UNIVERSITATEA POLITEHNICA BUCUREȘTI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

|  |  |
| --- | --- |
| upb | cs |

PROIECT DE DIPLOMĂ

TODO: Joc puzzle-adventure în Unity

TODO Subtitlu (ex: versiunea 2018)

Sergiu-Andrei Agachi

**Coordonator științific:**

Prof.dr.ing. Moldoveanu Alin Dragoș Bogdan

BUCUREŞTI

2019

UNIVERSITY POLITEHNICA OF BUCHAREST

FACULTY OF AUTOMATIC CONTROL AND COMPUTERS

COMPUTER SCIENCE DEPARTMENT

|  |  |
| --- | --- |
| upb | cs |

DIPLOMA PROJECT

TODO Puzzle-Adventure game in Unity

TODO Subtitle (ex: 2018 version)

Sergiu-Andrei Agachi

**Thesis advisor:**

Prof.dr.ing. Moldoveanu Alin Dragoș Bogdan

BUCHAREST

2019

**CUPRINS**

[Sinopsis 3](#_Toc18143716)

[Abstract 3](#_Toc18143717)

[1 Introducere 4](#_Toc18143718)

[1.1 Motivație 4](#_Toc18143719)

[1.2 Engine 4](#_Toc18143720)

[1.3 Modalitate de lucru 4](#_Toc18143721)

[2 STUDIUL DOMENIULUI 6](#_Toc18143722)

[2.1 Crypt of the NecroDancer 6](#_Toc18143723)

[2.2 Magirune 7](#_Toc18143724)

[2.3 The enchanted cave 2 8](#_Toc18143725)

[2.4 I Am Overburdened 10](#_Toc18143726)

[2.5 Concluzia studiului 11](#_Toc18143727)

[3 Funcționalitatea jocului 12](#_Toc18143728)

[3.1 Configurația terenului și elementele sale 13](#_Toc18143729)

[3.2 Indicatori vizuali 17](#_Toc18143730)

[4 Arhitectura si implementre 18](#_Toc18143731)

[4.1 Scipturi 18](#_Toc18143732)

[4.2 Scene 18](#_Toc18143733)

[4.3 Loader 19](#_Toc18143734)

[4.4 GameManager 19](#_Toc18143735)

[4.5 LivingObject și MovingObject 19](#_Toc18143736)

[4.6 Player 20](#_Toc18143737)

[4.7 Enemy 20](#_Toc18143738)

[4.8 HealthBar 21](#_Toc18143739)

[4.9 Scripturi pentru camera 21](#_Toc18143740)

[4.10 Alte scripturi 21](#_Toc18143741)

[5 Evaluare si Testare 21](#_Toc18143742)

[6 Concluzii 23](#_Toc18143743)

[7 Bibliografie 24](#_Toc18143744)

[8 Anexe 26](#_Toc18143745)

# Sinopsis

Sinopsisul proiectului are rol de introducere, conținând atât o descriere pe scurt a problemei abordate cât și o enumerare sumară a rezultatelor și a concluziilor. Se recomandă ca sinopsisul să fie redactat într-un limbaj accesibil unei persoane nefamiliarizate cu domeniul, dar în același timp destul de specific pentru a oferi rapid o vedere de ansamblu asupra proiectului prezentat.

Sinopsisul proiectului va fi redactat atât în română cât și în engleză. Ca dimensiunea recomandată aceasta secțiune va avea maxim 200 de cuvinte pentru fiecare variantă. Împreună, ambele variante se vor încadra într-o singură pagină.

# Abstract

The abstract has an introductory role and should engulf both a brief description of the issue at hand, as well as an overview of the obtained results and conclusions. The abstract should be formulated such that even somebody that is unfamiliar with the projects’ domain can grasp the objectives of the thesis while, at the same time, retaining a specificity level offering a bird’s eye view of the project.

The projects’ abstract will be elaborated in both Romanian and English. The recommended size for this section is limited to 200 words for each version. Together, both versions will fit in one page.

# Introducere

## Motivație

În ultimii ani, industria dezvoltării de jocuri video a intrat într-un stadiu de dezvoltare din ce în ce mai rapidă. Această creștere se reflectă atât în ceea ce privește tipul jocurilor, dar și a platformelor pentru care jocurile sunt create. Principalul aspect ceea ce a produs acest fenomen este raportul dintre costurile reduse de a produce un produs comparativ cu profitul considerabil obținut în urma lor. Astfel din ce în ce mai multe companii mari cât și dezvoltatori independenți au început să pătrundă pe această piață competitivă. În timp ce numărul consumatorilor acestui produs crește, în același timp crește și numărul celor care participă în procesul de producție al unui joc.

Unul dintre efectele cauzate de evoluția rapidă a domeniului acela că fiecare creator de jocuri (companie sau individual) dorește să creeze conținut influențat de propria viziune. Astfel sunt îmbinate diferite aspecte pozitive și sunt înlăturate alte aspecte negative conform perspectivei dezvoltatorului pentru a crea un produs unic și inovativ.

## Engine

În dezvoltarea unui joc video apare necesitatea folosirii unui produs software extern pentru a facilita procesul. Astfel au fost create diferite motoare de dezvoltare, care pe parcursul lucrării, vor fi adresate după denumirea lor în engleză: “game engine” sau pe scurt, doar “engine”. Odată cu creșterea industriei a apărut necesitatea dezvoltării a din ce în ce mai multe engine-uri ale căror funcționalități să satisfacă nevoi precum cea a dezvoltării produsului pe mai multe platforme precum Windows, Mac, Linux, Android, IOS, PS4.

Proiectul folosește ca și engine Unity, acesta fiind unu dintre cele mai populare dar în același timp și unul dintre cele mai accesibile pentru diferite niveluri de experiență în domeniu, platformele de dezvoltare vizate fiind Windows si Android. Sunt folosite funcționalități oferite de acesta precum: editorul vizual integrat, sistemul de scene multiple sau scripturi, pentru a interconecta fiecare componenta a jocului. Aspectul vizual al proiectului este construit în editorul vizual, iar funcționalitatea fiecărei componente se regăsește în scripturi ce sunt scrise în limbajul C#.

## Modalitate de lucru

Scopul acestei lucrări este de a evidenția procesul de dezvoltare al unui produs de tipul joc video de tipul puzzle-adventure utilizând engine-ul Unity. Obiectivul unui joc puzzle este acela de a îmbina diferite “piese” pentru a obține produsul final sau pentru a ajunge la o concluzie. Genul adventure are ca principii fundamentale libertatea de decizie a utilizatorului și progresul bazat pe o secvență de sarcini ce trebuie urmate, fiecare cu recompensa ei, pentru a atinge obiectivul. Combinând aceste două genuri obținem un format în care utilizatorul este pus în situația de a urmări o serie de evenimente și de a le ordona pentru a îndeplini un scop final.

