem diferite setări ale camerei cum ar fi: rezoluția camerei, dimensiunea lentilei (zoom), focusul pe un anumit obiect. Pentru a modifica mișcările camerei, în viewport se setează diferite locații cu ajutorul acelor cadre cheie. Deci avem locația inițială la 0 și alta la 4.5 secunde și se calculează o trajectorie invizibila a camerei. Se adaugă un actor numit „RobotArm” în care poți să-i modifici mișcările după preferințe și se salvează cu ajutorul cadrelor cheie.

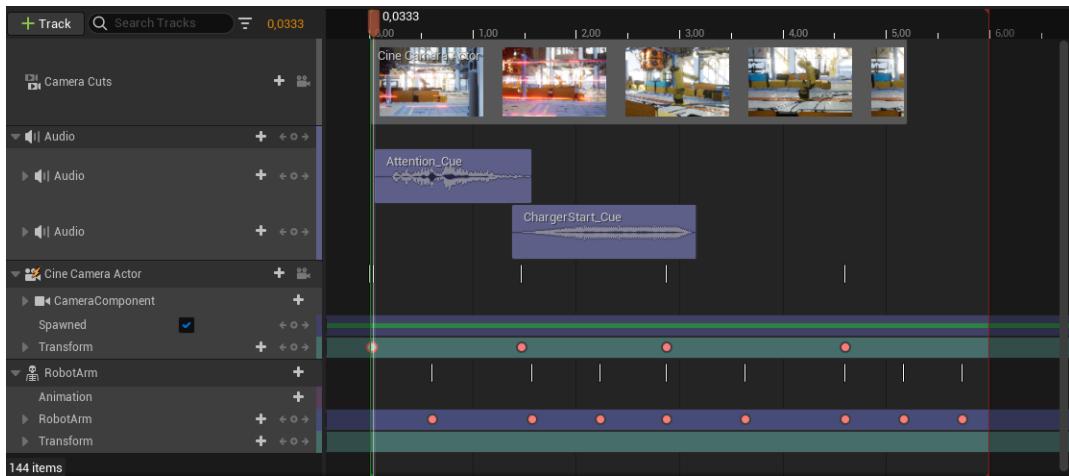


Figura 2.54 Exemplu de editare în cinematic

Pentru a apela acest cinematic se realizează o funcție prin care se setează variabila să fie animația. După care prin activarea evenimentului se va porni animația o singura data.

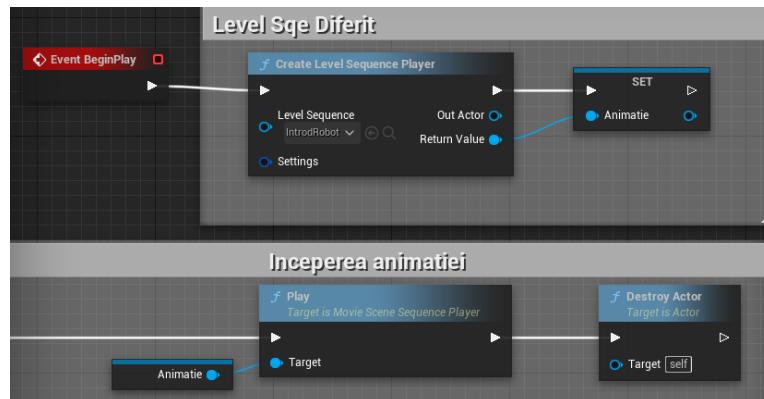


Figura 2.55 Scriptul de activare a cinematicului

2.2.3.7 Zona de activare a sunetului

Crearea unei zone de activare pentru a reda sunete în ordine se configurează folosind un actor de blueprint pe care îl poți amplasa de mai multe ori oriunde în nivel. Acesta va reda sunetul setat la acel declanșator, îl va dezactiva pentru a nu fi reîncercat și va activa următoarea zona care poate fi orice alt declanșator de sunet din nivel.

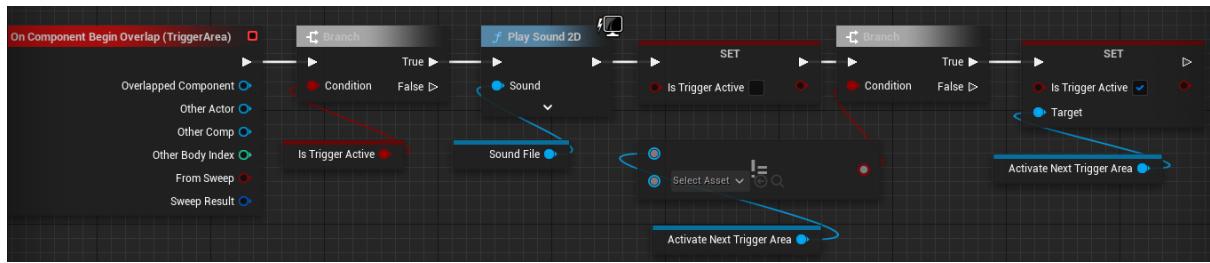


Figura 2.56 Scriptul de a succeda declanșarea sunetelor

2.2.3.8 Plasarea unui obiect pe o anumita direcție

Pentru a realiza un obiect care se mișcă pe o anumita direcție și viteza, avem nevoie de două blueprint-uri, una pentru obiect și cealaltă pentru direcție.

Primul blueprint este un actor numit „SplinePathBP”, acesta are rolul de controla calea obiectului. În interiorul actorului avem un sistem liniar denumit „Spline” care creează o caracteristica liniara. În al doilea blueprint situam un obiect pe care dorim să urmărească acea cale de la începutul acțiunii.

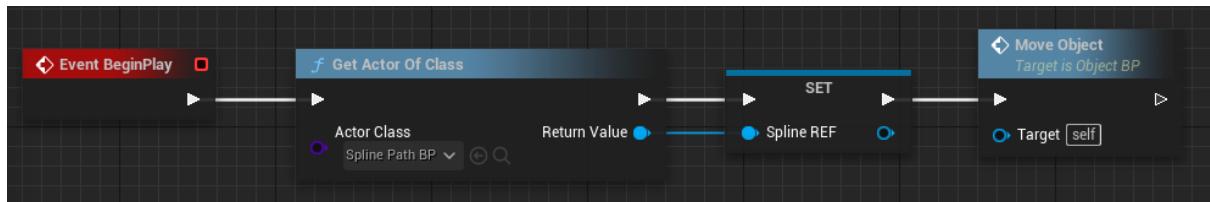


Figura 2.57 Spline Reference

Se stabilește timpul ciclului liniei și dacă se ajunge la sfârșit, imediat va apărea de la început noul obiect. Se actualizează constant rotația și locația obiectului pe acea linie, ca să nu rămână numai într-o singura poziție.

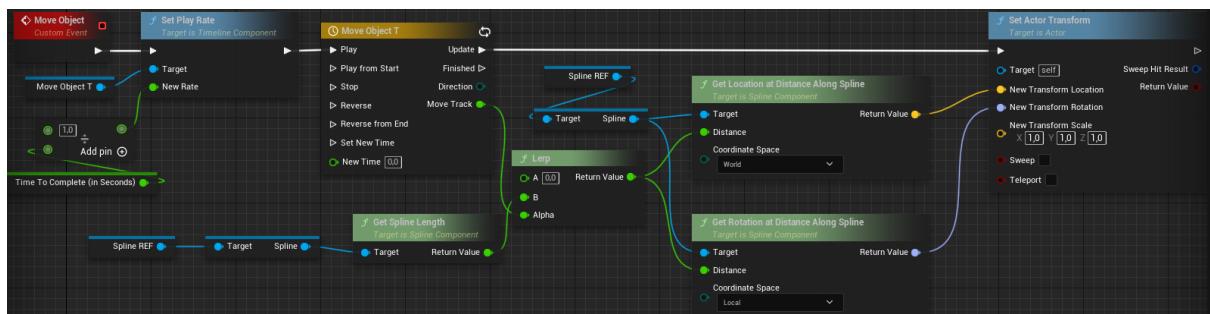


Figura 2.58 Scriptul pentru actualizarea constantă a obiectului pe acea linie

2.2.3.9 Realizarea portalelor

Datorita algoritmilor complecși din spatele procesării grafice afișate ca două portale, am creat posibilitatea de a călătorii în spațiul vizibil din nivelul curent. Aceasta mecanică adaugă o nota de aventură și perspectiva nouă asupra nivelului liniar.

Pentru a putea forma portalele și a interacționa cu ele avem nevoie de anumite seturi de reguli. Un perete special făcut pentru a putea forma portalul pe el, permiterea caracterului să se miște prin ele, să nu se creeze mai mult de două intrări.

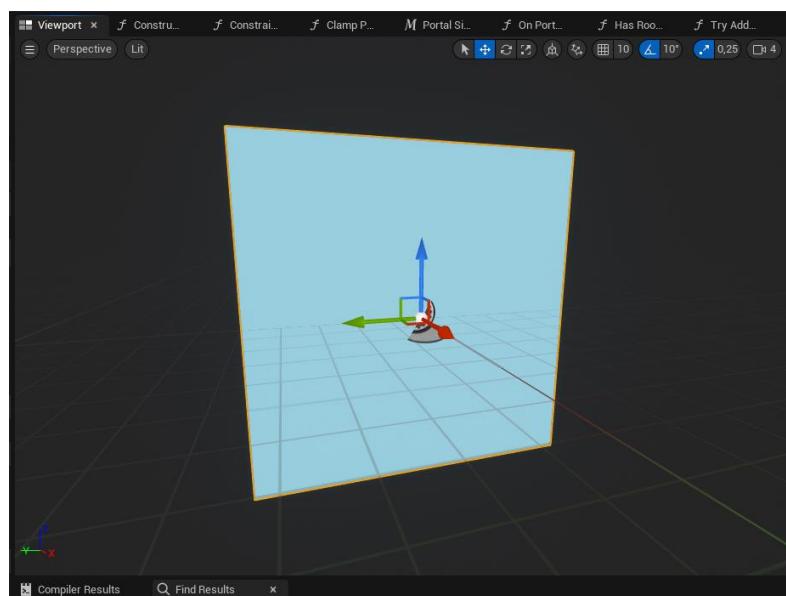


Figura 2.59 Peretele portalului

Acest obiect este peretele făcut special pentru a permite formarea portalelor asupra lui. Urmărește un set de reguli precum setarea lungimii și înălțimii pentru a nu permite cercului de la portal să depășească suprafața lui, detectează dacă portalul a fost activat și formează un orificiu pentru a permite trecerea personajului. Detectează și nu permite suprapunerea portalelor, verifică dacă există suficient spațiu în zona cea mai apropiată și va crea celălalt portal.

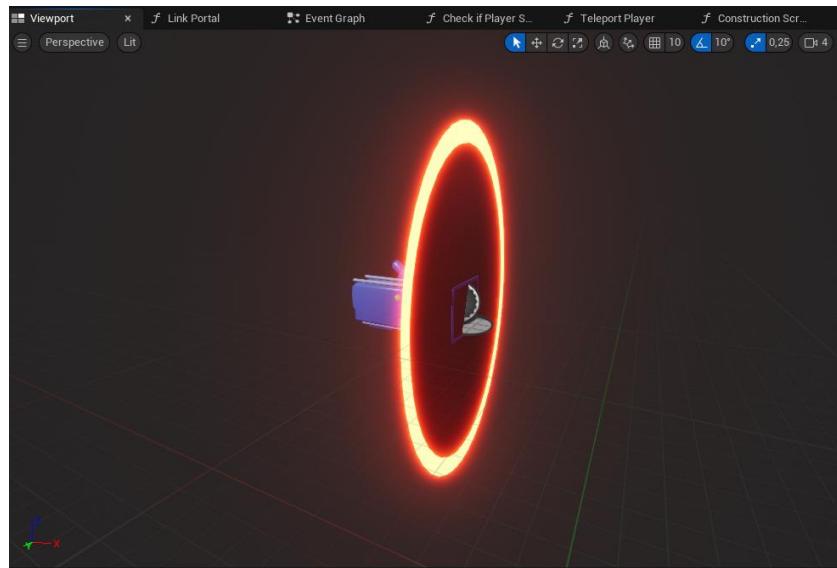


Figura 2.60 Portalul propriu-zis

Blueprint-ul la portal cuprinde mai multe caracteristici. În primul rând avem un inel de cerc care își schimba culoarea când apare celălalt portal. O camera care capturează scena în 2D și imprimă pe planul portalului ce perspectiva are caracterul când privește prin acesta. Planul portalului verifică dacă personajul s-a apropiat suficient de mult încât să-l transporte la celălalt capăt al portalului.

Redarea imaginii 2D prin portal a avut anumite probleme deoarece se redă la 1920 pe 1080 indiferent de rezoluția reală a ecranului, ceea ce făcea ca imaginea din portal să fie distorsionată. Realizam o funcție în C++ care forțează rezoluția portalului să urmărească pe cea a ecranului.

```
#include "CoreMinimal.h"
#include "Kismet/BlueprintFunctionLibrary.h"
#include "PortalFuntionLibrary.generated.h"

UCLASS()
class PORTALTEST_API UPortalFuntionLibrary : public UBlueprintFunctionLibrary
{
    GENERATED_BODY()

public:
    UFUNCTION(BlueprintCallable, Category = "PortalFuntionLibrary")
    static void ResizeRenderTarget(UTextureRenderTarget2D* render_target, float size_x, float size_y);
};
```

Figura 2.61 Funcția de chemarea rezoluției imaginii 2D

```

#include "PortalFunctionLibrary.h"
#include "Engine/TextureRenderTarget2D.h"

void UPortalFunctionLibrary::ResizeRenderTarget(UTextureRenderTarget2D* render_target, float size_x, float size_y)
{
    if (render_target == nullptr)
        return;

    render_target->ResizeTarget(size_x, size_y);
}

```

Figura 2.62 Funcția de actualizare a rezoluției imaginii 2D

Funcțiile definite mai sus sunt folosite pentru a actualiza dinamic lungimea și înălțimea actorului „BP_Portal” cu ajutorul algoritmilor de comparație returnând valorile sub forma de integer.

2.2.3.10 Mișcarea robotului

În următoarea parte vorbim despre modificarea scheletului unui robot care poate fi utilizat în cinematic. Prima dată, pentru a nu modifica setările de baza, selectăm scheletul acestuia și realizăm asupra lui un „control rig” denumit „RobotArm”.

În „RobotArm” realizam controalele specifice fiecărui os în parte și anume îl forțăm să nu aibă o alta mișcare decât cea aleasă. Se creează o ierarhie pentru controlul 2D pentru a verifica dacă acesta are mobilitatea adecvată într-o direcție definită. Se folosește un alt control la capătul scheletului numi „Robot1_End” pentru a-l observa cum se comportă scheletul acestuia când urmărește punctul final.

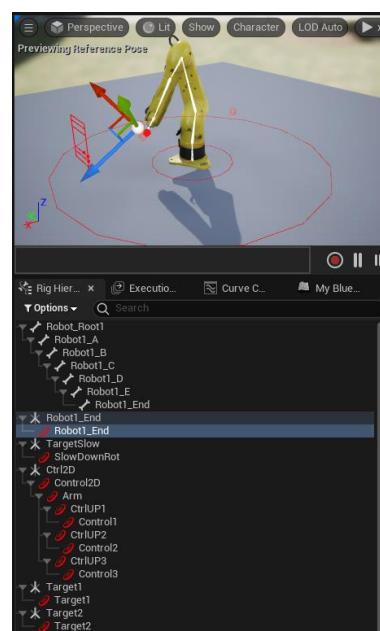


Figura 2.63 Setarea ierarhiei de control al robotului

În panoul de control al brațului robotic se va realiza scripturi pentru poziția inițială și poziția finală, reguli de mișcare și rotire pentru fiecare componentă a robotului, inclusiv limitele gradelor de libertate ale acestora. În exemplul de mai jos este prezentată funcția de ancorare a brațului de la baza robotului acestuia, însă având posibilitatea de a se rota liber în funcție de direcția de la capătul brațului.

