Harap-Alb

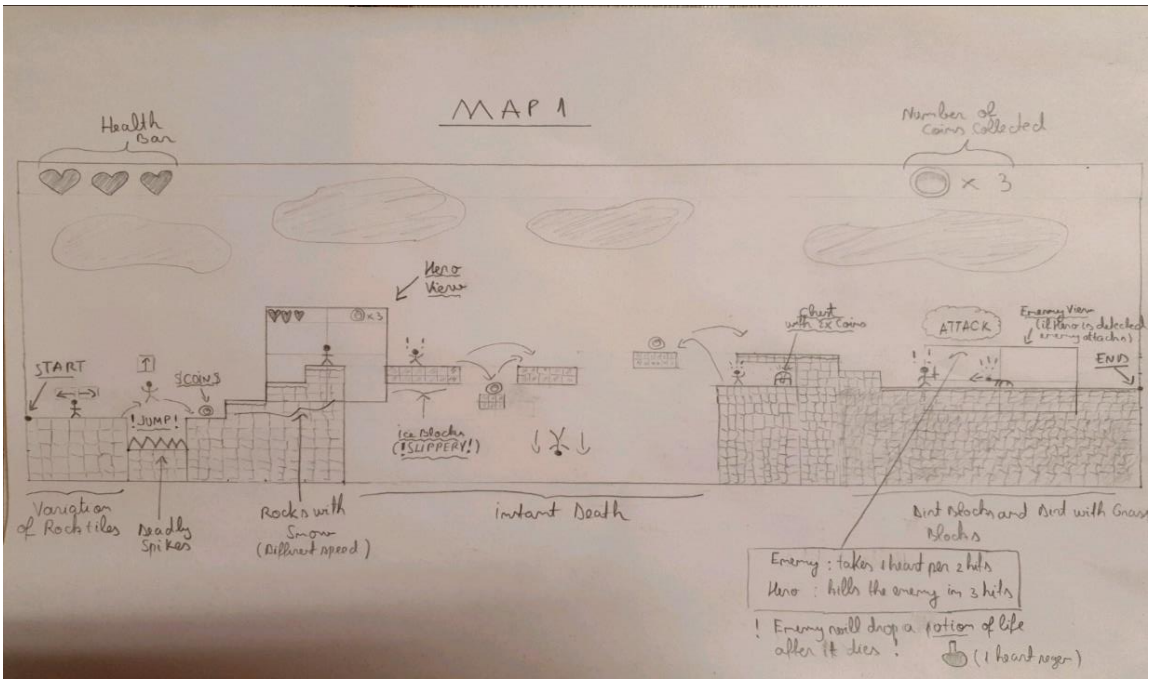
Gherasim Dragos-George

1207A

- **Anexa 1**

1. *Proiectarea contextului*

Jocul “Harap-Alb” este un action-adventure, single-player game, avand un stil minimalist, Pixel Art, iar perspectiva fiind Side View(vedere laterala). Povestea jocului urmareste aventurile tanarului Harap-Alb, intr-o lume imaginara, plina de magie si creaturi fantastice. Protagonistul nostru pleaca intr-o calatorie pentru a dobandi puture si intelepciune, in speranta de a castiga mana fiicei imparatului. In timpul aventurii sale, Harap-Alb se infrunta cu numeroase obstacole si este pus la incercare de unele personaje negative. Acesta are la indemana o sabie vrajita, foarte puternica, primita de la tatal lui, multa ambitie si dorinta de afirmare. Totusi, are nevoie de ajutorul tau pentru a trece peste toate hopurile ce ii ies in cale, esti vrednic sa iei parte la o asemenea experienta ?



2. *Proiectarea sistemului*

La inceperea hartii, se poate observa ca in coltul din stanga sus, se afla Health Bar-ul eroului (format din 3 inimi), iar in coltul din dreapta sus, numarul de monede colectate. Caracterul controlat de player se spawn-eaza la inceputul hartii si are abilitatile de a se misca stanga-dreapta (left-right arrow), poate sa sara peste obstacolele intalnite (up arrow) si poate ataca (space), pentru a-si invinge inamicii. Hartile au atmosfere diferite, in functie de drumul parcus de Harap-Alb si sunt formate din tiles-uri avand proprietati diferite. De exemplu, interactiunea eroului cu spikes-urile, duce la moartea instanta a acestuia, blocurile de gheata sau cele din pietra cu zapada, sunt mai alunecoase decat restul(forta de frecare pe axa x este mai mica decat la blocurile de pamant, de piatra). Aceasta lume magica respecta si o lege a fizicii, fundamentala lumii reale, in urma unei sarituri, caracterul este atras de forta gravitationala catre tiles-uri. Daca aceste reuseste sa cada de pe harta, soarta lui este pecetluita. Entitatile au propriul lor view, iar enamicii sunt activati sa atace daca eroul intra in raza lor de viziune. In functie de nivelul la care se afla jucatorul, obstacolele si inamicii pot avea o dificultate diferita (adversarii pot muri mai greu sau mai usor, pot lovi mai puternic sau nu). Dupa ce mor, inamicii vor lasa o potiune de viata, pe care Harap-Alb o va bea pentru a-si racapata viata pierduta. Pe harti vor exista monede care pot fi colectate de player pentru a realiza un scor cat mai mare, in functie si de timpul pe care acesta il realizeaza pentru a termina jocul. Vor exista unele locuri in care se vor ascunde niste cufere in care se vor gasi cate 2 monede. Pentru a trece la urmatorul nivel, toate entitatile inamice trebuiesc invise, iar caracterul trebuie sa ajunga la finalul hartii.