Tipul jocului fiind puzzle, am considerat că cea mai bună abordare grafică a jocului este de a crea vedere top-down asupra lumii, obiectele fiind bidimensionale. Astfel tabla de joc poate fi asemănată cu o tablă de șah, toate componentele se află la un moment dat într-o căsuță și niciun alt obiect nu se mai află în acel moment în aceeași căsuță.

Astfel proiectul va contura o imagine de ansamblu privind pașii necesari dezvoltării unui joc ce corespunde descrieri menționate mai sus, privind asupra dificultăților întâlnite alături de soluțiile acestora.

Capitolul 1 prezintă tema și scopul lucrării și contextul în care aceasta a fost realizată.

Capitolul 2 analizează concurența deja existentă pe piață și urmărește să analizeze în ce au excelat aceștia și unde abordarea nu a funcționat conform așteptărilor. Sunt analizate atât jocuri care au avut succes pe piață cât și jocuri considerate reușite, dar care nu au fascinat un public larg, ci doar un public de nișă.

Capitolul 3 explică funcționalitatea jocului. Sunt detaliate aici toate mecanismele prezente în joc împreună cu fiecare obiect în parte și rolul acestora pe parcursul jocului.

Capitolul 4 descrie arhitectura structurală și funcțională a proiectului. În acest capitol sunt enumerate detaliile tehnice și sunt argumentate deciziile de implementare împreună cu motivele pentru care alte metode nu ar fi funcționat.

Capitolul 5 urmărește să prezinte evaluare tehnică, dar și cea umană a jocului. Aici este prezentat feedback-ul obținut în urma terminării jocului de către voluntari. Astfel poate fi aproximat modul în care un public percepe acest joc în cazul în care el ar fi lansat pe piață.

Capitolul 6 formulează concluzia trasă în urma realizării acestui proiect, prezentând cunoștințe dobândite. Sunt incluse și sugestii de îmbunătățiri obținute din urma feedback-ului.

# STUDIUL DOMENIULUI

În orice domeniu există concurență, în cazul de față aceasta este reprezentată de diferitele jocuri care cad sub aceeași clasificare și/sau conțin aspecte conceptuale de natură comună cu produsul analizat în această lucrare. În continuare sunt prezentate câteva titluri, ale căror tematică se apropie foarte mult de tematica jocului dezvoltat, printr-o scurtă descriere și o analiză personală a punctelor tari și a punctelor slabe.

## Crypt of the NecroDancer

Povestea jocului prezintă o fată care pleacă în căutarea tatălui ei care a dispărut. În timpul acestui drum ea cade într-o criptă al cărei stăpân, un vrăjitor, îi fură inima fetei și o obligă să îi înfrunte armata de monștri în timp ce se mișcă pe ritmul bătăilor unei inimi pentru a-și recupera ceea ce i-a fost furat.

Jocul cade sub categoria roguelike, astfel că jucătorul este nevoit să exploreze diferitele niveluri ce sunt generate aleatoriu. Aspectul unic al jocului este oferit de ritm, astfel că în afara mecanismelor uzuale, toate deciziile trebuie luate în timp real în timp ce se respectă un ritm al acțiunilor, ritm determinat de melodia din fundal. În Figura 1 poate fi observat ritmul ce trebuie respectat. Dacă utilizatorul nu poate să respecte ritmul, acesta nu va primi o penalizare directă, însă acțiunea dorită nu va fi executată cu succes. Astfel apare necesitatea de a menține un ritm alert al acțiunilor îmbinat cu necesitatea de a deprinde tiparele predeterminate ale inamicilor. De precizat este că jocul permite jucătorului să adauge propriile melodii în joc pentru fiecare nivel, singura restricție este aceea că tempo-ul melodiilor trebuie sa fie ascendent de la cel mai ușor nivel până la cel mai dificil.



Figura 1 Acțiunile jocului trebui să fie realizate când barele de ritm ating inima[[1]](#footnote-1)

Puncte tari:

* joc ce inovează printr-un mecanism simplist de ritm
* caracteristici simpliste adecvate pentru platforme mobile, acesta fiind disponibil pe toate platformele de actualitate la momentul scrierii acestei lucrări

Puncte slabe:

* incipitul are un grad înalt de dificultate, tendință ce continuă pe tot parcursul poveștii, fapt ce duce jocul într-o regiune de nișă în ceea ce privește publicul țintă

## Magirune

Jocul spune povestea unui personaj care intră într-o peșteră pentru a căuta un cufăr. După ce acesta îl găsește își dă seama că este gol, dar când încearcă să iasă din peșteră, o piatră blochează drumul fapt ce servește ca scop al aventurii, să găsească un drum alternativ pentru a scăpa. Pentru a duce misiunea la bun sfârșit, personajul trebuie sa treacă prin diferite niveluri, să învingă inamicii pe care îi întâlnește, să colecteze diferite obiecte cum ar fi chei sau poțiuni și în final să rezolve diferitele puzzle-uri ce îi împiedică adesea drumul. Un asemenea nivel este prezentat in Figura 2.

O altă mecanică importantă a jocului este aceea a unui sistem de progres în experiență al personajului. De fiecare dată când personajul învinge un inamic pentru prima oară, acesta îi oferă puncte de experiență (din engleză, “experience points", sau pe scurt xp). Însă atunci când personajul părăsește încăperea, inamicii revin la viață, dar de data asta, dacă aceștia sunt învinși oferă un singur punct de experiență. Pentru a urmări progresul în experiență, dar și pentru a vedea ce inamici au fost învinși sunt puși la dispoziție niște indicatori vizuali, aceștia fiind ilustrați în Figura 3.

A picture containing indoor, building

Description automatically generated

Figura 2 Schema unui nivel în care inamicii învinși sunt transparenți

A black and silver text on a screen

Description automatically generated

Figura 3 Detalii despre inamicii existenți [[2]](#footnote-2)

Puncte tari:

* jocul este alcătuit din 46 de niveluri ceea ce îi conferă destul conținut pentru un joc de acest tip
* mai multe finalități ale jocului îi oferă jocului potențial de a fi rejucat

Puncte slabe:

* lipsa unor informații incipiente în legătură cu sistemele jocului
* finalul considerat cel mai bun presupune un număr aproape prestabilit de acțiuni, astfel că orice mică greșeală poate preveni obținerea rezultatului dorit
* sistemul de experiență este contra intuitiv pentru un sistem la prima vedere
* singura platformă pe care jocul este disponibil este Windows.