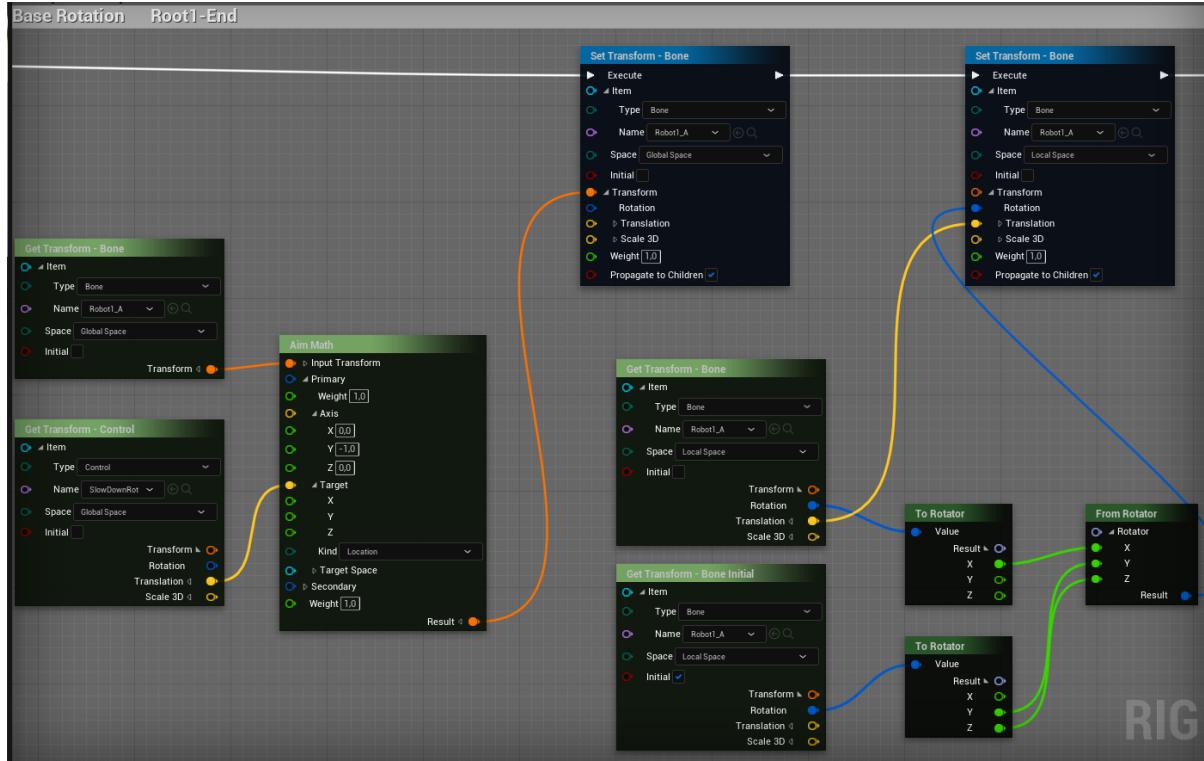


Figura 2.64 Setarea bazei pentru a urmări capătul final al robotului

În figura de mai jos este prezentată o funcție care creează gradul de libertate al robotului, ii permită să se miște pe o distanță între 150 și 500 de unități, pentru a nu se încrucișă vârful robotului cu baza acestuia.

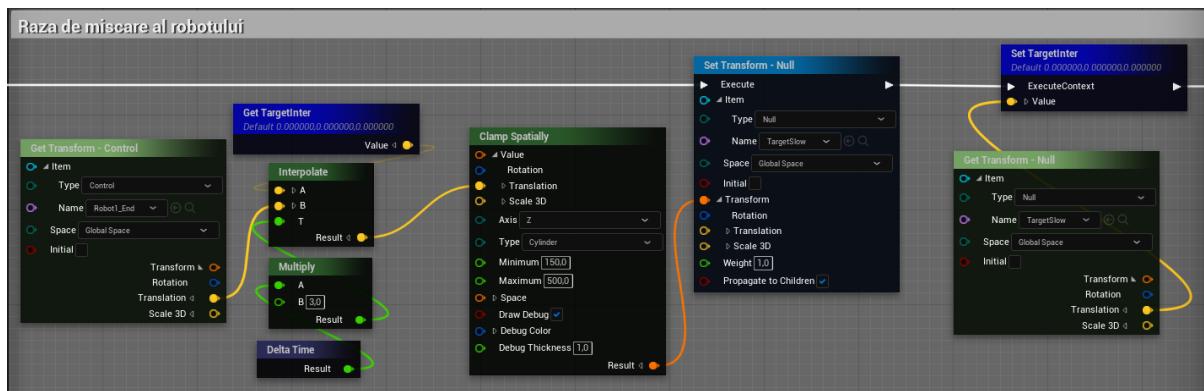


Figura 2.65 Gradul de libertate

2.2.3.11 Cadran

Este un puzzle cu mecanism similar lacătelor folosind o interfață de ceas ce, pentru a putea fi deschis, necesita ca săgeata din interiorul cadranului să se potrivească cu poziția prestabilită. Fiecare secțiune alba are o valoare de la 0 la 7 și pentru a selecta la ce număr se activează, setam asupra săgețăii un integer de la 0 la 7. Cadranul se poate multiplica în nivel pentru a fi asemănător unui cifru.

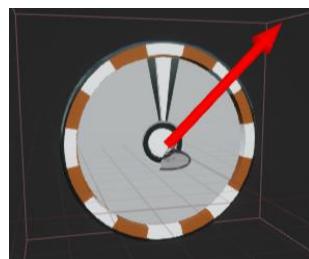


Figura 2.66 Cadranul folosit în joc

Figura de mai jos reprezintă scriptul de sunet și de calculare a mișcării săgeții la cate 45 de grade fata de situația curentă.

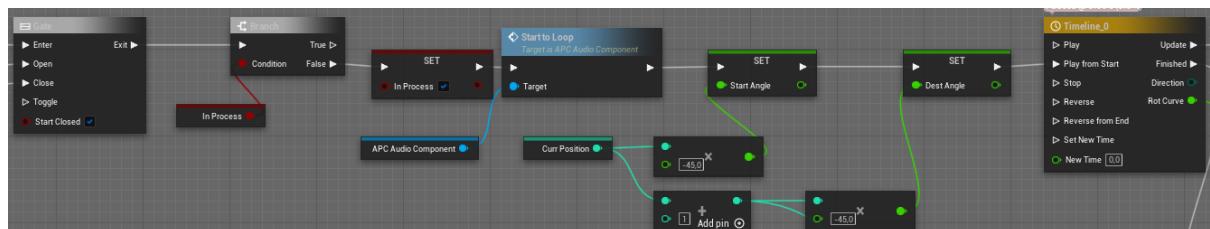


Figura 2.67 Mișcarea obiectului din interiorul cadranului

2.2.3.12 Peretele laser

Peretele laser reprezintă un obstacol în calea jucătorului. Jucătorul trebuie să rezolve puzzle-ul din nivel pentru a dezactiva laserele. Pentru a nu permite jucătorului să treacă prin el, acesta are incorporat un "Blocking volume". Laserele au multiple utilizări, fiind folosite atât pentru rezolvarea de puzzle-uri, cat și pentru ghidarea jucătorului. Modul de dezactivare alaserelor este diferit de la puzzle la puzzle pentru a nu crea sentimentul de liniaritate în jocul creat.

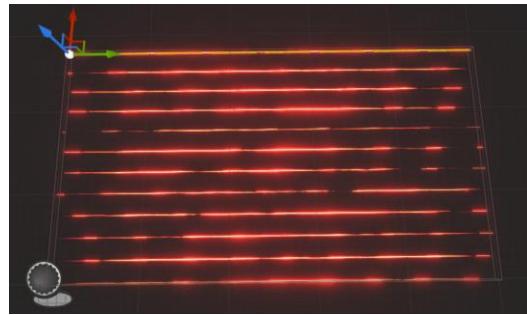


Figura 2.68 Peretele laser

2.2.3.12 Cutie fizica

Cutia colorată este un obiect fizic, care poate interacționa cu jucătorul și alte obiecte ale jocului, cum ar fi platforma buton și cea de teleport. Există trei variante de culoare ale cutiilor și fiecare culoare are o însușire diferita pentru a depinde de compatibilitatea cu platformele. În jurul ei există o cutie invizibila de coliziune care verifică constant dacă aceasta intră în contact cu obiectele și coincide cu tipul de material ales.

În timpul teleportării boxa are o animație de dezintegrare, se creează una invizibilă în cealaltă parte și cea inițială se sterge, având un efect de curgere.

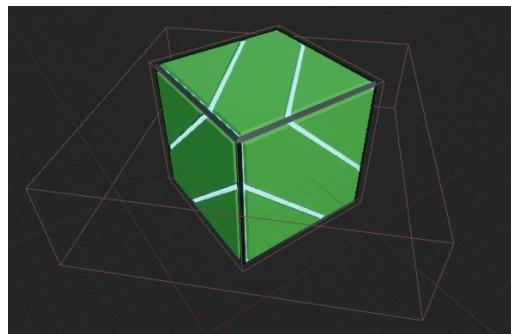


Figura 2.69 Cutia fizica

2.2.3.13 Platforma buton

Platforma este doar activată cu ajutorul cutiei fizice. Dacă daca platforma și cutia sunt de aceeași culoare, se va activa și executa acțiunea pentru actorii specificați. O singură cutie poate fi plasată pe buton. Butonul nu are nici un răspuns la coliziune și poate fi plasat oriunde pe o suprafață plană. Deasupra lui există o cutie invizibila care este setată ca în timpul interacției cu cubul o forțează în interiorul ei.

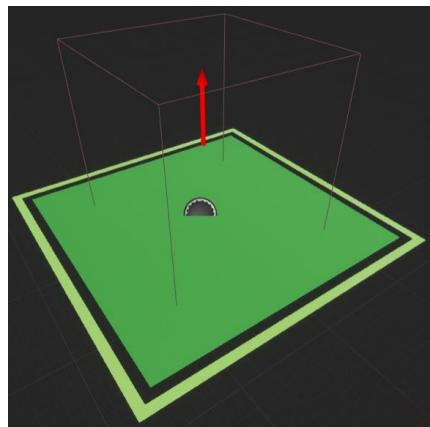


Figura 2.70 Platforma buton

2.2.3.14 Platforma de teleport

Teleportul este folosit pentru a trece un obiect dintr-o locație într-o altă locație setată. Setările sunt similare cu cele ale platformei buton singura diferență este că aceasta acceptă orice tip de cutie. „Spline” ajuta la realizarea unui efect de drum pe care particulele luminoase de diferite mărimi circula la următorul teleporter.

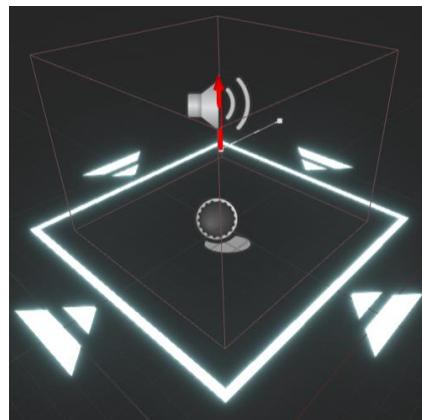


Figura 2.71 Platforma teleport

2.2.3.12 Placa de presiune

Placa este folosita pentru a detecta jucătorul și a activa de îndată evenimentul declarat. Are o animație și sunet care este redată în timpul acțiunii. De exemplu se pot utiliza mai multe platforme independente pentru a realiza un patern de urmărit când mergi printre ele.

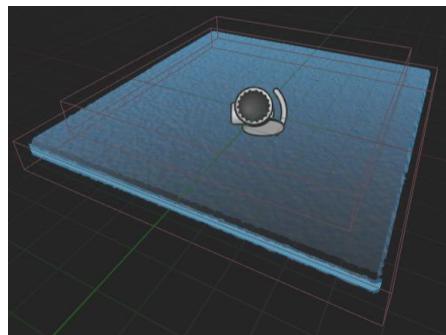


Figura 2.72 Placa de presiune

2.2.4 Ecranul de meniu și pauza

Meniul jocului a fost realizat într-un alt nivel de joc în mod 3D, unde cu ajutorul scripturilor s-a realizat interfața și legăturile dintre actori. Dispune de un actor tip camera și o interfață „widget”. Meniul dispune de patru opțiuni, fiecare având propriile atrbute.

În momentul apăsării butonului de „Options” se face o tranziție a camerei de la interfața meniului principal la cea a opțiunilor. Aici observam diverse butoane, unde în interiorul widget-ului s-au setat diferenți parametrii.

Interfața „widget” dispune de instrumente de editor care poate personaliza aspectul interfeței de utilizator prin combinarea componentelor simple, mai mici.

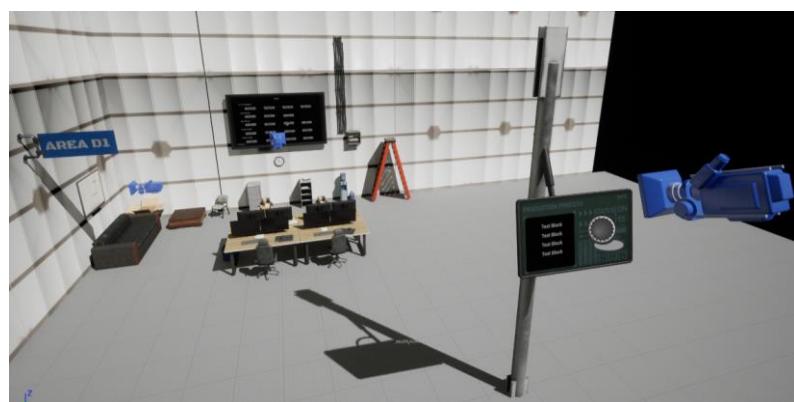


Figura 2.73 Previzualizare meniu 3D

Ecranul de pauza este un widget care poate fi accesat numai în interiorul jocului. Aceasta are un set de controale pe care jucătorul poate să le utilizeze pentru a schimba setările, a se întoarce în meniu sau a întrerupe jocul. S-a setat o regula ca la apăsarea butonului „escape” toate funcțiile din interiorul jocului se vor opri pana la revenirea acestora.

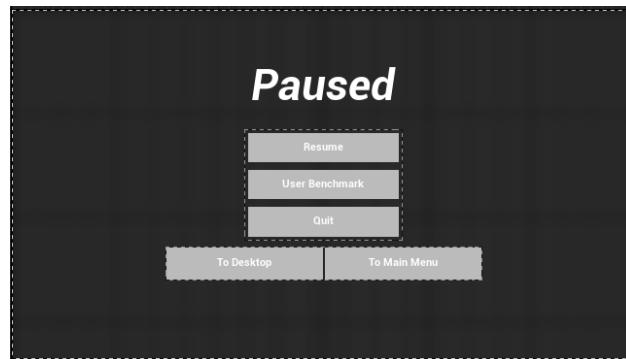


Figura 2.74 Previzualizare meniul de pauza

2.3. Scenariul de funcționare al aplicației

La pornirea aplicației jucătorul este întâmpinat de meniul principal , unde poate să alege una dintre opțiunile de pe ecran. În momentul apăsării butonului Start, se intră în ecranul de tranziție numit „loading screen” și se încarcă texturile și obiectele.

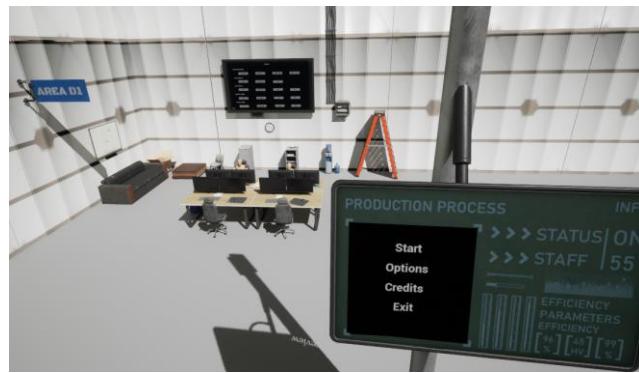


Figura 2.75 Meniul principal al jocului

După terminarea încărcării, apare o secvență animată ca o introducere a nivelului și se pune în prim-plan caracterul. El se poate mișca cu ajutorul tastaturii, poate crea portale, ridica și interacționa cu diverse obiecte.

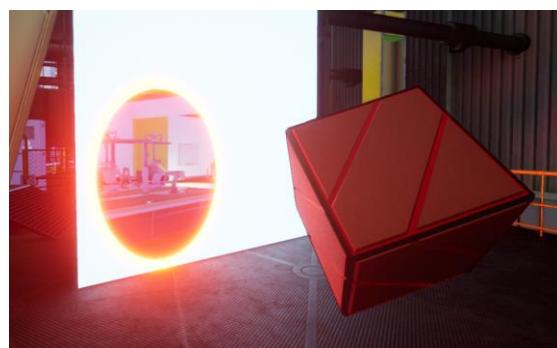


Figura 2.76 Portalul și obiectul ridicat

În nivel mai sunt elemente de animații care se activează cu ajutorul unui obiect sau când personajul intra într-o arie specifică. Pentru a progrăsa, jucătorul va trebui să rezolve anumite puzzle-uri. Fiecare obiect are o animație sau un sunet specific pentru a atenționa jucătorul dacă a fost activat.



Figura 2.77 Interacțiunea dintre portal și cub

2.4. Manual de instalare și utilizare

2.3.1 Cerințe software și hardware

Este importantă actualizarea driverelor calculatorului pentru o bună funcționare a jocului, deoarece consumă multe resurse pe partea grafică.

Pentru rularea aplicației este necesar doar un calculator, cu dispozitive I/O de bază (mouse, tastatura, monitor). De asemenea, este necesar un spațiu de stocare cu aproximativ 20GB de memorie liberă. Pentru o rulare optimă este indicată existența unei memorii RAM de minim 12GB și a unei placi video de minim 8GB.

În următoarea secțiune, vom pune în evidență categoriile de utilizatori ai jocului pe care ni-l dorim. Nu există o interfață de logare sau de creare user, prin urmare va exista un singur actor: utilizatorul (player-ul).

Utilizatorul pornește aplicația și poate vedea cele patru opțiuni ale meniului:

Butonul start începe jocul propriu-zis, player-ul va avansa în nivel prin apăsarea diferitelor taste pentru a se deplasa, sări, colecta obiecte. În joc jucătorul poate apăsa tasta de pauza unde vor apărea alte 3 opțiuni "Resume", "Benchmark" și "Exit".

În opțiuni se prezintă setările jocului, unde le poți schimba după bunul plac.

Butonul credit afișează interfață cu numele autorului iar cel de exit închide aplicația propriu-zisa.

