Informatică
2021-2022

Universitatea Politehnica din Timișoara

JOC EDUCATIV 2D PENTRU ELEVI

Candidat: Delia Bade

Coordonator științific: Ș.l.dr.ing Oana-Sorina Chirilă

Sesiunea: Iunie 2022

REZUMAT

Prezenta lucrare de licență constă într-un joc educativ 2D pentru elevi, în special elevii de clasele V-VIII, care conține întrebări de cultura generală din sfera materiilor studiate la orele de la școală și nu numai.

Jocul a fost realizat în platforma de dezvoltare Unity [1], care permite crearea unor jocuri atât 2D, cât și 3D. 2D înseamnă că jucătorul se poate deplasa doar în două direcții: sus și jos, dreapta și stânga. Alegerea dimensiunii jocului (2D, bidimensională) este o simplă preferință personală, având o înclinație pentru jocurile din acest domeniu.

Jocul cuprinde 3 nivele: începător, mediu și experimentat. Fiecare nivel conține întrebări de o dificultate specifică numelui și o hartă diferită pentru fiecare caz. Întrebările au fost formate, în mare parte, în urma unui studiu a programelor școlare specifice claselor de gimnaziu de pe site-ul oficial al Ministerului Educației Naționale [2], dar au fost adăugate și câteva întrebări din domeniile care mă pasionează, cum ar fi mitologia greacă.

Scopul jucătorului este să treacă prin fiecare nivel, să răspundă fiecărei persoane cu întrebări și să nu își piardă viața în același timp. Dacă s-au îndeplinit condițiile menționate anterior, jocul se va sfârși, iar punctajul total va fi afișat. Pentru fiecare întrebare răspunsă corect se acordă un punct. În cazul în care viața ajunge la 0, jocul se va termina și nu se va acorda niciun punctaj. Viața se resetează în momentul în care se trece la următorul nivel. Întrebările pentru fiecare NPC (non-player character) sunt păstrate în fișiere diferite de tip JSON, corespunzătoare NPC-ului cu același nume.

Harta fiecărui nivel a fost realizată cu ajutorul unor imagini speciale, specifice pentru jocurile 2D, denumite sprites. Acestea sunt, de fapt, o imagine creată din pixeli, care este și cauza aspectului pixelat al jocului, pe care eu îl consider un plus. Sprite-urile pot fi realizate prin intermediul unor programe software care permit prelucrarea pixelilor mai ușor, precum: Adobe Photoshop de la dezvoltatorul Adobe Inc., Aseprite, dezvoltat de David Capello, disponibil pe platforma Steam, etc. Prin intermediul editorului Aseprite, am realizat pictograma reprezentativă pentru numărul de persoane în fiecare nivel și imaginile cu butoane pentru tutorial [3].

Arta care înglobează crearea de sprite-uri noi se numește Arta Pixelilor și este o formă de artă digitală.

Unity mai pune la dispoziție elemente ce aparțin categoriei interfață cu utilizatorul (UI) și limbajul de programare C# pentru a comunica cu interfața. Astfel, cu ajutorul acestora, am putut realiza funcționalitatea jocului.

Cuprins

1 INTRODUCERE

1.1 ARGUMENTARE

Motivul pentru care am ales acest tip de proiect pentru lucrarea de licență este pasiunea mea pentru jocurile video, în special pentru jocurile 2D de tip RPG despre supraviețuire, construcție, explorare. Principalul joc care m-a inspirat și m-a făcut să-l îndrăgesc prin simplitatea sa, arta sa și povestea unică, destul de complexă, dar nu complicată, este Stardew Valley [4]. Alte jocuri pe care le-am jucat și care contribuie la motivul alegerii mele a proiectului de licență sunt: Terraria, Don't Starve, Don't Starve Together, Kynseed, Overcooked, Undertale, și multe altele.

1.2 TEMA

Temele alese pentru acest proiect au fost inspirate din temele jocurilor experimentate din trecut și până în prezent, și de tema aleasă în comun acord cu profesorul coordonator.

Astfel s-a ajuns la tema principală: educativă, care se combină cu tema explorării, tema reușitei, tema înfrângerii etc. Tema educativă apare prin crearea jocului cu întrebări de cultură generală, deoarece jucătorul își testează cunoștințele și în același timp dobândește informații noi de la NPC-uri. Întrebările sunt din domeniile următoare: limba și literatura română, matematică, limba engleză, limba franceză, geografie, istorie, biologie, chimie, mitologie greacă etc. Tema explorării se justifică prin descoperirea de locații noi, nivele noi și povești noi. Tema reușitei este prezentă datorită posibilității de a câștiga jocul prin răspunderea la toate întrebările disponibile fără a pierde toată viața, iar tema înfrângerii se datorează posibilității de a pierde jocul (viața jucătorului a ajuns la 0), caz în care se va afișa un mesaj corespunzător.

2 STUDIU ASUPRA JOCURILOR 2D

Pentru realizarea aplicației mele, am efectuat un studiu asupra unor jocuri din sfera 2D, pentru a putea identifica funcționalitățile pe care le oferă fiecare și pe care aș putea să le implementez și în jocul meu.

2.1 STARDEW VALLEY

Stardew Valley [4] este un joc de rol (RPG), în care ai rolul unui fermier care a primit ca moștenire din partea bunicului său, o fermă. Scopul jocului este de a-ți dezvolta propria fermă și de a renova centrul comunitar al orașului Pelican Town. Jocul nu are un final propriu-zis, jucătorul se poate juca până la nesfârșit, până a realizat tot ce se poate realiza în acest joc sau până când se plictisește, deoarece se poate opri în orice moment din a mai juca jocul.

Acest joc a fost conceput de la 0 de o singură persoană, Eric Barone, în decurs de 4 ani. Deși perioada de timp a fost una mai lungă, aceasta poate fi justificată din punctul de vedere a numărului de persoane minim în crearea jocului, cât și asupra faptului că intenția inițială a dezvoltatorului a fost să devină mai bun la programare [5]. Din punct de vedere personal, consider că Stardew Valley este unul dintre cele mai bine puse la punct jocuri de tip 2D. Jocul a vândut peste 10 milioane de copii, în contextul în care autorul ar fi fost fericit să ajungă la un număr de 10 mii: „*I remember saying, like, I would be happy if the game sold 10 thousands copies*” [6].

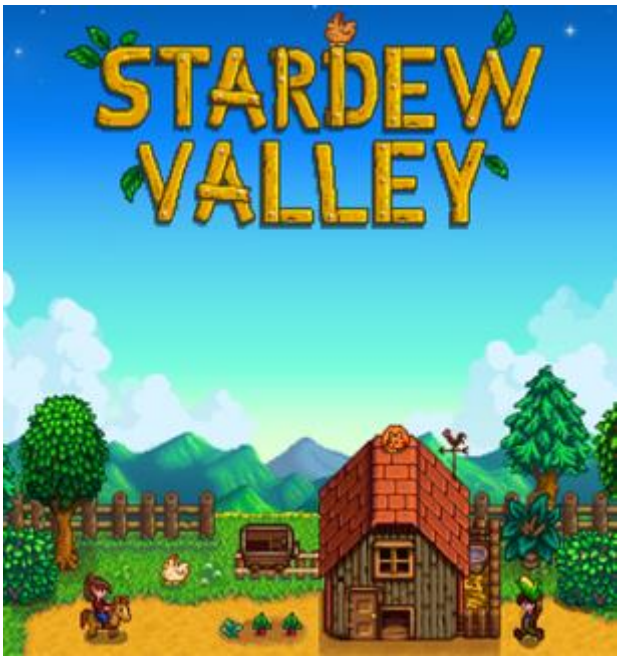

Figura 1 - Imagine reprezentativă a jocului Stardew Valley

Este de apreciat faptul că dezvoltatorul jocului, Eric Barone, a creat el însuși tot ce ține de joc, de la programare până la artă, muzică, poveste [5].

În prezent, Eric Barone lucrează la mai multe jocuri tot din sfera 2D, dintre care cel mai recent anunțat, în 8 octombrie 2021, este Haunted Chocolatier. Acest joc nu are o dată de lansare oficială [7].

2.2 TERRARIA

Terraria este un joc ce are ca temă principală supraviețuirea. Alte teme prezente sunt: explorarea, construcția, lupta și mineritul. Scopul jocului nu este unul predefinit, astfel, aș putea spune că principalul obiectiv este de a explora ceea ce poate oferi jocul. Jucătorului îi sunt oferite la început niște unelte cu care să înceapă și lângă el este generat un NPC, care îl va ajuta cu sfaturi despre progresie. Pe parcurs, se vor mai debloca iteme și vor mai apărea NPC-uri care vor asigura continuitatea în joc [8].

Un item, este un termen specific sferei jocurilor. El se referă la orice obiect care poate fi colectat de jucător, fie din lume, fie în urma completării unor misiuni, fie ca răsplată de la NPC-uri.

Conținutul este unul vast. Fiecare hartă este unică, fiind generată procedural. Din propria experiență, și având un număr de ore petrecute în joc care ajunge să atingă pragul de 100, pot confirma că numărul total de iteme pe care îl deține jocul este un număr imens. Acest lucru se datorează faptului că nu am reușit să descopăr toate itemele pe care acesta le poate oferi și nu am reușit să termin complet jocul încă, făcând o asociere și cu timpul petrecut în joc.

Figura 2 – Imagine reprezentativă a jocului Terraria

2.3 DON'T STARVE ȘI DON'T STARVE TOGETHER

Don't Starve și Don't Starve Together sunt 2 jocuri din sfera jocurilor 2D, de tip survival (supraviețuire) în sălbăticie, care au fost dezvoltate de către compania Klei Entertainment.

Cele două jocuri sunt similare, din punct de vedere al graficii, al modului de joc și al scopului, iar principala diferență este că primul joc, Don't Starve, este un joc singleplayer (care poate fi jucat doar de o singură persoană), iar Don't Starve Together, este un joc multiplayer (care poate fi jucat de mai multe persoane în același timp).

Tema principală a jocului, la fel ca și în cazul jocului prezentat anterior, este supraviețuirea, dar în cazul de față, aceasta este mult mai pronunțată. Sunt prezente elemente care țin de foame, de frig și de sănătate psihică. Scopul jucătorului este de a supraviețui monștrilor care se găsesc în lume, dar în același timp trebuie să aibă grijă de aceste aspecte de sănătate, care țin de propriul caracter [9].

Figura 3 – Imagini reprezentative pentru jocurile Don't Starve și Don't Starve Together

2.4 TABEL DE COMPARAȚIE

În urma realizării acestui studiu, am identificat o serie de funcționalități ce vor fi menționate în următorul tabel, și pe urmă analizate:

Tabel 1. Comparație în urma studiului asupra jocurilor 2D

	Stardew Valley	Terraria	Don't Starve & Don't Starve Together	Joc educativ 2D pentru elevi
Meniu	+	+	+	+
Sunete	+	+	+	+
Sistem viață	+	+	+	+
Sistem scor	-	-	-	+
Inventar obiecte	+	+	+	-
Existență NPC-uri	+	+	-	+
Posibilitatea de a construi	+	+	+	-
Monștri	+	+	+	+
Tutorial	+	+	-	+
Nivele multiple	+	-	-	+
Arme	+	+	+	-
Dialog	+	+	-	+

În urma acestui tabel se observă anumite caracteristici comune pentru toate cele 3 jocuri. Acestea fac parte dintr-o temă care le cuprinde pe toate, și anume lupta. Pentru a putea reprezenta lupta într-un joc, ai nevoie de inamici (monștri), arme, inventar de obiecte. În cazul proiectului meu de licență, tema aleasă nu include în totalitate elementele din această sferă, deoarece nu s-a pus accentul pe luptă. Singurul element prezent sunt NPC care iau forma unor monștri, dar aceștia au o personalitate prietenoasă și oferă jucătorului informații din sfera de divertisment. Totuși, aceste NPC-uri se găsesc într-o scenă adițională, secretă și nu modifică parcursul jocului dacă ar lipsi. Accentul a fost pus pe sistemul de dialog, pe sistemul de întrebări de tip grilă și pe sistemul de scor (scopul principal).

Meniul ar trebui să fie prima componentă a oricărui joc, așadar acesta este prezent în toate jocurile menționate.

Sunetele dau viață unui joc, pot fi sub diferite forme, dar în general în toate jocurile se rulează o muzică de fundal. În cazul jocului meu, muzica de fundal este diferită pentru fiecare scenă în funcție de atmosfera pe care o redă harta scenei.

Sistemul de viață este un element cu care se poate merge în diferite aspecte. În cazul jocului Stardew Valley, nu se pune accent pe acesta, bara de viață apărând doar în cazul în care jucătorul intră într-o zonă specială și începe să primească lovituri de la monștrii din jur, dar în cazul în care viața ajunge la 0, jocul nu se sfârșește. În cazul celorlalte jocuri, viața este un element definitoriu, deoarece acesta poate pune capăt jocului, sau va aduce pierderi majore. Cazul implementat în lucrare este cel de-al doilea, viața având un impact destul de mare asupra modului de joc.

Sistemul de scor este un pic generalizat în această situație. Fiecare joc în sine are un sistem pe care îl consideră scor (prin banii, bunurile, misiunile rezolvate), dar aici am făcut referință directă la punctajul obținut. În jocul creat de mine, pentru fiecare întrebare răspunsă corect se acordă 1 punct, iar suma acestor puncte va reprezenta scorul total. În cazul celorlalte jocuri, nu există un punctaj definit concret, dar se pune accentul pe altceva, cum ar fi: construcții și animale în cazul Stardew Valley, obiecte (arme, decorațiuni) în cazul Terraria și numărul de zile supraviețuite în cazul Don't Starve Together.

NPC-urile sunt prezente în majoritatea jocurilor, și sunt o caracteristică cu care se poate jongla destul de ușor, deoarece se pot reprezenta: o poveste fictivă a unui personaj, un tutorial prin interacțiune/dialog, ajutor în progresia jocului/caracterului etc. Acestea sunt prezente în aplicația mea pentru a ajuta jucătorul să progreseze, pentru a pune la dispoziție întrebările de cultură generală și pentru a împărtăși cunoștințe jucătorului.

Funcționalitatea de a construi diferite clădiri este o funcționalitate destul de complexă, care necesită mai mult timp de dezvoltare, dar care este o adiție impresionantă oricărui joc. Jocurile menționate mai sus dețin această funcționalitate, dar nu a fost implementată în lucrarea mea de licență. Însă, în aplicația mea, am creat case ale NPC-urilor cu care se poate interacționa (se poate intra înăuntru).