## The enchanted cave 2

Personajului îi este prezentat încă de la început scopul jocului, și anume să intre într-o peșteră ce conține o cantitate nesfârșită de comori. Resursele acestuia fiind limitate, el trebuie să își aleagă bine strategia pentru a maximiza câștigul obținut în urma expediției. Există două metode de a părăsi peștera , fie obține un artefact pentru așa ceva, după cum este exemplificat în Figura 5, fie este învins și este forțat să înceapă de la zero. Dacă acesta ajunge să fie învins în peșteră el va pierde tot echipamentul și armura dobândită în urma expediției, singurele lucruri ce sunt permanente fiind aurul și experiența. Figura 4 prezint cum în urma dobândirii a unei cantități suficiente de experiență, jucătorul va avansa la un nou nivel fapt ce îi oferă posibilitatea de a își îmbunătăți un atribut la alegere, jucătorul astfel fiind capabil să aleagă cum dorește să joace jocul.

“The Enchanted Cave 2” este un joc de tipul roguelike ceea ce determină un stil de joc bazat pe raportul dintre risc si recompensă pentru fiecare acțiune. Strategia este unul dintre cele mai importante aspecte ale acestui joc, acest aspect se răsfrânge asupra tuturor mecanismelor prezente fie ele legate de modul de alegere al echipamentului până la a ști când este momentul pentru retragere. Un alt aspect al unui joc roguelike este acela că de fiecare dată schema nivelurilor este generată aleatoriu ceea ce face unică fiecare încercare de a duce aventura la bun sfârșit.

A circuit board

Description automatically generated

Figura 4 Avansare în nivel

A screen shot of a computer

Description automatically generated

Figura 5 Salvarea din peșteră

Puncte tari:

* existența unui arbore cu abilități și al unui sistem de echipament oferă alt curs fiecărui joc distinct
* este disponibil pe diferite platforme Windows, Linux, MAC, dar și pe platformele mobile Android și IOS

Puncte slabe:

* mecanica de ieșire din peșteră este neiertătoare ceea ce oferă o complexitate ridicată jocului iar audiența către care este adresat jocul este astfel diminuată

## I Am Overburdened

“I am Overburdened” este un joc despre un vânător de recompense, al cărui capacitate de transportat obiecte este surprinzător de mare, ce pleacă într-o călătorie al cărei scop este să colecteze diferite artefacte dintr-o peșteră.

Ceea ce face acest joc unic este numărul foarte mare de obiecte colectabile în pe parcursul jocului. În Figura 6 se poate observa cum jucătorul poate purta în același timp 20 de obiecte fiecare de tipuri diferite, ceea ce determină un număr foarte mare și imprevizibil de acțiuni. Aceste aspecte sunt combinate cu o poveste umoristică pentru a face experiența mai plăcută. Jocul este menit să fie jucat în reprize scurte deoarece înfrângerea este permanentă și nimic dintr-o rundă nu se continuă în următoarea, cu excepția cunoștințelor despre artefacte a utilizatorului ce îl va ajuta în următoarele încercări.

A picture containing building, fence

Description automatically generated

Figura 6 Se observă în dreapta imaginii numărul de artefacte ce influențează jocul[[3]](#footnote-3)

Puncte tari:

* jocul are ca principal punct de atracție numărul mare de artefacte ce pot fi echipate în același timp conducând la o experiență non-repetitivă

Puncte slabe:

* nu sunt oferite informații la începutul jocului, astfel că scopul jocului nu este bine precizat
* progresul este greu de urmărit deoarece nu există niciun asemenea indicator

## Concluzia studiului

În urma acestei analize pot fi trasate niște concluzii ce servesc ca punct de ghidare în viitoarea implementare a proiectului. Toate titlurile mai sus menționate conțin în clasificarea lor genul “puzzle”, astfel că se observă apariția unor șabloane ce le determină succesul.

Unul dintre aceste trăsături aste aceea că jocurile sunt minimaliste și urmăresc să consume un interval mic de timp. Fiecare dintre aceste runde creează un mediu în care utilizatorii pot petrece o perioadă scurtă de câteva minute pentru a avansa pas cu pas spre final, abordare în special folositoare pentru dispozitivele mobile.

Este de evitat un model ce presupune o curbă de învățare prea abruptă. Utilizatorul obișnuit vrea să se relaxeze într-un mediu provocator, iar obstacolele ce urmează a fi parcurse trebuie să construiască treptat această complexitate globală a jocului.

Un aspect de evitat este lipsa de informare în raport cu scopul jocului. Unele dintre aceste titluri lasă jucătorul să deslușească toate detaliile astfel că o parte dintre aceștia pot abandona jocul fără a-i descoperii potențialul.

ar trebui să mai detaliez?

# Funcționalitatea jocului

În general, de la lansarea unui joc și până la conținutul efectiv al jocului, utilizatorii sunt întâmpinați cu un meniu principal. Acest meniu servește ca o barieră a fluxului jocului, astfel se evită ideea de a lansa jucătorul direct în esența jocului. După cum se observă în Figura 7, acest joc nu face excepție de la regulă și îi prezintă utilizatorului doua opțiuni: să înceapă jocul, sau în caz că s-a răzgândit, poate să îl închidă imediat.



Figura 7 Meniul principal al jocului

În cazul în care se apasă pe buton “Play Game”, jocul va intra într-o sesiune de *tutorial* [[4]](#footnote-4). Acesta va prezenta pe rând fiecare dintre obiectele ce pot apărea pe ecran la un moment dat în timpul jocului. În Figura 8 este prezentă o captură din acest tutorial. Alături de imaginea propriu zisă a obiectului este prezentă și o descriere sumară cu scop informativ, astfel că nu se înlătură elementul surpriză.

A close up of a logo

Description automatically generated

Figura 8 Fragment din ghidul de informare

La finalul indicațiilor incipiente, când jucătorul este pregătit, urmează o introducere în povestea jocului, explicându-se locația și motivul pentru care personajul se găsește la momentul respectiv în acea situație. Prin urmare, scopul acestuia este să iasă din turnul în care acesta a fost blocat, acțiune ce va deveni posibilă numai în urma soluționării unor probleme ce apar treptat pe parcursul aventurii.

După ce jucătorul devine conștient de contextul în care se află, este creată lumea descrisă, astfel că personajul se află la intrarea turnului în care este blocat. Se observă din Figura 9 că un etaj este de forma unui pătrat de dimensiunea 11 pe 11 dintre care 2 dintre blocuri sunt marginile blocante, ceea ce lasă ca spațiu de mișcare o suprafață de 9 pe 9. Vom numi acest spațiu, de 9 pe 9, “teren de joc”. Astfel că pe terenul de joc se vor situa toate obiectele ce au fost prezentate în incipitul jocului în partea de tutorial. Toate etajele au același format, motivul fiind ideea de turn vertical ce nu suferă modificări de structură pe parcursul ascensiunii.

