Universitatea Tehnică „Gh. Asachi" Iași

Facultatea de Automatică și Calculatoare

Sparking Ray

- Documentație -

Codău Eduard-Alexandru

Apetria George

Iași 2021

I. Rezumatul lucrării

1. Tema abordată

Având la bază tema cooperative multiplayer, ,,Sparking Ray" este un joc multiplayer 2D care a fost dezvoltat pornind de la ideea de a reuni doi jucători prin intermediul unei raze de interconectare cu scopul distrugerii unui număr cât mai mare de inamici.

2. Obiectivul

Obiectiv general: realizarea unei aplicații multiplayer utilizând platforma Unity, limbajul C# și pachetul Unity Photon 2 pentru jocuri multiplayer.

Obiective particulare:

- Delimitarea clară a caracteristicilor unui joc multiplayer, comparativ cu un joc singleplayer prin dezvoltarea unui meniu de încărcare multiplayer și al unui lobby.
- Crearea unor scripturi simple și particularizate utilizând limbajul de programare C# pentru dezvoltarea aplicației.
- Sincronizarea elementelor în rețea: script-uri, elemente grafice, texturi, animații, efecte sonore și funcții RPC (Remote Procedure Call) – crearea aplicației, divizarea acesteia în componente și ulterioara adăugare a protocolului de comunicație.
- Crearea unui controller pentru jucători cu o perspectivă de tip top-down prin implementarea unui cod care să ofere mișcări realiste la care se adaugă mecanismul screen wrap.
- Proiectarea unor inamici AI care vor urmări cel mai apropiat jucător și vor încerca să diminueze numărul de vieți al echipei formate din cei doi jucători.
- Implementarea unui sistem flexibil de spawning al inamicilor pentru a începe cu adevărat jocul, creând astfel provocări echipei prin intermediul numărului crescând de monștri.
- Adăugarea unui nume pentru jucător în cadrul meniului principal.
- Animarea și dinamizarea jocului prin inserarea unor efecte vizuale și sonore.

- Crearea unui sistem de înregistrare a scorului pentru a urmări performanțele jucătorilor.

3. Proiectarea aplicației

Aplicația a fost proiectată într-o manieră minimalistă, reunind o serie de componente de bază organizate într-un mecanism simplu:

- Server-ul Photon este responsabil de primirea și gestionarea cererilor sosite în vederea creării unei camere de către primul jucător care lansează aplicația în execuție (host) și alăturării jucătorului secund la camera anterior creată. Mai mult decât atât, coordonatele caracterelor care reprezintă clienții sunt gestionate tot prin intermediul serverului. Nu în ultimul rând, deconectarea oricărui jucător trebuie să fie prelucrată în timp real la nivelul server-ului.
- Clienții trimit către server date ce reprezintă numele camerei pentru a o iniția ori numele camerei la care vor să adere, precum și numele de jucător (nickname).
- Datele cu privire la scorul obținut în cadrul unui joc sunt exportate tot către serverul Photon, iar la cererea jucătorilor se poate furniza un clasament pe baza informațiilor anterior menționate, care au fost înregistrate de-a lungul desfășurării mai multor jocuri.

4. Modul de implementare

Sparking Ray a fost implemetat utilizând game-engine-ul Unity care oferă dezvoltatorului atât o interfață grafică amplă, precum și posibiliatea de a crea script-urile aferente în limbajul de programare C#. Deși mediul de dezvoltare permite utilizarea complementară a limbajului JavaScript, s-a optat doar pentru utilizarea limbajului C#, o abordare dualã neputând fi considerată un exemplu de bune practici, ci un impediment în înțelegerea și aprofundarea viziunii programării bazate pe entități de tip GameObject sau orientate-obiect.

Utilizând uneltele facilitate de mediul de dezvoltare, implementarea tuturor entităților existente în joc au urmat o ordine firească de la simplu la complex și de la general la particular. Astfel procesul de dezvoltare a presupus o ierarhizare între:

- componentele de bază necesare a fi create în stadiul incipient (meniuri – atât din punct de vedere grafic, cât și din punct de vedere al funcționalităților, de exemplu conectarea la server și entități – jucători, monștri)

- componentele ulterioare implementate în stadiul imediat următor unde s-a pus accentul pe tranziții, interacțiuni, contorizarea scorului, precum și alte elemente funcționale mai complexe.
- componentele de „rafinament” necesare la îmbunătățirea calității jocului; prin adăugarea unor efecte vizuale și sonore - una dintre etapele finale - s-a realizat animarea și dinamizarea aplicației.

În cele din urmă, etapa finală a constat din finisarea diferitelor elemente constitutive, dar și înlăturarea unor erori apărute de-a lungul procesului de dezvoltare.

5. Rezultatele obținute

Varianta finală a aplicației constă într-un joc funcțional în rețea, care depinde de o conexiune stabilă a jucătorilor la internet. Desfășurarea jocului cât si cazurile de utilizare vor fi detaliate într-un capitol separat al tezei.

II. Proiectarea aplicației

1. Analiza platformei hardware de execuție a aplicației & analiza implementării optime

În contextul apogeului tehnologic actual, dezvoltatorii de aplicații beneficiază de o gamă extinsă de platforme de dezvoltare ce includ opțiuni și direcții de implementare diverse. Gradul sporit de accesibilitate pentru toate aceste facilități create de mediile de dezvoltare, precum și materialele adiționale, fac posibilă implementarea propriilor soluții în care imaginația este singura limită.

De menționat este faptul că jocul nu necesită multe cerințe funcționale sau o platformă hardware foarte puternică, ci doar o conexiune la internet și un dispozitiv al cărui sistem de operare este Windows, oricare dintre versiunile 7, 8 sau 10. Așadar, orice sistem cu specificații minimale în raport cu dispozitivele de top aflate pe piață poate aduce bucuria unui joc împărtășit cu prietenii.

Implementarea întregii aplicații s-a dovedit a fi una optimă ținând cont de faptul că tehnologia utilizată a fost una complet nouă, iar timpul a fost limitat. Mai mult, procesul de dezvoltare a jocului s-a desfășurat în condiții bune datorită facilităților oferite de platforma Unity: interfață grafică puternică și posibiliatea de a crea script-urile aferente în limbajul de programare C# pentru aproape orice entitate. De subliniat este și importanța viziunii programării bazate pe entități de tip GameObject sau orientate-obiect care au permis structurarea jocului într-o manieră logică și ușor de urmărit pe parcursul întregului proces evolutive, după cum vom vedea în cele ce urmează la punctele de mai jos.