Modul audio va reda melodii relevante pentru starea curentă a jocului alături de diferitele sunete care să reprezinte efectele acțiunilor sale în interiorul aplicației.

Interfața utilizator este intuitiva și user-friendly, deplasarea player-ului se va realiza cu ajutorul tastaturii, dar nu există și posibilitatea de multiplayer, ci de un singur utilizator la momentul dat.

2.3.2 Instalare și rulare

Pe platforma „Epic Games”, poți descărca „Epic Game Launcher”. În acesta platforma poți gestiona toate versiunile de UE. Prima data vi se va solicita să va conectați cu contul de Epic, altfel nu puteți descărca sau utiliza motorul grafic. La instalarea motorului va apărea niște setări opționale precum: platforma de suport, conținut de început, depanare și codul sursa al motorului. La selectarea opțiunilor trebuie să ne asigurăm dacă avem suficient spațiu pe disc. În funcție de specificațiile sistemului și de viteza confeționii la internet, descărcarea motorului grafic poate dura între 10 și 40 de minute.

După finalizarea pașilor de mai sus, Lansatorul va începe verificarea instalării. De asemenea, va descărca orice componente opționale pe care le-ați selectat și pe care nu le poate găsi în folderul de instalare. Când acest proces se încheie, veți putea lansa această versiune a Unreal Engine din Epic Games Launcher. Pentru a accesa proiectul apăsați butonul „Browse” și selectați fișierul uproject.

CONCLUZII, CONTRIBUTII SI PERSPECTIVE DE VIITOR

Concluzii

Design-ul jocului este un proces intensiv care necesita planificare și dedicare. Nu putem nega și faptul că crearea unui joc este esențială pentru dezvoltarea pe plan intelectual și creativ. De-a lungul întregului proces de creare a jocului am reușit să testezi și să înțeleg designul jocului. Sunt necesare cunoștințe tehnice și artistice pentru a reuși pe partea de design și programare. A fost necesar mult timp pentru a înțelege cum să organizezi mai bine mecanica jocului deoarece nu totul poate fi planificat în avans și ca unele lucruri trebuie rafinate în timpul dezvoltării jocului. Proiectul jocului are potențialul de a se extinde și de a utiliza în continuare cu ajutorul informațiilor dobândite în cadrul acestuia. În plus, există încă mult potențial de atins și imersiunea jocului ar putea fi aprofundată. Documentațiile despre jocuri mi-au permis să învăț instrumentele și modul în care funcționează tehnologiile respective.

Contribuții personale

În dezvoltarea acestei aplicații, s-a realizat proiectarea și implementarea arhitecturii nivelului, a obiectelor și a acțiunilor atât pe partea vizuala cat și pe partea de programare. Jocul oferă o experiență plăcută, intuitivă și naturală prin îmbinarea elementelor vizuale și auditive cu simularea fizică și implementarea unor sisteme variate de puzzle-uri.

- Am creat interfața pentru a accesa meniul aplicației și implementarea acestuia în joc;
- Am realizat harta ce conține diverse elemente naturale;
- Realizarea unei fabrici în mediul prezentat de mai sus ce conține mașinarii, puzzle-uri și alte secrete;
- Folosind unealta „Sequence” am creat animațiile folosite pe parcursul jocului;
- Am implementat scripturi de interacțiuni cu obiecte și fizica acestora;

Perspective de viitor

Pentru faptul că aplicația a fost dezvoltată în motorul grafic Unreal Engine se pot realiza noi funcționalități pentru a atrage alți jucători prin adăugarea și editarea blueprint-urilor, funcții ușor de implementat.

Crearea unui sistem multiplayer este o posibilitate de dezvoltare viitoare în care se poate permite conectarea în simultan, pe aceeași lume, a doi jucători unde fiecare are alt sistem de puzzle realizat pentru a ii permite celuilalt jucător să progreseze.

BIBLIOGRAFIE

- [1] <https://docs.unrealengine.com/5.0/en-US/unreal-engine-5-0-release-notes/> 19.06.2022
- [2] https://en.wikipedia.org/wiki/Unreal_Engine 19.06.2022
- [3] https://en.wikipedia.org/wiki/Game_engine 19.06.2022
- [4] https://ro.wikipedia.org/wiki/Joc_video 19.06.2022
- [5] https://en.wikipedia.org/wiki/Video_game_design 22.06.2022
- [6] <https://docs.unrealengine.com/4.27/en- S/ProgrammingAndScripting/Blueprints/GettingStarted/> 22.06.2022
- [7] <https://docs.unrealengine.com/5.0/en-US/overview-of-blueprints-visual-scripting-in-unreal-engine/> 19.06.2022
- [8] https://en.wikipedia.org/wiki/Single-player_video_game 19.06.2022
- [9] <https://docs.unrealengine.com/5.0/en-US/plugins-in-unreal-engine/> 19.06.2022
- [10] <https://docs.unrealengine.com/5.0/en-US/unreal-editor-interface/> 19.06.2022
- [11] <https://docs.unrealengine.com/5.0/en-US/environmental-light-with-fog-clouds-sky-and-atmosphere-in-unreal-engine/> 19.06.2022
- [12] <https://docs.unrealengine.com/5.0/en-US/exponential-height-fog-in-unreal-engine/> 21.06.2022
- [13] <https://docs.unrealengine.com/5.0/en-US/level-blueprint-in-unreal-engine/> 20.06.2022
- [14] <https://docs.unrealengine.com/5.0/en-US/blueprint-class-assets-in-unreal-engine/> 21.06.2022
- [15] <https://docs.unrealengine.com/5.0/en-US/types-of-blueprints-in-unreal-engine/> 21.06.2022
- [16] <https://docs.unrealengine.com/5.0/en-US/landscape-sculpt-mode-in-unreal-engine/> 22.06.2022
- [17] <https://docs.unrealengine.com/5.0/en-US/procedural-foliage-tool-in-unreal-engine/> 22.06.2022
- [18] <https://docs.unrealengine.com/5.0/en-US/setting-up-a-character-in-unreal-engine/> 20.06.2022
- [19] <https://docs.unrealengine.com/5.0/en-US/unreal-engine-sequencer-movie-tool-overview/> 19.06.2022
- [20] <https://docs.unrealengine.com/5.0/en-US/content-browser-in-unreal-engine/> 19.06.2022

[21] <https://docs.unrealengine.com/5.0/en-US/control-rig-in-blueprints-in-unreal-engine/>
22.06.2022

[22] <https://docs.unrealengine.com/5.0/en-US/animation-blueprint-ccdk-in-unreal-engine/>
22.06.2022

[23] <https://docs.unrealengine.com/5.0/en-US/creating-a-main-menu-in-unreal-engine/>
21.06.2022

[24] <https://docs.unrealengine.com/5.0/en-US/creating-a-pause-menu-in-unreal-engine/>
21.06.2022

[25] <https://docs.unrealengine.com/5.0/en-US/physical-materials-in-unreal-engine/>
21.06.2022

[26] <https://docs.unrealengine.com/5.0/en-US/installing-unreal-engine/> 19.06.2022

[27] <https://stf3d.de/landscape-pro-tutorials/#start> 10.05.2022

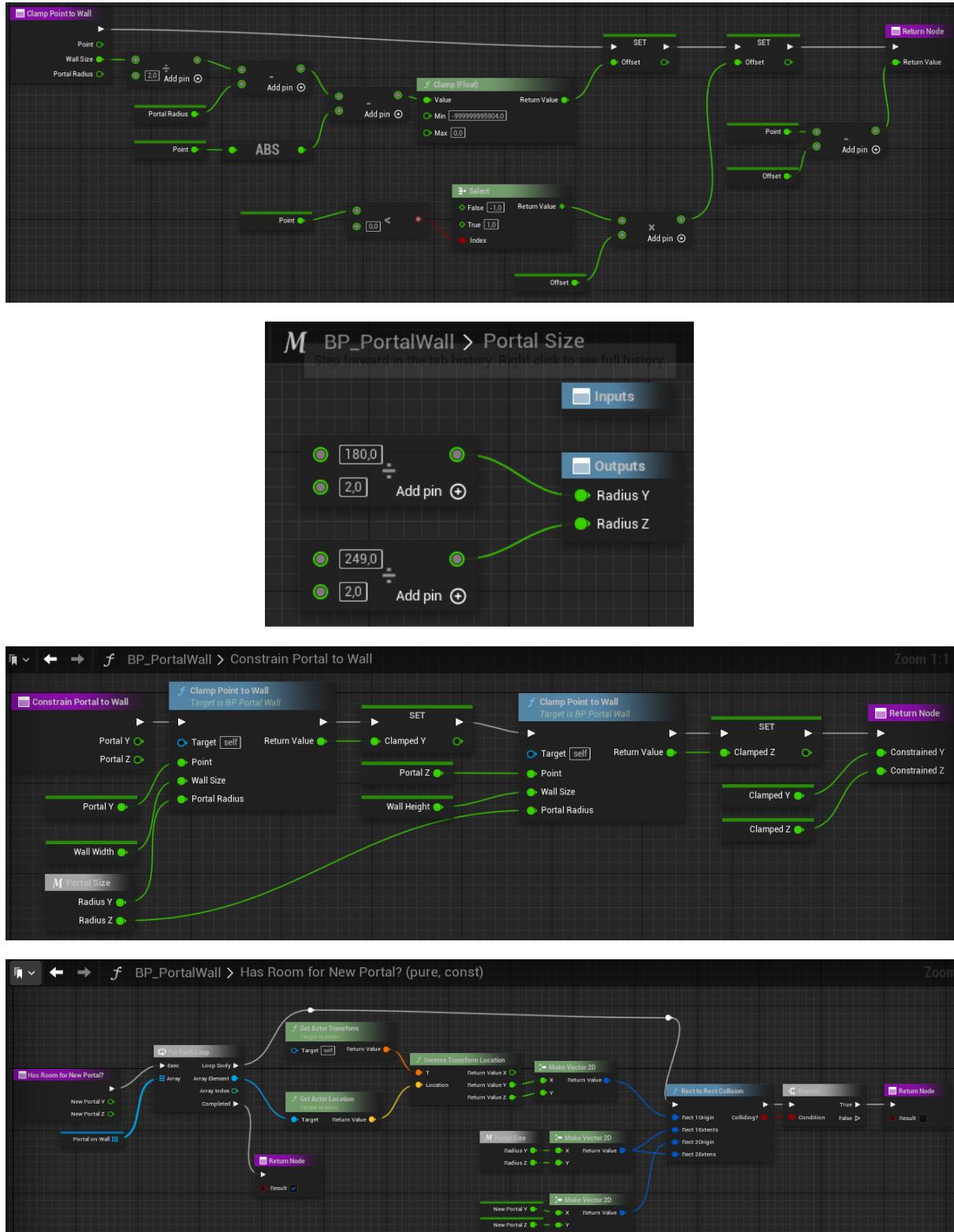
[28] <https://www.unrealengine.com/marketplace/en-US/product/adventure-character>
12.05.2022

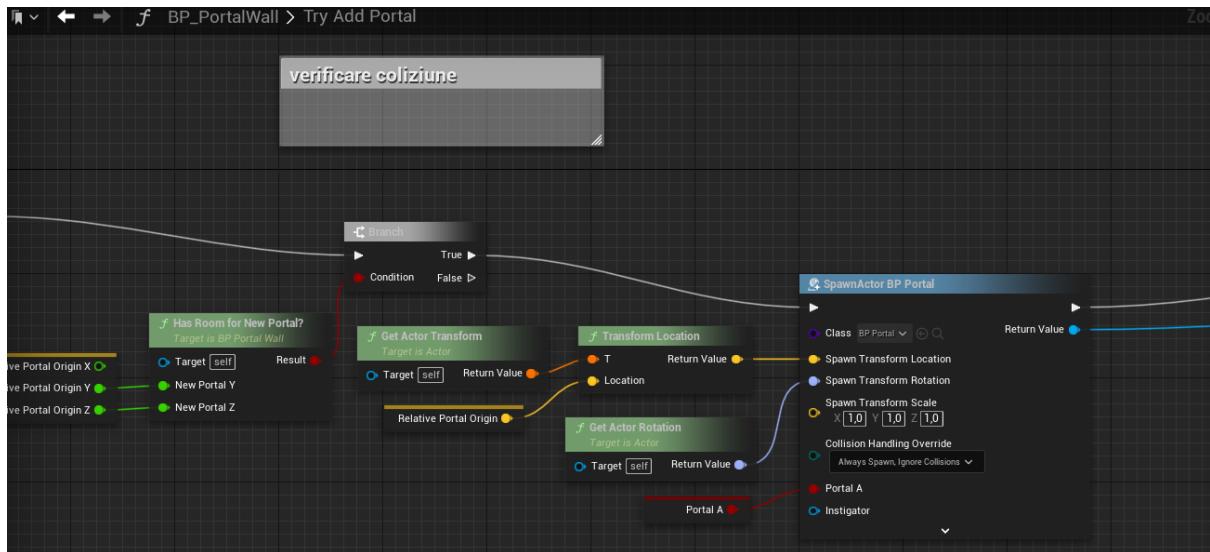
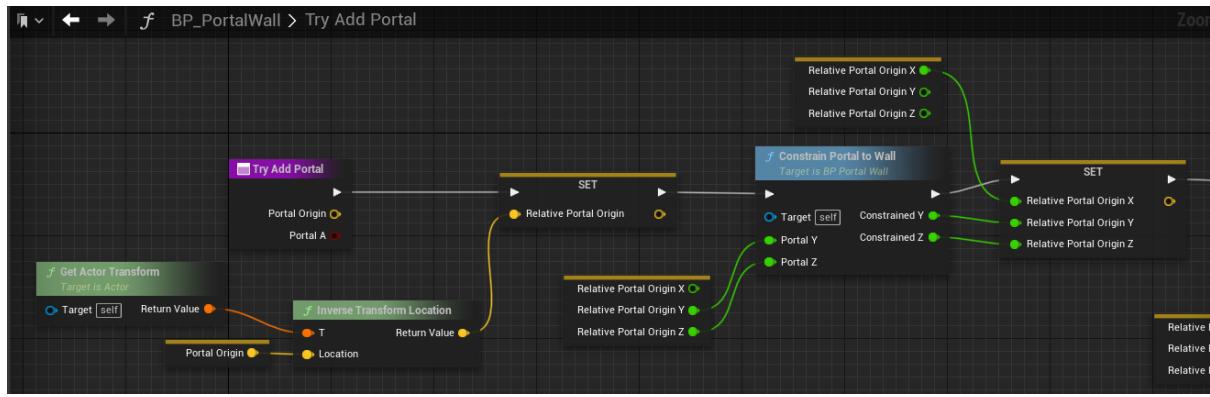
[29] <https://www.unrealengine.com/marketplace/en-US/product/factory-environment-collection> 14.05.2022

[30] <https://www.unrealengine.com/marketplace/en-US/product/advanced-puzzle-constructor>
20.05.2022