A circuit board

Description automatically generated

Figura 9 Schema terenului de joc al etajului 1

## Configurația terenului și elementele sale

Personajul, vizibil în Figura 10, este actorul principal al acestui joc și este controlat de utilizator. De asemenea este singurul element dinamic al jocului, astfel că el va interacționa pe rând cu celelalte obiecte. În urma acțiunilor sale unele elemente pot dispărea de pe ecran prin colectare sau prin distrugere. Fiind un joc top-down, mișcările personajului sunt limitate de cele două axe, verticală și orizontală, cea de-a doua determinându-i și orientarea. Prin urmare el se poate deplasa, la un moment de timp, doar într-o căsuță adiacentă (sus, jos, stânga, dreapta), în funcție de constrângerile impuse de teren. Dacă personajul se deplasează într-o căsuță care este ocupată deja acesta se va ciocni cu acesta și în funcție de obiect, rezultatul poate varia.

A picture containing indoor, building

Description automatically generated

Figura 10 Personajul controlat de jucător

Pentru a crea o provocare din aventura personajului, au fost create diferite obstacole, astfel că acesta va fi nevoit să le elimine din calea sa. Există 2 tipuri de obstacole ce se diferențiază prin capacitatea de a riposta sau nu. Dintre acestea diferențiem inamicii, care sunt capabil de a ataca personajul, și pereți ce doar blochează drumul. Toate obstacolele sunt statice și au nevoie de o anumită unealtă pentru a fi înlăturate.

Primul tip de obstacole este constituit de inamic. Ei sunt principalele obstacole ale jocului. Aceștia sunt de mai multe tipuri, astfel că diferențele sunt date de înfățișare, aspect vizibil în Figura 11, capacitatea de a supraviețui atacurilor și puterea de a ataca personajul. Ei pot fi combătuți și eliminați din calea personajului doar dacă acesta se află în posesia unei săbii. Un impact cu un inamic va iniția un atac între cei doi în urma căruia ambii vor suferi daune în funcție de capacitatea de atac a adversarului. Aceștia se vor regenera la trecerea dintr-un nivel în altul ceea ce obligă utilizatorul să își formeze o strategie de navigare a labirintului.



Figura 11 Tipurile de inamici

În afara inamicilor, care sunt obstacole ce ripostează, mai există și al doilea tip de obstacole și anume pereții din noroi și piatră. Acești pereți, prezentați în Figura 12, blochează accesul spre diferite rute ale unui etaj sau spre scurtături de întoarcere. Pentru a distruge aceste obstacole este nevoie de o lopată și respectiv de un târnăcop.



Figura 12 Pereți destructibili

Pe lângă obstacole, ce încurcă drumul personajului, există și obiecte menite să în ajute. Un astfel de tip de obiecte sunt uneltele. Ele sunt prezentate în Figura 13. Sunt 4 la număr (sabie, lopată, târnăcop și cheie) și se găsesc o singură dată într-un anumit punct al jocului, astfel că este considerată o provocare încercarea de a le găsi.



Figura 14 Uneltele ajutătoare

Figura 16 prezintă fructele ce servesc drept modalitate de a revitaliza personajul și a-i permite să continue fără să fie eliminat. Cel de-al doilea poate fi colectat pentru a fi utilizat ulterior, iar primul este consumat în momentul în care este cules.



Figura 15 Fructe ce revitalizează personajul

Deoarece jocul se desfășoară într-un turn, există modalități de a comuta între diferitele niveluri ale construcției. Acestea sunt niște scări iar aspectul lor vizual este prezentat în Figura 16. La interacțiunea personajului cu un asemenea obiect, nivelul se va schimba în funcție de tipul de scară, acesta aflându-se în continuare pe aceeași poziție pe teren, dar la un alt etaj.

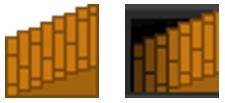


Figura 17 Scările între etajele turnului

Pentru a crea un labirint și pentru a spori dificultatea de orientare, sunt folosite niște obstacole indestructibile, astfel jucătorul fiind nevoit să găsească soluția problemei. În urma unui impact cu un perete imobil, personajul nu va interacționa cu el, ci doar se va roti dacă axa de deplasare este cea orizontală. Figura 18 prezintă aceste obiecte.

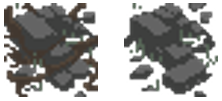


Figura 18 Pereți indestructibili

Încă de la început, jucătorul primește informația că pentru a-și îndeplini scopul și a scăpa din turn, el va trebui să găsească o comoară ce se află într-un cufăr. Interacțiunea cu acest cufăr este diferită dacă personajul deține sau nu cheia pentru acesta. Dacă acesta deține cheia, cufărul se va deschide și personajul poate colecta ce se află în interiorul acestuia. Cele două stări în care acesta se poate afla sunt prezentate în Figura 19. În caz contrar, va fi afișat un mesaj din care se deduce că acesta e închis și e nevoie de o cheie pentru a putea fi deschis.



Figura 19 Cele două stări ale cufărului

Pentru a oferi informații suplimentare jucătorului, există panouri de informare răspândite în primul nivel, aspectul lor vizibil se regăsește în Figura 20. Ciocnirea cu un asemenea indicator va demara afișarea informației ce se găsește scrisă pe acel indicator.



Figura 20 Semn de ajutor

În cazul în care personajul rămâne fără viață în urma atacului unui inamic, este considerat învins și se va întoarce din punctul de unde a început aventura, dar obiectele colectate pe drum vor rămâne în posesia acestuia. Pentru a nu pierde întreg progresul există un obiect ce salvează poziția în cazul înfrângerii. Dacă acesta este activ, culoarea lui se schimbă din roșu în verde conform Figurii 21.



Figura 21 Checkpoint[[5]](#footnote-5) inactiv și activ

În Figura 22 se regăsește ultimul obiect cu care poate interacționa personajul, ieșirea, ce inițial a fost închisă. Dacă acesta a îndeplinit toate cerințele și a colectat comoara necesară, ușa se va deschide și acesta va putea părăsi turnul. În urma acestei interacțiuni va fi afișată fereastra de final unde jucătorul este felicitat că a ajuns la sfârșitul aventurii.

A close up of a sign

Description automatically generated

Figura 22 Ieșirea din turn

## Indicatori vizuali

În afara terenului de joc sunt prezenți indicatori ce oferă informație suplimentară. Astfel, jucătorul poate să își formeze o privire de ansamblu pentru a-și crește șansele de reușită. Din Figura 23 putem observa că există 4 indicatori.

A circuit board

Description automatically generated

Figura 23 Indicatorii vizuali vizibili în lateralele ecranului

Primul indicator vizual se află în partea de sus stânga și este indicatorul pentru statisticile personajului. Prima statistică este cea ce afișează viața rămasă a personajului. De asemenea, acesta beneficiază de un sistem de avansare în nivel pe măsură ce acumulează experiență în lupta cu inamicii. De fiecare dată când avansează în nivel capacitatea de a ataca și capacitatea de a se apăra îi sunt îmbunătățite. Pentru a trece la următorul nivel de experiență, este necesar ca acesta să colecteze 100 de astfel de puncte.