2. Modulele generale ale aplicației & interacțiunile dintre ele

Aplicația constă într-un joc multiplayer în rețea, dezvoltată cu ajutorul platformei Unity (versiunea 2020.3.12f1 64-bit). Pe lângă partea de **gameplay**, alcătuită din componente Unity, texturi si scripturi C# client, componenta de **rețea** a jocului este realizată prin folosirea pachetului PUN (Photon Unity Networking) împreună cu o aplicație de tip server menținută în cloud ce oferă o capacitate de până la 20 de jucători concurenți. De asemenea, jucătorii pot trimite către serverul de baze de date informații cu privire la scorul dintr-un joc terminat, în vederea înregistrării într-un clasament.

După cum se poate observa și în fig. 1 de mai jos, principalele componete ale aplicației sunt:

- Serverul photon
- Clienții
- Server-ul de baze de date

În ceea ce privește interacțiunea dintre componente, distingem următoarele situații:

1) Server-ul photon primește cereri de la clienți în ceea ce privește:
   - crearea unei camere de către un jucător;
   - alăturarea unui jucator la o cameră existentă;
   - coordonatele caracterului corespunzator clientului într-un joc.
   - deconectarea unui jucator;
2) La rândul lor, clienții trimit datele precedente către serverul Photon.
3) De asemenea, clienții trimit date cu privire la scorul obținut într-un joc către serverul de baze de date.
4) La cererea clienților, serverul de baze de date afișează clasamentul jucătorilor pe baza informațiilor primite de la aceștia.

Prin clienți ne referim la orice instanță a jocului care rulează pe un dispozitiv conectat la internet a cărui sistem de operare este Windows, oricare dintre versiunile 7, 8 sau 10.

3. Avantajele și dezavantajele metodei alese

Unity este o platformă facilă pentru crearea jocurilor de dimensiuni mici și mijlocii, însă la pachet cu aceasta vin o serie de avantaje și dezavantaje cu care orice dezvoltator intră în contact mai devreme sau mai târziu în cadrul procesului de implementare a aplicației.

Un prim argument pro este faptul că platforma Unity nu necesită cunoștințe foarte vaste despre programare, iar dacă o situație rară impune acest lucru, tutorialele sunt la îndemâna oricui. Folosind o viziune bazată pe componente, obiectele sunt create (cum ar fi: jucătorii, inamicii, raza conectoare etc.) și ulterior acestora li se adaugă o serie de atribute și caracteristici (la nivel grafic, comportamental etc.). Mai mult, prin intermediul interfeței de tip drag&drop, editorul grafic permite desenarea unor elemente, plasarea în scenă a eroilor, rearanjarea entităților, precum și testarea ulterioară în timp real.

Un alt avantaj major al mediului de dezvoltare este suportul pentru diferite platforme (Windows, Linux, Android, iOS, Xbox etc), dintre care acum menționăm doar două: Windows și Linux. Cele două se află în plan principal datorită modului de a controla entitățile în timpul jocului. La toate acestea se

adaugă faptul că Unity acceptă DirectX și OpenGL, permițând adăugarea de efecte care sporesc dinamismul jocului.

Un alt punct forte pentru Unity este disponibilitatea. Fiind o platformă de dezvoltare gratuită, numărul adepților este unul considerabil, chiar dacă nu toate facilitățile sunt disponibile în varianta free.

Nu în ultimul rând, trebuie menționat faptul că Unity furnizează o bibliotecă vastă de active și plugin-uri prin intermediul cărora procesul de dezvoltare devine mai rapid și mai ușor.

În ceea ce privește dezavantajele metodei de dezvoltare alese, putem face referire la necesitatea unor cunoștințe vaste ale limbajului C# pentru jocuri de o complexitate considerabilă care inerent conduc la un timp de rulare din ce în ce mai mai mare. Astfel, un număr mare de componente pot conduce la performanțe scăzute în ceea ce privește necesarul resurselor hardware.

4. Limitele în care funcționează metoda

În cazul în care se dorește implementarea unei aplicații de dimensiuni mici sau mijlocii, platforma livrează perfect resursele necesare în procesul de dezvoltare. Astfel, un joc care reunește un număr relativ redus de module, tipar în care se încadrează și Sparking Ray, va fi ușor de creat și ușor de rulat. Începând cu jocurile care depășesc câteva sute de magaocteți, totul devine din ce în ce mai anevoios, iar multe dintre platformele mobile încep să iasă din calcule.

5. Proiectarea propriu-zisă (diagrame UML, scheme logice)

Reprezentările arhitecturale ale aplicației variază în funcție de aspectele discutate, însă toate folosesc diagrame ale limbajului de descriere și modelare UML.