Tutorialul reprezintă un ajutor venit jucătorului. Acesta poate fi sub diverse forme: se specifică controalele folosite pentru a realiza anumite acțiuni, se oferă îndrumări spre locul unde trebuie să se ajungă, se derulează un video explicativ etc. La fel ca în cazul de mai sus, și această funcționalitate este prezentă în majoritatea jocurilor, dar, spre exemplu, în cazul jocului Don't Starve Together nu este implementată, iar eu consider că lipsa tutorialului poate fi justificată de dorința dezvoltatorilor ca jucătorul să experimenteze singur ceea ce poate oferi aplicația, dar în același timp și de a adăuga jocului un grad de dificultate mai mare.

Proiectul meu de licență conține nivele multiple, și anume: începător, intermediar și experimentat. Fiecare nivel este reprezentat de o scenă diferită, de o grafică diferită și de alte NPC-uri cu întrebări de o dificultate corespunzătoare nivelului. Pe lângă cele 3 scene, mai există: o scenă secretă (pe care jucătorul o poate accesa prin intermediul unei uși ascunse în ultimul nivel, după un copac) și o scenă care conține toate zonele de interior ale clădirilor din ultima scenă. Acestea sunt adiționale și sunt legate de ultimul nivel.

3 FUNDAMENTARE TEORETICĂ

3.1 UNITY

Unity este un motor de joc dezvoltat de Unity Technologies. Acesta oferă posibilitatea de a crea jocuri atât 2D, cât și 3D. Există un număr mare de creatori de jocuri care folosesc Unity, iar 50% din jocurile de pe telefon, de pe PC și de pe consolă sunt realizate cu Unity.

Conform, site-ului oficial Unity, există peste 190 de țări în care se regăsesc creatori de jocuri video care utilizează Unity [10].

C# este principalul limbaj de programare folosit de motor [11]. Pentru jocurile 2D, Unity permite utilizatorului să importe sprite-uri și oferă un renderer al lumii 2D avansat.

Unity este, în același timp, gratuit și pune la dispoziție pe site-ul oficial o categorie de învățare, pentru a atrage cât mai multe persoane.

Unity pune la dispoziție clasa **ScriptableObject**, care permite stocarea de date ce sunt independente de instanțele claselor [12]. De asemenea, Unity oferă posibilitatea de a crea un obiect, modificarea lui în modul dorit și după salvarea obiectului în starea în care se află și pe urmă posibilitatea de a crea același obiect, cu aceleași modificări în aceeași scenă sau în altă scenă, prin intermediul sistemului de **Prefabs**. Acestea sunt o modalitate bună de a păstra obiectele modificate și de a putea crea în orice moment o nouă instanță, dacă este nevoie [13].

3.2 LIMBAJUL DE PROGRAMARE C#

Limbajul de programare C#, care se pronunță „C-sharp”, este un limbaj modern, orientat pe obiecte [11].

C# este un derivat al limbajului de programare C++, este similar din punct de vedere sintactic cu Java și este ușor de învățat pentru utilizatorii care au deja cunoștințe de C, C++ sau Java.

Limbajul C# prezintă numeroase motive pentru care argumentează de ce este un limbaj așa de popular și cerut, printre care cele de mai sus, dar și prin faptul că produce programe eficiente, poate fi compilat pe o varietate de platforme și este parte din .Net Framework [14].

C# poate fi caracterizat de următoarele adjective: simplu, modern, orientat pe obiecte, sigur, structurat, bogat în librării și rapid [15]. Fiind un limbaj care are la bază principiile de OOP (object-oriented programming), este o resursă importantă și adecvată în crearea jocurilor video.

3.3 JSON

JSON (**J**ava**S**cript **O**bject **N**otation) este un format de text pentru a stoca și transporta datele, este ușor de scris și de citit de către oameni, dar și ușor de analizat și generat de către mașini. Este independent de limbaj [16].

Un fișier JSON poate fi alcătuit din două structuri: de tip obiect și de tip vector. Sintaxa acestora se poate observa în figura 4.

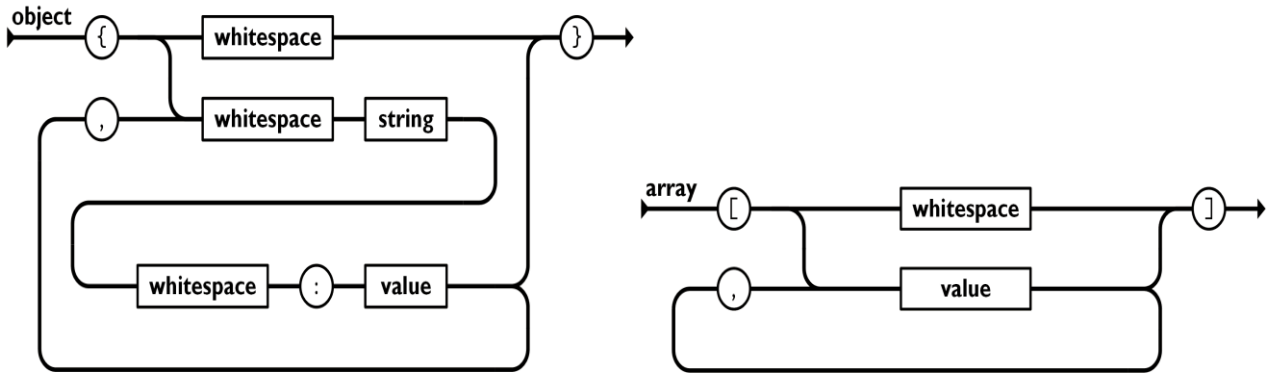

Figura 4 – Structura unui fișier de tip JSON

Valoarea (value) poate fi de mai multe tipuri, poate fi un șir de caractere, un număr întreg, un alt obiect, un vector de obiecte, poate avea valoare booleană, adică adevărat sau fals sau poate să nu aibă nicio valoare, adică null (figura 5).

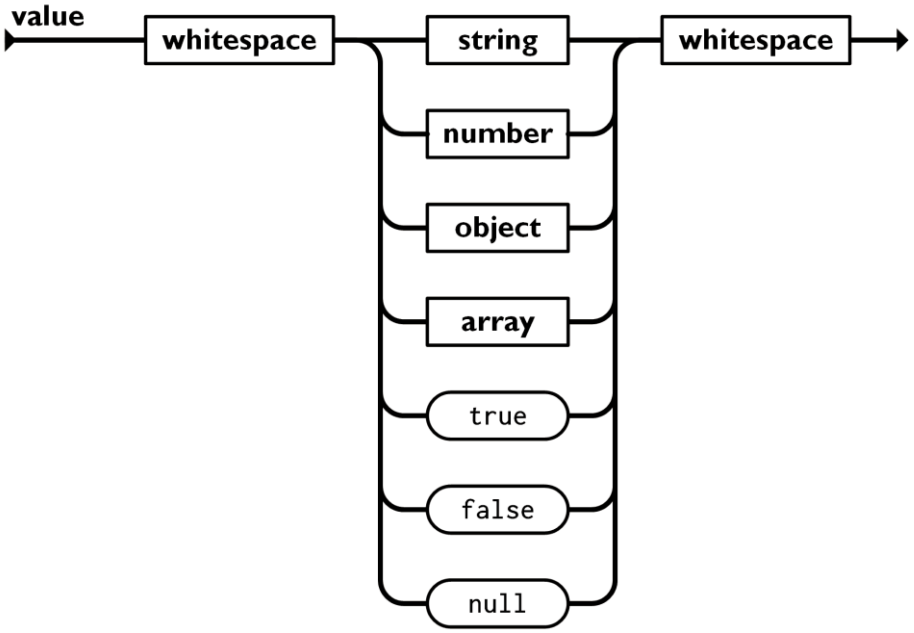

Figura 5 – Tipuri de valori

4 PROIECTAREA APLICAȚIEI

4.1 ARHITECTURA APLICAȚIEI

În figura 6, poate fi observat modul în care este structurat jocul:

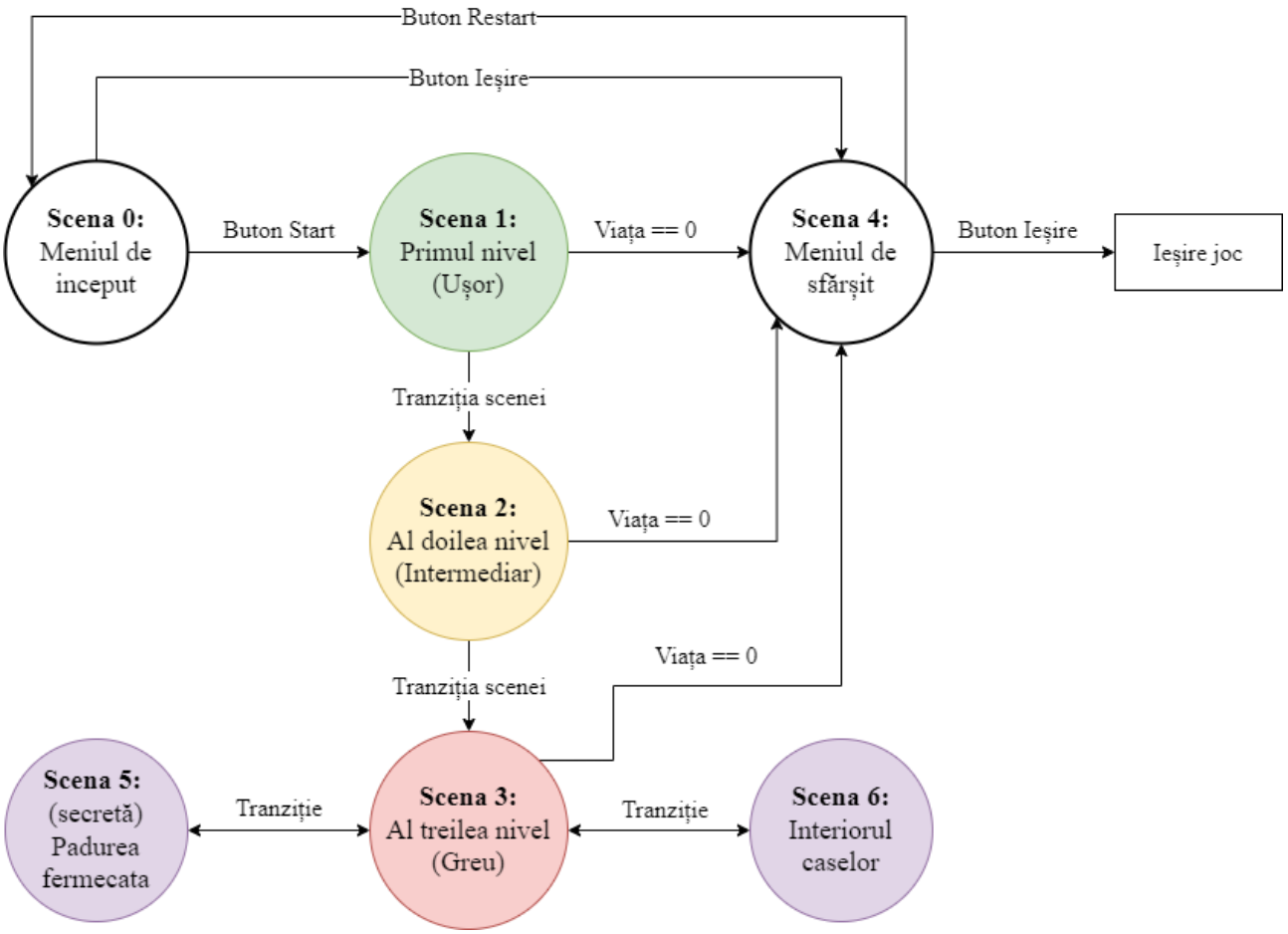

Figura 6 – Arhitectura jocului

4.2 STĂRILE CARACTERULUI

Stările caracterului pot fi de repaus, mișcare sau alergare, precum se observă în figura 7.

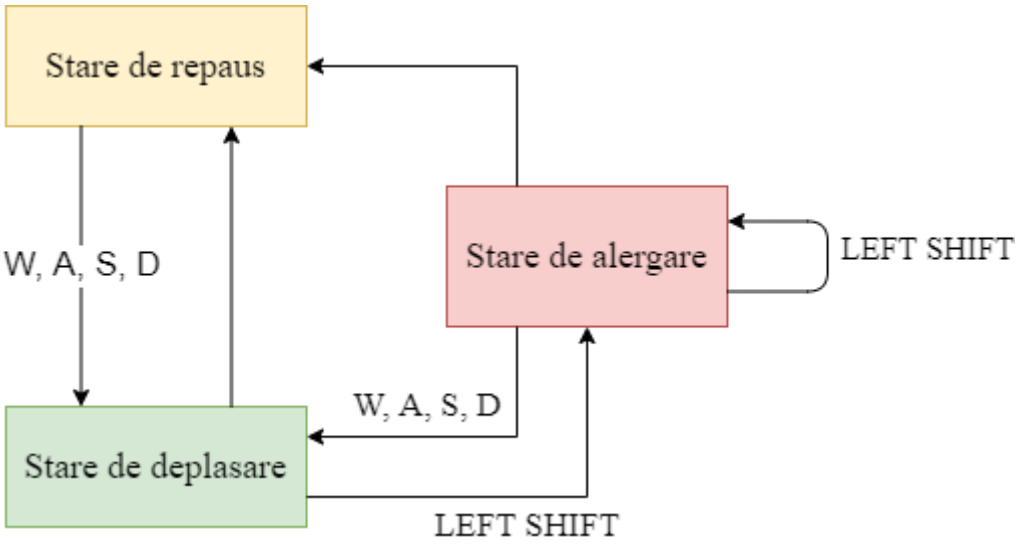

Figura 7 – Stările caracterului

Animațiile pentru stările de mers la pas și alergare sunt redate în funcție de direcția pe care o are caracterul. În următoarea figură, 8, este reprezentat fiecare caz.