3. *Proiectarea continutului*

Harap-Alb este personajul nostru principal, acesta este infatisat ca un cavaler, avand o armura si o sabie mistica, primite de la tatal sau. Primul sau inamic va fi un soarece malefic care vrea sa-i puna bete in roate, in incercarea tanarului viteaza de a trece la nivelul urmator (momentan jocul este in perioda de dezvolatare, mai multe entitati vor aparea pe parcusul realizarii urmatoarelor 2 nivele) .

Pentru realizarea animatiilor entitatilor (idle, move, jump, attack, die), s-a folosit urmatoarele Sprite-Sheets-uri:

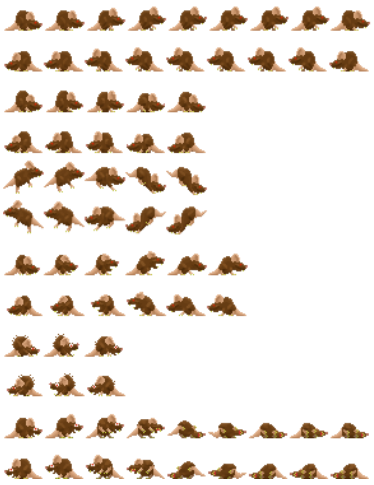
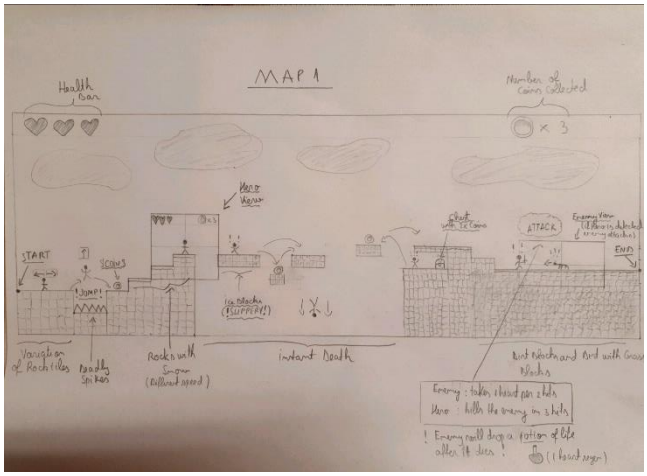


Tabla de joc este alcatuita din Tiles-uri pe un background specific atmosferei hartii. Pe harta se vor gasii obiecte cu care jucatorul va putea interactiona (cufere, monede si licoarea lasata dupa decesul unui inamic). Pentru interfata In-Game se va folosi o textura pentru inimile din Health Bar-ul eroului si prima imagine a monezii, pentru status-ul monedelor colectate. Mai jos se va regasi o imagine de ansamblu al texturilor folosite pentru crearea primului nivel:

4. *Proiectarea nivelurilor*

Primul nivel, asa cum se observa si din imaganiea de mai sus, de la sectiunea “Proiectarea continutului”, reprezinta drumul pe care Harap-Alb il parcurge dupa ce a plecat de la castelul tatalui sau, in cautarea intelepciunii si a iubirii adevarate. Pentru a ajunge la fata de imparat, acesta trebuie sa treaca mai intai pe o poteca anevoiasa la baza unui munte, la finalul careia il va astepta prima provocare “adevarata”, lupta cu un inamic.



Dupa ce trece de primul nivel, Harap-Alb ajunge in padurea blestamata, unde provocarile vor avea o dificultate sporita, fata de nivelul precedent. Aici, eroul nostru se va lupta cu mai multi inamici, pentru a capata experienta necesara, pentru a-si indeplini misiunea. In ultimul nivel, inainte de a ii cere mana fetei de imparat, Harap-Alb trebuie sa se dueleze cu un ultim inamic, care este si cel mai puternic de pana acum. Invingerea personajului negativ, va fi incununata cu finalul jocului, care il surprinde pe Harap-Alb aluturi de fiica de imparat.

5. *Proiectarea interfetei cu utilizatorul*

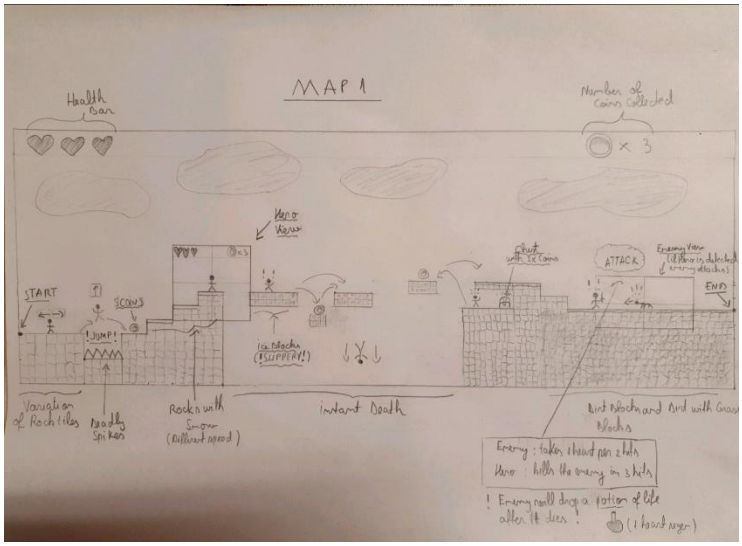
La deschiderea jocului propriu-zis, jucatorul va interactiona pentru prima data cu scena de intro (Logo Scene), in care se vor afisa pe un background sugestiv, numele jocului, care, timp de 5 secunde, va cobori catre mijlocul ecranului, iar apoi un mesaj de atentionare pentru player va aparea. Cand cele 5 secunde s-au scurs, jucatorul are posibilitatea de a trece la meniul principal. Mai jos o sa atasez cateva concepte pentru fiecare scena, pe care le-am folosit in jocul pentru C++(unele optiuni obligatorii nu se

regasesc, dar vor fi implementate, ex: in meniul principal posibilitatea de modifica setarile jocului, etc).