Al doilea indicator, aflat în partea de jos dreapta, oferă indicații despre un inamic dorit. Acest este invizibil în mod obișnuit, dar dacă utilizatorul mută cursorul pe un inamic, în acest chenar sunt prezentate informațiile despre acesta. Diferențele dintre acest indicator și cel al personajului sunt poza inamicului pentru confirmare și cantitatea de experiență care o va dobândi personajul dacă va învinge acest inamic.

Al treilea indicator este localizat sus dreapta și oferă informații despre principalele țelurile ce trebui atinse pentru a avansa în poveste. De fiecare dată când a fost nevoie de o unealtă de înlăturat un obstacol, a apărut o linie nouă ce descrie problema.

Ultimul indicator oferă informații despre obiectele colectate în timpul jocului. Odată colectat, imaginea obiectului este trimisă spre indicator, iar în cazul fructelor ceea ce este modificat este numărul celor rămase de consumat.

În Figura 24 se află meniul de final al jocului în care utilizatorul este felicitat că a ajuns la finalul jocului și îi sunt prezentate statisticile încercării sale.

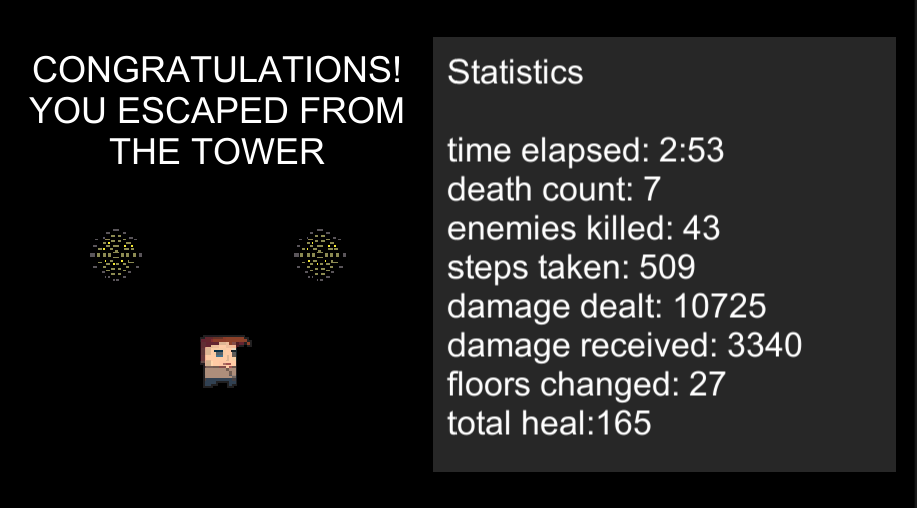
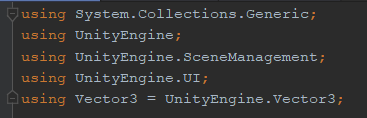


Figura 24 Meniul cu statistici de la finalul jocului

# Arhitectură și implementre

În capitolul anterior a fost prezentat aspectul vizual al proiectului, aspect ce a fost creat în editorul oferit de Unity. În acest capitol sunt oferite detalii despre implementarea în cod a jocului. Funcționalitatea este scrisă în interiorul unor scripturi ce sunt legate la obiectele de joc și asigură logica evenimentelor. Limbajul de programare folosit este C# peste care sunt adăugate biblioteci specifice Unity pentru a extinde funcționalitatea.



Biblioteci specifice Unity

## Scipturi

Pentru ca un fișier cu extensia .cs (C Sharp) să poate fi considerat un script și pentru a putea fi atașat unui obiect pentru a-i determina funcționalitatea, el trebuie să conțină în interiorul fișierului o clasă ce extinde clasa “MonoBehaviour”. Această ultimă clasă este considerată de Unity ca fiind clasa de bază de la care vor pleca toate celelalte. Conține informații de bază despre obiectele căror li se va atribui o componentă de tip script, cum ar fi numele obiectului, un indicator ce arată dacă obiectul este activ sau nu sau detalii de transformare ale acestui obiect (poziție, rotație, scalare) .

## Scene

Pentru a diferenția diferitele momente ale jocului, există o delimitare pe subdiviziuni numite scene. Aceste scene sunt alcătuite din obiecte ce pot sau nu fi create folosind niște schițe prefabricate. Când sunt create, toate obiectele apelează metoda incipientă “Start” sau “Awake” și sunt actualizate în fiecare frame[[6]](#footnote-6) al jocului în metoda “Update”.

Meniul principal este prima scenă a jocului, iar imaginea de final este ultima scenă. Acestea se pot schimba între ele folosind funcția “SceneManager.LoadScene(string sceneName, LoadSceneMode mode) ” unde modul poate fi de doua tipuri, aditiv, scena nouă se combină cu cea precedentă, sau singular, scena nouă înlocuind-o pe cea actuală.

## Loader

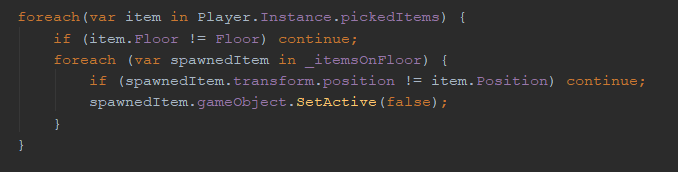
Un “loader” este acea componentă care asigură că instanțele unor obiecte au fost create. Acest script este primul care se execută odată cu începerea jocului propriu zis

Este util să existe o singură instanță a GameManager-ului și o singură instanță a Player-ului, deci în ambele cazuri este aplicat modelul de proiectare “singleton”. Prin urmare fiecare dintre aceste clase sunt instanțiate o singură dată în loader. Pentru ca obiectele astfel create să nu fie distruse atunci când se schimbă scena în modul single, în funcția de start a lor, trebuie creat un apel către funcția “DontDestroyOnLoad(GameObject object)” unde object este o referință către obiectul apelant.

## GameManager

Clasa care organizează fluxul jocului și care se ocupă de schimbarea scenelor după startul jocului este GameManager-ul. De asemenea are și rolul de container pentru variabile ce sunt necesare în alterarea fluxului actual al jocului. Un astfel de exemplu îl constituie poziția și etajul ultimului checkpoint activat. În momentul în care personajul reînvie, el se întoarce la ultimul checkpoint care poate să fie situat la alt etaj față de cel curent fapt ce necesită o urcare sau coborâre de nivel.

La schimbarea unui etaj este apelată funcția „LoadFloor(int floor)” ce conține un callback[[7]](#footnote-7) către funcția ce va reîmprospăta informațiile ce țin de etajul curent. Printre aceste informații se numără și obiectele ce pot fi colectate o singură dată pe etaj. Pentru a face acest lucru, se verifică, pe rând, dacă obiectele colectabile, de la etajul tocmai încărcat, se află în posesia jucătorului. Astfel cele colectate sunt imediat setate ca fiind inactive.



