

ANEXA 1

Oliver's Adventure

Martin Maria

1209A

Povestea jocului:

Jocul este situat în Pădurea Tunetului, în perioada în care felinele dețineau puterea lumii. Clanul Focului, stăpânul pădurii, se luptă cu inamicii săi, Clanul Piraților, pentru apărarea teritoriului și protejarea locuitorilor pădurii, războinicii având această datorie.

Pentru că niciodată nu sunt prea mulți războinici, în fiecare primăvară este ales un pisoi ucenic ce urmează un drum plin de provocări pentru a putea deveni, la rândul său, războinic.

Oliver este un pisoi ce se antrenează pentru a-și îndeplini visul: să devină un războinic și să lupte alături de războinicii clanului său pentru apărarea ținutului lor.

Ei a fost învățat să atace inamicii, iar pe parcursul drumului său găsește diferite obiecte "magice" și mâncare care îl vor ajuta să dobândească mai multă putere și să-și atingă scopul, dar există și obstacole ce vor îngreuna misiunea lui Oliver.

Pentru a putea primi titlul de Războinic al Clanului Focului, la finalul drumului va trebui să se lupte cu un inamic puternic, Dexter, o felină din clanul vecin. Dacă Oliver va câștiga, Regina Clanului Focului îl va numi, oficial, războinic, îndeplinindu-și astfel visul. Dacă inamicul îl va învinge pe Oliver, el va trebui să reînceapă misiunea.



Prezentare joc:

Jocul este pentru un singur jucător. Jucătorul trebuie să parcurgă nivelul colectând pești sau ghemotoace de lână ce îi vor aduce mai multe puncte și să se lupte cu inamicii, șobolani. Pentru a trece la următorul nivel, jucătorul va trebui să parcurgă harta până la găsirea unui portal. Pentru a putea câștiga, după ce a parcurs cele două nivele în care va colecta obiecte, va trebui să se lupte cu inamicul final. Dacă va pierde lupta finală, va trebui să înceapă jocul de la capăt.

Strategia de joc: SINGLEPLAYER

Reguli joc:

Jocul implică colectarea anumitor obiecte, fiecare având punctaje diferite, și lupta cu inamicii care, la finalul jocului, vor aduce puncte bonus. Dacă personajul a fost atins de către inamici va pierde câte o viață. Dacă personajul va rămâne fără vieți va începe jocul de la capăt. Pentru nivelul final, va trebui să se lupte cu inamicul cel puternic, "the final boss". Jucătorul va avea o limită de 5 vieți, iar inamicul 10, iar scopul jucătorului este de a-l ataca pe inamic, până când va rămâne fără vieți. Viețile se vor pierde, pe rând, la fiecare atac primit. În caz, contrar, dacă inamicul va bate jucătorul, acesta va începe tot jocul de la capăt.

Personajele jocului:

- **Oliver** este protagonistul și jucătorul-personaj. El este de obicei descris ca fiind tipul eroic, vesel, visător, ambicioz și este un adevărat luptător.



- **Dexter** este antagonistul, mereu în căutare de noi aventuri și conflicte. Totodată el este loial clanului său și este un personaj puternic.



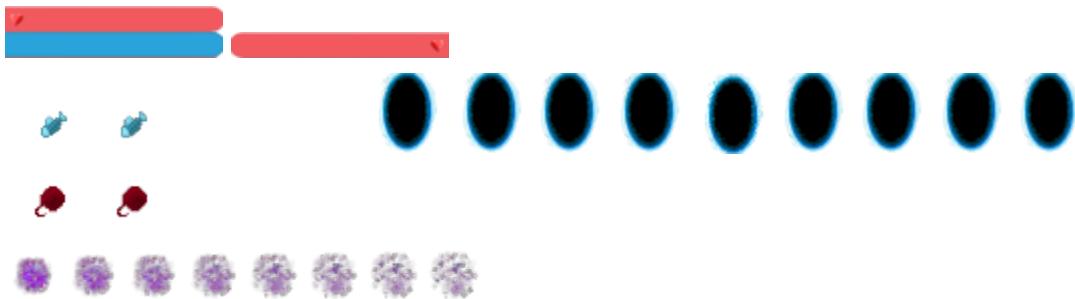
- **Şobolanii**, inamicii care îl împiedica pe erou să-şi îndeplinească misiunea.