In meniu principal, jucatorul va avea mai multe optiune pe care sa le aleaga:

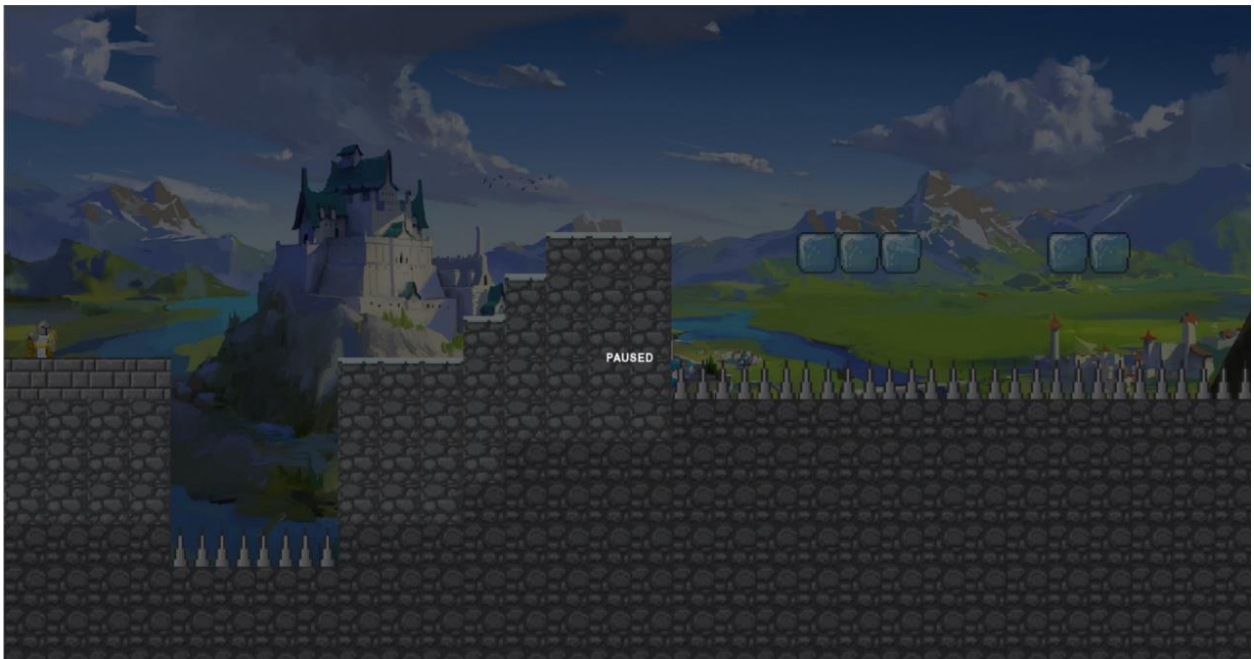
- *Resume* = reporneste nivelul activ, in caz ca s-a parasit nivelul actual si se doreste revenirea la el;
- *New Game* = porneste un joc nou, de la primul nivel, indifferent de nivelul la care s-a ajuns la ultimul run;
- *Load Game* = reporneste jocul de la ultimul nivel salvat;
- *Settings* = trimiterea jucatorului catre scena de setari, unde acesta va putea sa-si schimbe butoanele pentru controlul eroului;
- *Exit* = parasirea jocului.

Scena de game este reprezentata de interfata din cadrul jocului, cu nivelele configurate, un health bar in coltul din stanga sus si o reprezentarea a numarului de monede colectate.