Bucla ce determină dacă obiectele de la etajul curent au fost colectate anterior

## LivingObject și MovingObject

O clasă abstractă nu are ca scop implementarea directă a întregii funcționalități, ci servește în a fi un șablon pentru clasele o ce vor extinde. Astfel ea conține metode abstracte, ce conțin doar definiția, nu și implementarea, dar poate conține și metode concrete, ce prezintă funcționalități standard și pot fi extinse la rândul lor, ale căror corp este prezent.

LivingObject este clasa abstractă ce stă la baza tuturor obiectelor de joc ce vor avea viață și vor putea ataca. De asemenea tot LivingObject, fiind o clasă de bază, va colecta toate informațiile necesare despre componentele din Unity ale obiectului, cum ar fi SpriteRenderer, componenta ce desenează imaginea obiectului, sau RigidBody2D, componentă ce asigura buna funcționare din punctul de vedere al fizicii și determină coliziunea obiectelor.

În Unity există o funcționalitate numită Layers. Acest mecanism este folosit pentru a grupa cu ușurință obiecte ale căror caracteristici sunt împărtășite. În joc, acest mecanism este folosit pentru a grupa obiectele imobile și deci pentru a opri obiectele ce se pot mișca din a-și schimba poziția dacă în urma acestei mutări ele ar intra în coliziune cu un bloc imobil.

Clasa ce extinde LivingObject și îi conferă obiectului abilitatea de a se mișca pe hartă este MovingObect, o altă clasă abstractă. Mișcarea obiectelor are la bază 4 metode.

Prima metodă este “AttemptMove” care încearcă să miște obiectul în direcția dorită. Tot în această metodă are și loc rotația obiectului pe orizontală în funcție de sensul de mers.

Pentru a verifica dacă tranziția pe noua căsuța va crea vreo coliziune se apelează metoda “Move”. În interiorul acestei metode este apelată funcția din Unity numită “Physics2D.Linecast (Vector2 start, Vector2 end, int layerMask)“. Linecast va trasa o linie imaginară cu capetele în start și end, adică poziția în care se află obiectul și poziția în care obiectul dorește să ajungă, iar layer-ul este grupul de coliziune în care se află obiectul ce inițiază mișcarea. Dacă nu a fost detectat niciun obiect atunci se pornește corutina[[8]](#footnote-8) ce va realiza mișcarea propriu zisă, cu câte o unitate pe frame, creându-se astfel senzația de mișcare netedă, fluidă.

Pentru a crea această mișcare continuă, obiectul trebuie să se miște în fiecare frame o mică porțiunea din mișcarea totală. Astfel la fiecare iterație se verifică dacă distanța totală a fost parcursă, iar în caz contrar obiectul este mișcat cu o unitate și distanța rămasă este recalculată.

Dacă mișcarea în direcția dorită a fost oprită de un obiect imobil, atunci este apelată metoda “OnCantMove”, ea fiind abstractă, deoarece fiecare obiect mobil poate decide cum tratează cazul în care acesta nu se poate mișca.

## Player

Personajul este cu siguranță obiectul cel mai prezent pe parcursul întregului joc, ceea ce face ca scriptul ce îi determină acțiunile să fie cel mai complex. Clasa principală din acest script este numită “Player” și extinde clasa detaliată mai sus, MovingObject.

Conține 2 clase interne specifice acestui tip de obiect, „ExtendedParameters”, ce conține toți parametrii specifici (nivel, armură, experiența acumulată etc) și „Quest”, ce stochează informații despre obiectivele existente în joc.

Toate datele statistice ce sunt afișate la finalul jocului sunt înregistrate fiecare într-un câmp private și sunt transmise odată cu terminarea jocului. De asemenea, este ținută câte o referință către toate textele de tip statistică pentru a le putea actualiza pe măsură ce acestea sunt modificate. De aceeași abordare beneficiază sistemul de obiective, la fiecare pas sunt verificate dacă sunt descoperite, dar nu și îndeplinite, iar dacă condițiile sunt validate acestea sunt afișate.

Deoarece Player extinde MovingObject mișcarea personajului are loc în mai multe frame-uri astfel apare nevoia unui flag[[9]](#footnote-9) care determină când acestuia îi este permis să se deplaseze din nou. Astfel, dacă acesta se află în mișcare, nicio altă acțiune nu îi este permisă. Modul în care se comandă deplasarea depinde de dispozitivul pe care este rulat jocul. Astfel, pe PC sunt folosite cele două combinații uzuale de taste ”WASD” sau săgețile direcționale, iar pe Android este folosit swipe-ul spre direcția dorită pentru deplasare. Deplasarea pe diagonală este evitată, iar dacă două taste pe direcții diferite sunt apăsate în același timp, prioritate are axa orizontală.

Funcția OnCantMove, este extinsă în clasa player pentru a descrie funcționalitatea personajului în urma contactului cu un obiect imobil. Dacă a avut loc o coliziune cu un inamic, interacțiunea este dependentă de un obiect al personajului și anume sabia. În lipsă sabiei, quest-ul ce presupune obținerea sabiei va fi setat ca descoperit dar necompletat. Dacă sabia este prezentă, inamicul apelează funcția sa de a primi daune în funcție de capacitatea de atac a inamicului. Dacă inamicul este învins, acesta nu ripostează, dar întoarce valoarea sa în puncte de experiență. Personajul asimilează experiența și în cazul în care acesta a acumulat 100 de puncte, va crește în nivel și își va îmbunătăți capacitatea de atac și de apărare conform variabilelor constant: “AttackPointsPerLevel” și “ArmorPerLevel”. Rezultatul este asemănător în cazul pereților destructibili, singurele diferențe fiind date de experiența oferită, lipsa ripostei și unealta necesară pentru a le distruge. Dacă a avut loc o coliziune cu un semn de informație, acestuia îi va fi accesat câmpul ”info” pentru a citi informația setată. Coliziunea dintre cufăr și personaj poate determina 3 acțiuni, obținerea unui quest, deschiderea cufărului în prezența cheii sau obținerea comorii dacă acesta este deschis. În urma oricărei dintre aceste coliziuni se va porni o nouă corutină ce obligă personajul să aștepte o perioadă scurtă de 0,2 secunde pentru a simula o mișcare reușită, și să se aștepte aceeași perioadă de timp.