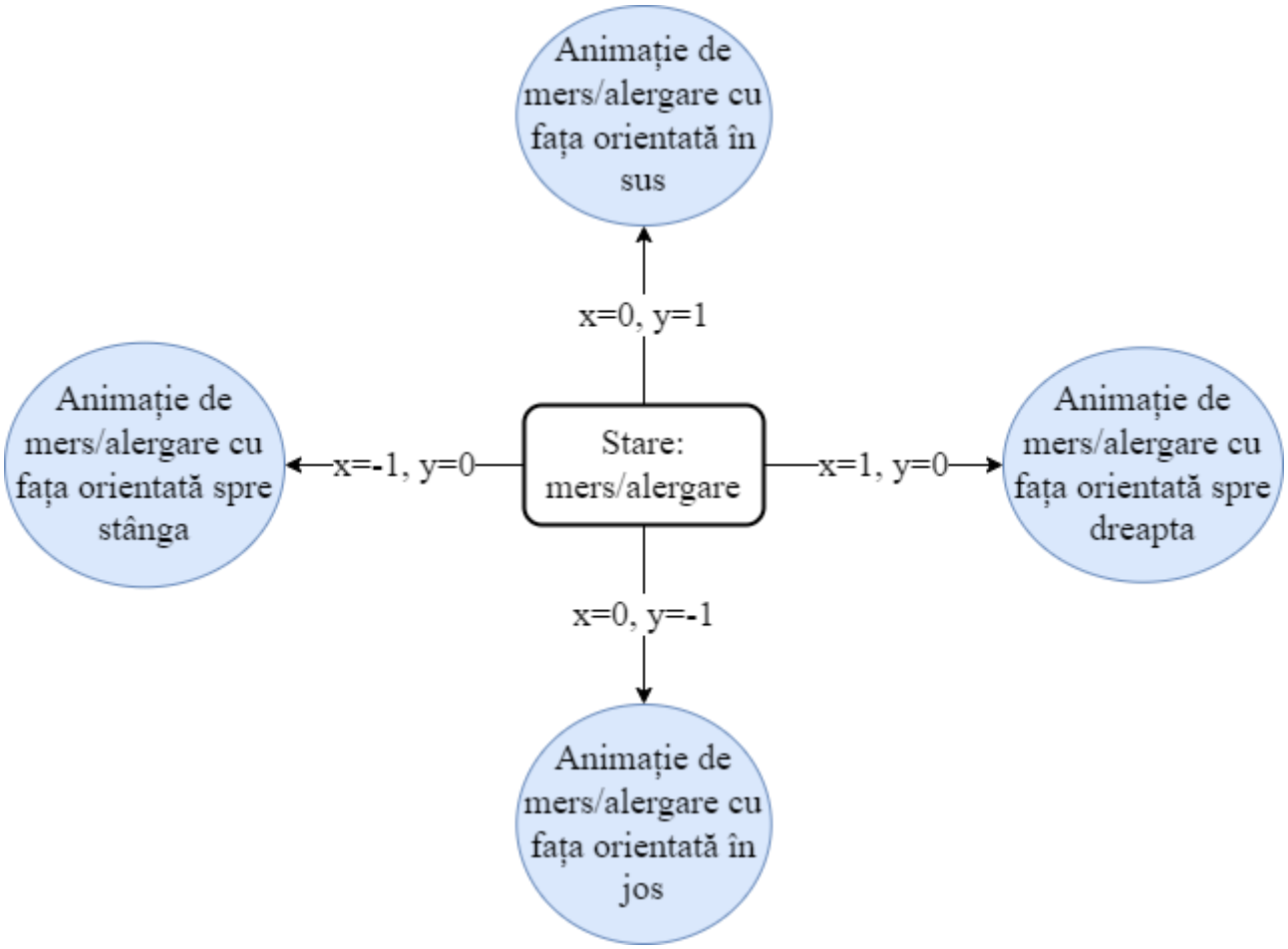

Figura 8 – Animațiile caracterului

4.3 ARHITECTURA SISTEMULUI DIALOG ȘI SISTEMULUI ÎNTREBĂRI

Cele două sisteme, pentru dialog și pentru întrebări, sunt în strânsă legătură, deoarece sistemul de întrebări poate fi declanșat doar de sistemul de dialog. Astfel, cele 2 sunt legate între ele. În următoarea figura, 9, am prezentat modul în care se declanșează cele două sisteme.

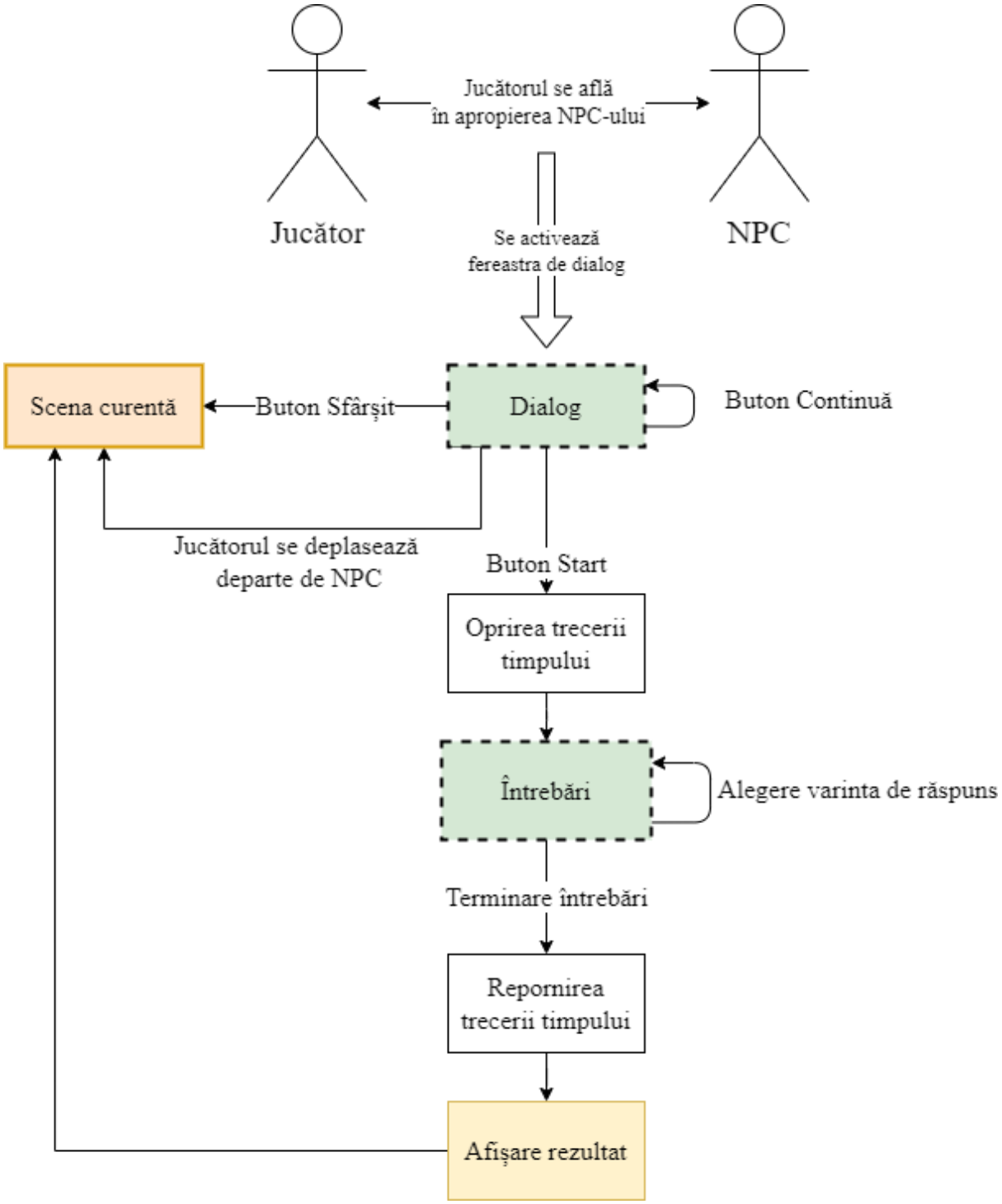

Figura 9 – Funcționalitatea sistemului dialog și întrebări

5 FUNCȚIONALITATEA APLICAȚIEI

În acest capitol vor fi detaliate componentele și funcționalitatea jocului. Librăria care ușurează interacțiunea cu utilizatorul, prin punerea la dispoziție a mai multor elemente care țin de UI este UnityEngine.UI. Aceasta este indispensabilă atunci când se lucrează cu interfața cu utilizatorul, și de aceea este inclusă în majoritatea scripturilor folosite în aplicație.

5.1 STRUCTURA SCENELOR

În cadrul unui joc, scenele sunt asociate cu pânza pictorului pe care se lucrează în Unity, pentru a crea o interfață cu utilizatorul (cel care se joacă) și prin intermediul cărora jocul prinde viață. Scenele sunt create cu ajutorul sprite-urilor.

În cazul jocului meu, am creat mai multe scene, pe care le voi clasifica în următorul mod:

- principale: Level 1 (Easy), Level 2 (Medium), Level 3 (Hard)
- secundare: MainMenu, EndMenu
- adiționale: Secret, Interior

Scenele principale sunt cele mai importante, deoarece în cadrul acestora are loc acțiunea propriu-zisă, adică explorarea lumii și răspunderea la întrebările oferite de NPC-uri. Scenele secundare sunt scenele care conțin meniul de început și meniul de sfârșit al jocului. Scenele adiționale sunt scene care pot lipsi din joc, fără a avea un impact major în funcționalitatea aplicației, dar contribuie la poveste.

Fiecare dintre aceste scene are atribuit un buildIndex – un index de creare, care ajută la managementul scenelor prin intermediul codului și reprezintă, de fapt, și ordinea în care au fost adăugate scenele în setările de construcție ale aplicației. Indicele începe de la 0 și se termină la numărul de scene - 1. Fiecare scenă primește un indice crescător de la 0, în ordinea adăugării lor [17].

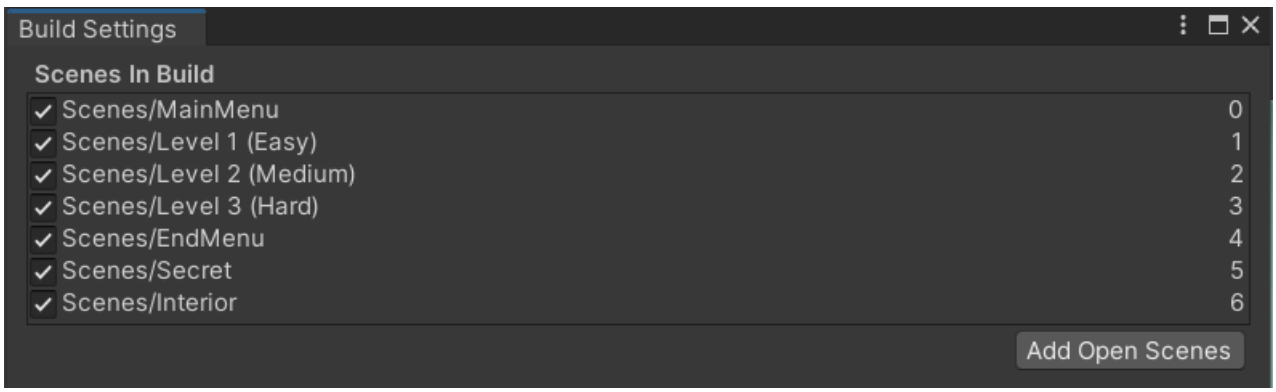

Figura 10 – Captură de ecran din setări

Prima scenă, pe care am denumit-o sugestiv MainMenu, având indicele 0, este scena de început, de start, și ea este primul lucru cu care interacționează jucătorul.

Aceasta conține (Figura 11): camera principală - *Main Camera*, o componentă prefabricată - *AudioManager*, și, predominant, componente de interfață cu utilizatorul (UI), cuprinse în obiectul *Canvas*, alături de care este creat automat obiectul *EventSystem*, care este responsabil pentru procesarea și gestionarea evenimentelor. În lipsa acestuia, componentele UI nu vor avea niciun efect, deoarece evenimentele nu vor fi procesate și gestionate. Dacă acest obiect lipsește, se poate adăuga prin click dreapta în zona de ierarhie a scenei, și din meniu se va alege UI -> Event System. Este important de menționat că este nevoie de doar o singură componentă de tip Event System într-o scenă [18].

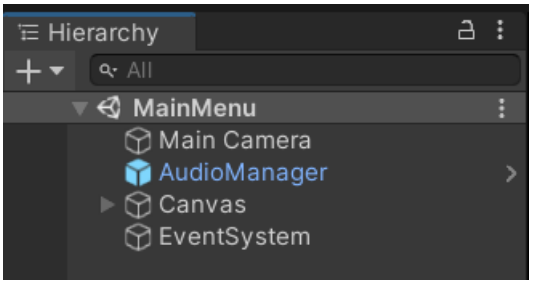

Figura 11 – Captură de ecran a componentelor scenei de început (MainMenu)

Obiectul care cuprinde toate elemente de UI, precum am precizat mai sus, se numește pânză (Canvas din figura de mai sus).

În interiorul pânzei, figura 12, se află o imagine ce servește drept imagine de fundal pentru meniu, BackgroundPanel, și încă 2 obiecte, MainMenuContainer și OptionsMenuContainer, care cuprind butoane, text sau slider ce au atribuite evenimente. Evenimentele activate prin apăsare a butoanelor se numesc OnClick.

Doar un singur obiect este activat la un moment dat, OptionsMenuContainer este cel dezactivat și se poate observa prin culoarea gri a textului, iar atunci când se face schimbarea între meniuri, meniul curent va fi dezactivat și celălalt meniu va fi activat.

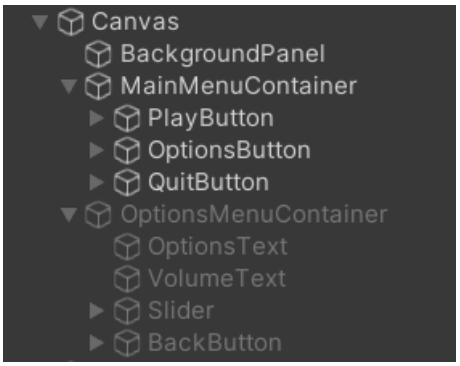

Figura 12 – Componența obiectului Canvas

În figura 13, butonul PLAY va lansa în execuție jocul, prin încărcarea primei hărți, Level 1 (Easy). Butonul OPTIONS, va face vizibil meniu de opțiuni (Figura 15). Butonul QUIT va închide aplicația.

Figura 13 – Captură de ecran a scenei de început (MainMenu)

În figura 14, este prezentat meniul de opțiuni. Prin intermediul slider-ului, obiectului care se poate trage în dreapta sau în stânga, se poate modifica volumul sunetului de fundal pentru tot parcursul jocului, iar butonul BACK va încărca meniul de start din figura anterioară.