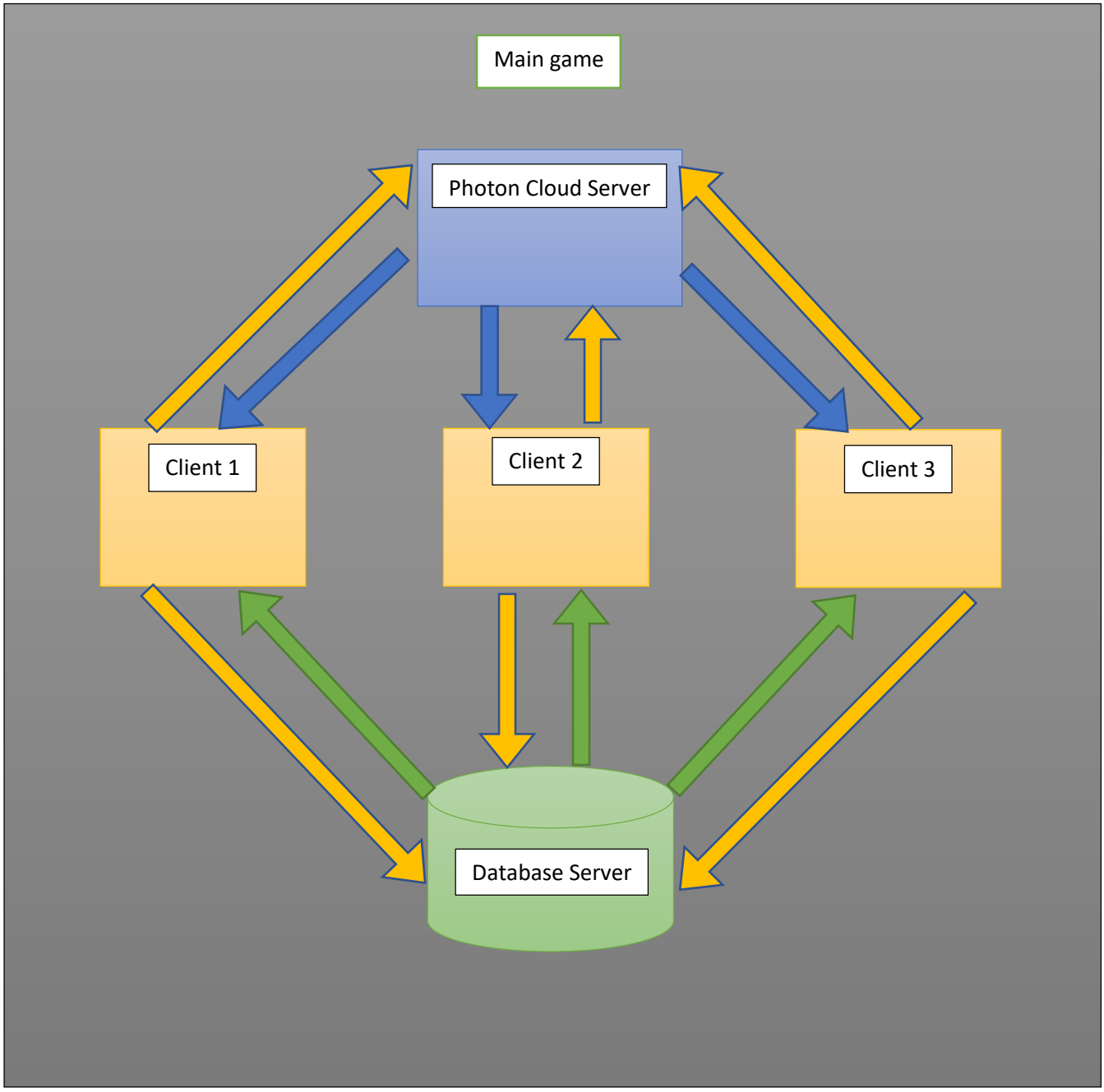


Fig.1 – componentele principale ale aplicației și interacțiunea dintre acestea

Ținând cont de gradul de complexitate a proiectului, se vor prezenta în continuare câteva reprezentări specifica modului de organizare și funcționare a aplicației.

Diagrama cazurilor de utilizare

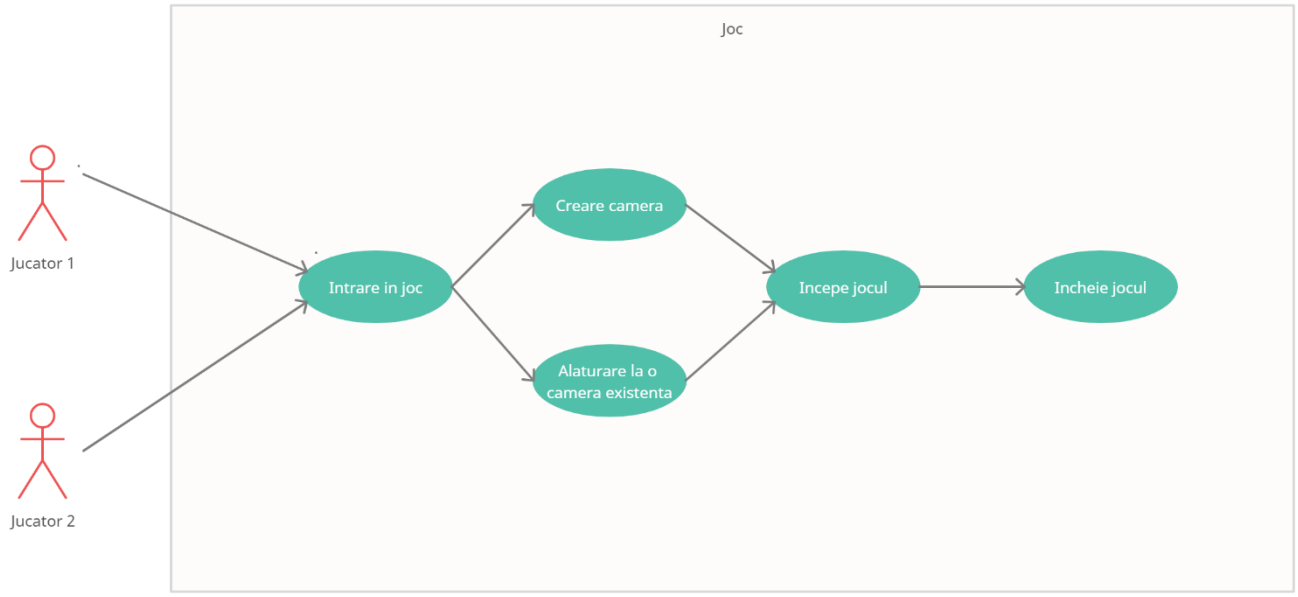


Fig. 2 – cazurile de utilizare ale aplicației împreună cu actorii

Intrare în joc: pe lângă punctul principal de intrare în aplicație, reprezintă și meniul în care se vor face alegerile următoare;

Creare camera: permite crearea unei camere de joc prin introducerea următoarelor date: nume pentru cameră și nume pentru jucător; ulterior se realizează confirmarea alegerii;

Alăturare la o cameră existentă: dacă un alt jucător a creat deja o cameră de joc, jucătorul curent se poate alătura introducând numai numele camerei, și numele de jucător;

Începe jocul: odată ce jucătorii s-au alăturat unei camere (minim 2), jocul poate începe, iar inamicii încep să fie generați pe hartă;

Încheie jocul: în acest caz se ajunge dacă numărul de puncte vitale a ajuns la zero; se permite începerea unui nou joc.