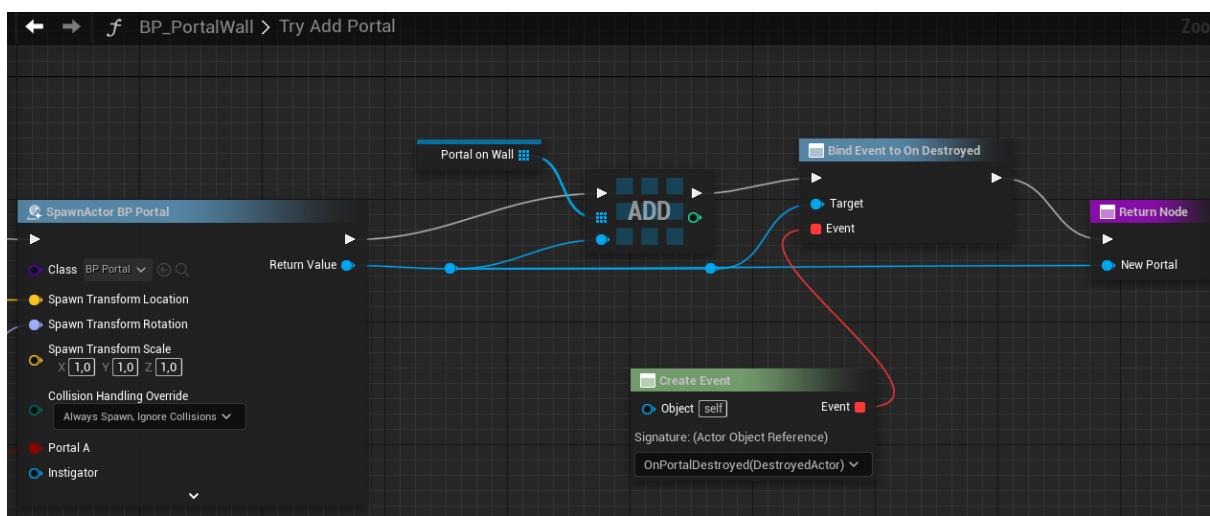
ANEXE

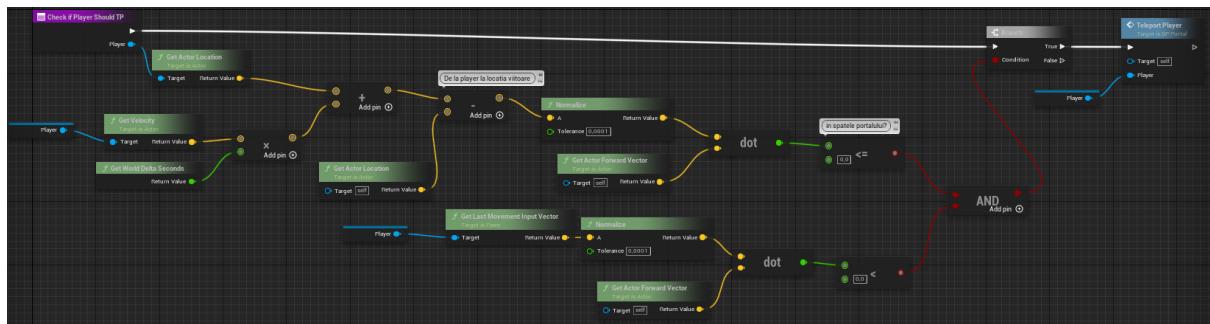
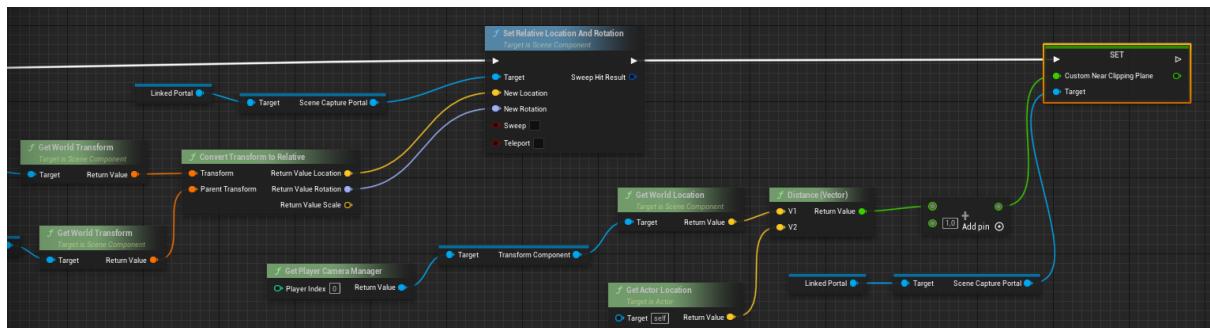
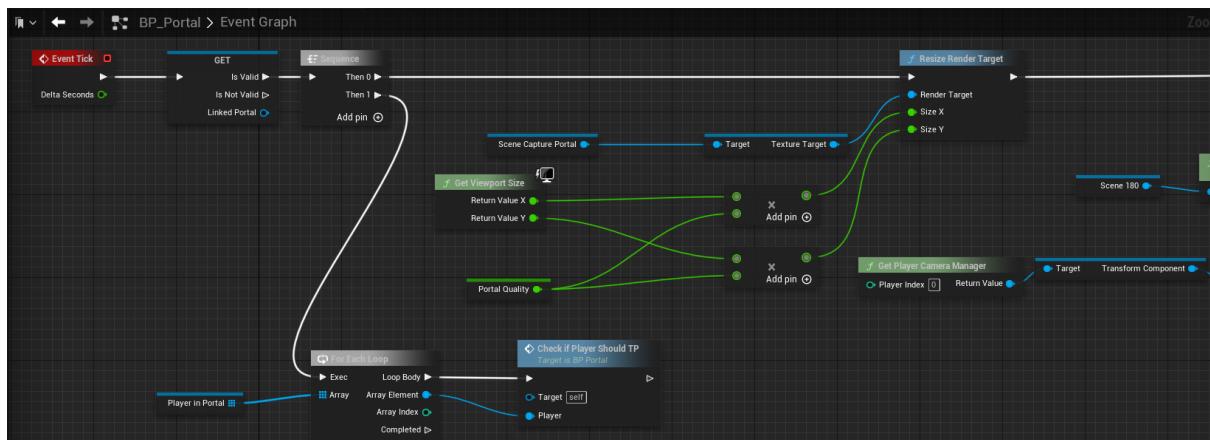
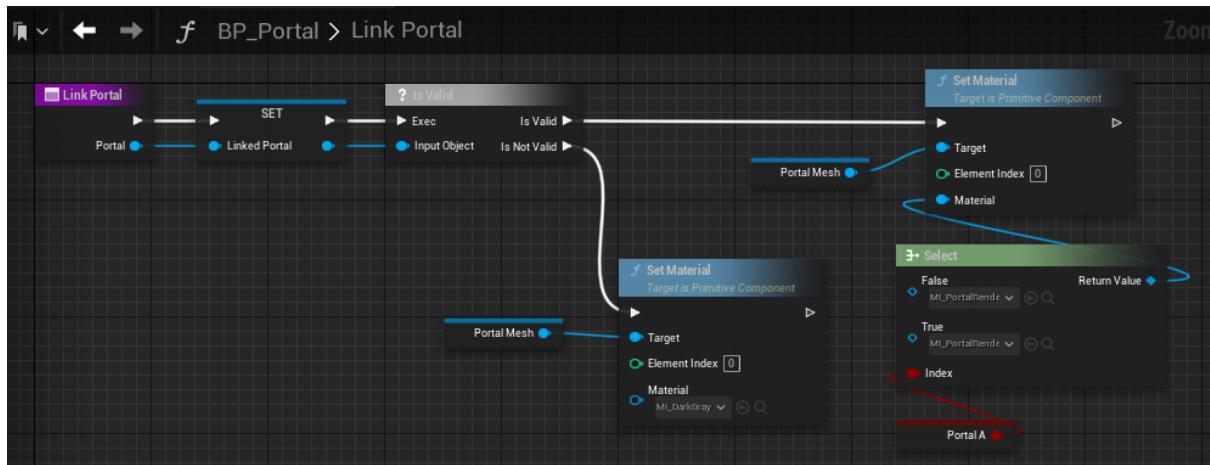
BP_PortalWall

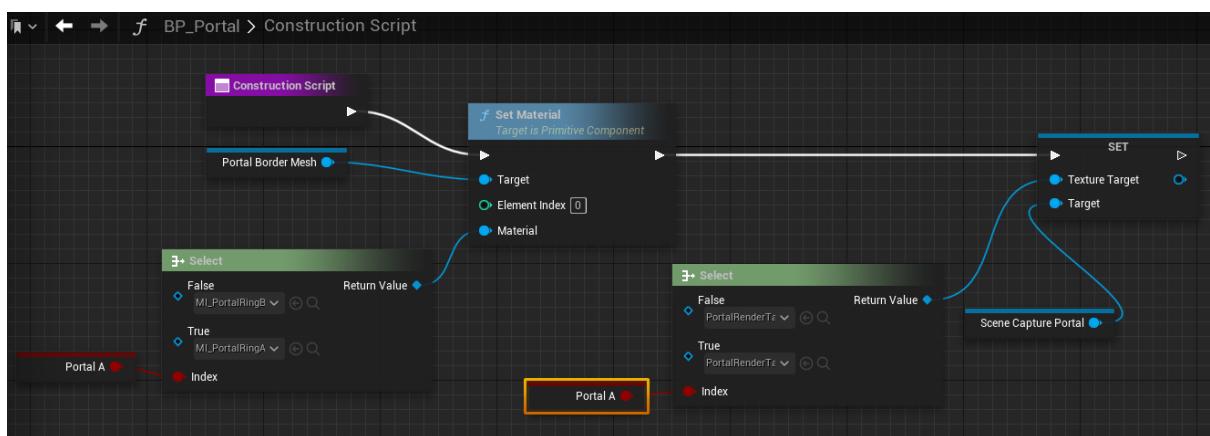
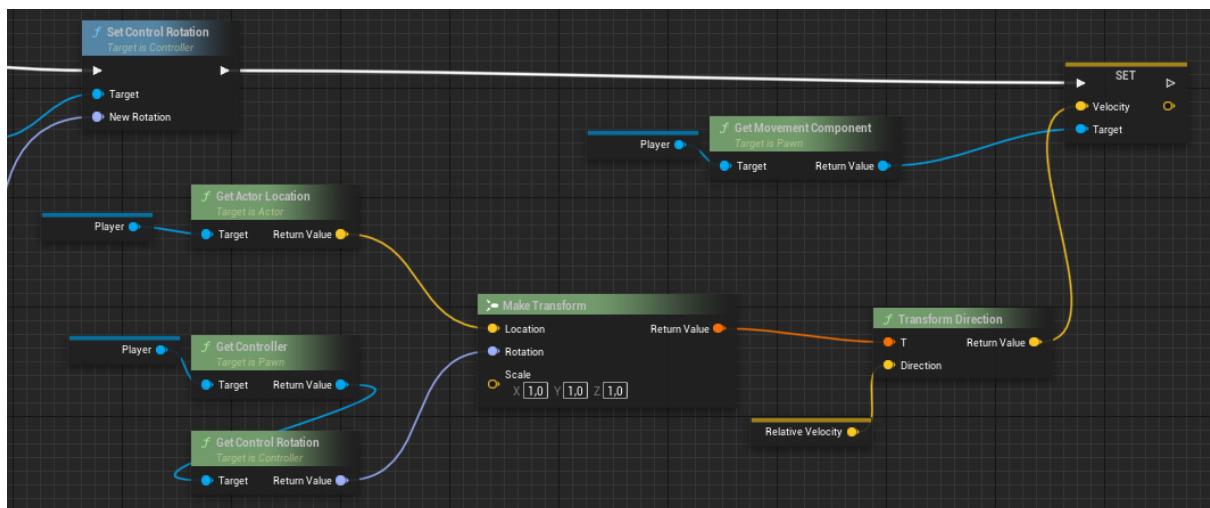
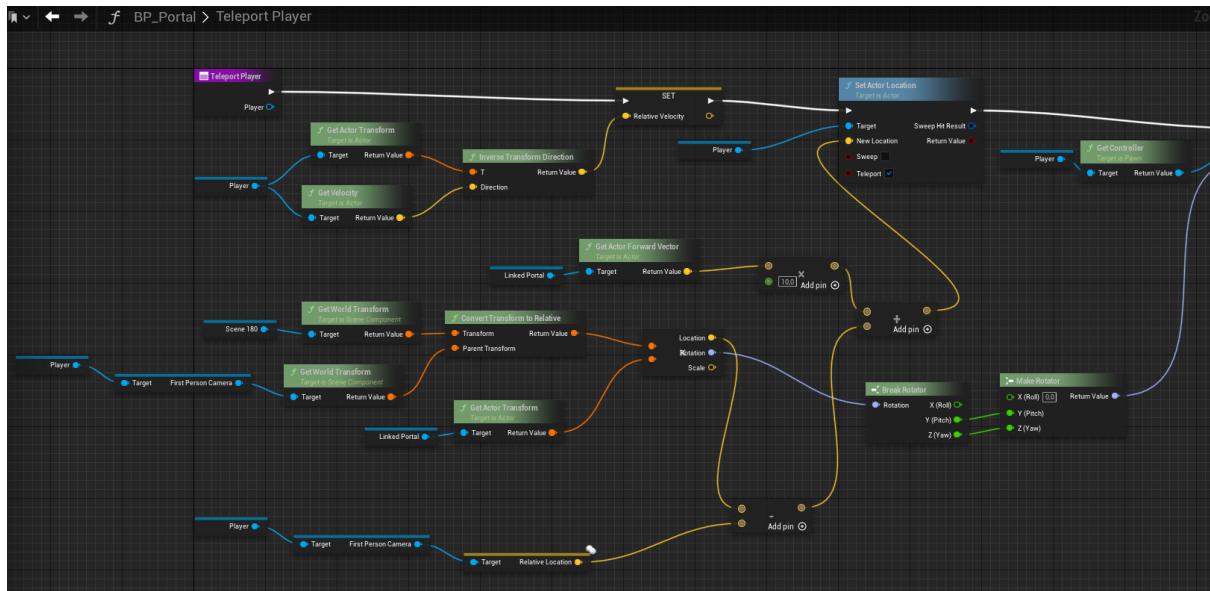