Mai sus a fost detaliată coliziunea cu obiecte imobile, însă nu toate obiectele aparțin acestei categorii. Interacțiunea cu celelalte obiecte este de doua tipuri: de suprapunere sau de colectare. Sub tipul suprapunerii există 3 obiecte, exit, checkpoint și stairs. Contactul dintre exit și personaj este diferit doar dacă acesta deține comoara, caz în care jocul s-a terminat și este afișat ecranul de final, în caz contrat se afișează un mesaj de informare și căsuța unde se află exit este considerată a fi vacantă, așa ca personajul se va deplasa. În cazul suprapunerii cu un checkpoint, acesta din urmă va deveni activ schimbându-și sprite[[10]](#footnote-10)-ul și GameManager îi va salva locația (etaj și poziție) pentru a întoarce personajul pe aceeași poziție în caz ca este învins. Contactul cu scările determină un apel spre GameManager pentru a schimba etajul curent. În urma schimbării jucătorul se va afla tot pe scări, dar la un alt etaj ceea ce ar determina o buclă infinită de schimb între două etaje. Acest lucru este evitat folosind un flag ce este setat la contactul cu scările și resetat când personajul a părăsit scările. Un detaliu legat de consistență este dat de rotirea personajului în direcția urcării scărilor.

Obiectele ce pot fi colectate sunt sabia, lopata, târnăcopul și fructele. După ce acestea sunt colectate, ele nu vor fi recreate odată cu schimbarea nivelului. Această funcționalitate este oferită de scriptul și clasa UniqueItem. Astfel când obiectul este obținut este returnată o instanță a clasei ItemLocation ce deține toate informațiile de localizare ale obiectului în cauză, iar el nu va mai fi creat conform algoritmului specificat în subcapitolul 4.4.

Primele trei obiecte unice sunt uneltele care în urma colectării vor fi bifate ca fiind deținute în câmpurile destinate acestui lucru și se va oferii un feedback vizual. Fructele sunt și ele de două tipuri în funcție de momentul în care acestea recapătă viața pierdută de utilizator, ele fiind singura resursă limitată, dar în același timp jocul poate fi dus la capăt fără colectarea lor, spre deosebire de unelte.

## Enemy

O altă clasă ce extinde LivingObject este Enemy. Deoarece inamicii sunt principalul tip de obstacole, coliziunea cu ei este inevitabilă, coliziune în urma căreia personajul va pierde viață. Pentru ca utilizatorul să poată să-și creeze o strategie de a avansa în experiență în timp ce păstrează numărul înfrângerilor cât mai mic, este adăugat un indicator vizual ce oferă informații despre inamici. Aici sunt folosite funcțiile eveniment ”OnMouseOver” și ”OnMouseExit”, ce sunt disponibile și pe mobil, pentru a face indicatorul să dispară sau să apară. Pentru a nu dezactiva și activa la loc acest indicator format din o colecție de obiecte, o metoda mai eficientă este aceea de a crea un chenar de acoperire de aceeași dimensiune cu indicatorul și de aceeași culoare cu fundalul. Astfel în loc de a schimba starea unei grupări de obiecte, este schimbată starea unui singur obiect, iar gruparea de obiecte este doar actualizată cu informațiile necesare.

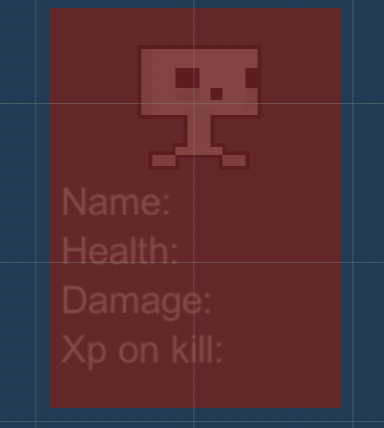


Figura 25 Chenar de acoperire

În Figura 25 este exemplificat modelul de proiectare al acestui indicator, dar chenarul nu are opacitatea maximă și nici culoarea fundalului (în joc este negru) din motive exemplificative.

## HealthBar

Ceea ce diferențiază un LivingObject de celelalte este prezența parametrului Health. Odată ce acest parametru a scăzut sub 0, înseamnă că obiectul a fost învins și va dispărea, în cazul inamicilor sau va reveni la checkpoint, în cazul personajului. Pentru a evidenția mai bine acest aspect, fiecare obiect are deasupra sprite-ului o bară ce indică viața rămasă, conform Figurii 26.

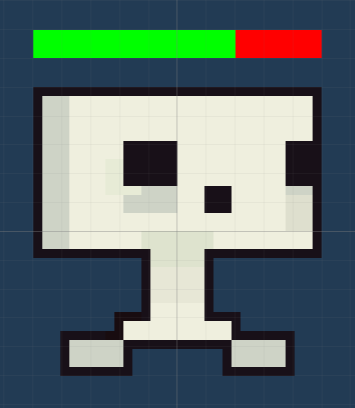


Figura 26 Bara de viață prezentă deasupra sprite-ului

Această bară este rezultatul suprapunerii a două bare (verde peste roșu). Bara verde însă este în interiorul unui container ce nu este redat vizual și are centrul în extremitatea stângă a barei roșii. Mecanismul de scalare funcționează astfel încât obiectul ce urmează a suferi această transformare își va modifica dimensiunea, dar își va păstra centrul. Din acest motiv bara verde este conținută ca și copil într-un container. O modificare asupra containerului determină modificarea relativă asupra copiilor. Se stabilește intervalul de scalare, de la 0 la dimensiunea dorită a barei. Acest interval reprezintă viața rămasă a obiectului, deci la 100% din viață cele două se suprapun, iar la 0% bara verde este invizibilă. Acest mecanism este explicat în Figura 27.

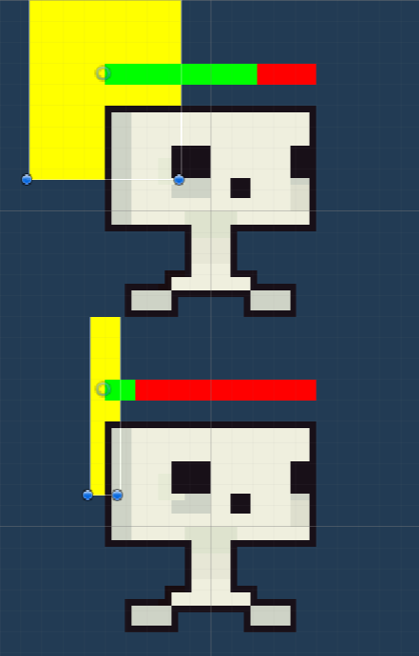


Figura 27 Explicația vizuală din spatele mecanismului de bară de viață

## Scripturi pentru camera

În partea de început a jocului și la finalul acestuia, se găsesc scene de prezentare în care utilizatorul nu are nicio influență asupra ecranului, singura acțiune posibilă este de a schimba ecranul și a trece mai departe. Astfel în aceste scene nu este fondată folosirea GameManager-ului pentru a modera acțiunile. În acest caz se creează un script specific pentru scena respectivă și se atribuie camerei ce redă jocul.