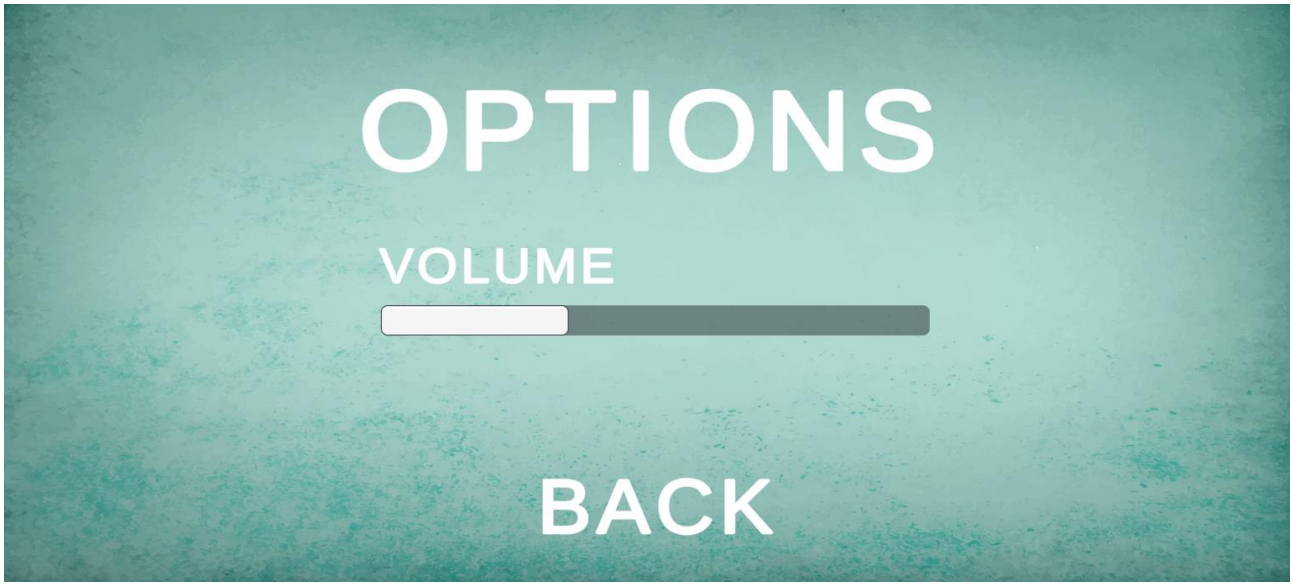

Figura 14 – Captură de ecran a meniului de opțiuni

A doua scenă, cu index 1, este Level 1 (Easy). Această scenă reprezintă primul nivel cu care se va confrunta jucătorul. Gradul de dificultate este ușor. Harta scenei poate fi văzută în figura 15.

Figura 15 – Captură de ecran a primului nivel

A treia scenă, cu index 2, este Level 2 (Medium). Această scenă este următoarea scenă care va fi încărcată. Conține întrebări de un grad de dificultate intermediar/mediu. Harta scenei poate fi observată în figura 16.

Figura 16 – Captură de ecran a nivelului 2

A patra scenă, cu index 3, este Level 3 (Hard). Harta scenei se poate observa în Figura 17. Din cadrul acestei scene se poate merge în scenele adiționale, scenele cu index 5 și 6, și se poate reveni înapoi.

Figura 17 – Captură de ecran a nivelului 3

A cincea scenă, cu index 4, este scena care reprezintă meniul de sfârșit. La fel ca și scena care conține meniul de început, ea este alcătuită identic, din: Main Camera, AudioManager, Canvas, EventSystem.

Pe pânză, adică înăuntrul obiectului Canvas, au fost create 2 butoane: RESTART, care în urma apăsării acestuia se rulează jocul de la început, și QUIT, care va determina ieșirea din joc.

Figura 18 – Captură de ecran a meniului de sfârșit

A șasea scenă, cu index 5, este una dintre scenele adiționale. Această scenă are o intrare secretă în scena precedentă, în partea dreaptă sus, în spatele unui copac. Conținutul scenei poate fi observat în figura 19. În interiorul acestei scene se găsesc NPC-uri monștri, dar care sunt prietenoși. Aceștia nu pun întrebări, dar dialogul lor se concentrează pe oferirea de informații sfera jocurilor, filmelor, serialelor. Copacii din această

scenă sunt generați în mod aleatoriu, prin intermediul codului, pentru a dezvălui aerul misterios, fantastic și neobișnuit al scenei, în care niciun copac nu a fost cum era înainte.

Figura 19 – Captură de ecran a scenei secrete

A șaptea scenă, cu index 6, este a doua dintre scenele adiționale. În această scenă sunt reprezentate toate interioarele caselor din scena cu ultimul grad de dificultate. Tranziția dintre cele două scene se face prin ușa deschisă. Captura de ecran a scenei este figura 20.

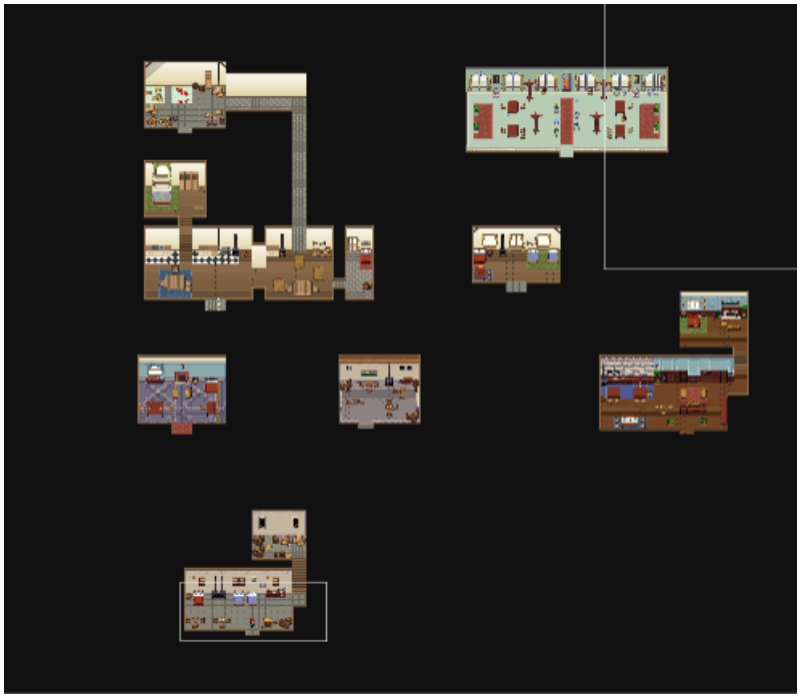

Figura 20 – Captură de ecran a scenei Interior

5.2 CREAREA SCENELOR PRINCIPALE

Crearea scenelor principale [19] a fost realizabilă utilizând sprite-uri gratuite atât de pe site-ul opengameart.org, cât și din Unity Asset Store, care este o librărie de unde se poate descărca și utiliza orice resursă disponibilă care este gratuită sau se pot cumpăra resurse.

Site-ul opengameart, precum îi spune și numele, pune la dispoziție majoritatea resurselor ca fiind gratuite de descărcat și utilizat, doar unii creatori cer a fi amintiți în cazul în care se utilizează arta creată de aceștia. Voi aminti fiecare autor, împreună cu sprite-urile folosite de la ei, într-un capitol de la finalul documentului curent.

Unele dintre setările sprite-urilor au fost modificate, astfel încât să corespundă setărilor necesare pentru a crea un joc 2D, astfel:

- Texture Type: Sprite (2D and UI)
- Sprite Mode: Multiple
- Pixels Per Unit: depinde la fiecare caz (16, 32, 64 Pixels Per Unit)
- Filter Mode: Point (no filter)
- Compression: None

Înainte de a crea o hartă, este necesar a avea pachetul 2D Tilemap Editor, care ar trebui să fie instalat automat la crearea unui proiect nou de tip 2D, în Unity. Dacă acesta nu apare, poate fi instalat manual din Unity Package Manager. Fereastra Package Manager permite vizualizarea pachetelor și caracteristicilor disponibile de instalat în proiectul curent și modificarea lor [20].

O hartă este creată din mai multe tilemap-uri. Acestea se creează prin click dreapta în zona de ierarhie și se alege 2D Object -> Tilemap -> Rectangular. Toate acestea sunt incluse într-un obiect denumit Grid. Pentru fiecare nivel s-au creat mai multe tilemap-uri, deoarece pentru a putea reprezenta obiecte deasupra altor obiecte a fost necesar crearea mai multor straturi, iar acestea reprezintă tilemap-urile. Pentru a putea afișa un obiect deasupra altui obiect, în componenta Tilemap Renderer au fost schimbate valorile pentru Order in Layer, astfel încât obiectul care se dorește a apărea deasupra are un număr mai mare. Astfel în majoritatea nivelelor au fost definite tilemap-uri pentru: sol, apă, decorațiuni, stâncă etc.

Pentru a putea picta scena, sprite-urile au fost adăugate în componenta Tile Palette (vezi Figura 21), unde au fost create palete de sprite-uri, cu care apoi a fost creată scena. Procedeul de creare a unei palete, constă în tragere și plasare a sprite-ului în interiorul componentei Tile Palette. Pentru a picta scena, se selectează unealta Brush, se alege tile-ul corespunzator și se pune în scena curentă.

Un alt aspect care a contat în crearea hărții au fost obiectele prin care, în mod normal, nu se poate trece în realitate ca și zidurile, stâncile etc. Toate acestea au fost incluse în întregime într-un tilemap sau mai multe, deoarece pentru a crea iluzia de a nu putea trece

prin ele, au fost adăugate anumite componente, la nivel de tilemap. Componentele adăugate la nivel de tilemap, au efect pentru fiecare tile care face parte din acesta.

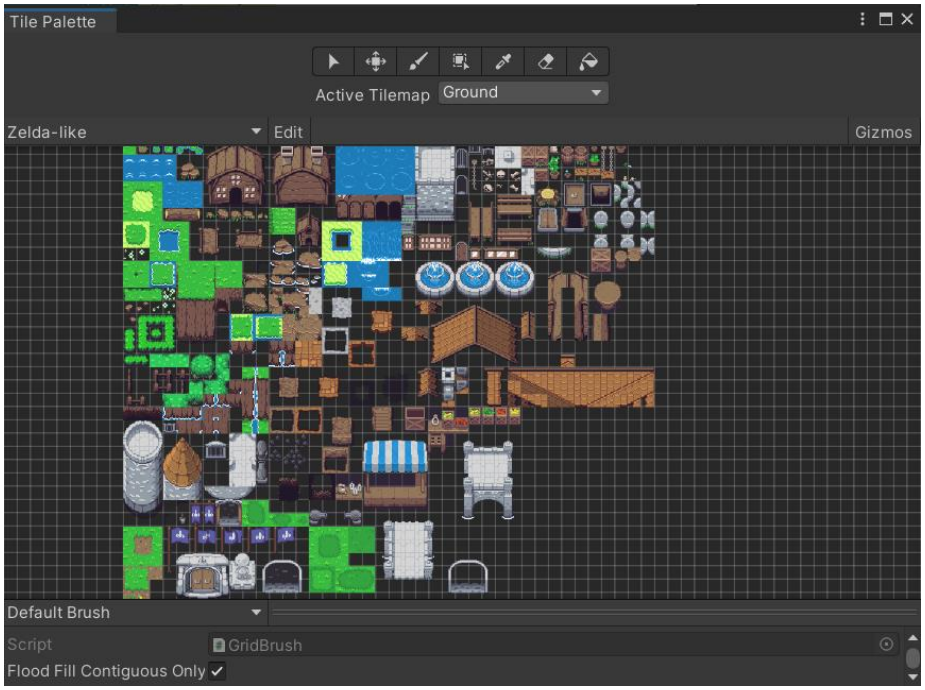

Figura 21 – Captură de ecran a componentei Tile Palette

Astfel, pentru a crea iluzia de a nu putea trece printr-un obiect, au fost adăugate componentele: Tilemap Collider 2D, Rigidbody 2D, Composite Collider 2D.

Componenta Rigidbody 2D are rolul de a face tile-ul să acționeze după legile fizicii [21]. Prin schimbarea proprietății Body Type a componentei Rigidbody 2D în Static, obiectul nu va mai cădea de pe hartă, iar prin prezența celorlalte două componente, Tilemap Collider 2D și Composite Collider 2D, jucătorul nu va mai putea trece prin acesta. Rolul lui Composite Collider 2D este de a îmbina componentele de tip Colliderele separate generate de Tilemap Collider 2D, astfel existând un singur Collider pentru toate tile-urile adiacente. Conform manualului oferit de Unity, acest lucru contribuie la performanța jocului [22].

5.3 MIȘCAREA CARACTERULUI

Mișcarea caracterului [23] a fost realizată prin script-ul PlayerMovement atașat obiectului cu numele Player.

Codul folosit pentru a realiza deplasarea jucătorului:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public float moveSpeed;
```

```
public Rigidbody2D rigidBody;
public Animator animator;

Vector2 movement;

/* Player walk in house */
public VectorValue startingPosition;

private void Start()
{
    transform.position = startingPosition.positionAfterTransition;
}

// Update is called once per frame
void Update()
{
    movement.x = Input.GetAxisRaw("Horizontal");
    movement.y = Input.GetAxisRaw("Vertical");

    animator.SetFloat("Horizontal", movement.x);
    animator.SetFloat("Vertical", movement.y);
    animator.SetFloat("Speed", movement.sqrMagnitude);

    if(movement != Vector2.zero)
    {
        if(Mathf.Abs(movement.y) > Mathf.Abs(movement.x))
        {
            //north
            if (movement.y > 0)
                animator.SetFloat("MoveDirection", 0);

            //south
            else
                animator.SetFloat("MoveDirection", 1);
        }
        else
        {
            //east
            if (movement.x > 0)
                animator.SetFloat("MoveDirection", 2);
            //west
            else
                animator.SetFloat("MoveDirection", 3);
        }

        //for running
        if(Input.GetButton("Left Shift"))
        {
            animator.SetBool("Run", true);
        }
        else
        {
            animator.SetBool("Run", false);
        }
    }
    else
    {
        animator.SetBool("Run", false);
    }
}
```

```
    void FixedUpdate()
    {
        // walk
        if(animator.GetBool("Run") == false)
        {
            rigidBody.MovePosition(rigidBody.position + movement * moveSpeed *
Time.fixedDeltaTime);
        }
        // run
        else
        {
            rigidBody.MovePosition(rigidBody.position + movement * 2*moveSpeed *
Time.fixedDeltaTime);
        }
    }
}
```

Variabila startingPosition este utilă atunci când se trece într-o scenă nouă, iar poziția de start diferă în funcție de locul de unde a avut loc tranziția. Astfel variabilei startingPosition i se va atribui poziția la care trebuie să se genereze jucătorul în metoda Start. La începutul jocului, variabila positionAfterTransition este setată și ar trebui să rămână setată pe (0,0), adică x=0, y=0.

În funcția Update, se setează parametrii componentei Animator, care este responsabilă de afișarea animațiilor corespunzătoare, în funcție de direcția sau tasta apăsată. Dacă se ține apăsat tasta Left Shift, jucătorul se află în starea de alergare.

În funcția FixedUpdate, se realizează deplasarea prin modificarea poziției componentei rigidBody de care este atașat scriptul. Viteza de deplasare pentru starea de alergare va fi de 2 ori mai rapidă decât mersul.

5.4 DIALOG

La baza funcționalității dialogului stau 2 scripturi: DialogueTrigger și DialogueManager. DialogueTrigger are rolul de a declanșa pornirea dialogului [24].

Codul fișierului DialogueTrigger:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DialogueTrigger : MonoBehaviour
{
    public Dialogue dialogue;
    public TextAsset myJSONFile = null;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            if(dialogue.hasQuestions == false)
            {
                FindObjectOfType<DialogueManager>().StartDialogue(dialogue, "No
questions");
            }
```

```
            else if (dialogue.hasQuestions == true)
            {
                FindObjectOfType<DialogueManager>().StartDialogue(dialogue,
myJSONFile.ToString());
            }
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if(collision.CompareTag("Player"))
        {
            if (dialogue.questionsDisplayed == true)
            {
                this.GetComponent<BoxCollider2D>().enabled = false;
            }

            FindObjectOfType<DialogueManager>().EndDialogue();
        }
    }
}
```

Funcția OnTriggeredEnter2D este chemată atunci când un alt obiect intră în raza collider-ului care are proprietatea Is Trigger activată. Aceasta verifică dacă obiectul care a declanșat funcția este jucătorul, prin compararea tag-ului asociat acestuia – CompareTag(“Player”). În caz afirmativ, se declanșează pornirea dialogului prin chemarea funcției StartDialogue, la care se trimite dialogul ce se dorește a fi afișat și fișierul de unde să se citească întrebările, dacă există.