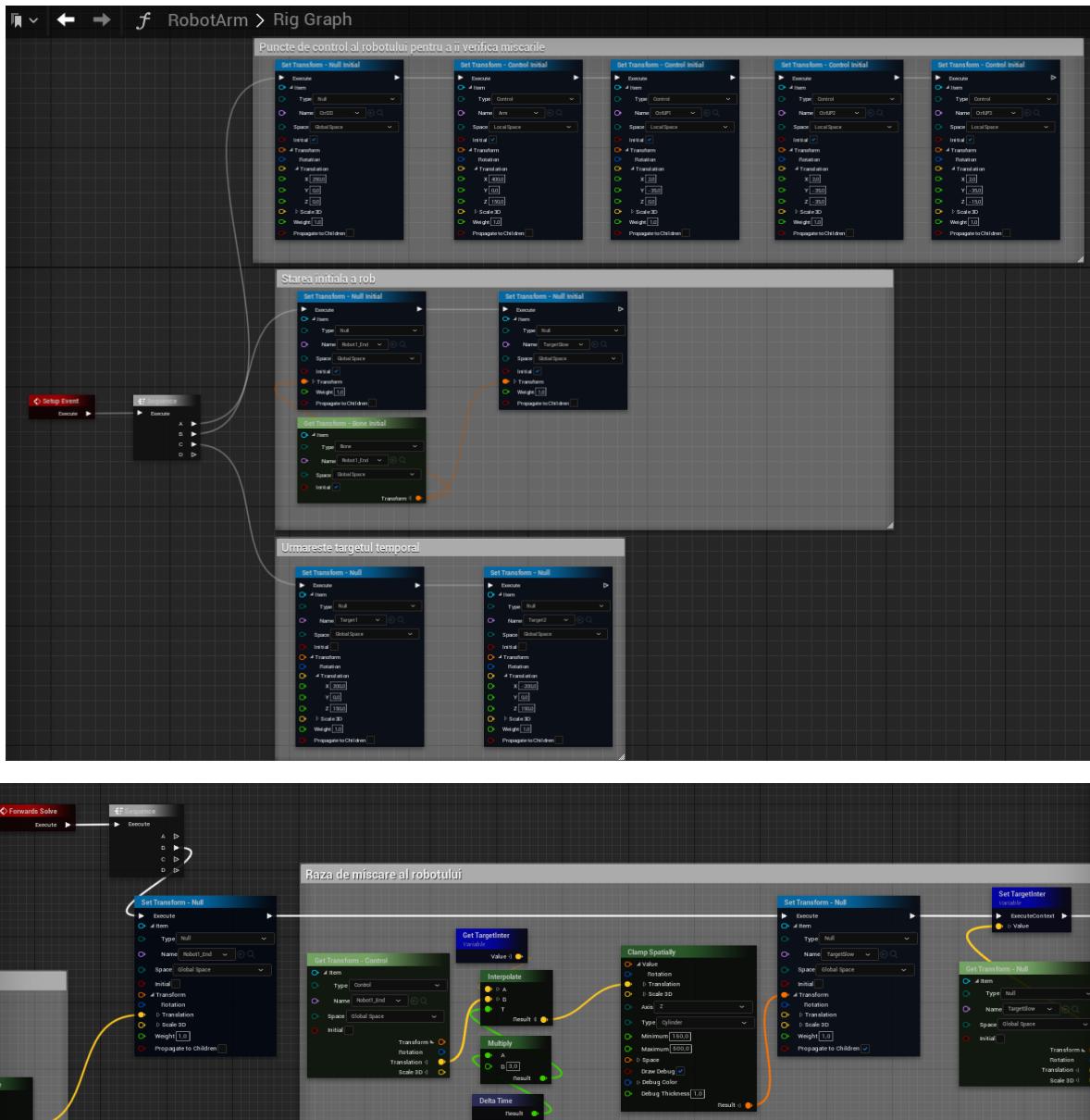
Portal

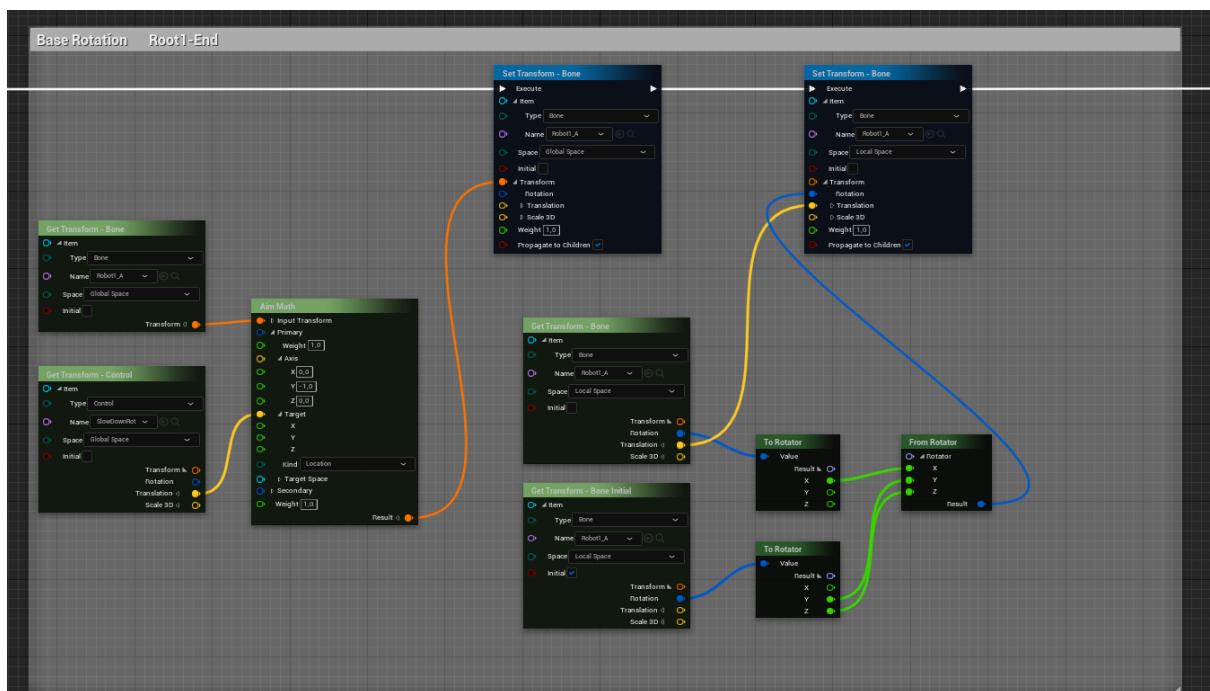
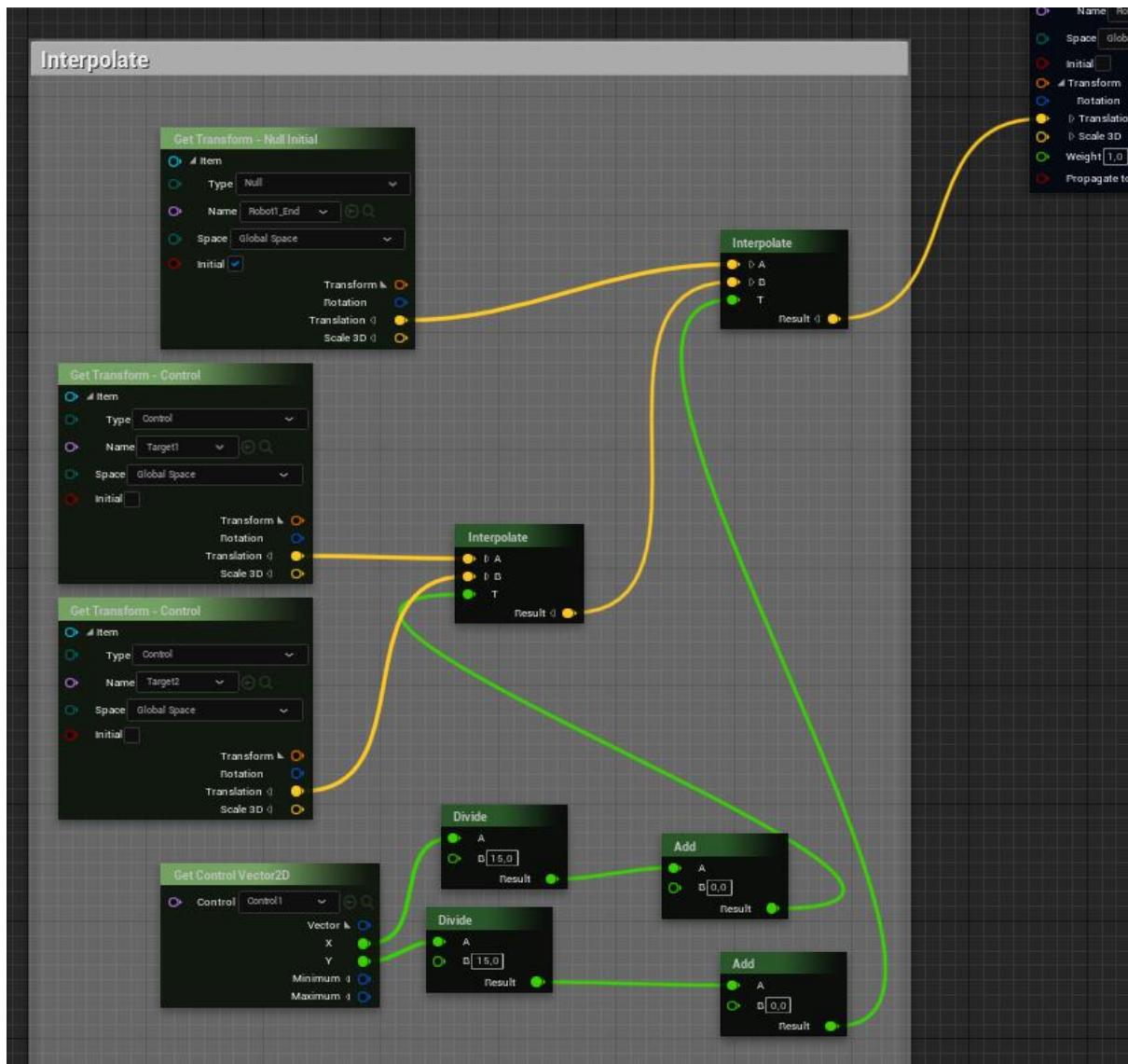


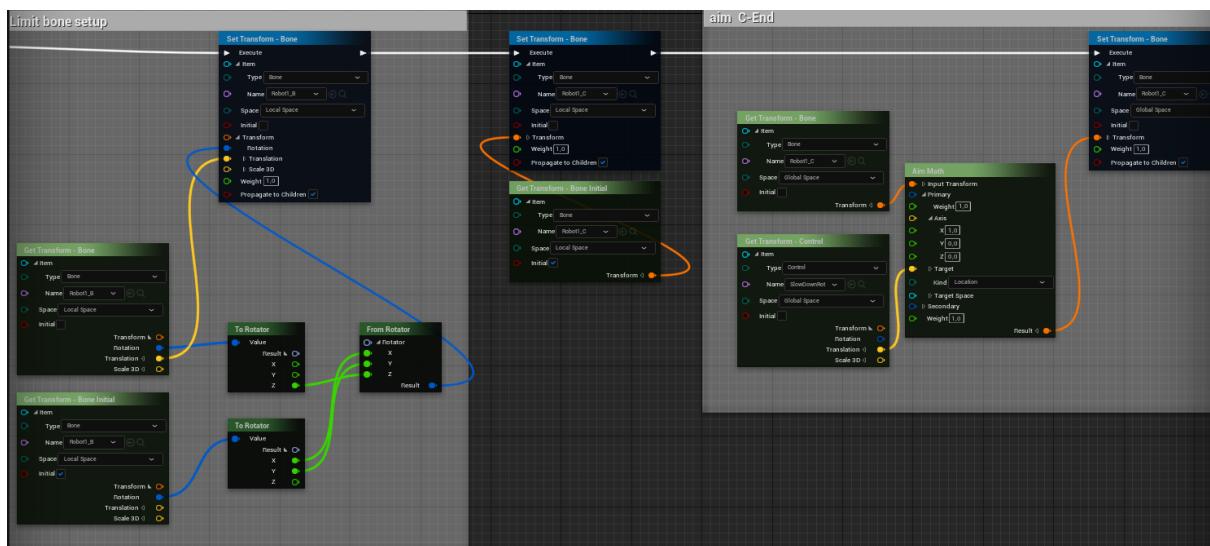
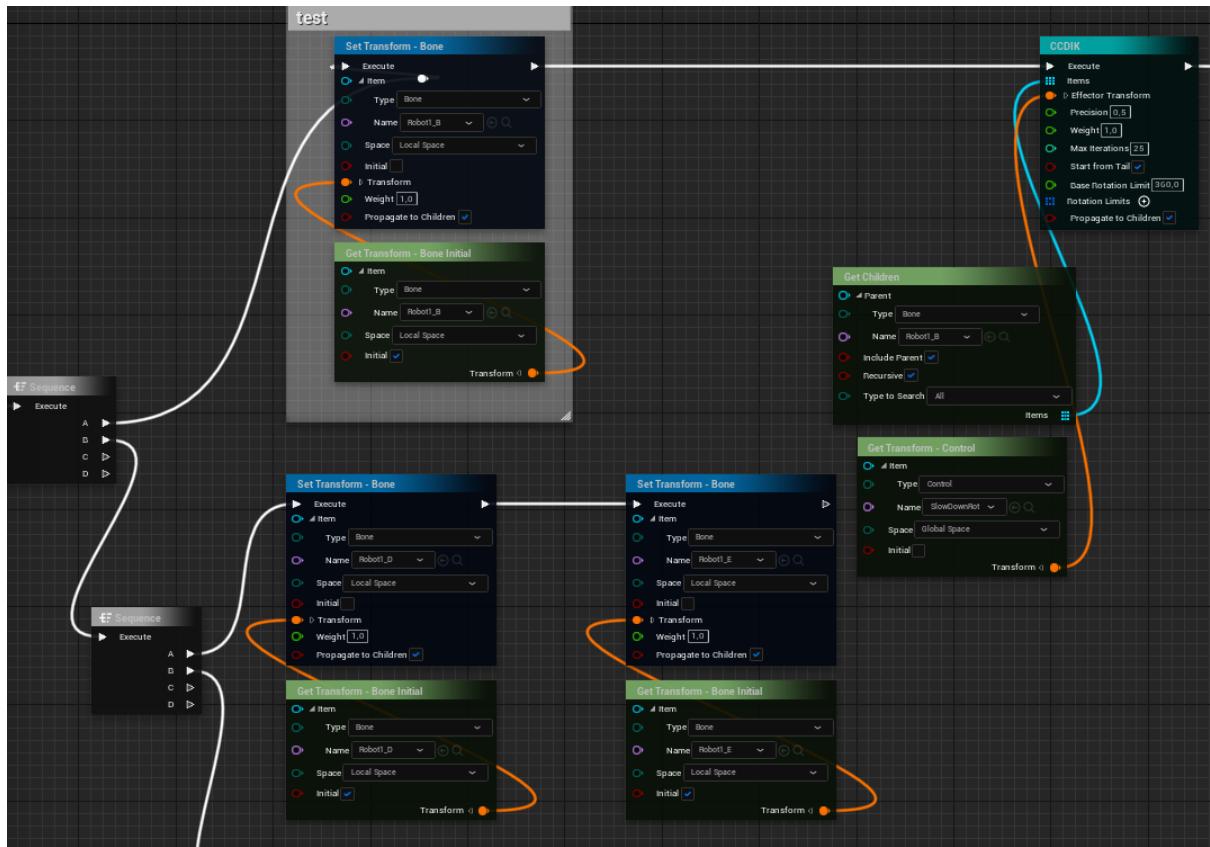


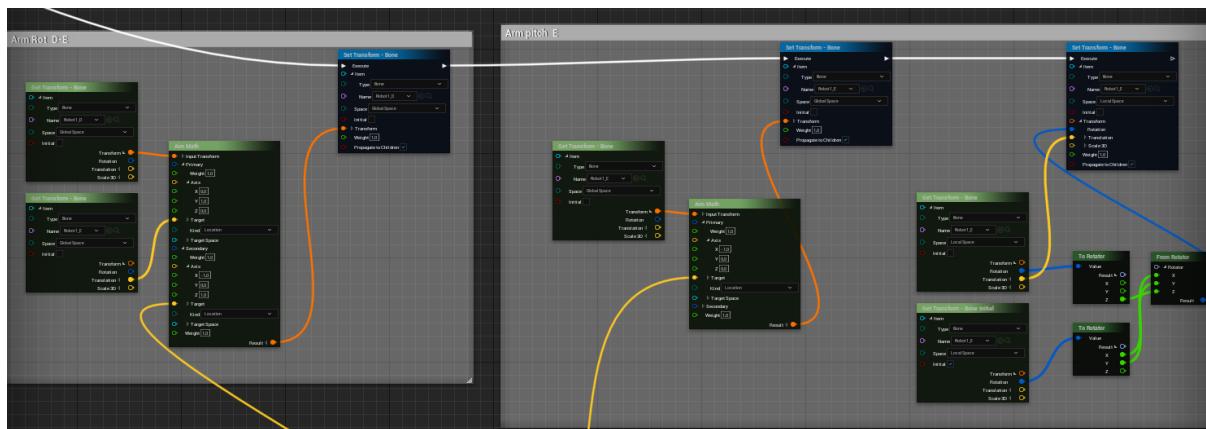
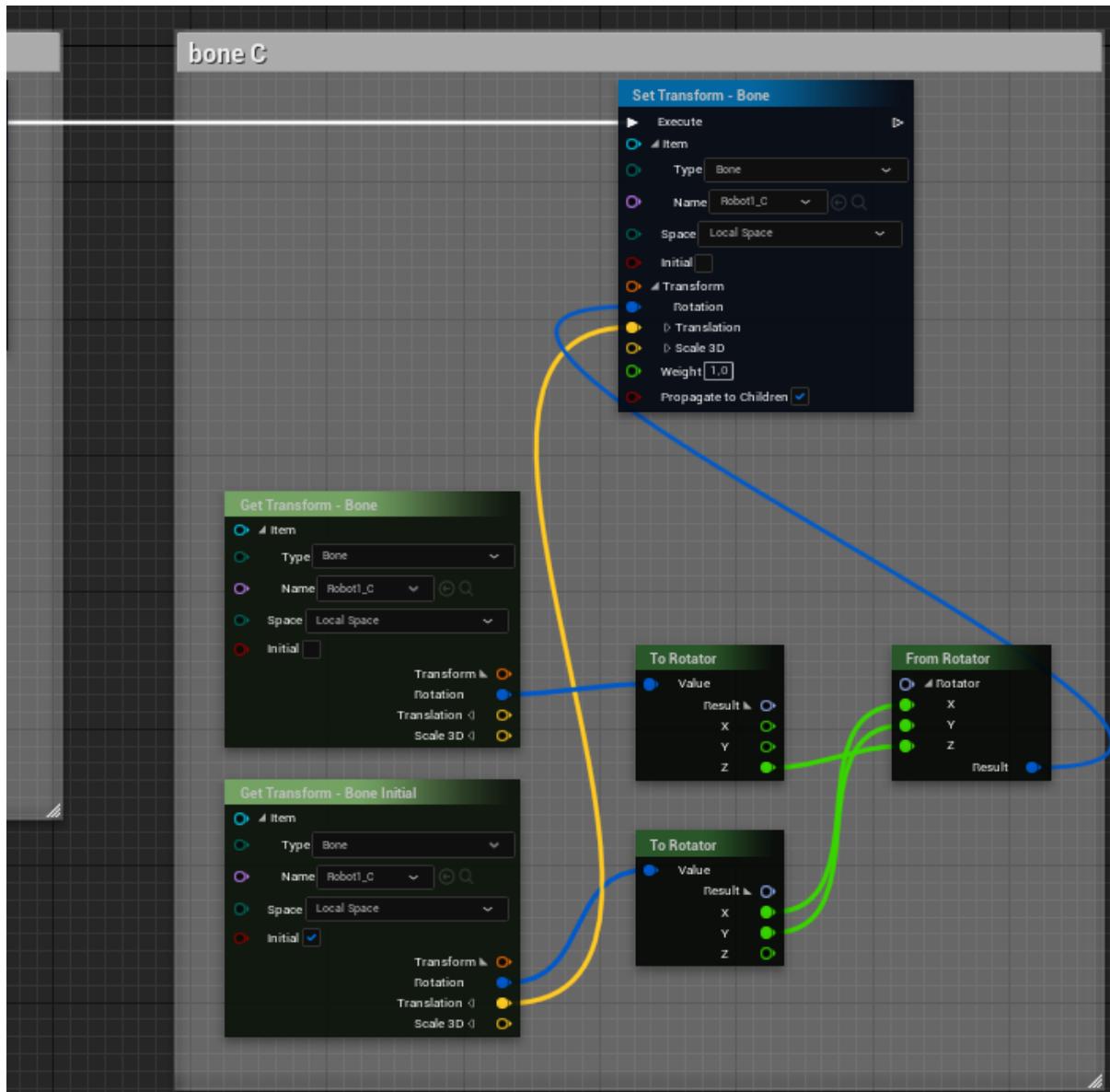