Diagrama de clase

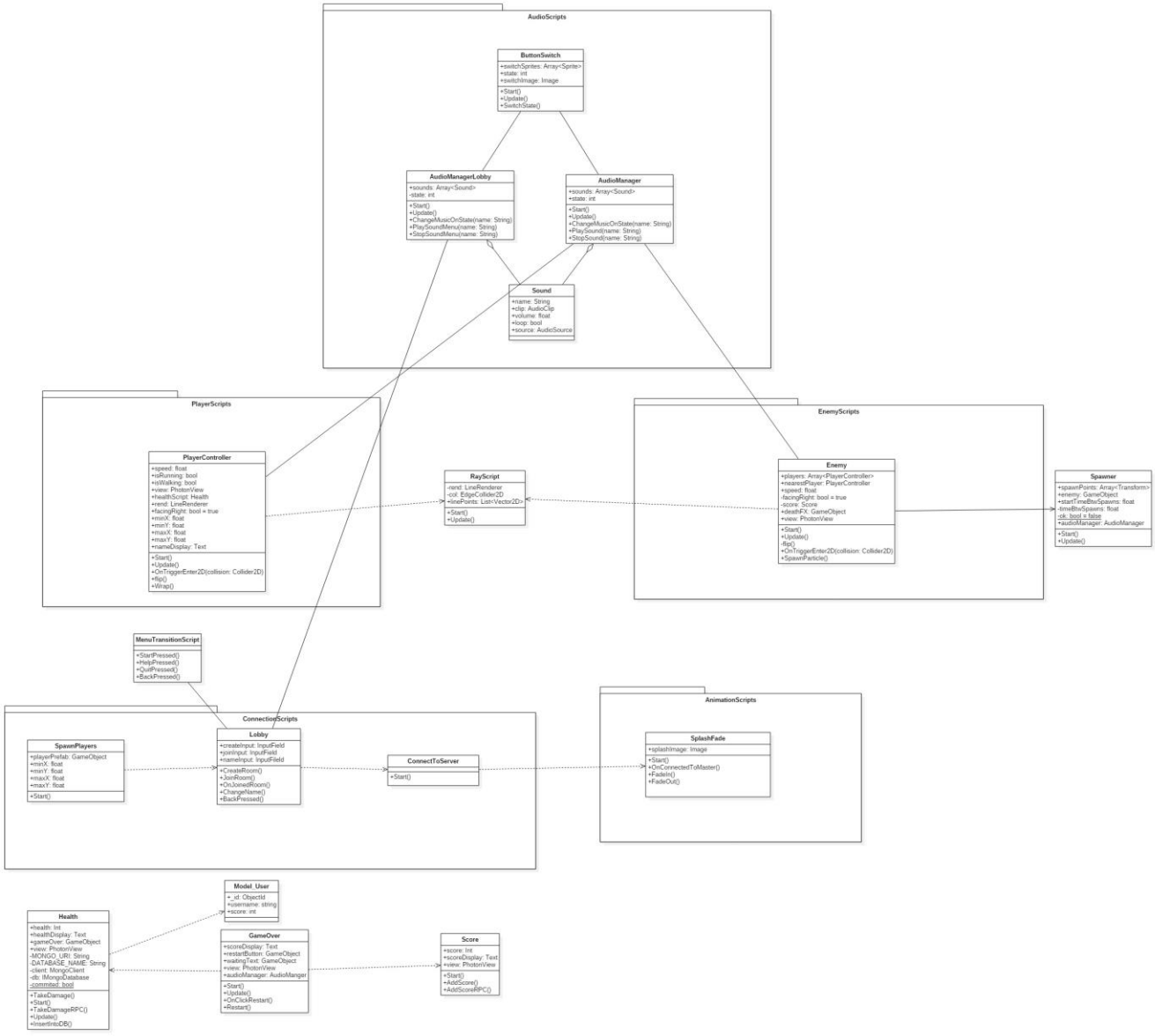


Fig. 3 – componentele principale ale aplicației și interacțiunea dintre acestea

În faza de proiectare s-a ținut cont de următoarele aspecte:

- organizarea claselor în pachete cu scopul de a face distincția între unitățile logice ale aplicației;

- pe masură ce vor fi implementate și testate, claselor existente li se vor adăuga membri și metode noi, dar și relații cu noi clase.

Pachetele împart aplicația în următoarele unități logice:

- PlayerScripts: asigură instanțierea caracterului jucătorului, afișarea acestuia în scenă și interacțiunea cu celelalte componente: inamici, scor, puncte de viață;

- ConnectionScripts: clasele din acest pachet sunt responsabile de întreținerea conexiunii în rețea dintre jucatori;

- EnemyScripts: conține toate clasele care descriu comportamentul inamicilor, pe lângă instanțiere și punere în scenă;

- AnimationScript: cuprinde clase responsabile de tranziții și scene din meniul jocului.