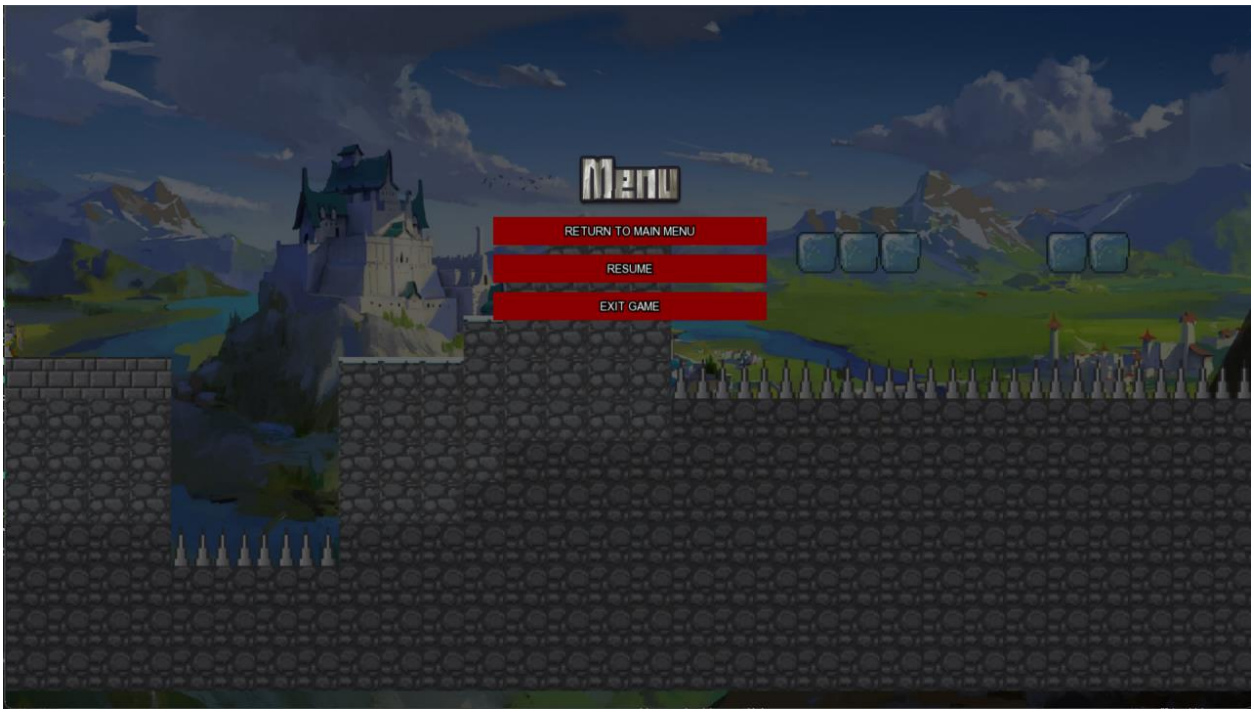


In scena de game, jucatorul poate comuta cu alte doua scene:

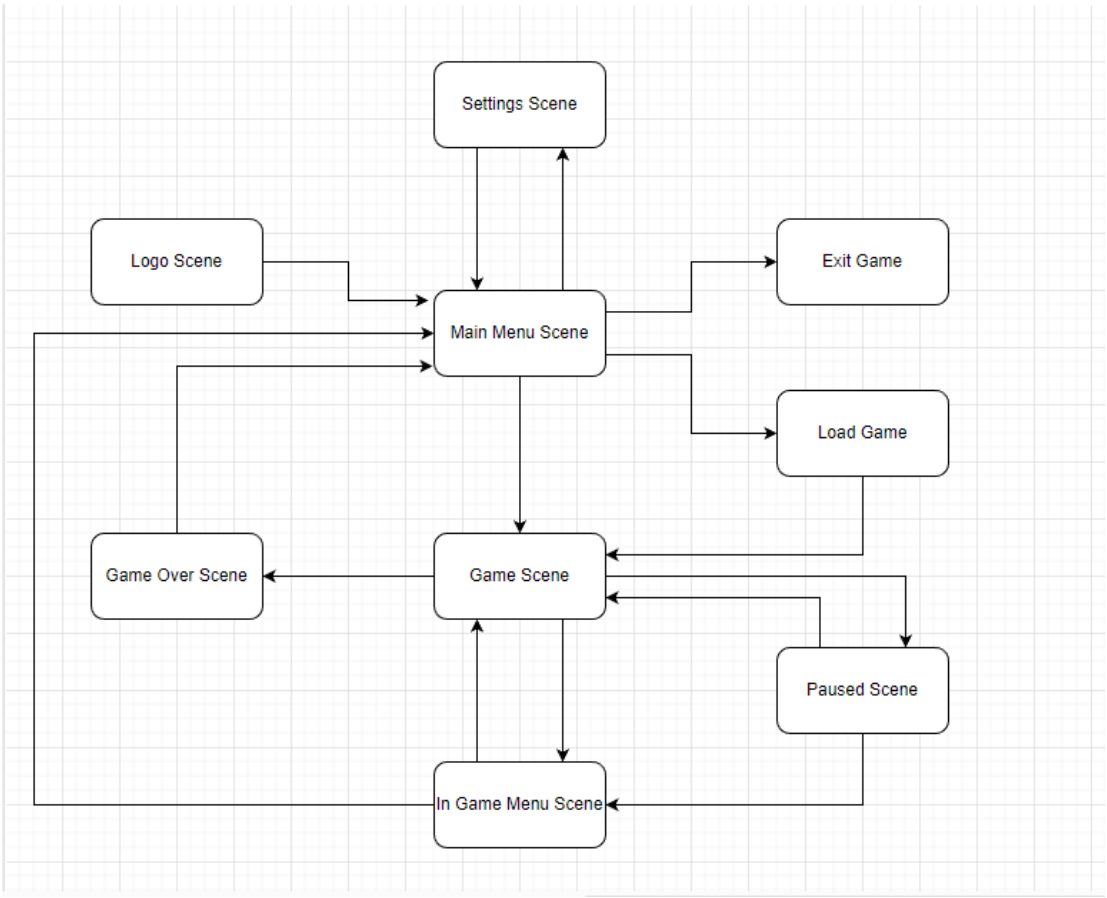
- *Scena de Pauza* = in care scena de joc este “inghetata”(scena de game nu upload-eaza);



- *Scena de In Game Menu* = in care scena de game nu este “inghetata” (scena de game se upload-eaza). Se aseamana cu meniul principal, doar ca aici va aparea si posibilitatea de salvare a nivelului actual;