Funcția OnTriggeredExit2D are rolul de a termina dialogul. În cazul în care au fost afișate întrebările, funcția mai are rolul de a dezactiva collider-ul atașat de NPC-ul responsabil de pornirea dialogului, astfel încât jucătorul să nu poată accesa întrebările de la același NPC de 2 ori.

Codul fișierului DialogueManager:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DialogueManager : MonoBehaviour
{
    public Text nameText;
    public Text dialogueText;
    public Button continueButton;
    public Button startButton;
    public Button endButton;

    public Animator animator;

    private Queue<string> sentences;
    private bool hasQuestions;

    private Dialogue dialogue;

    //jsonFile
```

```
    private string myJSONFile = null;

    //questionsManager
    public GameObject questionsPanel;

    // Start is called before the first frame update
    void Start()
    {
        sentences = new Queue<string>();
    }

    public void StartDialogue(Dialogue dialogue, string jsonFile)
    {
        animator.SetBool("IsOpen", true);

        myJSONFile = jsonFile;

        this.dialogue = dialogue;

        nameText.text = dialogue.name;
        hasQuestions = dialogue.hasQuestions;

        continueButton.gameObject.SetActive(true);
        startButton.gameObject.SetActive(false);
        endButton.gameObject.SetActive(false);

        sentences.Clear();

        foreach(string sentence in dialogue.sentences)
        {
            sentences.Enqueue(sentence);
        }

        DisplayNextSentence();
    }

    public void DisplayNextSentence()
    {
        if (sentences.Count == 1 && hasQuestions) //display start button to start the
questions dialogue.
        {
            continueButton.gameObject.SetActive(false);
            startButton.gameObject.SetActive(true);
        }
        else if (sentences.Count == 1 && !hasQuestions)
        {
            continueButton.gameObject.SetActive(false);
            endButton.gameObject.SetActive(true);
        }

        string sentence = sentences.Dequeue();
        StopAllCoroutines();                        //in case the user press
"continue" before the whole sentence was display letter by letter
        StartCoroutine(TypeSentence(sentence));
    }

    //coroutine for displaying text letter by letter
    IEnumerator TypeSentence(string sentence)
    {
        dialogueText.text = "";

        foreach(char letter in sentence.ToCharArray())
```

```
        {
            dialogueText.text += letter;
            yield return null;                          // wait a single frame
        }
    }

    public void EndDialogue()
    {
        animator.SetBool("IsOpen", false);
    }

    public void TriggerQuestions()
    {
        //freeze the scene to display questions
        Time.timeScale = 0;

        dialogue.questionsDisplayed = true;

        FindObjectOfType<QuestionsManager>().StartQuestions(myJSONFile);
    }
}
```

Propozițiile dialogului sunt păstrate în variabila sentences, care este un vector de string-uri. Pentru a afișa dialogul, fiecare propoziție din sentences, va fi adăugată într-o colecție de obiecte Queue, prin intermediul instrucțiunii foreach. Queue are proprietatea FIFO – first in, first out (adică primul intrat este și primul ieșit). Astfel, este mult mai ușor a scrie dialogul în ordine naturală, ca pe urmă acesta să fie preluat și afișat identic, adică în ordinea în care a fost scris.

Funcția DisplayNextSentence are rolul de a asigura continuitatea afișării dialogului, prin afișarea următorului enunț, cu ajutorul funcției Dequeue (a scoate din coadă).

Funcția TypeSentence este o funcție de tip corutină, adică o funcție care permite ca rezultatul unui cod să se desfășoare pe parcursul a mai multor frame-uri [25]. Am utilizat această funcție pentru a afișa propozițiile dialogului literă cu literă. Aceasta funcționalitate este opțională, dar aduce un avantaj vizual jocului.

Funcția EndDialogue setează un parametru al componentei Animator ce este responsabilă cu închiderea ferestrei de dialog atunci când parametrul de tip boolean IsOpen devine fals.

Ultima, și nu cea din urmă, TriggerQuestions, precum îi spune și numele, este funcția care declanșează întrebările, dacă acestea există în cazul NPC-ului curent cu care se vorbește. Pentru a îngheța jocul, cu scopul de a împiedica jucătorul să poată să se deplaseze departe de NPC în timp ce întrebările încep a fi derulate, s-a oprit timpul cu instrucțiunea Time.timeScale [26]. Pentru a începe afișarea întrebărilor, se cheamă funcția StartQuestions, ce primește ca parametru un string cu numele fișierului JSON din care se vor citi întrebările.

5.5 ÎNTREBĂRI GRILĂ

Sistemul de management al întrebărilor a fost conceput în mod similar cu cel al dialogului. Există o funcție TriggerQuestions, care declanșează apariția panoului de întrebări și derularea întrebărilor pe ecran. Această funcție se regăsește în fișierul C#

DialogueManager, deoarece dialogul va declanșa întrebările în cazul unui NPC cu întrebări. Script-ul care se va ocupa de afișarea întrebărilor, a variantelor de răspuns și a răspunsului corect, este QuestionsManager.

Codul sursă a scriptului:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class QuestionsManager : MonoBehaviour
{
    public GameObject questionsPanel;
    public Question currentQuestion;
    public Questions questionsInJSON;

    //aux to count nr of questions displayed
    private int questionsDisplayCounter;

    //variable which retains the number of questions to display from the JSON file,
(by default - 5)
    private int nrOfQuestionsToDisplay = 5;

    //array of listed questions
    public int []questionAlreadyDisplayed;

    //QuestionsPanel UI
    public Text questionText;
    public Button choice1;
    public Button choice2;
    public Button choice3;
    public Button choice4;

    //var aux to retain the initial number of points
    int nr_initial, nr_end, total;

    //for result
    public GameObject result;

    public void StartQuestions(string questionsJSONFile)
    {
        questionsPanel.SetActive(true);

        //retain the current number of points
        nr_initial = FindObjectOfType<PointsBar>().ReturnPoints();

        questionsDisplayCounter = 0;

        //read the questions from JSON file
        questionsInJSON = JsonUtility.FromJson<Questions>(questionsJSONFile);

        questionAlreadyDisplayed = new int[questionsInJSON.questions.Count];

        GenerateQuestion();
    }

    public void GenerateQuestion()
    {
```

```
        //if no more questions left
        if(questionsDisplayCounter == questionsInJSON.questions.Count)
        {
            EndQuestions();
        }

        Question q;

        do
        {
            q = questionsInJSON.questions[Random.Range(0,
questionsInJSON.questions.Count)];

        } while (questionAlreadyDisplayed[q.nr] == 1 && questionsDisplayCounter !=
questionsInJSON.questions.Count && questionsDisplayCounter < nrOfQuestionsToDisplay);

        questionsDisplayCounter++;
        questionAlreadyDisplayed[q.nr] = 1;

        //modify the correspondent UI
        questionText.text = q.text;
        choice1.GetComponentInChildren<Text>().text = q.choices[0];
        choice2.GetComponentInChildren<Text>().text = q.choices[1];
        choice3.GetComponentInChildren<Text>().text = q.choices[2];
        choice4.GetComponentInChildren<Text>().text = q.choices[3];

        choice1.GetComponent<Answer>().correctAnswer = false;
        choice2.GetComponent<Answer>().correctAnswer = false;
        choice3.GetComponent<Answer>().correctAnswer = false;
        choice4.GetComponent<Answer>().correctAnswer = false;

        //set the bool to true for the correct answer
        switch (q.correctAnswer)
        {
            case 1:
                choice1.GetComponent<Answer>().correctAnswer = true;
                break;
            case 2:
                choice2.GetComponent<Answer>().correctAnswer = true;
                break;
            case 3:
                choice3.GetComponent<Answer>().correctAnswer = true;
                break;
            case 4:
                choice4.GetComponent<Answer>().correctAnswer = true;
                break;
            default:
                break;
        }
    }

    public void EndQuestions()
    {
        // Calculate the number of points gained
        FindObjectOfType<QuestionResult>().DisplayNothing();
        nr_end = FindObjectOfType<PointsBar>().ReturnPoints();
        total = nr_end - nr_initial;

        // Display the result on UI object
        FindObjectOfType<Message>().DisplayResult(total,
questionsInJSON.questions.Count);
```

```
        // Increase number of npc's found
        FindObjectOfType<PeopleBar>().IncreasePersonsFound();

        // Unfreeze the scene
        Time.timeScale = 1;

        questionsPanel.SetActive(false);
    }
}
```

Precum am menționat mai sus, funcția StartQuestions are rolul de a face panoul cu întrebări activ prin instrucțiunea SetActive(true) și de a inițializa variabilele de care este nevoie. Variabila questionsInJSON este o instanță a clasei Questions care se regăsește în fișierul script Question. Aceasta este de fapt o listă de obiecte de clasa Question, care cuprinde ca atribute: nr (numărul întrebării), text (textul întrebării), choices (variantele de răspuns pe care le are întrebarea – în total 4) și correctAnswer (un număr întreg care reprezintă varianta de răspuns corectă a întrebării). Fișierul mai conține și un array de numere întregi, initializate cu 0, questionAlreadyDisplayed, care va fi util la generarea unei întrebări noi, pentru a nu afișa aceeași întrebare de 2 ori. La finalul acestei funcții, este chemată funcția GenerateQuestion.

În cadrul metodei GenerateQuestion, are loc generarea unei întrebări din fișierul sursă JSON și modificarea elementelor ce țin de interfața cu utilizatorul în mod corespunzător. Instrucțiunea do while selectează o întrebare din variabila menționată mai sus, questionsInJSON (care conține toate întrebările citite din fișier) în mod aleator prin secvența de cod Random.Range(0, questionsInJSON.questions.Count). Intervalul fiind între 0 și numărul total de întrebări. Pe urmă, are loc modificarea elementelor UI-ului, care a fost posibilă cu ajutorul funcțiilor GetComponentInChildren și GetComponent. În cadrul instrucțiunii switch, pentru fiecare întrebare, se actualizează variabila de tip intreg a clasei Question, correctAnswer, care reprezintă, de fapt, răspunsul corect la întrebare.

Ultima funcție, EndQuestions, care este chemată din cod, are rolul de a dezactiva panoul cu întrebări, și de a face calculele și acțiunile necesare. Se resetează mesajul care îți indică dacă întrebarea la care s-a răspuns anterior a fost răspunsă corect sau greșit. Script-ul responsabil de această funcționalitate este QuestionResult, astfel se cheamă funcția DisplayNothing a acestuia. Se calculează numărul de puncte acumulat în linia de cod: total = nr_end – nr_initial, unde nr_end reprezintă numărul final de puncte, iar nr_initial reprezintă numărul de puncte curent înainte de începerea afișării întrebărilor:

```
        // Display the result on UI object
                FindObjectOfType<Message>().DisplayResult(total,
questionsInJSON.questions.Count);
```

Message este un element UI din interiorul pânzei, fiind o poză ce conține și un text unde va fi afișat rezultatul. Se mai actualizează un element UI, și anume PeopleBar, care reprezintă numărul de persoane cu întrebări găsite din numărul total de persoane cu întrebări necesar a fi găsite pentru a trece la următorul nivel.

La final se repornește trecerea timpului în joc, tot cu ajutorul parametrului timeScale, și se dezactivează panoul cu instrucțiunea SetActive(false).

5.6 SETAREA LIMITELOR VIZUALE ALE JOCULUI

Pentru a seta limitele vizuale ale jocului, am creat un script, CameraFollow, care este atașat de componenta Main Camera a fiecărei scene [27].

Codul aferent fișierului:

```
using UnityEngine;

public class CameraFollow : MonoBehaviour
{
    public Transform target;
    public float smoothing;
    public Vector2 minPosition;
    public Vector2 maxPosition;

    void LateUpdate()
    {
        if (transform.position != target.position)
        {
            Vector3 targetPosition = new Vector3(target.position.x, target.position.y,
transform.position.z);

            targetPosition.x = Mathf.Clamp(target.position.x, minPosition.x,
maxPosition.x);
            targetPosition.y = Mathf.Clamp(target.position.y, minPosition.y,
maxPosition.y);

            transform.position = Vector3.Lerp(transform.position, targetPosition,
smoothing);
        }
    }
}
```

Principiul după care funcționează scriptul este: obiectul de tip Main Camera urmărește tot timpul caracterul jucătorului, dar acesta se oprește atunci când atinge ori valorile minime, ori valorile maxime, care îi sunt admise.

Astfel, deplasarea camerei se poate face doar între cele 2 seturi de valori: minPosition.x, minPosition.y și maxPosition.x, maxPosition.y. Aceste valori trebuie introduse manual, și se obțin prin deplasarea camerei în locul din stânga jos, unde se consideră a fi poziția minimă pe care o poate atinge camera, și dreapta sus, unde se consideră a fi poziția maximă pe care o poate atinge camera.

În cazul în care camera a atins una dintre cele 2 limite și se oprește din a se mai deplasa, aceasta revine în deplasare în momentul în care jucătorul se deplasează în sensul opus colțului de limită.

Funcția Mathf.Clamp are rolul de a returna valoarea poziției jucătorului, dar doar dacă aceasta se află între intervalul dintre valorile reținute minPosition, maxPosition, iar în caz contrar, returnează cea mai apropiată valoare din acel interval închis [28].