- AudioScripts: clase responsabile de rularea conținutului audio

Diagrama de secvențe

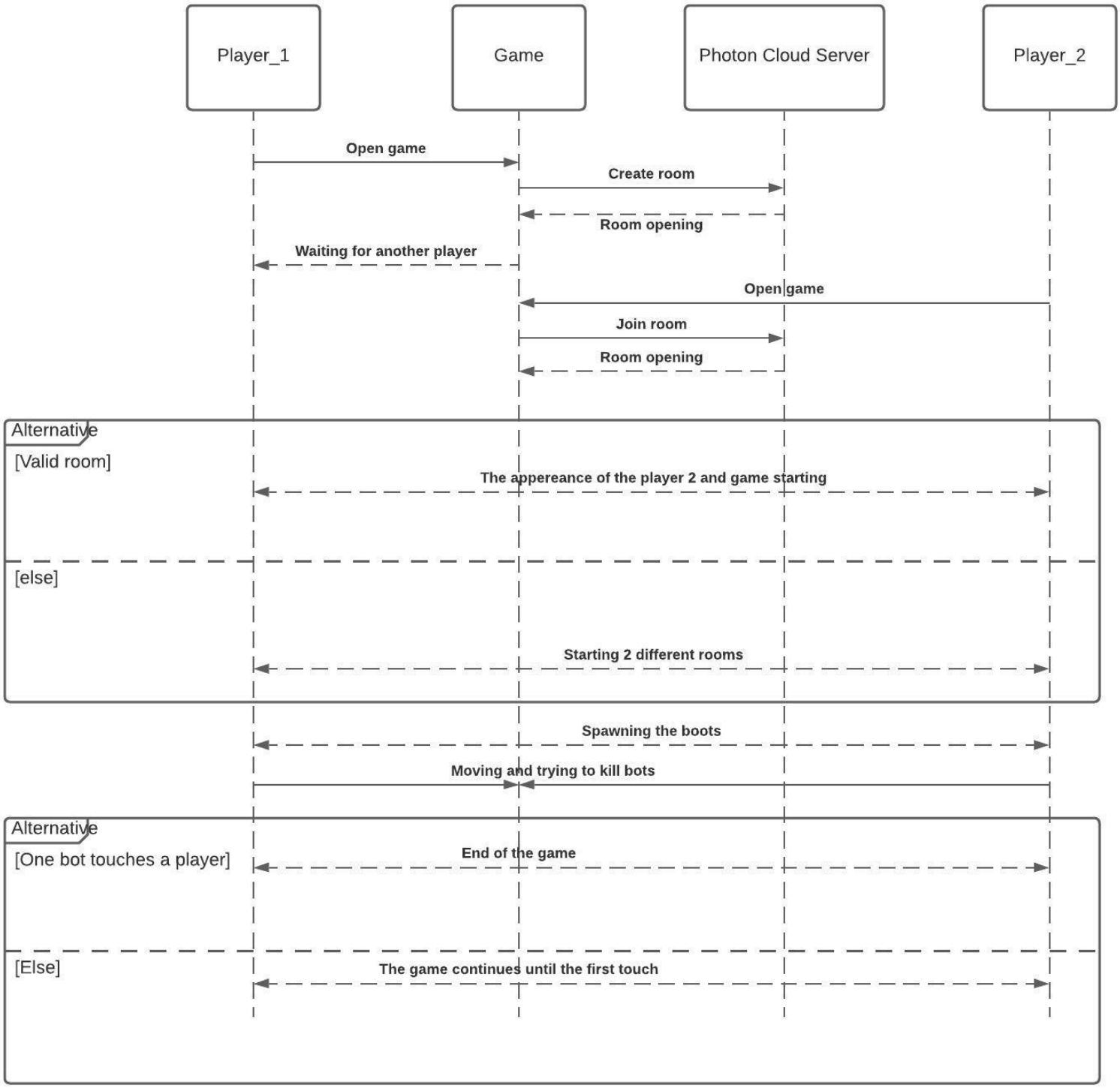


Fig. 4 – Diagrama de secvențe prin care se prezintă cronologic interacțiunea componentelor aplicației

Diagrama de secvențe reunește totalitatea etapelor parcurse în ordine cronologică după cum urmează:
- în primă fază, un jucător lansează aplicația în execuție și ulterior creează o cameră căreia trebuie să îi fie atribuit un nume; tot în această etapă și jucătorul trebuie să primească un nume.
- în pasul imediat următor se realizează conexiunea la serverul Photon, iar camera este deschisă și pentru alți utlizatori care doresc să se conecteze și cunosc numele camerei;
- în acest stadiu, utlizatorul care a creat camera este într-o stare de așteptare a unui alt player pentru a putea începe jocul propriu-zis;
- după conectarea celui de-al doilea utilizator în camera creată cu datele valide, începe întreaga acțiune: se realizează procesul de „bot spawning”, iar cei doi coechipieri trebuie să distrugă cu raza care îi conectează cât mai mulți monștri;
- jocul durează până când unul dintre cei doi playeri este atins de un bot, iar la final se afișează scorul cumulat prin distrugerea boților pe parcursul rundei respective.

Diagrama de deployment

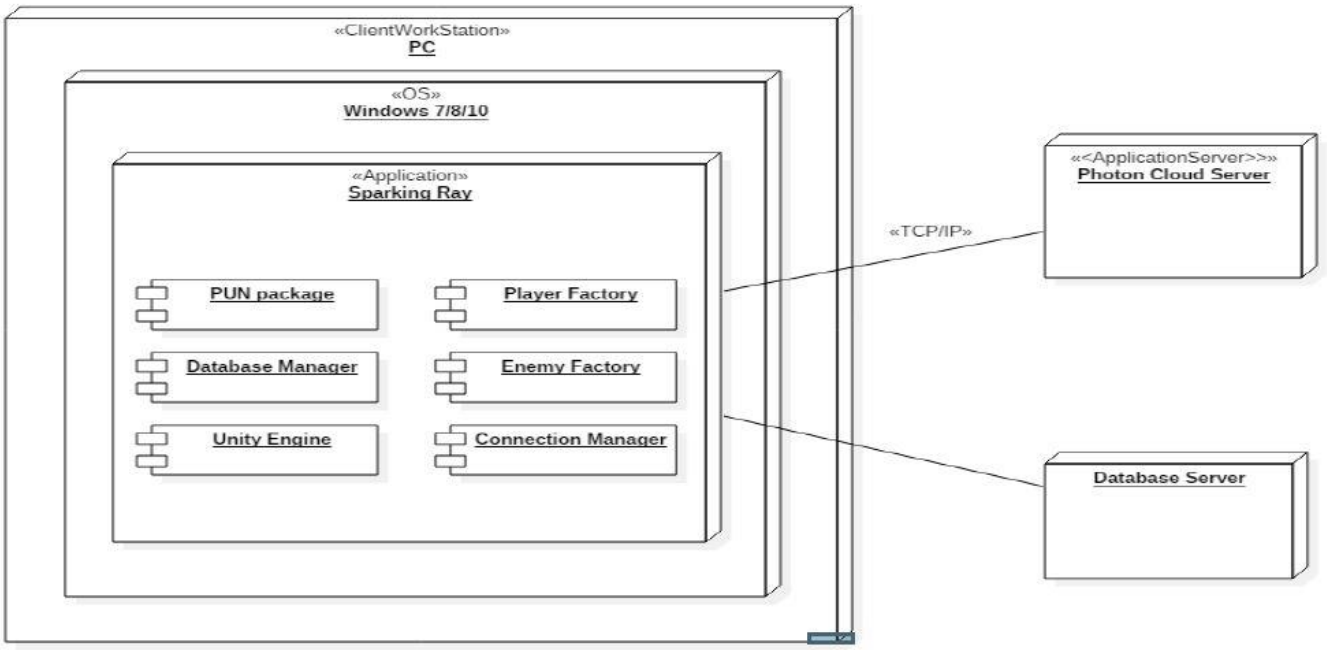


Fig. 5 – Diagrama de deployment

Se remarca faptul că jocul nu necesită multe cerințe funcționale, doar o conexiune la internet și un dispozitiv al cărui sistem de operare este Windows, versiunile 7/8/10.