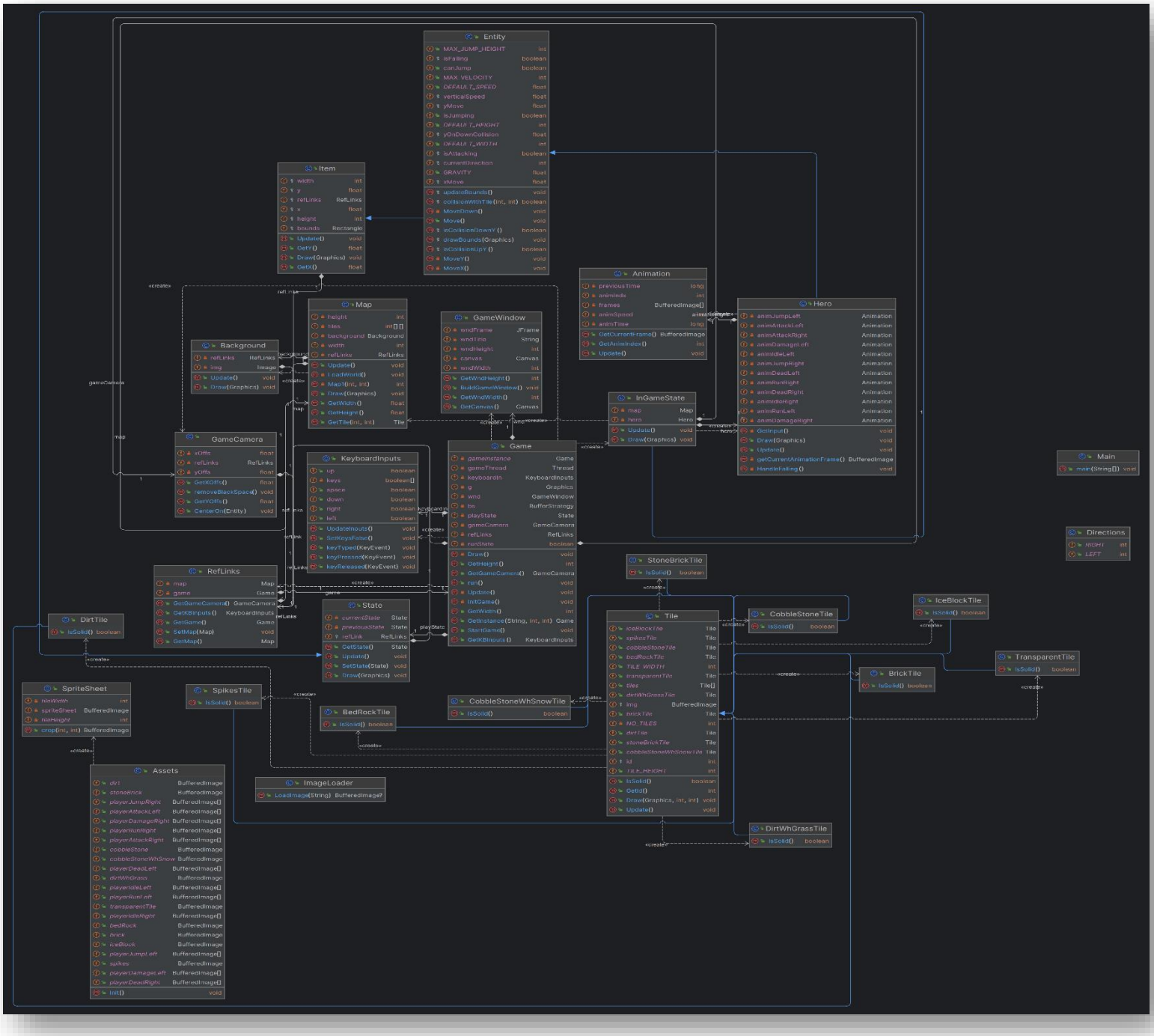


- *Scena de Game Over* = dupa ce eroul isi face animatia de moarte, apare scena de Game Over, iar jucatorul este redirectionat catre meniul principal.


Settings Scene
Logo Scene
Exit Game
Main Menu Scene
Load Game
Game Over Scene
Game Scene
Paused Scene
In Game Menu Scene

- **Anexa 2**

1. *Arhitectural Design Document (UML)*



2. Descriere Clase

Entities Package

- *Item* : Implementeaza notiunea abstracta de entitate activa din joc, "element cu care se poate interactiona: monstru, turn etc.

- *Entity* : Defineste notiunea abstracta de caracter/individ/fiinta din joc. Notiunea este definita doar de viata, viteza de deplasare si distanta cu care trebuie sa se miste/deplaseze in urma calculelor.

- *Hero* : Implementeaza notiunea de erou/player (caracterul controlat de jucator)

GameWindow Package

- *GameWindow* : Implementeaza notiunea de fereastra a jocului. Membrul wndFrame este un obiect de tip JFrame care va avea utilitatea unei ferestre grafice si totodata si cea a unui container (toate elementele grafice vor fi continute de fereastra).

Graphics Package

- *Animation* : Implementeaza o animație care este formată dintr-un set de cadre (frames) de imagine, cu o viteză de animare specificată de utilizator. Conține câteva variabile de stare, cum ar fi "frames" pentru a stoca imaginile de animație, "animSpeed" pentru a stoca viteza de animație și "animIndx" pentru a ține evidența poziției curente în cadrul animației.