Tabla de joc:

- Componente pasive: platforme de iarbă (personajul va putea sări pe ele)
- Componente active:
 - ghemotoc de lână, pești: obiecte pe care personajul le va colecta pentru a obține puncte
- Structura tablei de joc :
 - Pentru nivelele în care se vor colecta obiecte, există diferite platforme de iarbă dispuse aleatoriu pe hartă, cele înalte putând fi accesate sărind. Pe unele platforme se află obiectele pe care Oliver le va colecta. Obiectele pot fi pe sol sau în aer. În colțul din stânga sus se va ține evidența punctajului și a vieților.
 - Pentru nivelul în care protagonistul se va lupta cu inamicul, există platforme pe care poate să sără. În partea de sus, se va ține evidența vieților pe care personajele le dețin.

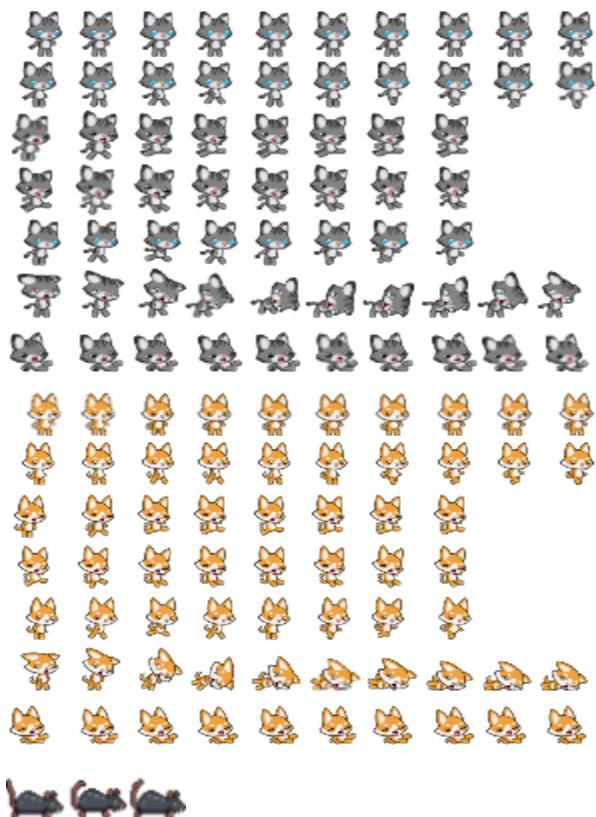




Mecanica jocului:

- Game Points:
 - Pești: 25 puncte;
 - Ghemotoc de lână: 50 puncte;
- Game Points BONUS(se vor adauga la finalul jocului):
 - Vieți rămase: 10 puncte;
 - Inamici uciși: 15 puncte;
 - Boss Final: 1000 puncte;
- Keys:
 - Săgeată stânga: personajul va merge spre stânga;
 - Săgeată dreapta: personajul va merge spre dreapta;
 - Space: personajul va face un salt;
 - R + săgeată stânga: personajul va alerga spre stânga;
 - R + săgeată dreapta: personajul va alerga spre dreapta;
 - H: personajul va aluneca, lovind inamicul

Game sprite:

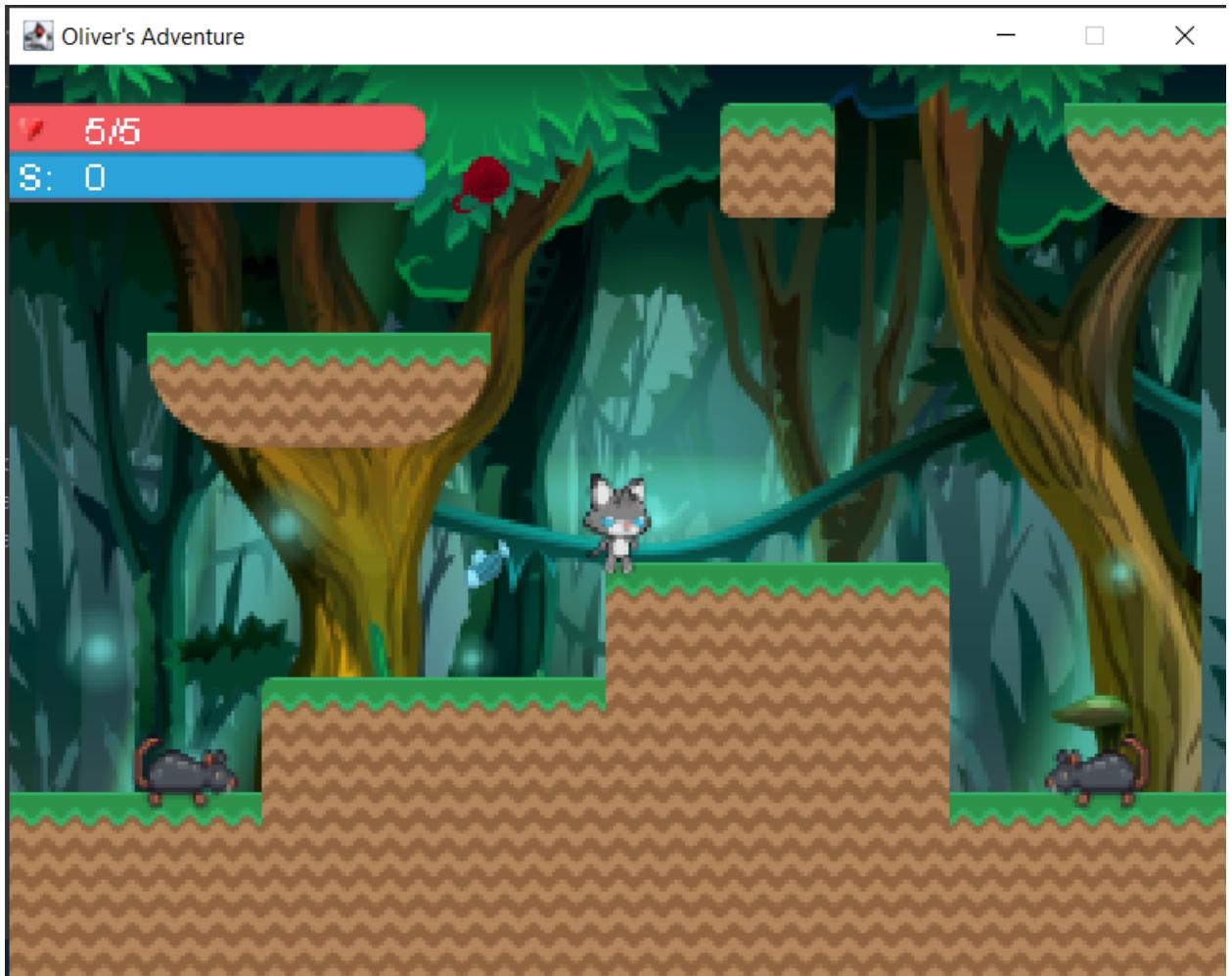


Descriere fiecare nivel

Primele două nivele ale jocului sunt de tip platformer, iar ultimul tip acțiune.