Diagrama de stare

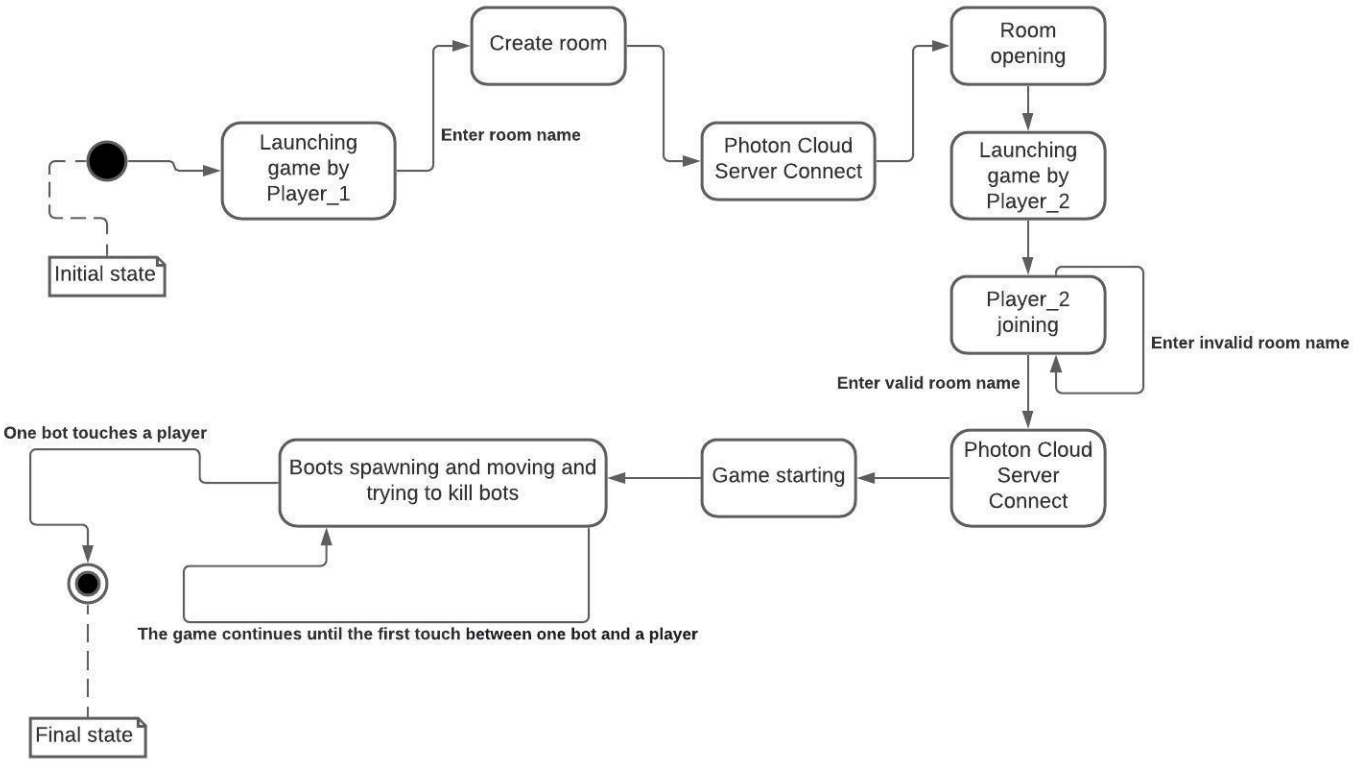


Fig. 5 – Diagrama de stare – totalitatea stărilor, condițiilor și posibilităților de trecere de la o stare la alta

Diagrama de stare are la bază două concepte fundamentale: stările care descriu obiecte sau seturi de obiecte, dar și evenimentele care constau în stimuli externi, ce acționează asupra obiectelor și determină tranziția de la o stare la alta. Cu alte cuvinte, diagrama de mai sus nu reprezintă altceva decât un graf, în care nodurile sunt stări și arcele direcționate reprezintă tranzițiile care pot conține etichete care specifică numele evenimentelor care le-au provocat. De asemenea, se precizează acțiunile rezultate din schimbări. La fel ca în cazul diagramei de secvențe, se expune succesiunea stărilor parcurse (considerăm redundantă o prezentare identică cu cea din cazul diagramei de secvențe) de la lansarea în execuție a jocului și până la terminarea unei runde. Se remarcă necesitatea condiției de inserare a numelui valid în cazul camerei pentru ca jocul să poată să înceapă, precum și faptul că un singur contact între un bot și un player conduce la terminarea jocului.

Diagrama ER

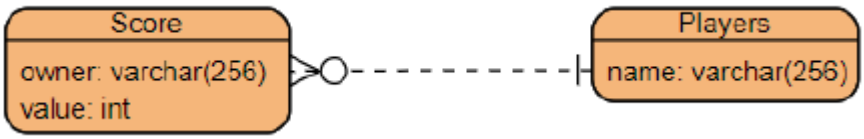


Fig. 6 – Diagrama ER

Serverul de baze de date va fi menținut în cloud, cu ajutorul unui Database-as-a-service: MongoDB Atlas.