- *Assets* : Clasa incarca fiecare element grafic necesar jocului. Game assets include tot ce este folosit intr-un joc: imagini, sunete, harti etc.

- *GameCamera* : Implementeaza notiunea de cameră de joc care urmărește și afișează o anumită entitate pe ecran. Ea este responsabilă de setarea și actualizarea poziției camerei de joc, astfel încât entitatea selectată să fie mereu centrată pe ecran.

- *ImageLoader* : Clasa ce contine o metoda statica pentru incarcarea unei imagini in memorie.

- *SpriteSheet* : Clasa retine o referinta catre o imagine formata din dale (sprite sheet). Metoda crop() returneaza o dala de dimensiuni fixe (o subimagine) din sprite sheet de la adresa (x * latimeDala, y * inaltimeDala).

Inputs Package

- *KeyboardInputs* : Gestioneaza intrarea (input-ul) de tastatura. Clasa citeste daca au fost apasata o tasta, stabiliteste ce tasta a fost actionata si seteaza corespunzator un flag. In program trebuie sa se tina cont de flagul aferent tastei de interes. Daca flagul respectiv este true inseamna ca tasta respectiva a fost apasata si false nu a fost apasata.

Main Package

- *Game* : Clasa principala a intregului proiect. Implementeaza Game - Loop (Update -> Draw). Interfata este utilizata pentru a crea un nou fir de executie avand ca argument clasa Game. Clasa Game trebuie sa aiba definita metoda "public void run()", metoda ce va fi apelata in noul thread(fir de executie). Mai multe explicatii veti primi la curs.

- *RefLinks* : Clasa ce retine o serie de referinte ale unor elemente pentru a fi usor accesibile. Altfel ar trebui ca functiile respective sa aiba o serie intreaga de parametri si ar ingreuna programarea.

- *Main* : "Main" este folosită pentru a porni jocul și a inițializa toate resursele necesare pentru jocul în sine.

Maps Package

- *Background* : Implementeaza notiunea de fereastra a jocului.

- *Map* : Implementeaza notiunea de harta a jocului.

States Package

- *State* : Implementeaza notiunea abstracta de stare a jocului/programului. Un joc odata ce este lansat in executie nu trebuie "sa arunce jucatorul direct in lupta", este nevoie de un meniu care sa contine optiuni: New Game, Load Game, Settings, About etc. Toate aceste optiuni nu sunt altceva decat stari ale programului (jocului) ce trebuiesc incarcate si afisate functie de starea curenta.

- *InGameState* : Implementeaza/controleaza jocul.

Tiles Package

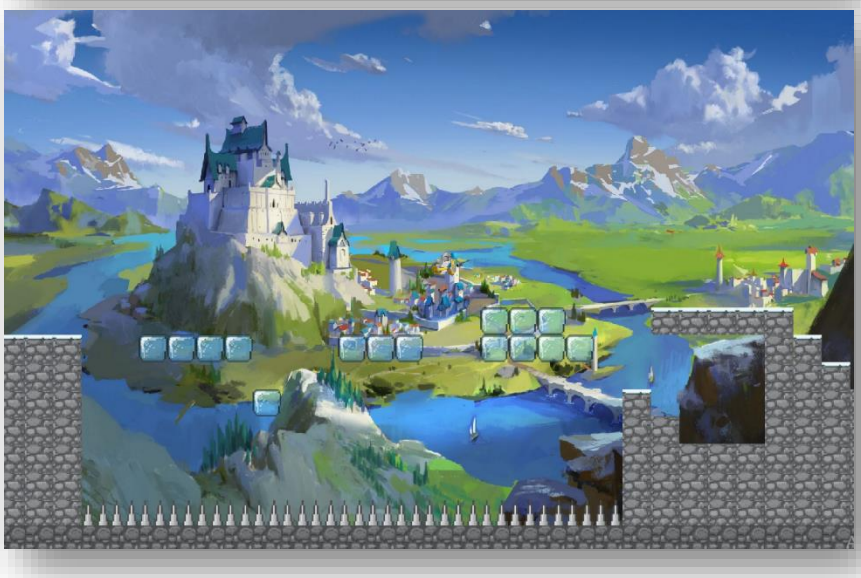
- *Tile* : Retine toate dalele intr-un vector si ofera posibilitatea regasirii dupa un id.

- *BedRockTile, BrickTile, CobbleStoneTile, CobbleStoneWhSnowTile, DirtTile, DirtWhGrassTile, IceBlockTile, SpikesTile, StoneBrickTile, TransparentTile* : Abstractizeaza notiunea de dala pentru fiecare tip specific.

Utils Package

- *Directions* : Implementeaza directiile pe care le poate lua eroul (Stanga, Dreapta).