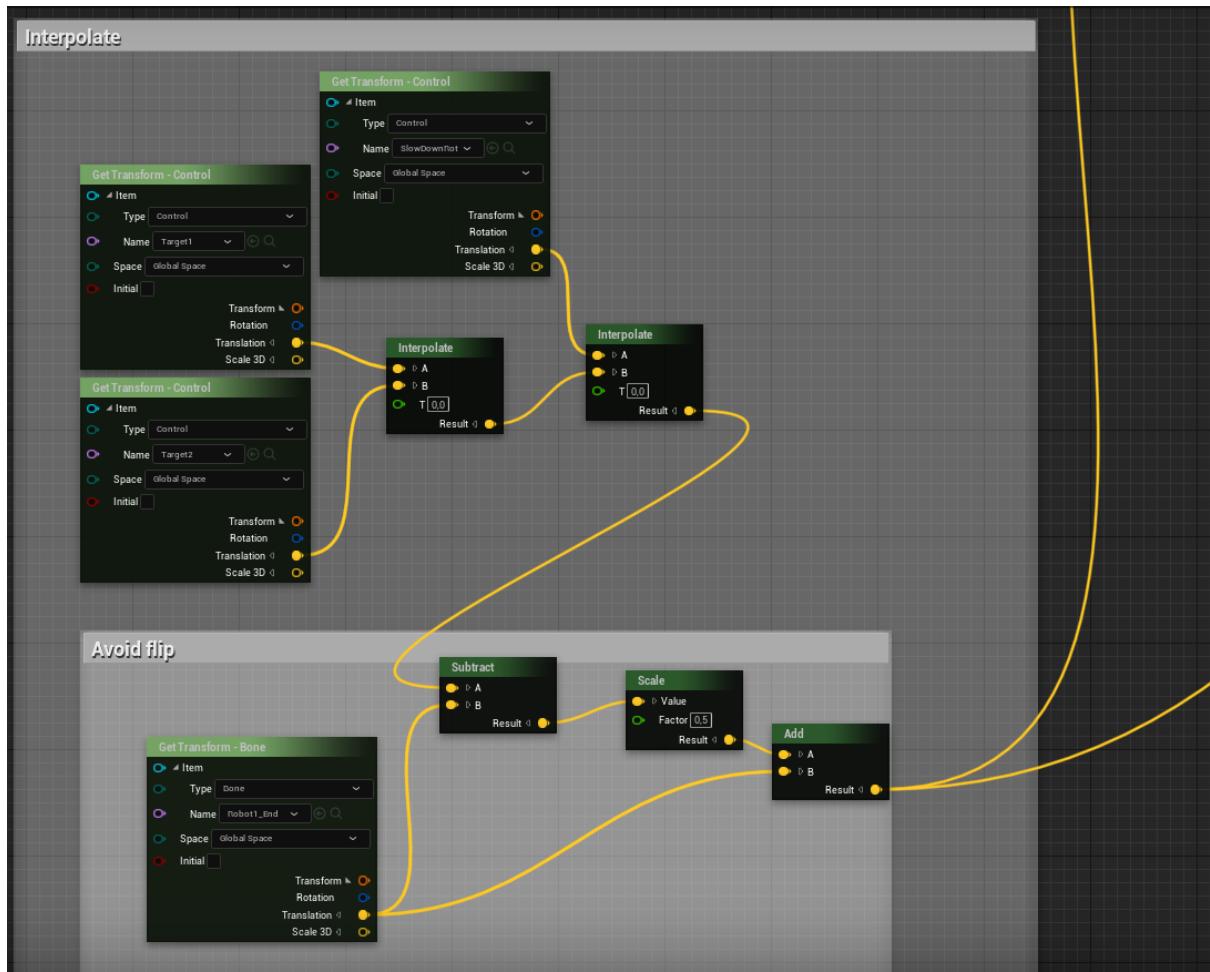
Brat Robot



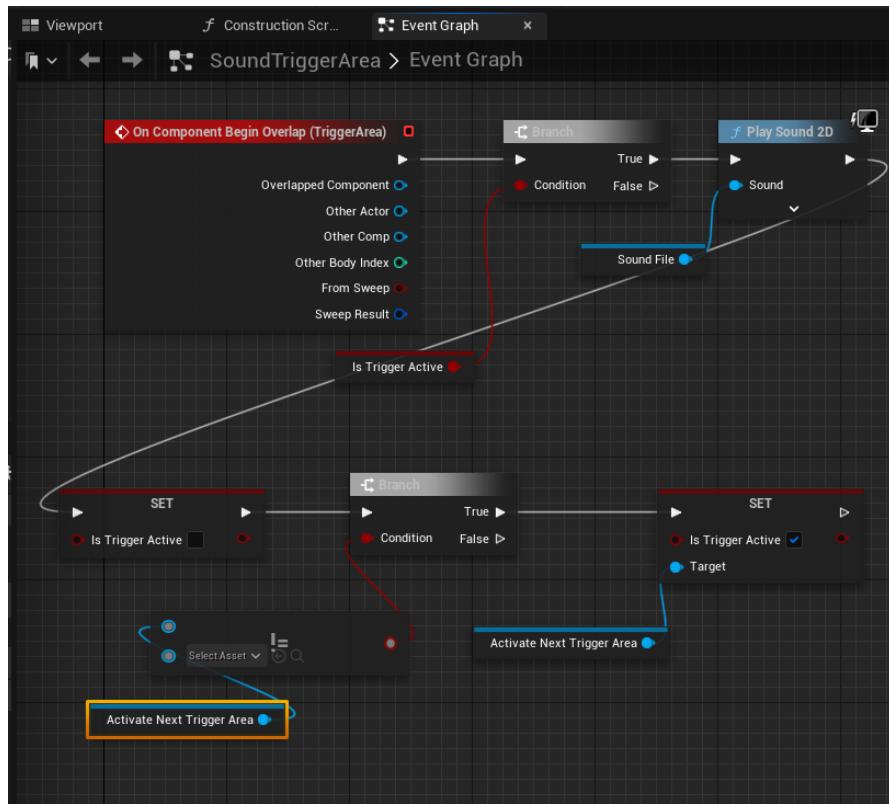




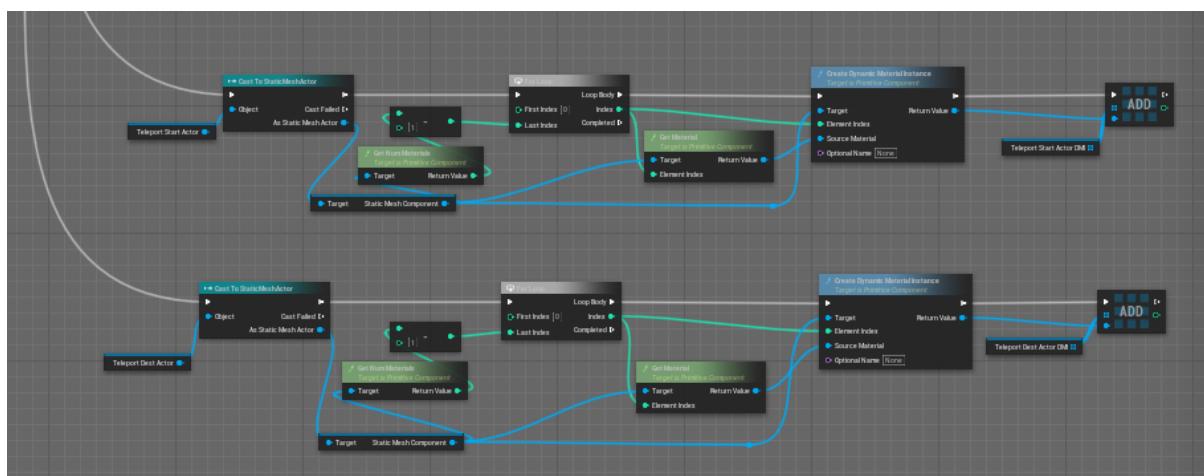
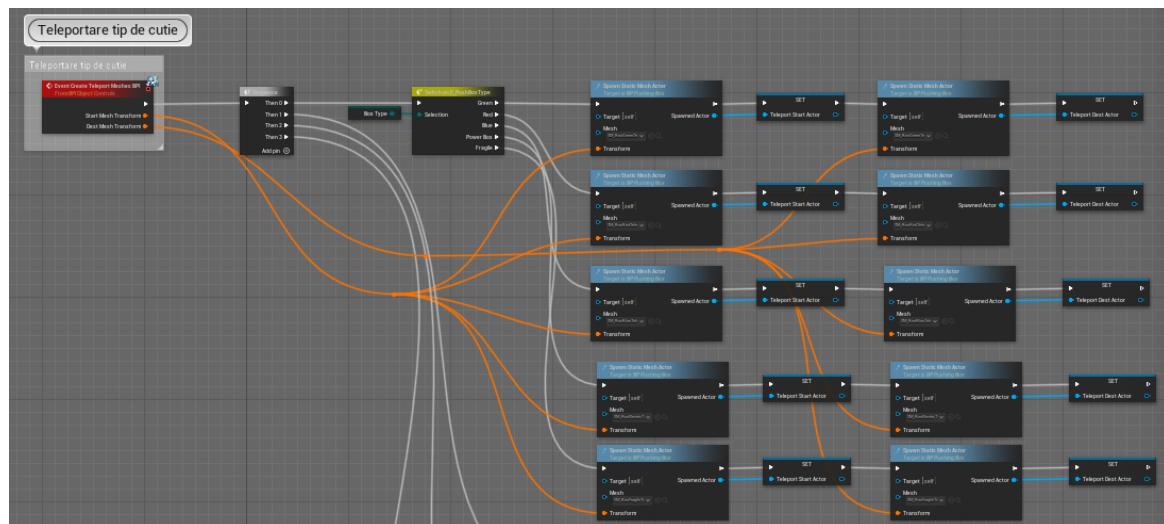
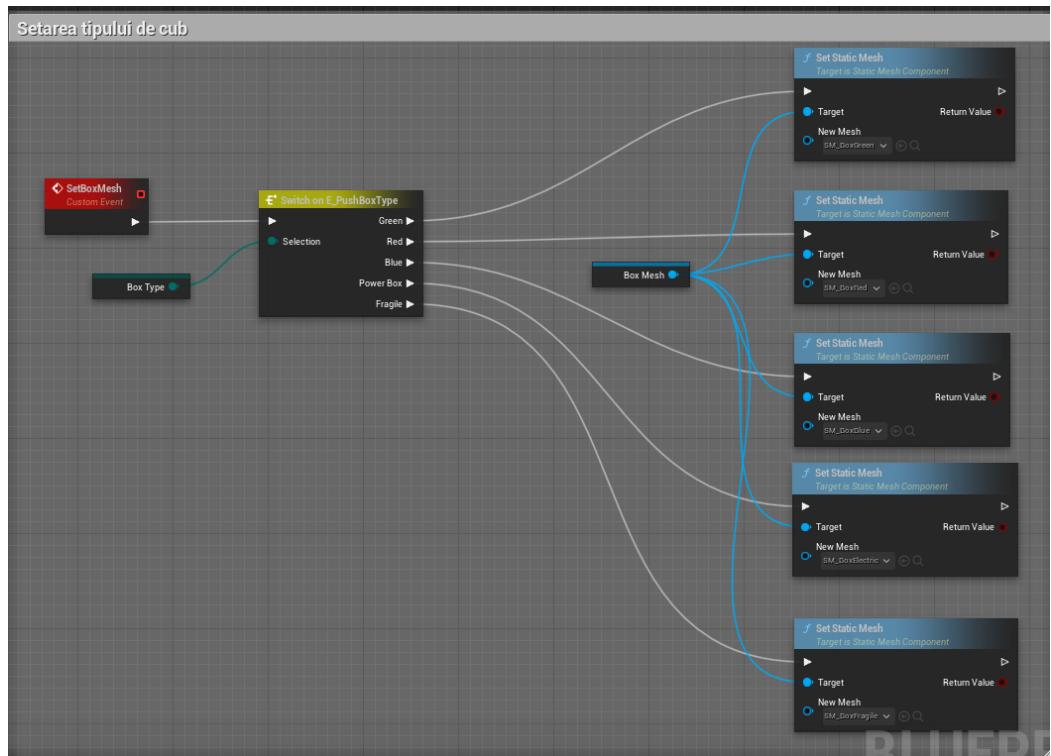