6. Tehnologii utilizate în implementare

Sparking Ray a fost implemetat utilizând **game-engine-ul Unity** care oferă dezvoltatorului o serie de beneficii majore, prezentate anterior la punctul 3, dintre care remarcăm: interfața grafică puternică și posibiliatea de a crea script-urile aferente în **limbajul de programare C#** pentru aproape orice entitate prezentă în proiect. Limbajul oferă diverse variante de rezolvare a problemelor datorită viziunii programării bazate pe entități de tip GameObject sau orientate-obiect. De evidențiat este utilizarea pachetului **PUN (Photon Unity Networking)** împreună cu o aplicație de tip server menținută în cloud ce oferă o capacitate de până la 20 de jucători concurenți. În plus, jucătorii pot trimite către serverul de baze de date informații cu privire la scorul dintr-un joc terminat, în vederea înregistrării într-un clasament. Nu în ultimul rând, precizăm faptul că serverul de baze de date va fi menținut în cloud, cu ajutorul unui Database-as-a-service: **MongoDB Atlas**.

7. Descrierea claselor

Din diagrama claselor reies urmatoarele clase principale:

- **Lobby**: permite crearea unei camere de joc sau alăturarea la o cameră existentă, cât și alegerea unui nume de jucător, care va fi utilizat mai departe in reținerea scorurilor în baza de date;
- **MenuTransitionScript**: prin metodele oferite de această clasă se realizează trecerea de la o scena la alta in cadrul meniului principal;
- **PlayerController**: pe lângă controlul miscărilor si al animațiilor corespunzătoare acestora, clasa descrie si metode de interacțiune a jucătorilor cu inamicii, în special pagubele suferite;
- **Enemy**: descrie interacțiunea monștrilor cu jucătorii(direcție de deplasare, poziționare in scenă, distrugere in contact cu raza, efecte audio/vizuale);
- **Spawner**: prezintă mijloacele necesare de instanțiere a inamicilor în scenă, pe baza unor puncte predefinite.

III. Testarea aplicaţiei şi rezultate experimentale

1. Lansarea aplicatiei, elementele de configurare si instalare;

Lansarea cu succes a aplicatiei necesita prezenta pe masina utilizatorului a structurii de fisiere din figura urmatoare:

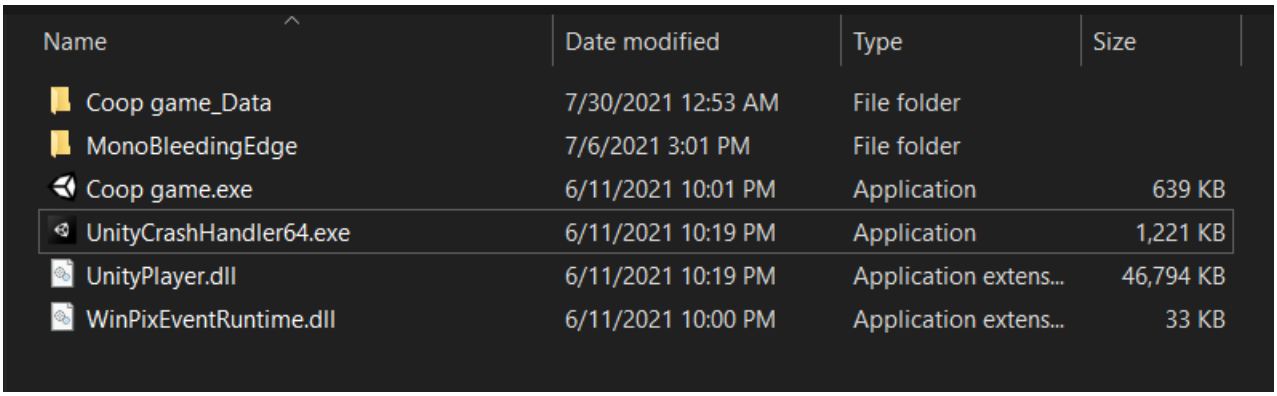


Fig. 7 – Structura directorului din care are loc lansarea aplicatiei

Aceasta rezultă in urma compilării proiectului. Lansarea efectiva se face ruland executabilul “Coop game.exe” care va deschide o fereastra nouă. In continuare utilizatorului ii sunt expuse cateva cadre din meniul principal.