3. *Secvente joc*

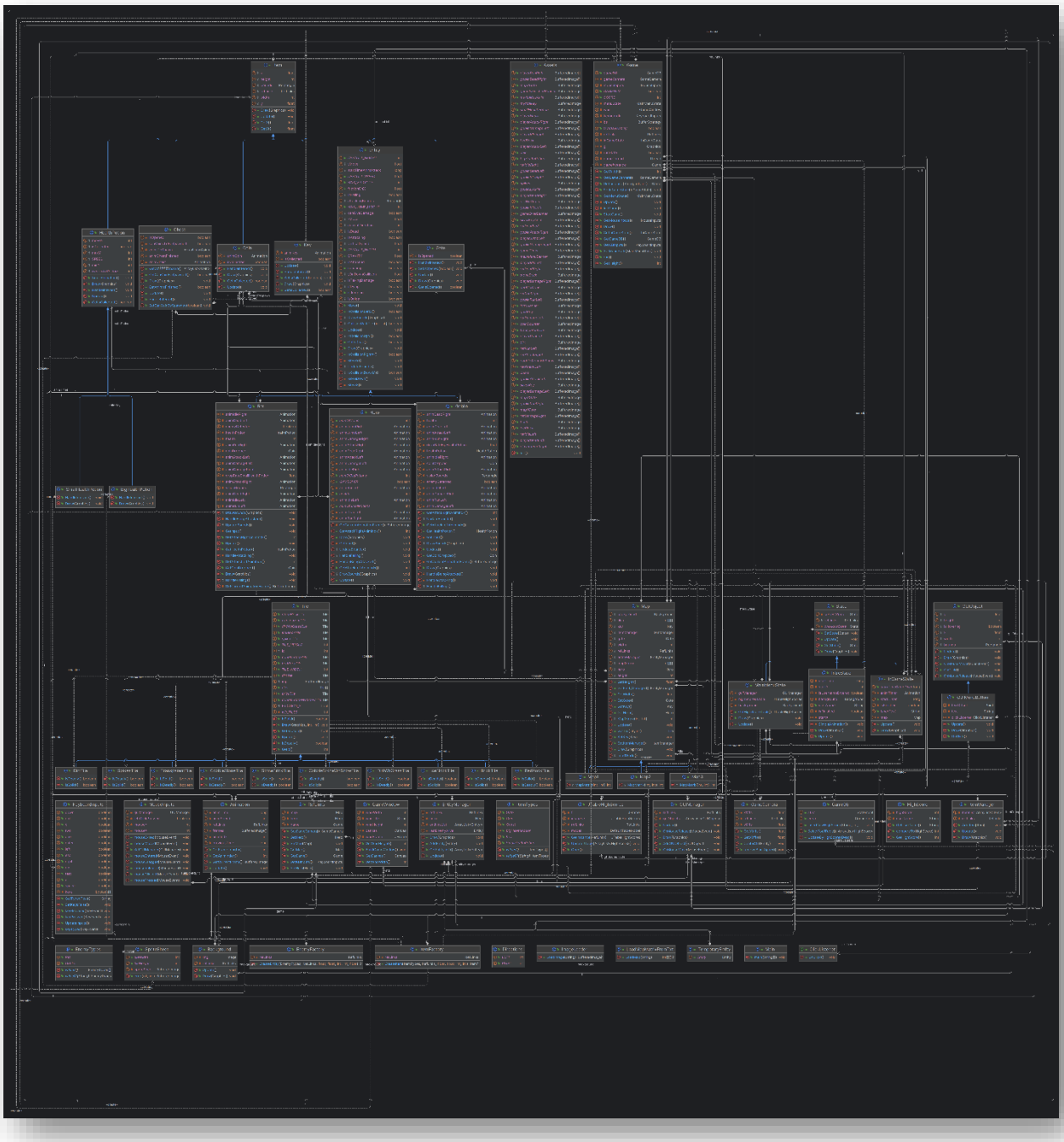


4. *Design Pattern*

- *Singleton* : Am folosit pentru clasa Game deoarece doar o instanta a jocului trebuie sa existe la un moment dat.

• Anexa 3

1. *Arhitectural Design Document (UML) (Actualizare)*



2. *Descriere Clase (Actualizare)*

Entities Package

- *HealthPotion* : Clasa abstractă care servește drept clasă de bază pentru diferite tipuri de poțiuni de vindecare în jocul "Harap Alb". Această clasă gestionează aspecte comune ale poțiunilor de vindecare și oferă funcționalitate de bază pentru interacțiunea cu acestea;

- *BigHealthPotion* : Este o subclasă a clasei HealthPotion și reprezintă o entitate a jocului care reprezintă o poțiune mare de vindecare. Această clasă se ocupă de comportamentul și aspectul grafic al acestei poțiuni în joc;

- *SmallHealthPotion* : Este o subclasă a clasei HealthPotion și reprezintă o entitate a jocului care reprezintă o poțiune mică de vindecare;

- *Chest* : Este o clasă care reprezintă un cufar în jocul "Harap Alb". Această clasă se ocupă de comportamentul și aspectul grafic al cufarului;

- *Coin* : Este o clasă care reprezintă o moneda în jocul "Harap Alb". Această clasă se ocupă de comportamentul și aspectul grafic al unei monede;

- *Gate* : Este o clasă care reprezintă o poarta în jocul "Harap Alb". Această clasă se ocupă de comportamentul și aspectul grafic al unei porti;

- *Key* : Este o clasă care reprezintă o cheie în jocul "Harap Alb". Această clasă se ocupă de comportamentul și aspectul grafic al unei chei;

- *Rat* : Este o clasă care reprezintă unul dintre personajele negative în jocul "Harap Alb". Această clasă se ocupă de comportamentul și aspectul grafic al sobolanului;

- *Goblin* : Este o clasă care reprezintă unul dintre personajele negative în jocul "Harap Alb". Această clasă se ocupă de comportamentul și aspectul grafic al goblinului;

- *EntityManager* : Este responsabilă de gestionarea entităților (erou și inamici) din joc;

- *ItemManager* : Este responsabilă de gestionarea itemelor de pe harta (coins, gate, potions).