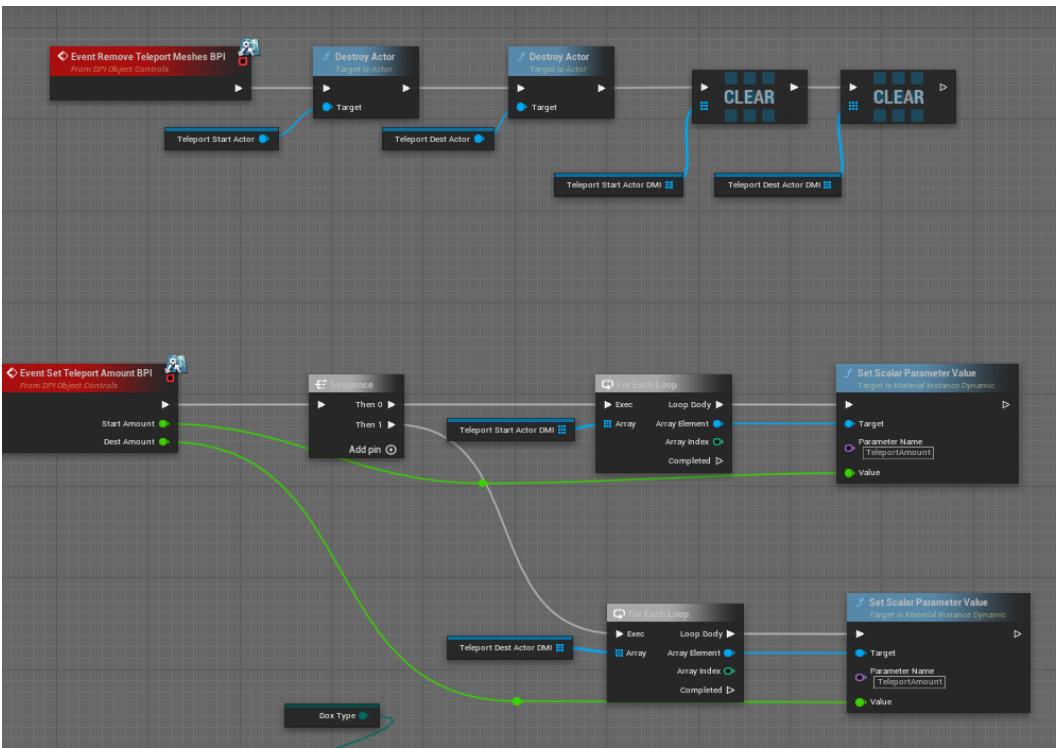


Sound trigger

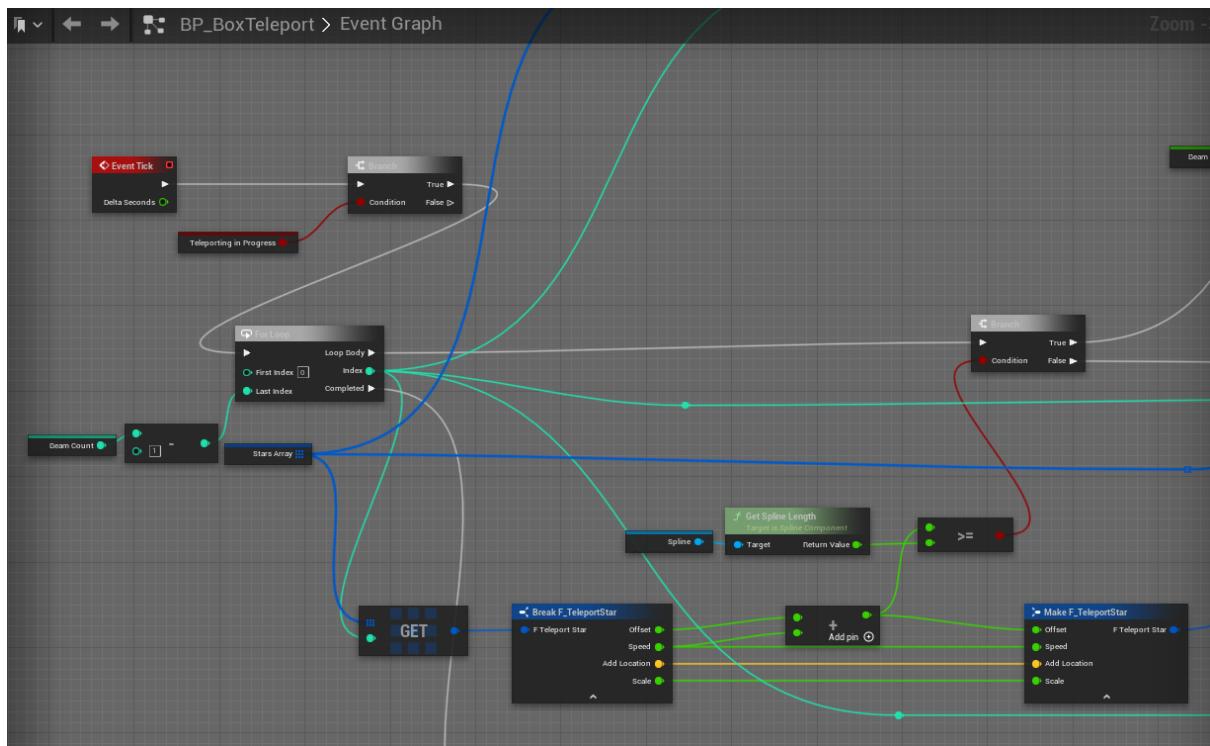


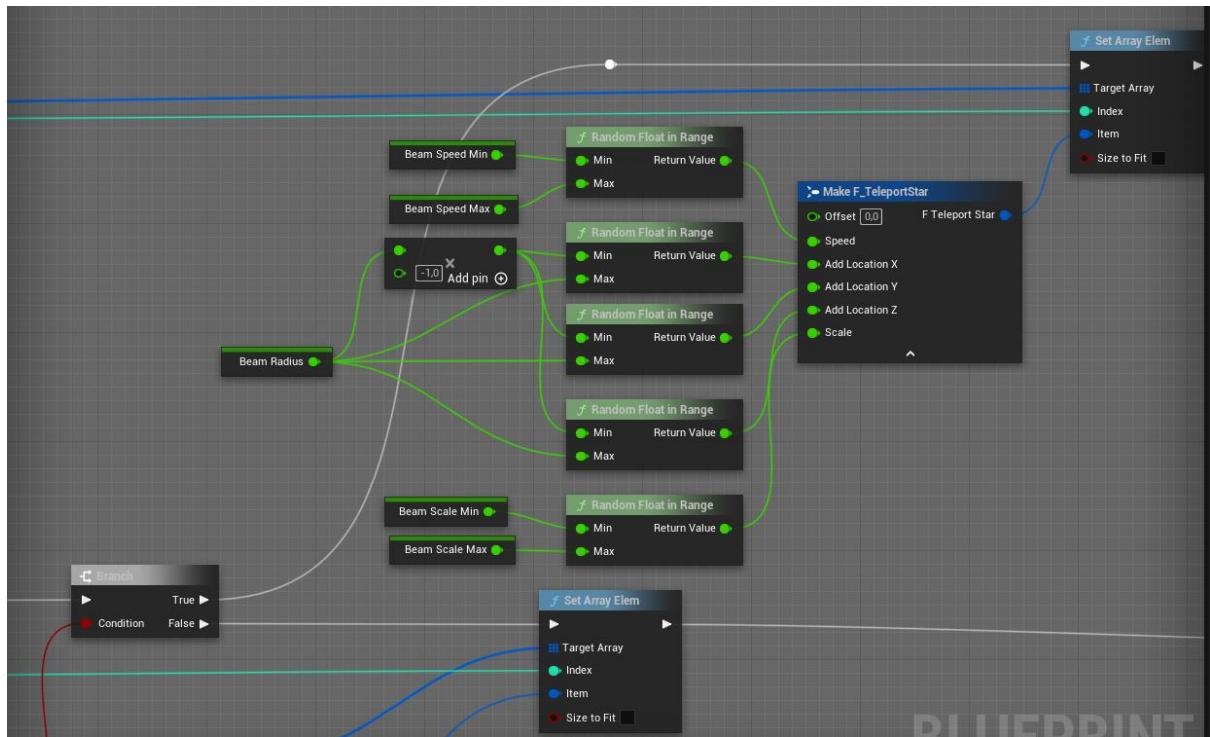
Interactiune cub



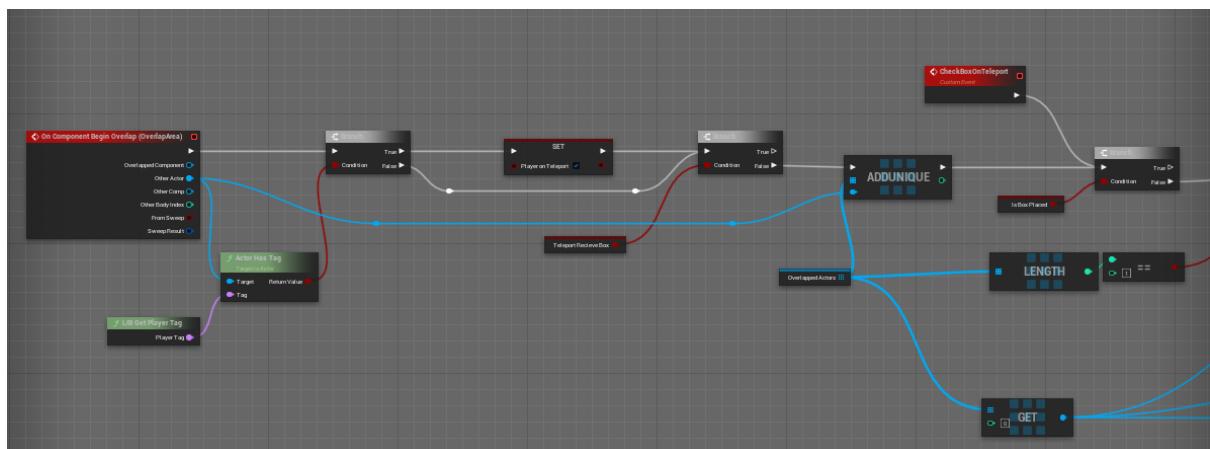
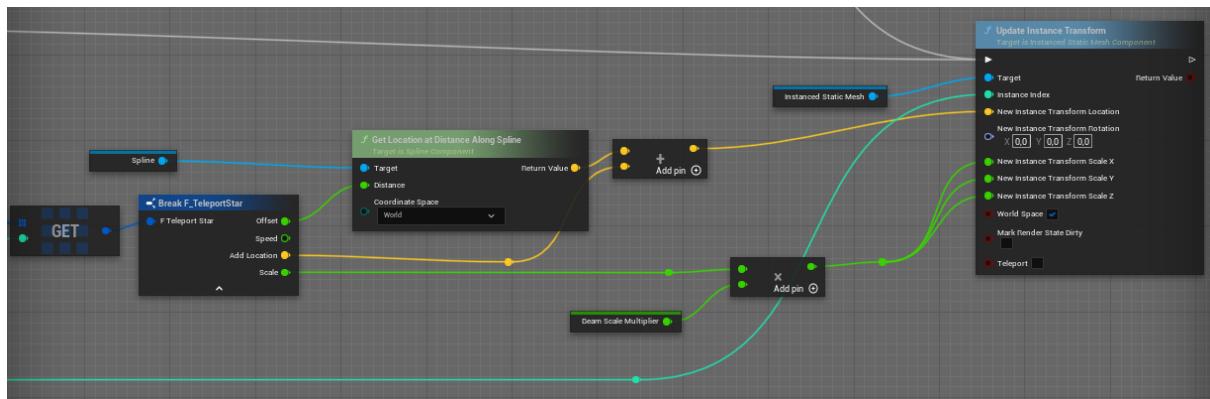


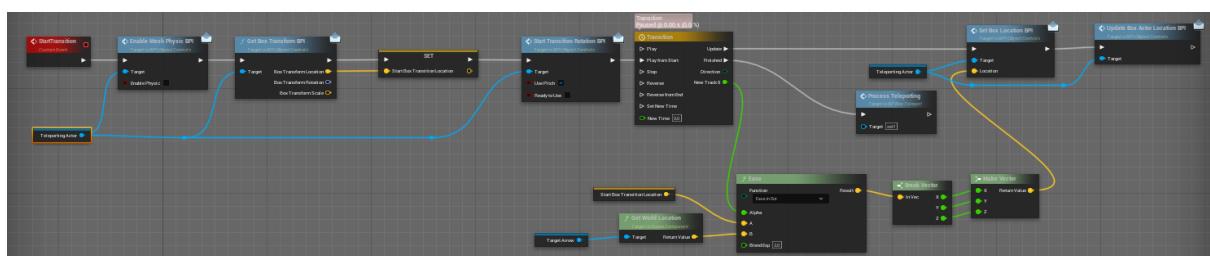
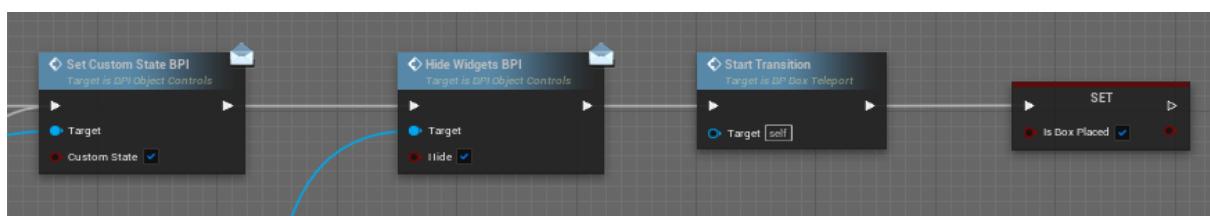
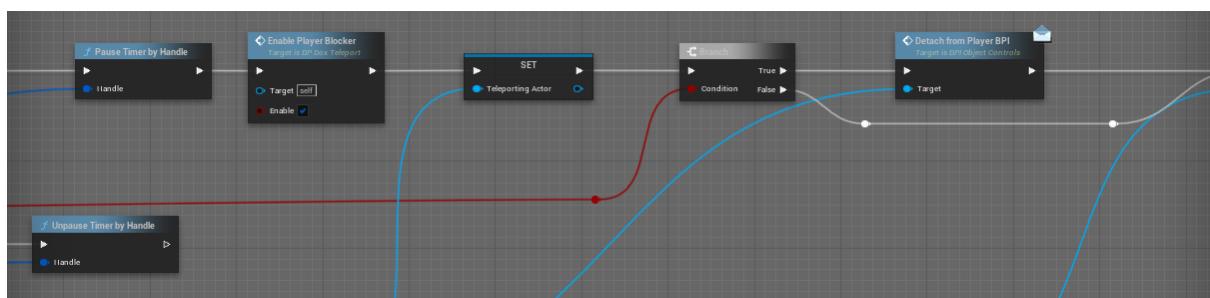
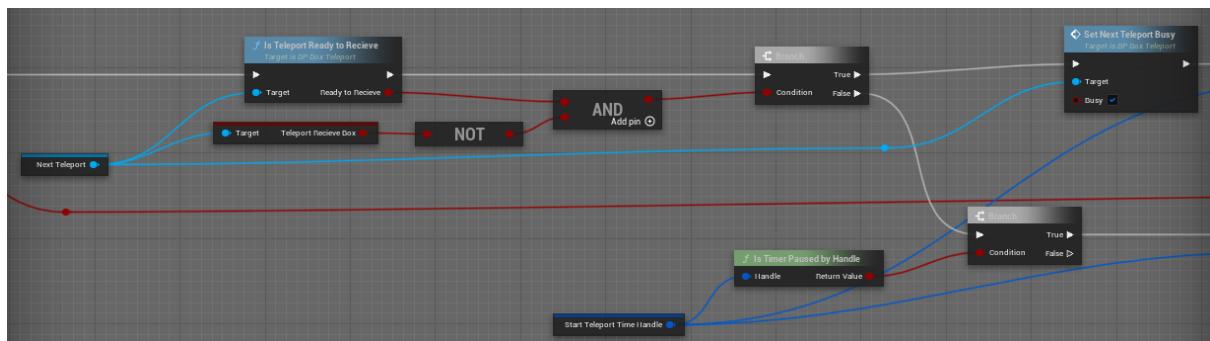
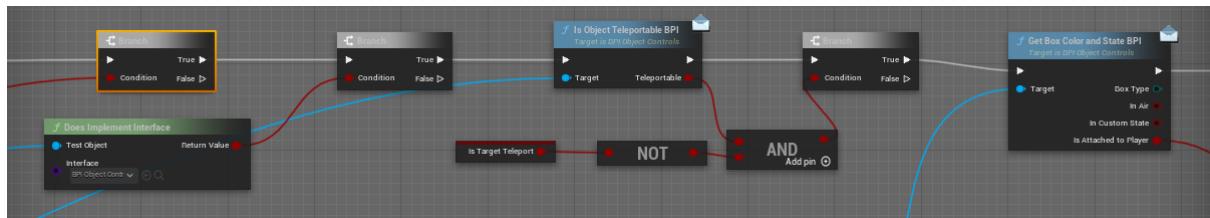
Teleport