Funcția Vector3.Lerp are rolul de a interpola valorile reprezentate de poziția camerei si poziția jucătorului pe care îl urmărește, cu valoarea folosită pentru interpolare, smoothing, care este trimisă ca parametru și având o valoare introdusă manual în Inspector [29].

5.7 SUNET

Pentru redarea sunetelor în joc, am creat un sistem de sunet [30]. Script-ul aferent, care se ocupă cu manipularea sunetelor se numește AudioManager.

Codul aferent acestuia:

```
using UnityEngine.Audio;
using UnityEngine;
using System;
using UnityEngine.UI;

public class AudioManager : MonoBehaviour
{
    public float sliderValue;       // variable to change the volume of sounds playing

    public Sound[] sounds;

    public static AudioManager instance;

    // Awake method is similar to Start method, except it is called right before
    void Awake()
    {
        // if there is no object in the scene, create one.
        if (instance == null)
        {
            instance = this;
        }
        // if there is already an object in the scene, remove it.
        else
        {
            Destroy(gameObject);
            return;
        }

        DontDestroyOnLoad(gameObject);      // the object it's not destroyed when
loading a new scene

        foreach (Sound s in sounds)
        {
            s.source = gameObject.AddComponent<AudioSource>();
            s.source.clip = s.clip;
            s.source.volume = s.volume;
            s.source.pitch = s.pitch;
            s.source.loop = s.loop;
        }
    }

    public void Play (string name)
    {
        Sound s = Array.Find(sounds, s => s.name == name);

        if (s == null)
        {
            Debug.LogWarning("Sound \" " + name + " \" could not be found!");
            return;
        }
        else
        {
```

```
            s.source.Play();
        }
    }

    public void Stop(string name)
    {
        Sound s = Array.Find(sounds, s => s.name == name);

        if (s == null)
        {
            Debug.LogWarning("Sound \" " + name + " \" could not be found!");
            return;
        }
        else
        {
            s.source.Stop();
        }
    }

    public void SetVolume()
    {
        sliderValue = GameObject.Find("Slider").GetComponent<Slider>().value;

        foreach(Sound s in sounds)
        {
            s.source.volume = sliderValue;
        }
    }
}
```

Toate fișierele pentru sunet sunt păstrate într-un vector cu obiecte de tipul clasei Sound. Conținutul clasei se poate observa în următoarea parte:

```
[System.Serializable]       // this makes the class to appear in Inspector
public class Sound
{
    public string name;         // name of the sound

    public AudioClip clip;      // clip that contains the sound

    [Range(0f, 1f)]
    public float volume;        // the volume of the sound

    [Range (0.1f, 3f)]
    public float pitch;         // the pitch of the sound

    public bool loop;

    [HideInInspector]
    public AudioSource source;

}
```

Funcția Awake are rolul de a pregăti sunetele jocului, prin parcurgerea fiecărui sunet adăugat în vector cu ajutorul instrucțiunii for și adăugarea lui în AudioSource. De asemenea, se modifică și atributele ce țin de sunet, precum ar fi volumul acestuia.

Rolul funcțiilor următoare, Play, Stop și SetVolume poate fi dedus încă din numele acestora. În ordinea precizată, fiecare funcție în parte: redă o melodie, oprește o melodie

sau actualizează volumul melodiei din exterior, cu ajutorul componentei de tip Slider. Această componentă se găsește în meniul de Opțiuni a jocului.

Fișierul AudioManager este atașat de un obiect cu același nume, care se găsește în fiecare scenă a jocului.

5.8 VIAȚA

Viața caracterului este reprezentată de bara de viață [31] din stânga sus de pe ecran. Fișierul care se ocupă de funcționalitatea acesteia este HealthBar.

Codul fișierului:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class HealthBar : MonoBehaviour
{
    public Slider slider;

    public static HealthBar instance;
    public Points playerPoints;

    public void FixedUpdate()
    {
        if (slider.value == 0)
        {
            playerPoints.healthNumber = 0;

            // Load EndMenu scene
            SceneManager.LoadScene(4);
        }
    }

    public void SetMaxHealth(int health)
    {
        slider.maxValue = health;
        slider.value = health;
    }

    public void SetHealth(int health)
    {
        slider.value = health;
    }

    public void DecreaseHealth(int nr)
    {
        slider.value -= nr;
    }

    public void IncreaseHealth(int nr)
    {
        slider.value += nr;
    }
```

```
    public int GetHealth()
    {
        return (int)slider.value;
    }
}
```

Metoda FixedUpdate are rolul de a verifica într-un mod consistent dacă viața jucătorului a ajuns la 0, caz în care se actualizează variabila din ScriptableObject, playerPoints.healthNumber, care va contribui la afișarea mesajului corespunzător în meniul de sfârșit, și, desigur, de a încărca scena ce conține meniul de sfârșit. Scena cu buildIndex 4 este scena de sfârșit.

Celelalte funcții au următorul rol: SetMaxHealth - setează valoarea maximă a vieții, SetHealth - setează viața la o anumită valoare, DecreaseHealth – scade viața cu un anumit număr, IncreaseHealth - crește viața cu un anumit număr, și, GetHealth - returnează valoarea vieții reprezentată de un număr float, care va fi convertit într-un număr întreg, deoarece am realizat calcule doar cu numere întregi în ceea ce privește viața jucătorului.

5.9 SCOR

Script-ul care se preocupă de administrarea scorului este PointsBar. Această bară de puncte a fost creată în mod similar cu bara de viață.

Codul fișierului PointsBar:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PointsBar : MonoBehaviour
{
    public Text numberOfPoints;
    int pointsTotal;

    public void IncreasePoints(int points)
    {
        pointsTotal = int.Parse(numberOfPoints.text) + points;
        numberOfPoints.text = pointsTotal.ToString();
    }

    public int ReturnPoints()
    {
        return int.Parse(numberOfPoints.text);
    }
}
```

Este nevoie la bara de punctaj, care se realizează prin declararea unui obiect de tip Text, numberOfPoints, sub forma căruia se va afișa punctajul actualizat în timp real al jucătorului.

Funcția IncreasePoints are rolul de a mări punctajul cu un număr ce este reținut în parametrul points, și care reprezintă numărul de puncte obținut. Acest număr este adunat în variabila pointsTotal, și apoi este setat ca fiind textul corespunzător barei de puncte.

Funcția ReturnPoints are rolul de a returna punctajul curent, ca un număr întreg.

5.10 TRANZIȚIA SCENEI

Tranziția de la o scenă la alta se face cu ajutorul unui obiect denumit Scene Transition prin intermediul script-ului cu același nume atașat de acesta, SceneTransition [32].

Codul fișierului C#, SceneTransition

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneTransition : MonoBehaviour
{
    public string sceneToLoad;
    public Vector2 positionToLoadTo;          // position the player to move to (in
the next scene)
    public VectorValue playerMemory;        // it memories the player's location
before the transition

    public Points pointsStorage;
    public PointsBar pointsBar;
    public PeopleBar peopleBar;
    public HealthBar healthBar;

    public GameObject popUp;

    public void OnTriggerEnter2D(Collider2D collision)
    {
        //if there is a scene to load
        if (!sceneToLoad.Equals("None"))
        {
            playerMemory.positionAfterTransition = positionToLoadTo;

            if (sceneToLoad.Equals("Next Level"))
            {
                pointsStorage.pointsNumber = pointsBar.numberOfPoints.text;
                SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
            }
            else if (collision.CompareTag("Player") && !collision.isTrigger)
            {
                if (sceneToLoad.Equals("Interior") || sceneToLoad.Equals("Secret") ||
sceneToLoad.Equals("Level 3 (Hard)"))
                {
                    pointsStorage.pointsNumber = pointsBar.numberOfPoints.text;
                    pointsStorage.personsNumber = peopleBar.personsFound.text;
                    pointsStorage.healthNumber = healthBar.GetHealth();
                }
                else if (sceneToLoad.Equals("EndMenu"))
                {
```

```
                    pointsStorage.pointsNumber = pointsBar.numberOfPoints.text;
                    pointsStorage.healthNumber = healthBar.GetHealth();
                }
                SceneManager.LoadScene(sceneToLoad);
            }
        }
        else if (sceneToLoad.Equals("None"))
        {
            popUp.GetComponent<Animator>().SetBool("PopUp", true);
        }
    }

    public void OnTriggerExit2D(Collider2D collision)
    {
        if(sceneToLoad.Equals("None"))
        {
            popUp.GetComponent<Animator>().SetBool("PopUp", false);
        }
    }
}
```

Principiul de funcționare: în momentul în care collider-ul jucătorului și collider-ul tranziției se întâlnesc este chemată funcția OnTriggerEnter2D, trebuie să se încarce următoarea scenă, asta în cazul în care este deblocată tranziția prin schimbarea valorii variabilei de tip string, sceneToLoad, cu scena către care să se realizeze tranziția.

În următoarea parte, sunt prezente o serie de instrucțiuni if, deoarece la trecerea normală între nivelele jocului trebuie să se memoreze doar scorul jucătorului - viața și numărul de persoane resetându-se la fiecare nivel, dar în cazul trecerii de la nivelul 3 la scenele adiționale, interior sau secret, trebuie să se memoreze, pe lângă scor, și viața și numărul de persoane găsite. Astfel, avem nevoie de referințele următoare:

```
    public Points pointsStorage;
    public PointsBar pointsBar;
    public PeopleBar peopleBar;
    public HealthBar healthBar;
```

În variabila pointsStorage se vor reține valorile variabilelor după fiecare caz, și acestea se vor actualiza în alt script. Celelalte variabile, sunt reprezentate de bara de punctaj, bara de persoane și bara de viață a jocului.

5.11 DECOLORAREA COPACILOR

Decolorarea copacilor este o funcționalitate care a fost implementată pentru o mai bună experiență a jocului. Această funcționalitate se referă la obiectele ce sunt asemănate cu niște copaci, care se vor decolora dacă jucătorul se deplasează în spatele copacului. Rolul acestei decolorări a obiectului este de a putea vedea cu ușurință poziția jucătorului. Funcționalitatea de acest gen a mai fost întâlnită pe parcurs în jocurile jucate și am decis să o implementez în aplicația mea.

Cu ajutorul clasei SpriteRenderer se poate accesa proprietatea color a obiectului, iar prin modificarea valorii acestei proprietăți se obține un obiect vizual mai transparent sau opac [33].

Codul aferent:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Fading : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            Color faded = GetComponent<SpriteRenderer>().color;
            faded.a = 0.8f;
            GetComponent<SpriteRenderer>().color = faded;
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            Color faded = GetComponent<SpriteRenderer>().color;
            faded.a = 1f;
            GetComponent<SpriteRenderer>().color = faded;
        }
    }
}
```

Fiecare obiect de tip copac, are atașat o componentă de tip Collider ce are proprietatea Is Trigger true, pentru a putea folosi funcțiile menționate mai sus. Principiul de funcționare este destul de simplu, în momentul în care cele două componente de tip Collider, cel de pe jucător și cel de pe copac, se întâlnesc, este declanșată prima funcție. OnTriggerEntered2D decolorează obiectul, astfel încât culoarea acestuia reprezintă doar 80% din total. În momentul în care jucătorul se deplasează înafara copacului, adică cele două componente de tip Collider nu se mai ating, este declanșată a doua funcție, OnTriggerExit2D. Această funcție realizează operația inversă, resetează culoare obiectului de formă copac la 100%.

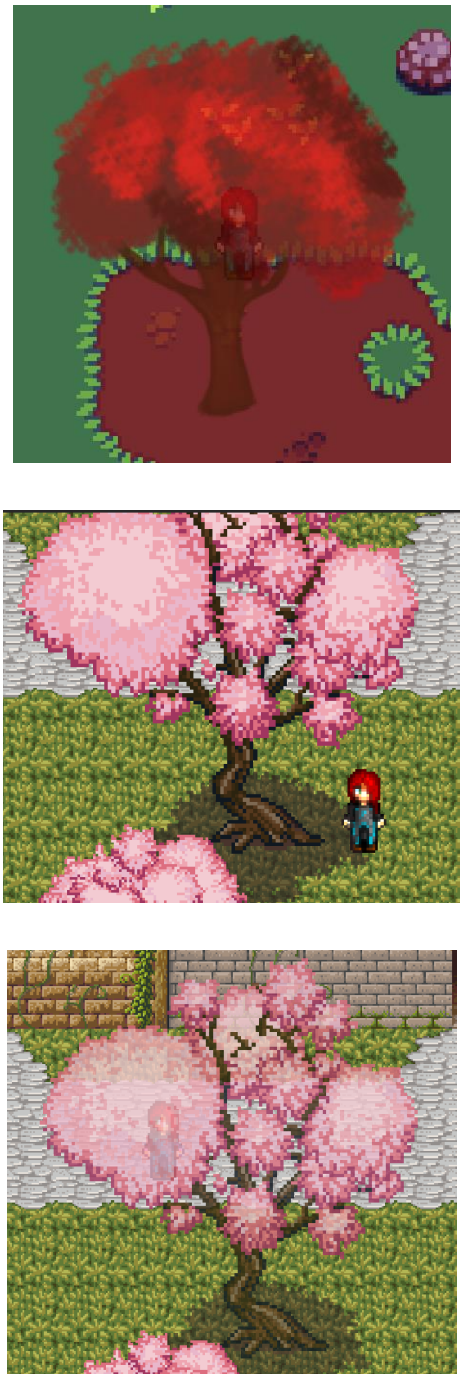

Figura 22 – Capturi de ecran reprezentative pentru funcționalitatea de decolorare

5.12 BARA DE PERSOANE

Bara de persoane se referă la bara care contorizează numărul de persoane cu întrebări găsite în nivelul actual, și aceasta se găsește în partea stângă sus a ecranului. Scriptul care se ocupă cu funcționalitatea acestei bări este:

Cod aferent scriptului PeopleBar:

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PeopleBar : MonoBehaviour
{
    public Text personsFound;
    public Text totalPersons;

    // Number of NPC with questions
    public int npcs;

    private int nr = 0;
    private bool msgDisplayed = false;

    private GameObject nextLevelTransition;

    private void Start()
    {
        totalPersons.text = npcs.ToString();
    }

    private void Update()
    {
        if (personsFound.text == totalPersons.text && msgDisplayed == false)
        {
            // When all people with questions are found, unlock next level
            FindObjectOfType<Message>().DisplayMessage("Felicitari! Ai deblocat
urmatorul nivel! Il poti accesa la locul de start al acestui nivel!");
            nextLevelTransition = GameObject.Find("Scene Transition (Next Level)");
            nextLevelTransition.GetComponent<SceneTransition>().sceneToLoad = "Next
Level";
            msgDisplayed = true;
        }
    }

    public void IncreasePersonsFound()
    {
        nr = int.Parse(personsFound.text);
        nr++;
        personsFound.text = nr.ToString();
    }
}
```

În metoda Start se setează numărul total de persoane cu întrebări din nivelul curent. Acestă valoare a fost introdusă manual în Inspector.

În metoda Update, se verifică dacă s-a atins numărul total de persoane necesar a fi găsite, iar în cazul adevărat, se va afișa un mesaj care indică jucătorului că nivelul următor a fost deblocat, prin apelarea metodei DisplayMessage și trimiterea prin intermediul parametrului a mesajului de afișat:

```
FindObjectOfType<Message>().DisplayMessage("Felicitari! Ai deblocat urmatorul nivel!
Il poti accesa la locul de start al acestui nivel!");
```

În pasul următor, trebuie deblocată tranziția către nivelul următor. Acest lucru se realizează prin căutarea obiectului responsabil de tranziția către următorul nivel, adică:

```
nextLevelTransition = GameObject.Find("Scene Transition (Next Level)");
```

Obiectul găsit este salvat în obiectul nextLevelTransition, pentru a putea modiifca textul variabilei sceneToLoad:

```
nextLevelTransition.GetComponent<SceneTransition>().sceneToLoad = "Next Level";
```

Textul tranziției este foarte important, deoarece tranziția se realizează în alt script, pe baza acestuia.

Ultima funcție, IncreasePersonsFound, are rolul de a incrementa numărul de persoane găsite. Această funcție este chemată din QuestionsManager, în funcția EndQuestions, care realizează operațiile necesare la terminarea întrebărilor.

6 TESTARE

În ceea ce privește testarea, aceasta a fost făcută gradual, fiecare funcționalitate, sau fiecare obiect nou adăugat a fost testat pentru a verifica existența posibilelor erori, deoarece scopul testării este de a identifica ce nu funcționează corespunzător. După testarea individuală a componentelor, am realizat o testare de ansamblu, pentru a asigura că totul funcționează cum trebuie și în modul intenționat.

O primă problemă întâmpinată a reprezentat-o componentele de tip Collider pentru elementele de pe hartă. Neavând experiență în crearea unei hărți 2D în Unity, unele elemente au fost desenate pe tilemap-uri necorespunzătoare, adică dacă un element trebuia să conțină componenta de tip Collider, acesta a fost pus din greșeală într-un tilemap care nu conținea acea componentă. Dar, după sesizarea acestei erori, a fost realizată o structură mai bună a tilemap-urilor și a conținutului acestora.

O altă inconvenientă întâmpinată a reprezentat faptul că sistemul de întrebări funcționa, inițial, doar cu un singur fișier JSON general pentru toate NPC-urile. Aceasta se datora faptului că sistemul de întrebări era declanșat direct de la butonul de start din dialog prin activarea panoului de întrebări. Intenția mea, a fost ca fiecare NPC să ofere întrebări diferite și dintr-un domeniu diferit. Soluția pe care am găsit-o în acest caz, a fost să creez o variabilă de tip TextAsset, atașată script-ului care declanșează dialogul și care este atașat de fiecare NPC, unde am introdus numele fișierului de unde să se genereze întrebările pentru NPC-ul în cauză. Astfel, pentru a putea trimite numele fișierului în scriptul responsabil de administrarea întrebărilor, buton start din panoul de dialog nu va mai activa direct panoul de întrebări, ci va chema o funcție, TriggerQuestions, care va declanșa întrebările prin chemarea funcției StartQuestions, la care va fi trimis ca parametru, numele fișierului de format JSON.

O altă problemă întâmpinată a fost legată de bara de scor și bara de viață. Principiul acestora este de a se reseta atunci când jucătorul trece la următorul nivel (trece dintr-o scenă în alta). Problema apare la ultimul nivel, level 3, când jucătorul se poate deplasa în scena care conține interiorul caselor sau în scena secretă. În mod normal, viața, scorul și numărul de persoane cu întrebări găsite în nivel nu ar trebui să se modifice la aceste tranziții. Astfel, am creat un Scriptable Object, PlayerPoints, ce are la baza script-ul Points, unde rețin aceste valori, iar în momentul tranziției se actualizează valorile reținute și în momentul de început a scenei se setează la valoarea corectă, dar doar în cazul scenelor în consecință.

De asemenea, pe parcursul timpului pe care l-am petrecut pentru a alcătui harta lumii din joc și de a adăuga mai multe obiecte, mai mulți NPC, am observat o eroare de afișare a câtorva elemente dintre acestea. Eroarea s-a datorat sortării necorespunzătoare a straturilor și a ordinii din strat. Pentru a rezolva această problemă, am modificat proprietatea Order in Layer a diferitelor obiecte.

În cazul testării de ansamblu, am observat că totul funcționează în mod intenționat, dar a apărut o singură inconvenientă reprezentată de niște linii care apar în diferite momente pe hartă și în cazul diferitor sprite-uri utilizate. Nu am reușit să găsesc o soluție pentru această problemă, deoarece nu este o problemă specifică motorului de joc Unity.

7 MANUAL DE UTILIZARE

În capitolul curent, voi detalia pașii care trebuie urmați pentru a putea juca jocul:

1. Accesarea meniului de opțiuni pentru a modifica volumul sunetului de fundal pe tot parcursul jocului. (Opțional)

Figura 23 – Butonul de accesare a meniului de opțiuni

Figura 24 – Meniul de opțiuni

3. Pentru a începe jocul, se apasă butonul PLAY, iar pentru a închide jocul se apasă butonul QUIT.

Figura 25 – Buton start și buton ieșire

4. Pentru a vedea tutorialul, se accesează meniul de pauză, prin apăsarea butonului PAUSE, de sub sistemul de scor. (Opțional)

Figura 26 – Buton pauză

Figura 27 – Meniu pauză

Pentru a reveni la joc, se apasă butonul Continua, pentru a afișa tutorialul se apasă butonul Tutorial, iar pentru a ieși din aplicație se apasă butonul Quit. Fereastra de tutorial se poate observa în figura 28.

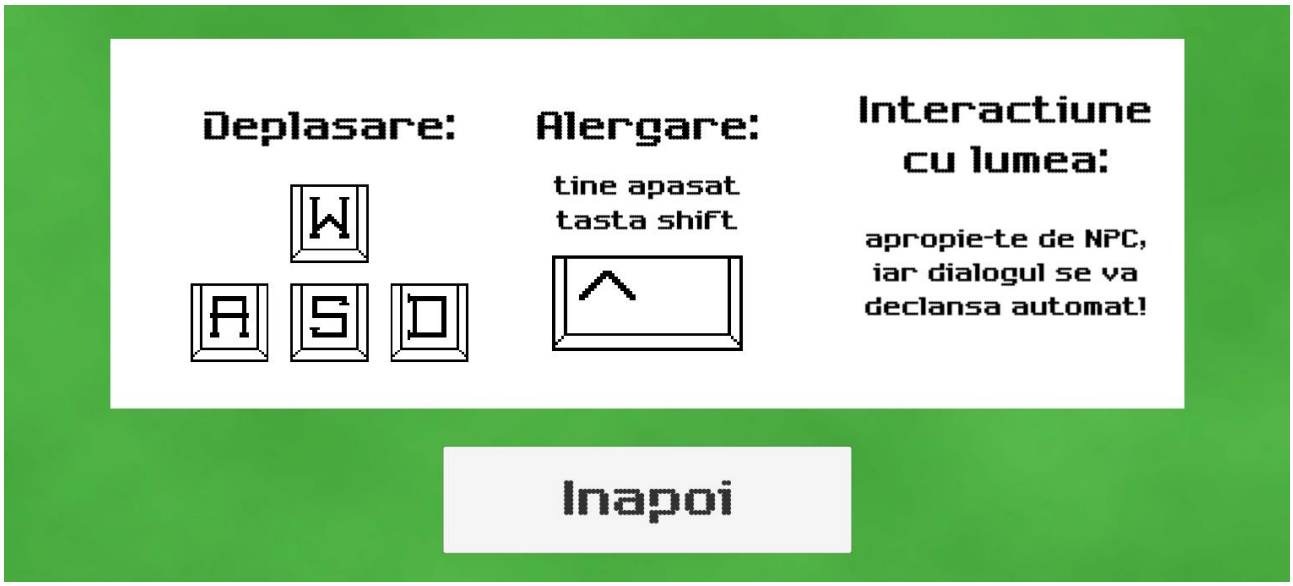

Figura 28 – Captură de ecran a tutorialului

5. Pentru a trece a trece la nivelul următor, trebuie explorată harta, trebuie găsite toate persoanele care au întrebări și trebuie oferite răspunsuri la toate aceste întrebări.

Figura 29 – Exemplu persoană cu întrebări

Numărul de persoane cu întrebări găsite din numărul total de persoane existente în nivel poate fi observat în partea stânga sus a ecranului.

Figura 30 – Numărul de persoane găsite / numărul total

6. Viața este reprezentată tot în partea stânga sus a ecranului. Aceasta scade când se răspunde greșit la o întrebare și crește puțin când se răspunde corect la o altă întrebare.

Figura 31 – Bara de viață a jucătorului în diferite momente

7. În cazul în care se atinge numărul de persoane de găsit, se va debloca următorul nivel prin afișarea următorului mesaj:

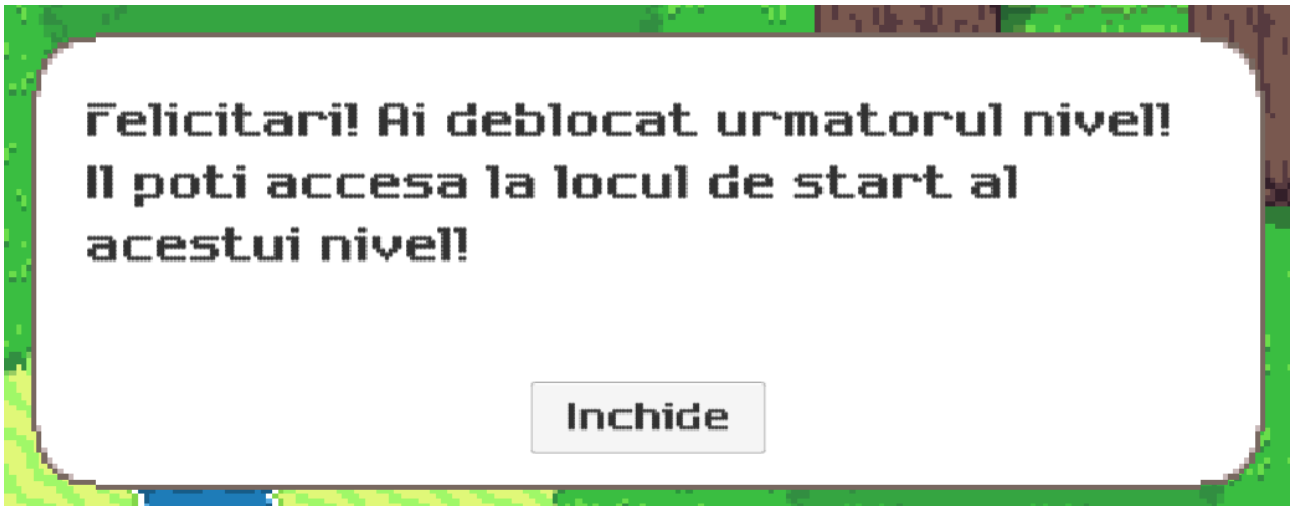

Figura 32 – Mesaj de notificare a deblocării următorului nivel

Pentru a trece la următorul nivel, trebuie să se caute tranziția care face această trecere. Aceasta este asemănătoare unei portițe de ieșire, ca o deschizătură înafara hărții.

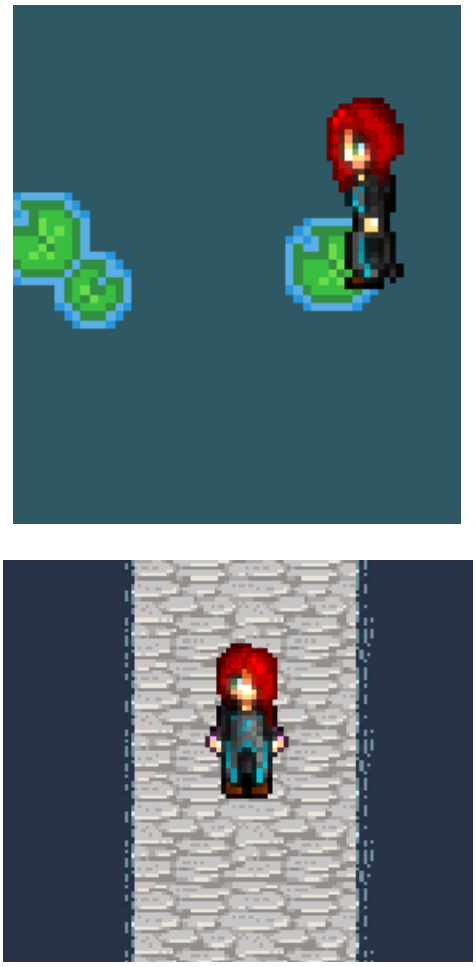

Figura 33 – Toate cele 3 tranziții de nivel ale jocului

8. În cazul în care viața ajunge la 0, jocul se va sfârși și se afișează meniul de sfârșit. Sari la punctul 10.

Figura 34 – Meniul de sfârșit în cazul pierderii jocului

9. Dacă s-a ajuns la ultimul nivel, și se trece prin tranziția de nivel, jocul se va sfârși și se va afișa punctajul total obținut.

Figura 35 – Meniul de sfârșit în cazul câștigării jocului

10. Pentru a reîncepe jocul, se va da click pe butonul RESTART, iar pentru a ieși din aplicație, se va da click pe butonul QUIT.

Figura 36 – Buton restart și buton ieșire

8 CONCLUZII

În urma finalizării aplicației am ajuns la niște concluzii pe care le voi menționa în următoarele paragrafe.

Crearea unui joc în Unity a reprezentat un exercițiu bun în ceea ce privește domeniul de dezvoltare a jocurilor video. Jocurile video din ziua de azi, sunt dezvoltate de companii mari, care sunt reprezentate de mii de angajați, iar fiecare echipa/angajat al firmei se ocupă de anumite aspecte ale jocului. Dezvoltarea unui întreg joc, de către o singură persoană, poate fi o provocare, dar pe care eu am decis să o accept.

Jocul va aduce un sentiment de familiaritate prin grafica cu care a fost realizat. Acesta conține funcționalități similare cu majoritatea jocurilor de tip 2D de pe piață, de exemplu: animații, sistem de scor, sistem de viață, meniuri, tutorial etc. Astfel, nu ar trebui să reprezinte o piedică încercarea de a-l juca.

Fișierele de format JSON, care au fost utilizate pentru citirea întrebărilor, au un format concret, ușor de folosit și ușor de înțeles, astfel nici utilizarea lor nu a reprezentat o piedică.

Aspectul care mi s-a părut cel mai greu în crearea jocului nu a constat în ceea ce trebuie să fac, ci în ceea ce vreau să fac. Posibilitatea de a merge în foarte multe direcții cu jocul este un element, pe cât de avantajos, pe atât de dezavantajos. Astfel, gândirea în ansamblu a tot ceea ce reprezintă jocul, este cea mai grea parte, dar într-un final jocul a primit viață.

Un alt aspect care mi s-a părut mai dificil, a fost preluarea și programarea într-un nou mediu de dezvoltare. La început, totul pare străin, dificil, dar, după un timp mediul de dezvoltare Unity îți devine familiar.

Un aspect care m-a impresionat într-un mod plăcut și care m-a atras din prima, a fost modul de creare a unei hărți de tip 2D. Asocierea cu paleta cu care pictorul pictează o pânză, iar modul prin care niște simple imagini ajung să alcătuiască o lume fictivă mi se pare un aspect foarte interesant. Având și anumite inclinații artistice, crearea scenelor a reprezentat o plăcere.

Din punct de vedere al funcționalităților jocului, crearea scripturilor în parte, folosind limbajul de programare C#, nu a fost ceva dificil, partea puțin mai grea a devenit legarea acestora.

Într-un final, eu consider că lucrarea de licență, reprezentată de jocul de tip 2D, educativ, cu lumea și povestea lui fictivă, originală, este un joc care trebuie încercat, cel puțin o dată, pentru că oferă multe aspecte plăcute. Starea jocului este o stare destul de solidă, dar, precum am menționat, direcțiile în care se poate duce un joc sunt infinite. Există multe alte funcționalități care ar putea fi implementate, multe alte resurse de folosit, și într-un final, multe aspecte care pot fi adăugate sau actualizate. Domeniul jocurilor video este un domeniu în dezvoltare continuă, dar eu consider că ideea jocului meu este o idee bună și merită a fi luată în considerare.

BIBLIOGRAFIE

[1] Unity Technologies, Unity User Manual 2021.3 (LTS), Site oficial Unity, 2021, Disponibil: https://docs.unity3d.com/Manual/index.html, [Accesat în data: 06.11.2021]

[2] Ministerul Educației Naționale, GIMNAZIU - PROGRAME ȘCOLARE 2017, 2017 Disponibil: http://programe.ise.ro/Actuale/Gimnaziu-2017.aspx, [Accesat în data: 03.05.2022

[3] Igara Studio S.A., Site oficial Aseprite, Disponibil: https://www.aseprite.org/, [Accesat în data: 10.11.2021]

[4] „ConcernedApe" - Eric Barone, Stardew Valley, Site oficial Stardew Valley, Disponibil: https://www.stardewvalley.net/, [Accesat în data: 27.05.2022]

[5] King 5, „Stardew Valley developer is an Auburn native - KING 5 Evening", Youtube, Disponibil: https://www.youtube.com/watch?v=9YqdutJLIi8, [Accesat în data de: 27.05.2022]