Graphics GUI

- *ClickListener* : Reprezintă un contract pentru un ascultător de evenimente de click;

- *GUIManager* : Este responsabilă de gestionarea și afișarea obiectelor interfeței grafice (GUI) în joc;

- *GUIMenuButton* : Extinde clasa GUIObject și reprezintă un obiect de tip buton pentru meniul interfeței grafice;

- *GUIObject* : Este o clasă abstractă care servește ca bază pentru obiectele interfeței grafice.

Inputs Package

- *MouseInputs* : Este responsabilă de este responsabilă de gestionarea evenimentelor legate de mouse gestionarea evenimentelor legate de mouse.

Maps Package

- *Map* : Este o clasă abstractă care servește ca bază pentru crearea și gestionarea hărților din jocul "Harap Alb";

- *Map1 , Map2, Map3* : Este o clasă abstractă care servește ca bază pentru crearea și gestionarea hărților din jocul "Harap Alb".

SQL Package

- *GameDB* : Este responsabilă de gestionarea bazei de date SQLite pentru stocarea și manipularea scorurilor în jocul "Harap Alb";

- *HighScore* : Reprezintă un obiect care stochează informații despre scor într-un joc (userName, highScore);

- *JTableHighScores* : Este responsabilă de afișarea și actualizarea unui tabel cu scoruri dintr-o baza de date, într-o interfață grafică utilizând componentele Swing.

States Package

- *IntroState* : Este o subclasă a clasei "State" și reprezintă starea de introducere a jocului în care utilizatorul poate introduce un nume de utilizator;

- *MainMenuState* : Este o subclasă a clasei "State" și reprezintă starea meniului principal al jocului.

States Utils

- *Directions* : Implementeaza directiile pe care le poate lua eroul (Stanga, Dreapta);

- *EnemyFactory* : Responsabilă de crearea și returnarea obiectelor de tip entitate (enemy) în funcție de tipul specificat;

- *EnemyTypes* : Este o enumerare care definește tipurile posibile de inamici (enemies) în jocul "HarapAlb";

- *ItemFactory* : Este responsabilă de crearea și returnarea obiectelor de tip obiect (item) în funcție de tipul specificat;

- *EnemyTypes* : Este o enumerare care definește tipurile posibile de iteme în jocul "HarapAlb";

- *LoadMapMatrixFromTxt* : Oferă o metodă statică "LoadMap" pentru a încărca o matrice de hărți dintr-un fișier text;

- *TemporaryEntity* : Oferă o metodă statică "LoadMap" pentru a încărca o matrice pentru o harta dintr-un fișier text;

4. *Design Pattern (Actualizare)*

- *Factory* : Am folosit acest pattern pentru a creea entitatile inamice si itemele care urmeaza sa fie adaugate in EntityManager, respectiv ItemManager.

5. *Algoritmi Utilizati*

Tratarea Coliziunilor se realizeaza prin urmatoarele etape generale:

1. Verificarea direcției de mișcare:
    - Dacă playerul se deplasează spre dreapta (xMove > 0), se verifică coliziunea pe marginea dreaptă a dreptunghiului de coliziune al playerului.
    - Dacă playerul se deplasează spre stânga (xMove < 0), se verifică coliziunea pe marginea stângă a dreptunghiului de coliziune al playerului.
    - Dacă playerul se deplasează în sus (yMove < 0), se verifică coliziunea pe marginea superioară a dreptunghiului de coliziune al playerului.
    - Dacă playerul se deplasează în jos (yMove > 0), se verifică coliziunea pe marginea inferioară a dreptunghiului de coliziune al playerului.
2. Determinarea tile-urilor cu care se produce coliziunea:
    - Se calculează poziția tile-ului corespunzător marginei verificate (pe baza coordonatelor x și y și dimensiunilor dreptunghiului de coliziune).
    - Se verifică coliziunea cu tile-urile în vecinătatea respectivei margini.
3. Gestionarea coliziunii:
    - Dacă nu există coliziune pe marginea verificată, playerul poate continua să se deplaseze în direcția respectivă.
    - Dacă există coliziune, se ajustează poziția playerului astfel încât să fie aliniat corespunzător cu marginea tile-ului pentru a evita spațiul liber între player și tile.
4. Alte acțiuni:
    - În cazul mișcării în jos/sus, se verifică și o condiție specifică în care playerul să nu poată să sari mai mult decât o anumită înălțime (MAX_JUMP_HEIGHT) față de o poziție de referință (yOnDownCollision).

Gestionarea obiecte/inventar:

Clasa ItemManager păstrează o listă de obiecte (itemList) care reprezintă toate obiectele (itemele) de joc existente într-o anumită harta. Metoda Update() este responsabilă de actualizarea stării fiecărui obiect din lista itemList. Aceasta parcurge fiecare obiect și, în funcție de tipul acestuia, apelează metoda Update() corespunzătoare pentru a actualiza starea obiectului.

Deplasarea inamicilor:

Metoda GetInput() este responsabilă de obținerea intrărilor pentru deplasarea inamicului. În funcție de detecția eroului și de coliziunile cu mediul înconjurător, se determină direcția și viteza de deplasare a șobolanului. Dacă eroul este detectat în raza de acțiune a inamicului, acesta își va îndrepta atenția către erou și va încerca să se apropie de acesta. În caz contrar, inamicul va rămâne într-o stare de patrulare, deplasându-se înainte și înapoi.

Bibliografie

- Backyard Ninja Design – Free Stuff : Link
- OpenGameArt.org : Link