Scriptul Tutorial este cel ce afișează sprite-urile și informații despre acestea pe ecran. Această scenă este formată din 3 obiecte ce conțin doar un sprite pentru a desena imaginea lor și un text, iar conținutul lor este schimbat în funcție de valoarea unei variabile ce este incrementată la apăsarea oricărui buton.

## Alte scripturi

Pentru alte obiecte de joc ale căror rol este episodic, sunt create scripturi ce îndeplinesc o funcție simplă și setează parametrii necesari în constructor. Această abordare este preferată în defavoarea uneia în care GameManager-ul al face acest lucru, deoarece astfel se păstrează încapsularea, determină codul să fie mai coerent și ușor de menținut și extins.

## Graphic User Interface (GUI) în Unity

Pentru a crea indicatorii vizuali, Unity pune la dispoziție un obiect de tipul ”Canvas” ce joacă rolul de ”pânză de desen”. Utilizând această metodă, se pot ancora obiecte în spațiul ecranului ceea ce determină ca aspectul jocului să rămână același pe orice dispozitiv cu aceeași perspectivă (16:9). În Figura 28 este exemplificat cum sunt ancorate chenarele cu informații.

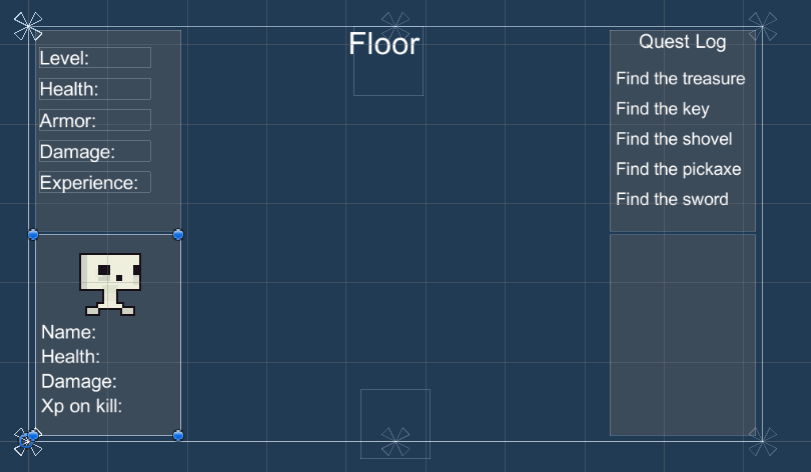


Figura 28 Indicatorii vizuali ancorați în spațiul ecranului

# Evaluare si Testare

Acest capitol trebuie să răspundă, în principiu, la **2 întrebări** și să se încheie cu **o discuție** a rezultatelor obținute. Cele doua întrebări la care trebuie sa se răspundă sunt:

1) **Merge corect**? (Conform specificațiilor extrase în capitolul 2);

Evaluarea dacă merge corect se face pe baza cerințelor identificate în capitolele anterioare.

2) Cât de bine merge / cum se compară cu soluțiile existente? (pe baza unor metrici clare).

Evaluarea cât de bine merge trebuie să fie bazată pe procente, timpi, cantitate, numere, **comparativ cu soluțiile prezentate în capitolul 3**. Poate fi vorba de performanță, overhead, resurse consumate, scalabilitate

În realizarea discuției, se vor utiliza tabele cu procente, rezultate numerice și grafice. În mod obișnuit, aici se fac comparații și teste comparative cu alte proiecte similare (dacă există) și se extrag puncte tari și puncte slabe. Se ține cont de avantajele menționate și se demonstrează viabilitatea abordării / aplicației, de dorit prin comparație cu alte abordări (dacă acest lucru este posibil). Cuvântul cheie la evaluare este „metrică“: trebuie să aveți noțiuni măsurabile și cuantificabile. În cadrul procesului de evaluare, explicați datele, tabelele și graficele pe care le prezentați și insistați pe relevanța lor, în următorul stil: „este de preferat ... deoarece …“; explicați cititorului nu doar datele ci și semnificația lor și cum sunt acestea interpretate. Din această interpretare trebuie să rezulte poziționarea proiectului vostru printre alternativele existente, precum și cum poate fi acesta îmbunătățit în continuare.

# Concluzii

În acest capitol este sumarizat întreg proiectul, de la obiective, la implementare, si la relevanta rezultatelor obținute. În finalul capitolului poate exista o subsecțiune de „Dezvoltări ulterioare“.

joc scurt pentru a studia piata cu un asemenea produs, in caz de succes poate fi creat mai mult content.

# Bibliografie

|  |  |
| --- | --- |
| [1] | J. Silva-Martinez, "ELEN-325. Introduction to Electronic Circuits: A Design Approach," 2008. [Online]. Available: http://www.ece.tamu.edu/~spalermo/ecen325/Section%20III.pdf. |
| [2] | H. Baali, H. Djelouat, A. Amira and F. Bensaali, "Empowering Technology Enabled Care Using IoT and Smart Devices: A Review," *IEEE Sensors Journal,* vol. 18, no. 5, pp. 1790-1809, 2018. |
| [3] | A. Haroon, S. Akram, M. A. Shah and A. Wahid, "E-Lithe: A Lightweight Secure DTLS for IoT," in *IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Toronto, 2017. |
| [4] | A. K. Jain and R. C. Dubes., Algorithms for Clustering Data, Upper Saddle River: Prentice-Hall, Inc., 1988. |
| [5] | Unity Documentation, 2019. Unity Manual, 2D, Physics Reference 2D, Rigidbody 2D https://docs.unity3d.com/Manual/class-Rigidbody2D.html |
| [6] | Unity Documentation, 2019. Unity Manual, 2D, Sprites https://docs.unity3d.com/Manual/Sprites.html |

1. https://braceyourselfgames.com/crypt-of-the-necrodancer/ [↑](#footnote-ref-1)
2. https://fluffy-lotus.itch.io/magirune [↑](#footnote-ref-2)
3. https://spidi.itch.io/i-am-overburdened [↑](#footnote-ref-3)
4. Tutorial; ansamblu de indicații ce deservește la familiarizarea cu anumite concepte noi [↑](#footnote-ref-4)
5. Checkpoint; punct în care este salvat progresul într-un joc [↑](#footnote-ref-5)
6. frame; moment de timp în care imaginea este “desenată” pe ecran [↑](#footnote-ref-6)
7. callback; apel către o altă funcție atunci când anumite condiții sunt îndeplinite [↑](#footnote-ref-7)
8. corutină; funcție scrisă într-un limbaj de programare ce funcționează într-un regim particular, la care poate apela o altă funcție [↑](#footnote-ref-8)
9. flag; o valoare folosită pentru a stabili fluxul de urmat al unei funcții [↑](#footnote-ref-9)
10. sprite; imaginea oferită unui obiect [↑](#footnote-ref-10)