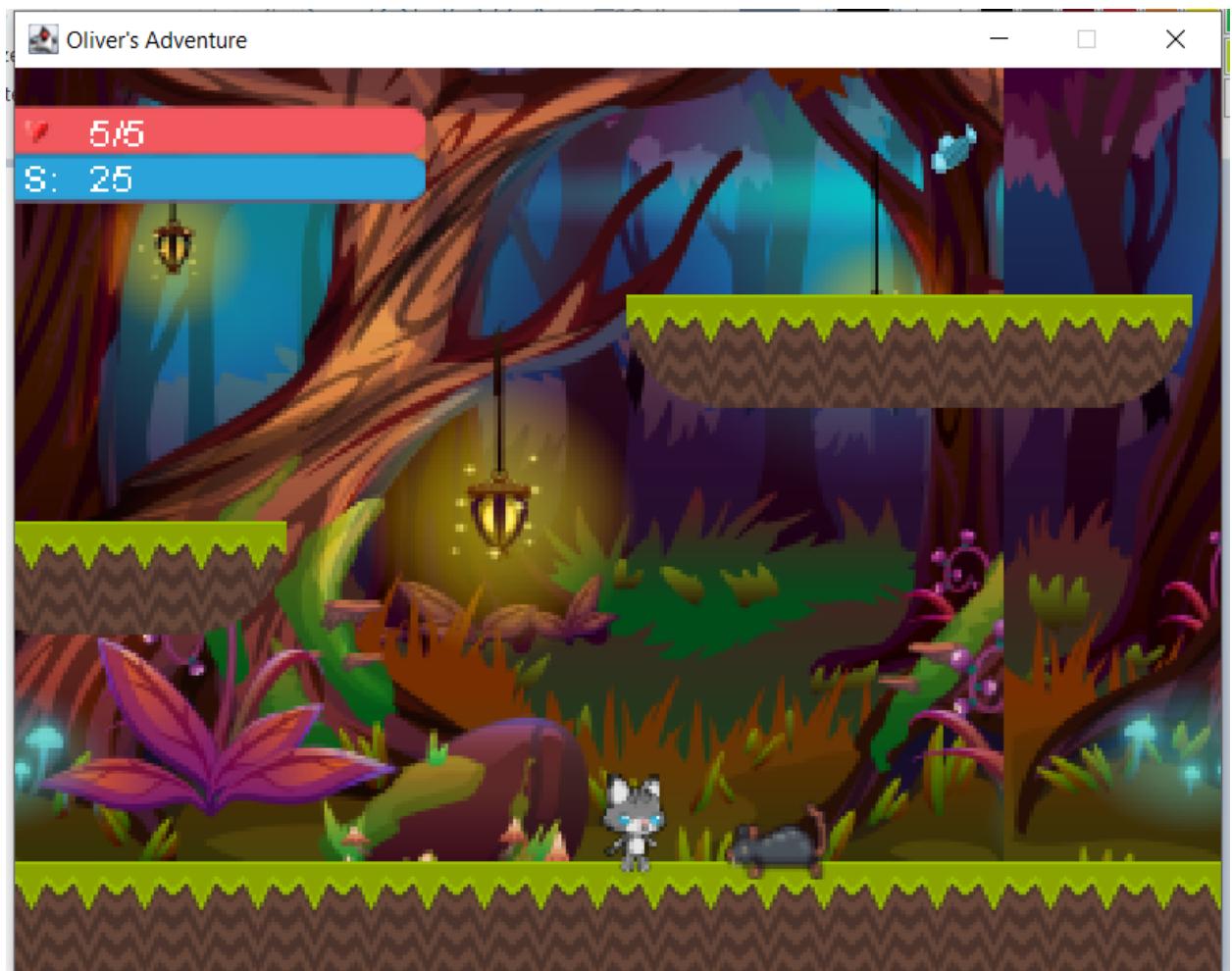
Nivelul 1:

Jucătorul va trebui să colecteze de pe hartă obiecte și trebuie să găsească portalul ce va face posibilă trecerea la noul nivel, având grija la inamicii ce îl vor împiedica.