Fig. 8 – Meniul principal

Fig. 9 – Fereastra de ajutor asociată aplicației

Fig. 10 – Fereastra de tip lobby a jocului

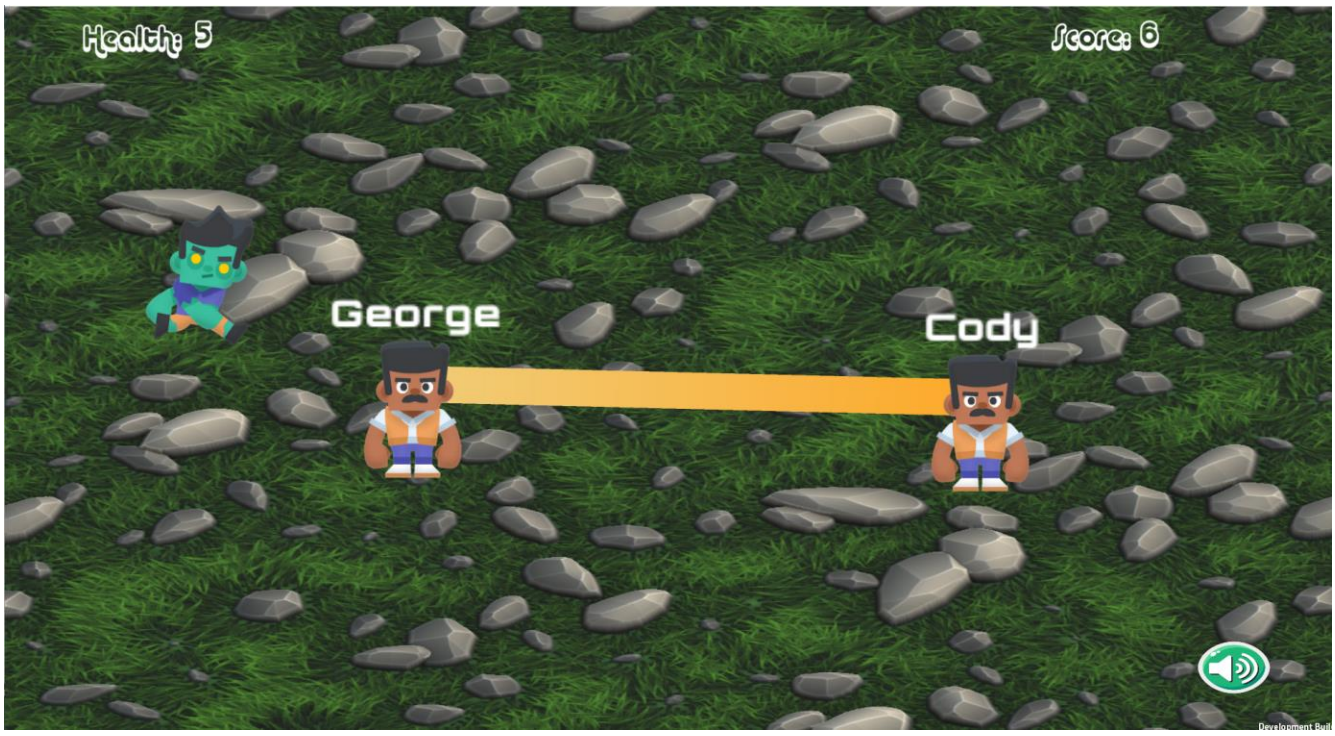


Fig. 11 – Desfasurarea jocului

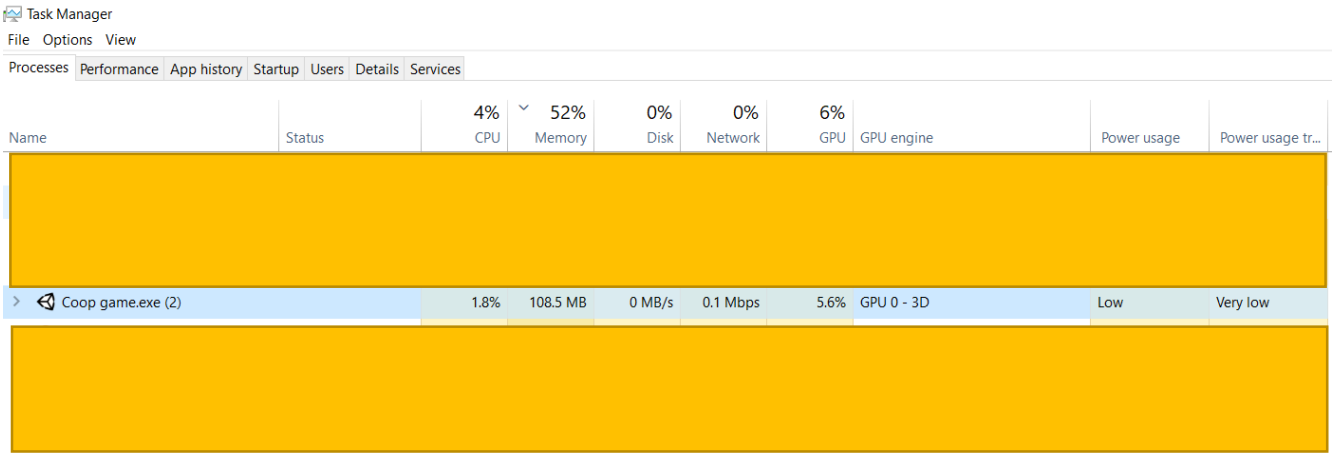
Jocul se incheie atunci când punctele vitale ating pragul nul. Începerea unui joc nou este stabilită exclusiv de creatorul camerei, clientului care s-a alăturat ulterior ii va fi afișat un mesaj de informare.

Fig. 12 – Fereastra sfârșitului de joc pentru master

Fig. 13 – Fereastra sfârșitului de joc pentru client

2. Aspecte legate de încărcarea procesorului, memoriei,limitări în ce privește transmisia datelor/comunicarea

Conform datelor prezentate de managerul de resurse Windows 10, aplicatia nu supune sistemul unei incarcari mari, rezultand ca poate rula pe orice statie cu cerinte minime de sistem.



3. Aspecte legate de fiabilitate/securitate

Fiabilitatea aplicației este asigurată de disponibilitatea serverelor de joc si de baze de date iar securitatea este intreținută de framework-ul Photon care pune la dispoziție apeluri sigure in rețea.

Concluzii

1. Gradul in care s-a realizat tema propusă

Tema propusă a fost realizată într-un procent majoritar, deciziile de proiectare si implementare care au fost luate pentru a ajunge in acest punct au fost legate de modificarea diagramei de clase.

2. Direcții de dezvoltare

Deși a fost realizată complet funcțională, aplicația poate fi cel puțin finisată atât prin corectarea unor eventuale erori care pot aparea pe parcurs, cât si integrarea unor sisteme de tip lobby management, care ofera posibilitatea utilizatorilor de a vizualiza camerele disponibile pe server.

De asemenea, tot in lobby poate fi adăugat un sistem de chat, care să permită comunicarea jucătorilor din aceeași cameră.

Alte imbunătățiri pot fi aduse in desfășurarea jocului, prin creșterea graduală a dificultății după trecerea unei perioade de timp sau depășirea unui prag de scor. Adăugarea de obiecte care pot oferi un avantaj temporar jucătorilor reprezintă o altă direcție de dezvoltare benefică. La fel și folosirea datelor din baza de date pentru afișarea unui clasament.

3. Responsabilitățile fiecărui membru al echipei

Analiza

- stabilirea tematicii jocului: împreună
- stabilirea arhitecturii aplicației:
    - stabilirea mijloacelor de comunicare în rețea: Eduard Codău
    - stabilirea modelului de comunicație: George Apetria

Proiectare

- diagrama de clase: Eduard Codău
- diagrama cazurilor de utilizare: Eduard Codău
- diagrama de secvențe: George Apetria
- diagrama de deployment: Eduard Codău
- diagrama de stare: George Apetria
- diagrama ER: Eduard Codău

Implementare

- codul necesar conectării jucătorilor în rețea: Eduard Codău
- asset-urile pentru jucători si inamici: Eduard Codău
- asset-urile pentru fundal si rază: George Apetria
- efecte vizuale si audio: George Apetria
- meniul principal:
    - fereastra help: George Apetria
    - fereastra de tip lobby: Eduard Codău
- interacțiune jucător-scenă: Eduard Codău
- interacțiune jucător-inamici: Eduard Codău
- interacțiune jucător-jucător: Eduard Codău
- fereastră sfârsit de joc: Eduard Codău
- nume utilizator: George Apetria
- mijloacele de comunicare cu serverul de baze de date: Eduard Codău