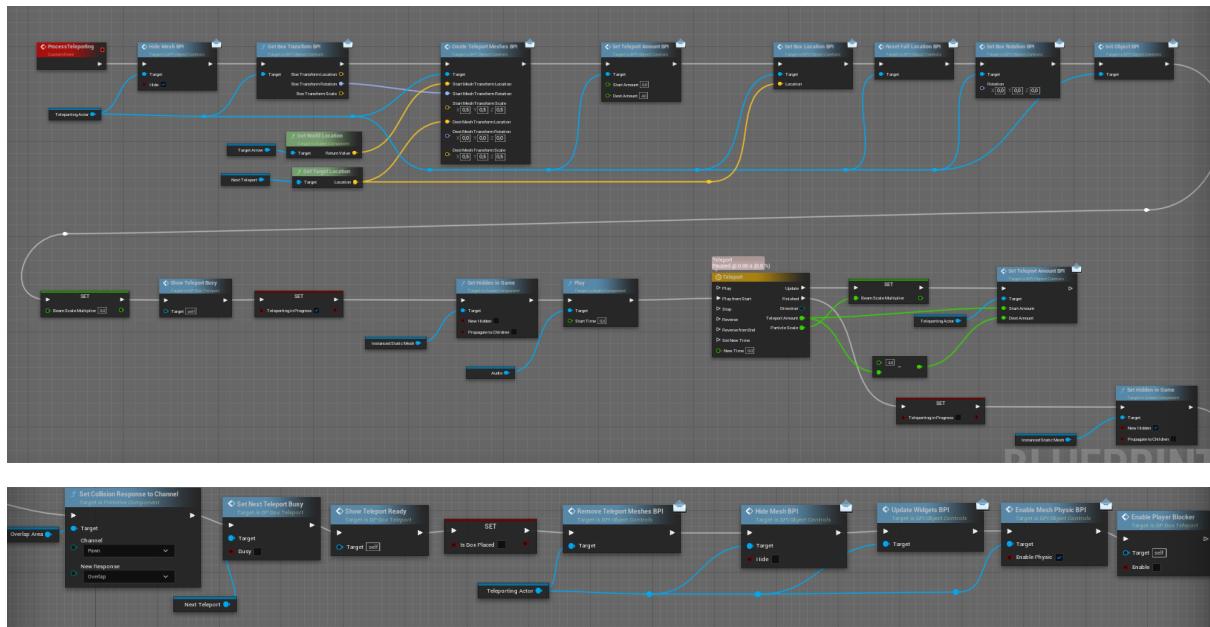




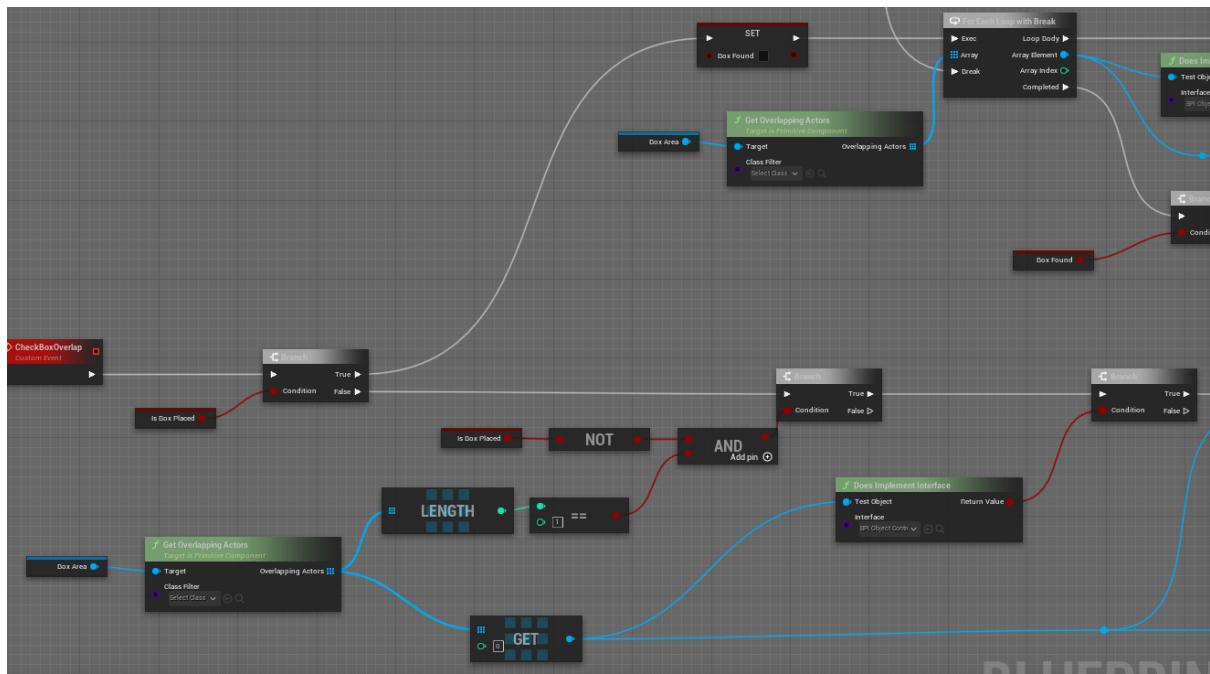
BEAMEDPRINT

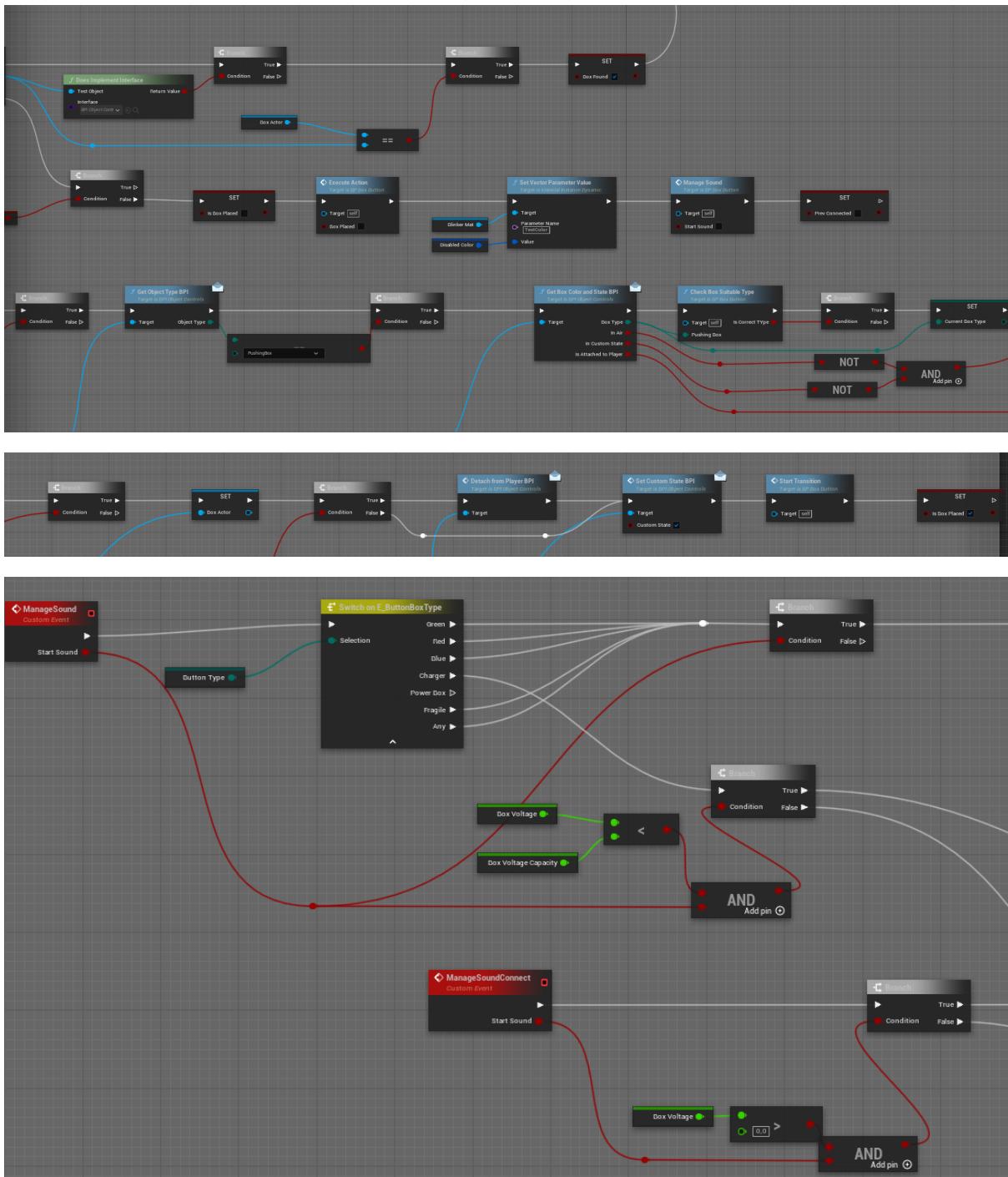


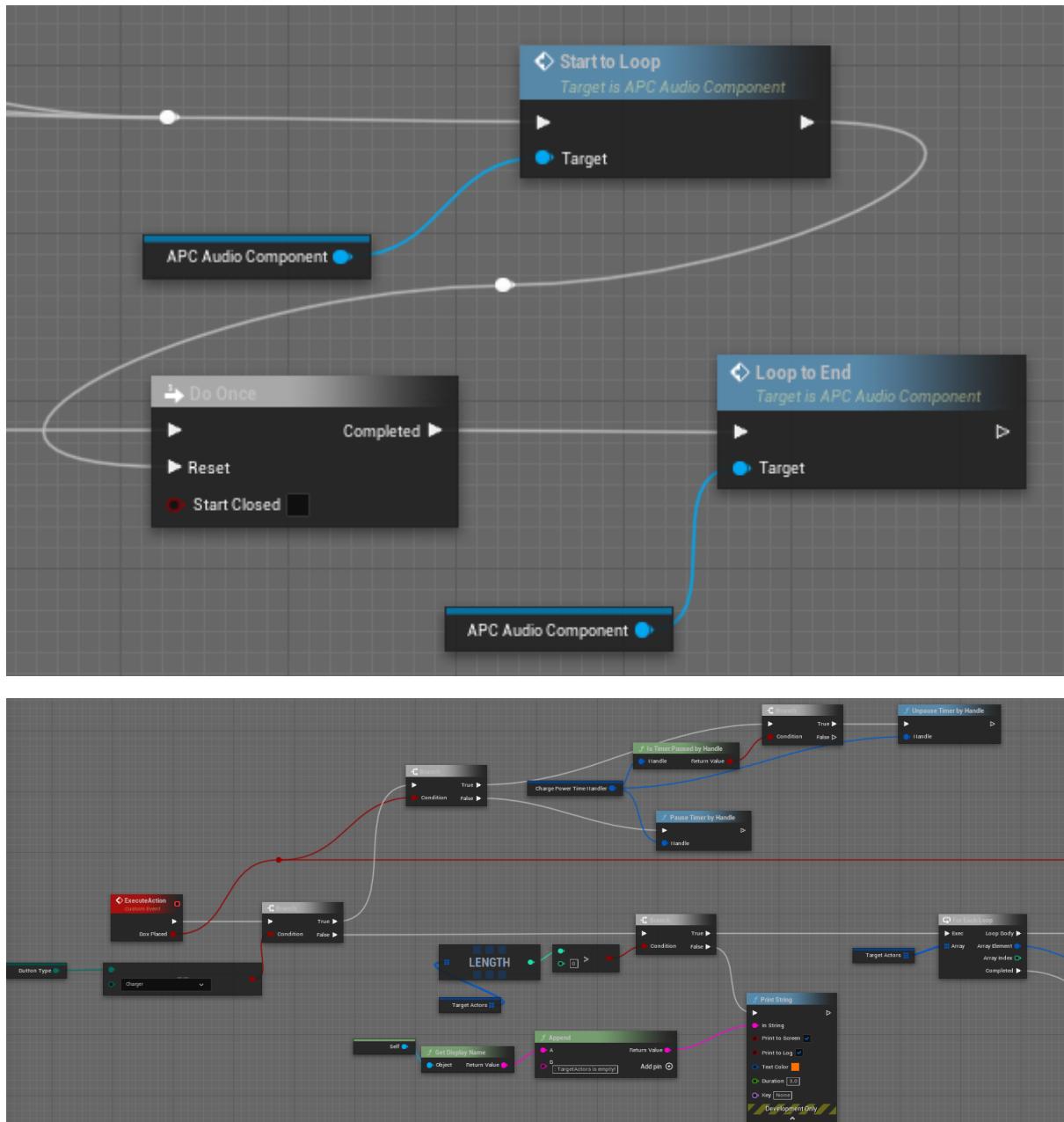


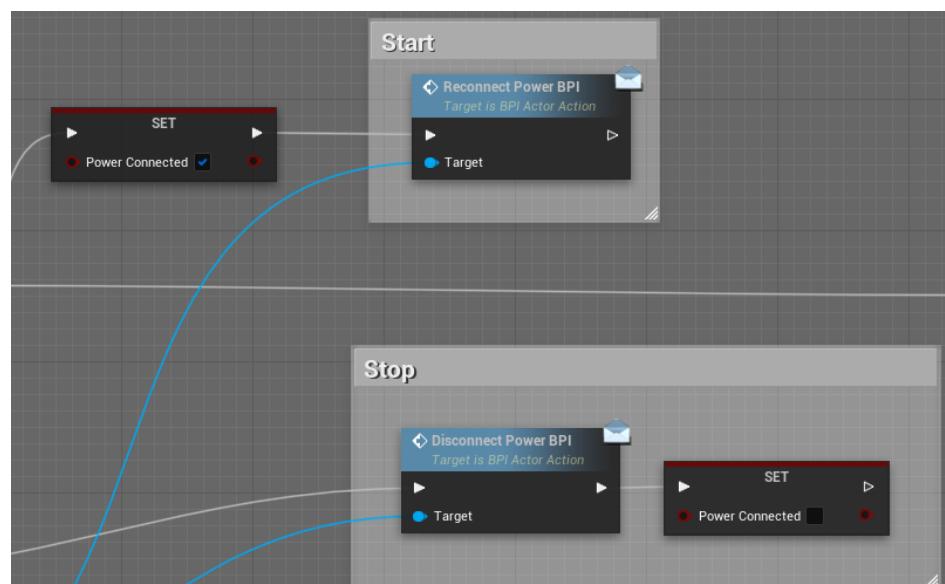
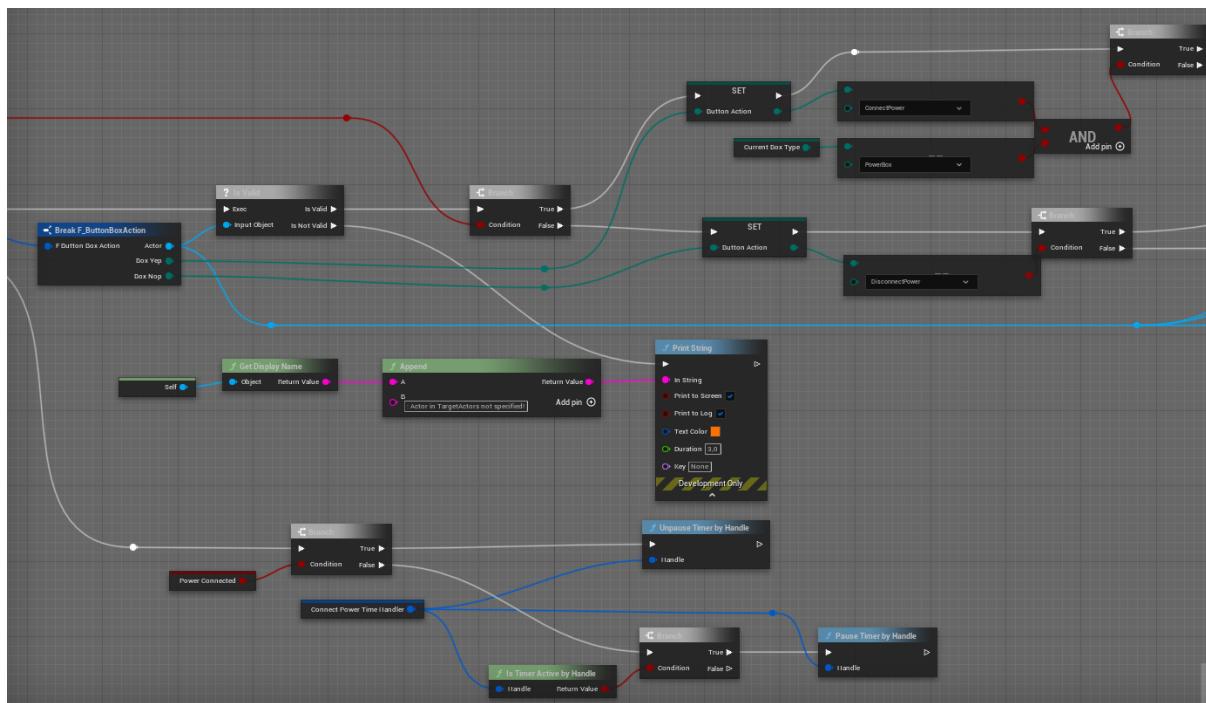


Platforma button

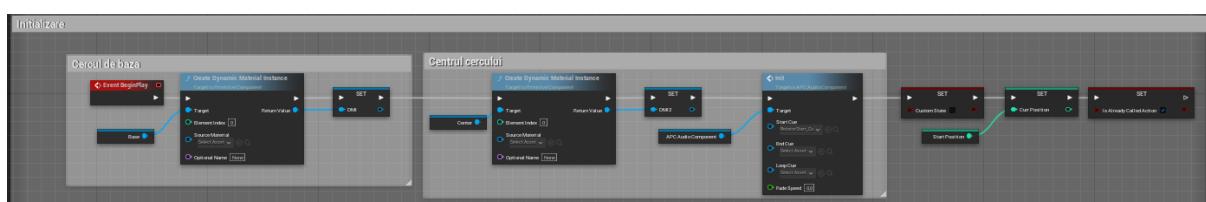


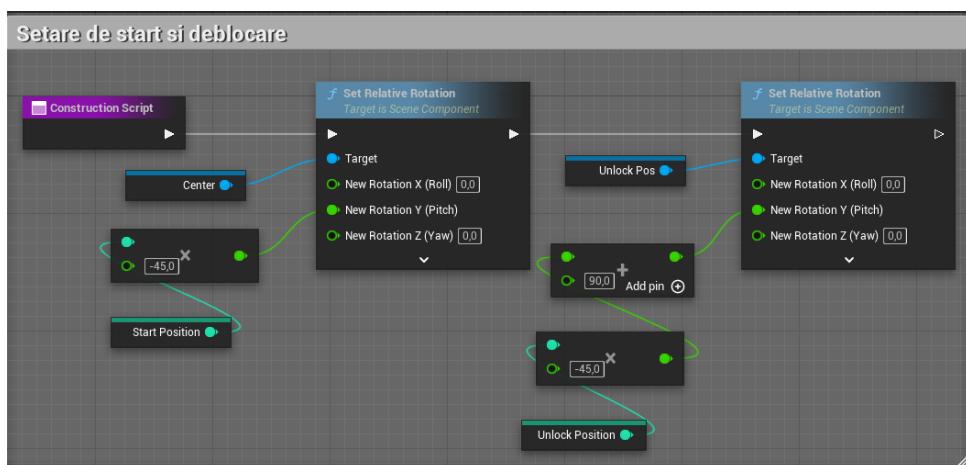
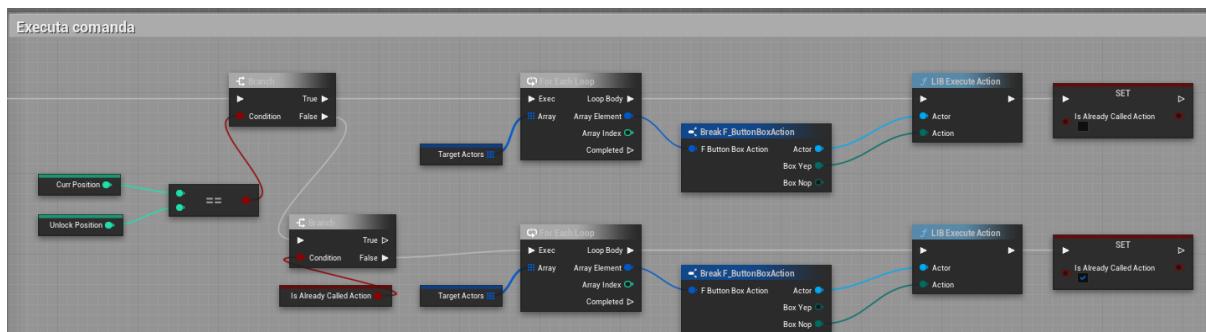
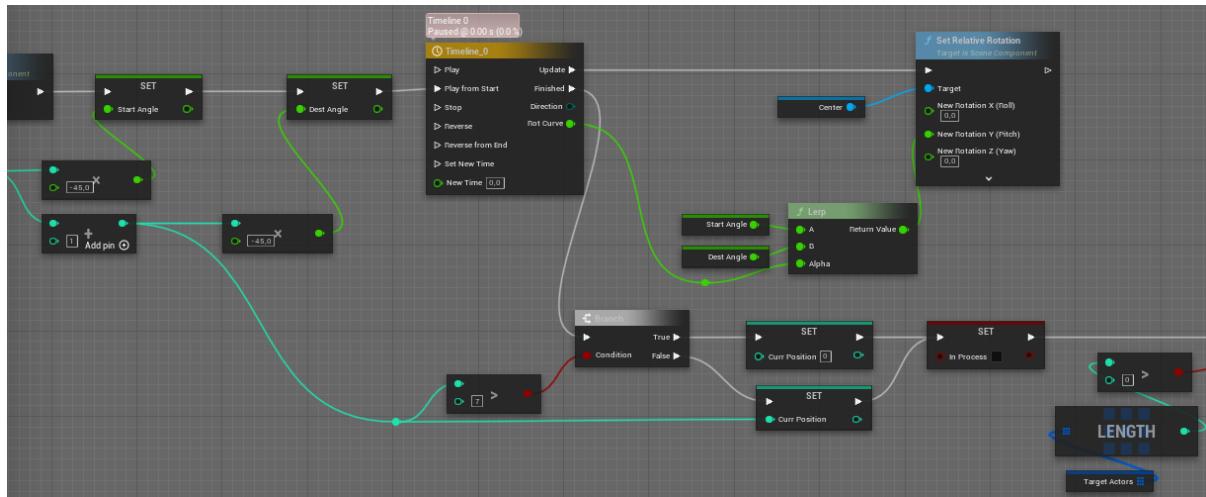
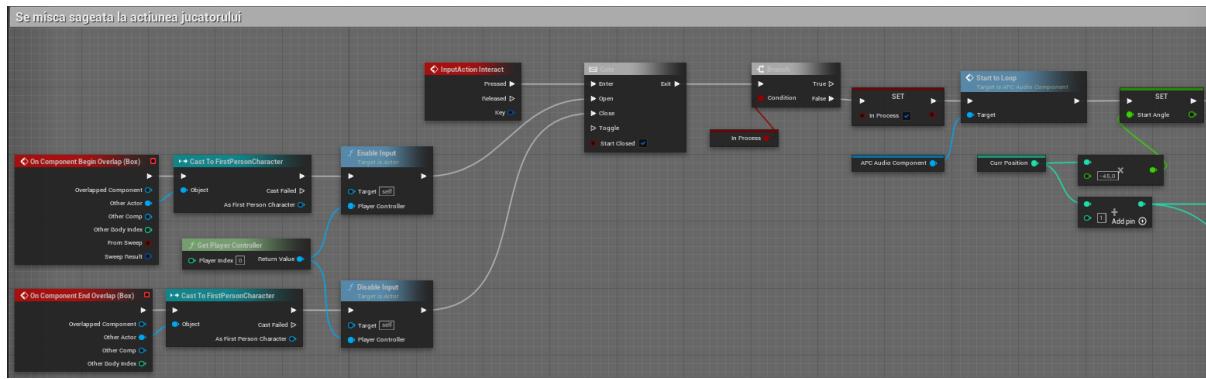






Cadran





Placa de presiune