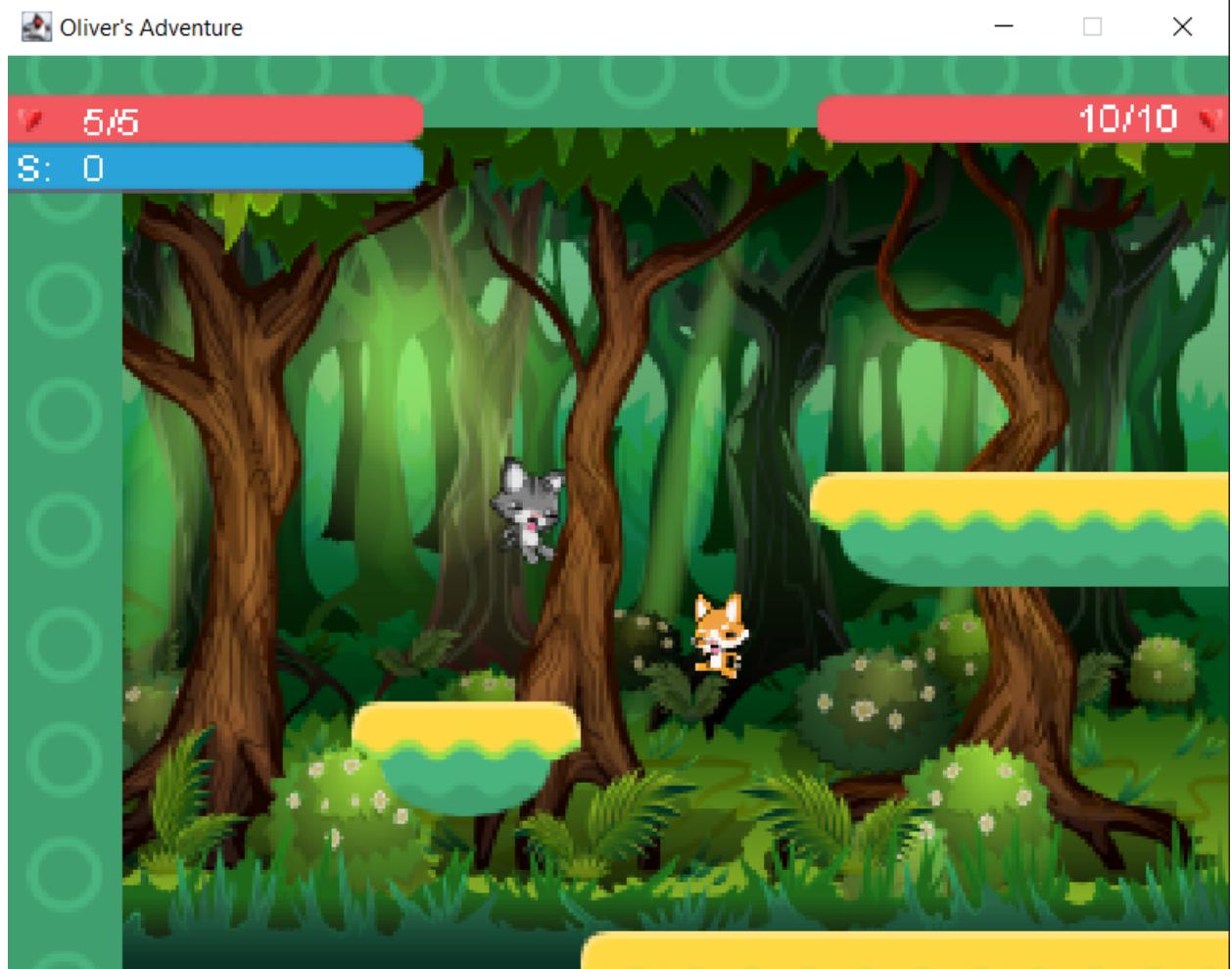
Nivelul 2:

Acet nivl, este asemnitor cu primul. Diferenea const în gradul de dificultate, obiectele fiind mai greu de colectat.



Nivelul 3:

În acest nivel, jucătorul se confruntă cu inamicul său, Dexter. El trebuie să lupte contra acestuia, lovindu-l de 10 ori și evitând atacurile sale. Dacă acesta câștigă, va fi numit războinic al Clanului Focului. Dacă va pierde, jucătorul va începe tot jocul de la capăt.