[6] Eric Barone, Despre așteptările asupra jocului minutul 0:50, Youtube, Disponibil: https://www.youtube.com/watch?v=9YqdutJLIi8 [Accesat în data de: 27.05.2022]

[7] „ConcernedApe" - Eric Barone, „Announcing my new Game", Site oficial Haunted Chocolatier, 8 octombrie 2021, Disponibil: https://www.hauntedchocolatier.net/2021/10/08/hello-world/ [Accesat în data: 27.05.2022]

[8] Andrew "Redigit" Spinks, Terraria, Site oficial Terraria, 2021, Disponibil: https://terraria.org/ [Accesat în data: 28.05.2022]

[9] Klei Entertainment, Don't Starve Together, Site oficial Klei Entertainment, 2022, Disponibil: https://www.klei.com/games/dont-starve, [Accesat în data: 04.06.2022]

[10] Unity Technologies, „Our impact by the numbers", Site official Unity, 2021, Disponibil: https://unity.com/our-company, [Accesat în data: 05.06.2022]

[11] Refsnes Data, C# Tutorial, W3Schools, 2022, Disponibil: https://www.w3schools.com/cs/index.php, [Accesat în data: 11.12.2021]

[12] Unity Technologies, ScriptableObject, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/Manual/class-ScriptableObject.html, [Accesat în data: 03.06.2022]

[13] Unity Technologies, Prefabs, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/Manual/Prefabs.html, [Accesat în data: 23.04.2022]

[14] Oana Sorina Chirilă, Curs „Medii și Tehnologii de Programare", Facultatea de Automatică și Calculatoare, Universitatea Politehnica Timișoara, an universitar 2021-2022

[15] JavaTpoint, C# Features, Site oficial JavaTpoint, 2021, Disponibil: https://www.javatpoint.com/csharp-features, [Accesat în data: 14.06.2022]

[16] „Introducing JSON", site json.org, Disponibil: https://www.json.org/json-en.html, [Accesat în data de: 28.05.2022]

[17] Unity Technologies, Scene.buildIndex, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/ScriptReference/SceneManagement.Scene-buildIndex.html, [Accesat în data: 08.06.2022]

[18] Unity Technologies, Event System, Unity Manual, 2020, Disponibil: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/EventSystem.html, [Accesat în data: 10.06.2022]

[19] Velvary, „Creating Tilemaps For Your 2D Game in Unity 2021 – Tutorial", Youtube, Disponibil: https://www.youtube.com/watch?v=DTp5zi8_u1U, [Accesat în data: 06.12.2021]

[20] Unity Technologies, Tilemap, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/Manual/class-Tilemap.html, [Accesat în data: 08.12.2021]

[21] Unity Technologies, Rigidbody2D, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/ScriptReference/Rigidbody2D.html, [Accesat în data: 07.12.2021]

[22] Unity Technologies, Tilemap Collider 2D, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/Manual/class-TilemapCollider2D.html, accesat pentru prima oară în data de 11.12.2021

[23] Brackeys, „TOP DOWN MOVEMENT in Unity!", Youtube, Disponibil: https://www.youtube.com/watch?v=whzomFgjT50, [Accesat în data de 06.11.2021]

[24] Brackeys, „How to make a Dialogue System in Unity", Youtube, Disponibil: https://www.youtube.com/watch?v=_nRzoTzeyxU, [Accesat în data de: 18.03.2021]

[25] Unity Technologies, Coroutines, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/Manual/Coroutines.html, [Accesat în data de: 20.03.2022]

[26] Unity Technologies, Time.timeScale, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/ScriptReference/Time-timeScale.html, [Accesat în data de: 30.05.2022]

[27] Mister Taft Creates, „Part 6 - Bounding the Camera: Make a game like Legend of Zelda with Unity and C#", Youtube, Disponibil: https://www.youtube.com/watch?v=OWJa6lcFTXk, [Accesat în data de 15.12.2021]

[28] Unity Technologies, Mathf.Clamp, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/ScriptReference/Mathf.Clamp.html, [Accesat în data: 15.12.2021]

[29] Unity Technologies, Vector3.Lerp, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/ScriptReference/Vector3.Lerp.html, [Accesat în data: 15.12.2021]

[30] Brackeys, „Introduction to AUDIO in Unity”, Youtube, Disponibil: https://www.youtube.com/watch?v=6OT43pvUyfY, [Accesat în data de: 14.04.2022]

[31] Brackeys, „How to make a HEALTH BAR in Unity!”, Youtube, Disponibil: https://www.youtube.com/watch?v=BLfNP4Sc_iA, [Accesat în data de: 09.04.2022]

[32] Mister Taft Creates, „Part 28 - Switching Scenes: Make a game like Zelda using Unity and C#”, Youtube, Disponibil: https://www.youtube.com/watch?v=wNl--exin90, [Accesat în data de: 25.05.2022]

[33] Unity Technologies, SpriteRenderer, Unity Manual, 2021, Disponibil: https://docs.unity3d.com/ScriptReference/SpriteRenderer.html, [Accesat în data de: 09.05.2022]

CREDITE PENTRU RESURSE FOLOSITE

- Zelda-like tilesets and sprites, https://opengameart.org/content/zelda-like-tilesets-and-sprites - realizat de ArMM1998
- Whispers of Avalon: Grassland Tileset, https://opengameart.org/content/whispers-of-avalon-grassland-tileset - realizat de Leonard Pabin
- Mage City Arcanos, https://opengameart.org/content/mage-city-arcanos - realizat de Hyptosis, link personal www.lorestrome.com
- Outdoor tiles, again, https://opengameart.org/content/outdoor-tiles-again - realizat de Michele "Buch" Bucelli, link pagină de profil https://opengameart.org/users/buch
- Assets Free: Nature Sprites [Trees, Shrubs], https://opengameart.org/content/assets-free-nature-sprites-trees-shrubs - realizat de Wenrexa
- Golden UI, https://opengameart.org/content/golden-ui - realizat de Buch, link pagină de profil, https://opengameart.org/users/buch
- 6809 Chargen, Font, https://www.dafont.com/6809-chargen.font - realizat de Typodermic Fonts Inc.
- [LPC] Plant Repack, https://opengameart.org/content/lpc-plant-repack - realizat de Barbara Rivera, Casper Nilsson, Johann Charlot, Chris Phillips, Lanea Zimmerman, mai multe detalii se regăsesc în folderul Assets\Sprites\[LPC] Plant Repack
- Interior Tileset 16x16, https://opengameart.org/content/interior-tileset-16x16 - realizat de Bonsaiheldin
- LPC: Modified base tiles, https://opengameart.org/content/lpc-modified-base-tiles - realizat de Lanea Zimmerman și mulțumiri speciale lui William Thompson
- TinyRpg Stranger Forest Pack, https://opengameart.org/content/tinyrpg-stranger-forest-pack - realizat de ansimuz
- Fumiko Complete Charset, https://opengameart.org/content/fumiko-complete-charset - realizat de sylvius fischer
- LPC Tile Atlas2, https://opengameart.org/content/lpc-tile-atlas2 - mai multe detalii despre autori se regăsesc în fișierul Attribution2.txt din locația Assets\Sprites\LPC Tile Atlas2\Atlas2

- Frogatto & Friends - NPC Pack, https://opengameart.org/content/frogatto-friends-npc-pack - realizat de Guido Bos și depus de marcavis
- Sunet „Woodland Fantasy”, https://opengameart.org/content/woodland-fantasy - realizat de Matthew Pablo, link pagină proprie, https://matthewpablo.com/services/
- Sunet „forest”, https://opengameart.org/content/forest - realizat de syncopika
- Sunet „one”, https://opengameart.org/content/one - realizat de pheonton
- Sunet „caravan”, https://opengameart.org/content/desert-theme - realizat de yd
- Sunet „awesomeness”, https://opengameart.org/content/menu-music - realizat de mrpoly
- Edited and Extended 24x32 Character Pack, https://opengameart.org/content/edited-and-extended-24x32-character-pack - realizat de diamonddmgirl, creator original: Svetlana Kushnariova, adresă de e-mail: lana-chan@yandex.ru
- 24x32 characters with faces (big pack), https://opengameart.org/content/24x32-characters-with-faces-big-pack - realizat de Svetlana Kushnariova, adresă de e-mail: lana-chan@yandex.ru
- Warrior Free Asset, https://assetstore.unity.com/packages/2d/characters/warrior-free-asset-195707#description – realizat de Clembod

INDEX FIGURI

INDEX TABELE

ABREVIERI FOLOSITE ȘI TERMENI DIN DOMENIU EXPLICAȚI

Tabel 2. Termeni abreviați

Abreviere	Termen complet
NPC	Non-player character
RPG	Role-playing game
UI	User Interface

Tabel 3. Termeni din domeniu explicați

Termen	Explicație
Colliders	Aceste componente definesc o anumită formă invizibilă în jurul obiectului atașat, de exemplu: pătrat, poligon, sferă, cerc etc. Scopul acestora este trata coliziunile fizice ale obiectelor
Indie	Este prescurtarea de la independent video game, iar un joc de tip indie este, în general, un joc creat de o singură persoană sau de o echipă de dezvoltatori mică
Item	Este orice obiect care poate fi reprezentat în joc. Spre exemplu: arme, mobilier, flori etc.
Multiplayer	Definește un tip de joc care se poate juca de una sau mai multe persoane simultan
Non-player character	Un NPC este un obiect de tip caracter dintr-un joc care nu este controlat de către jucător. Acesta este programat cu un scop și comportamentul lui este limitat de codul după care a fost programat
Renderer	Este componenta care are rolul și ocupația de a reda obiectele pe ecran
Singleplayer	Definește un tip de joc care se poate juca doar de o singură persoană
Sprites	Sunt imagini, care reprezintă o resursă a jocului. Cu ajutorul lor se creează obiectele vizuale care apar în joc
Tutorial	Reprezintă un ghid oferit jucătorului ce cuprinde detalii, instrucțiuni, informații despre joc, având scopul de îndrumare

DECLARAŢIE DE AUTENTICITATE A LUCRĂRII DE FINALIZARE A STUDIILOR *

Subsemnatul BADE DELIA,

legitimat cu CI seria ZH nr. 213210,

CNP 6001001055078

autorul lucrării JOC EDUCATIV 2D PENTRU ELEVI

elaborată în vederea susţinerii examenului de finalizare a studiilor de LICENȚĂ organizat de către Facultatea AUTOMATICĂ ȘI CALCULATOARE din cadrul Universităţii Politehnica Timişoara, sesiunea IUNIE a anului universitar 2021-2022, coordonator OANA-SORINA CHIRILĂ, luând în considerare conţinutul art. 34 din *Regulamentul privind organizarea și desfășurarea examenelor de licență/diplomă și disertație*, aprobat prin HS nr. 109/14.05.2020 și cunoscând faptul că în cazul constatării ulterioare a unor declaraţii false, voi suporta sancțiunea administrativă prevăzută de art. 146 din Legea nr. 1/2011 – legea educației naționale și anume anularea diplomei de studii, declar pe proprie răspundere, că:

- această lucrare este rezultatul propriei activități intelectuale,
- lucrarea nu conține texte, date sau elemente de grafică din alte lucrări sau din alte surse fără ca acestea să nu fie citate, inclusiv situația în care sursa o reprezintă o altă lucrare/alte lucrări ale subsemnatului.
- sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.
- această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență/diplomă/disertație.

Timişoara,

Data
18.06.2022

Semnătura

* Declaraţia se completează *„de mână"* şi se inserează în lucrarea de finalizare a studiilor, la sfârşitul acesteia, ca parte integrantă.