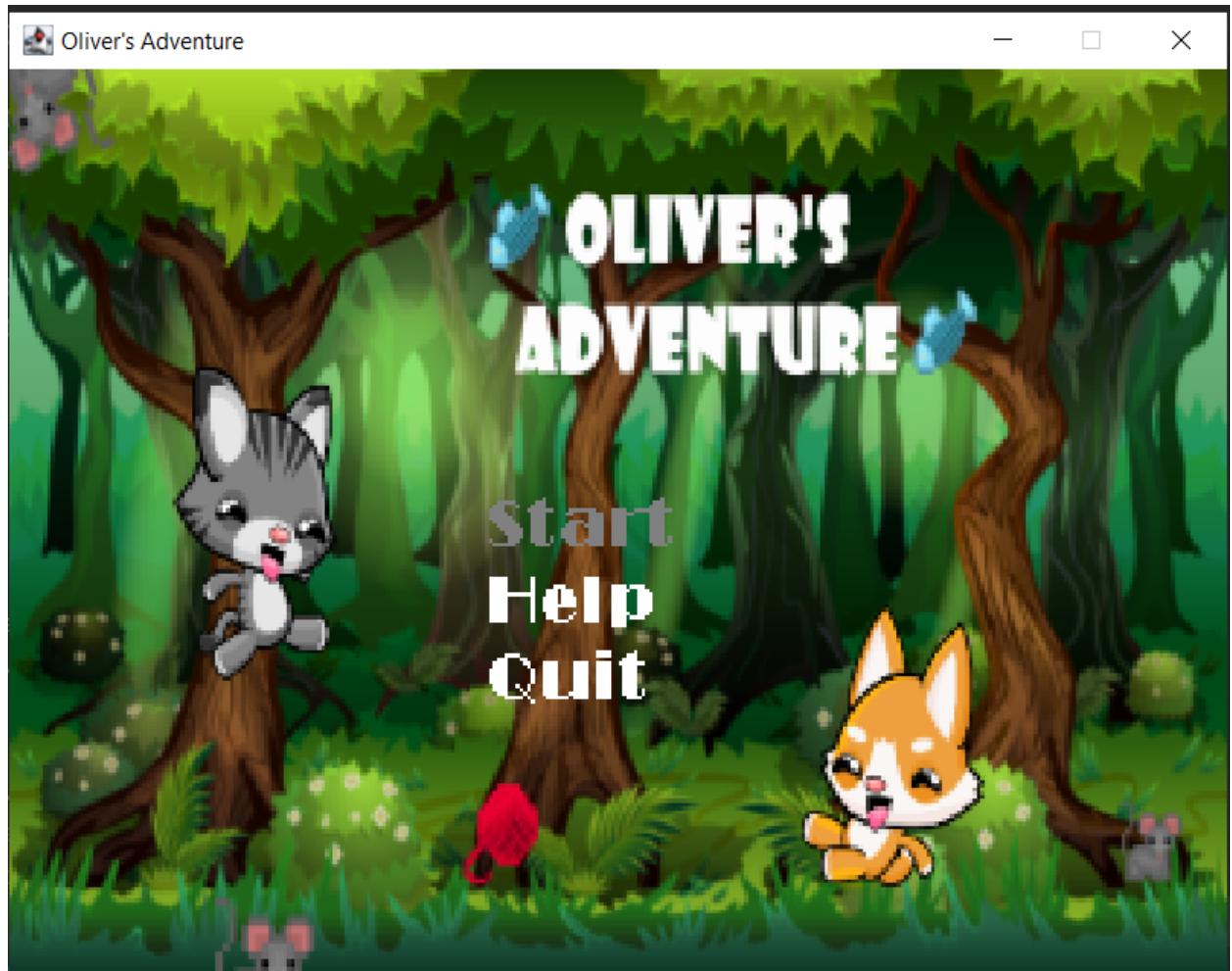


Descriere meniu

Meniu Principal:

Meniul va fi central. Acesta va putea fi accesat la începutul jocului, dacă jucătorul va pierde toate viețile sau la finalul jocului.

Meniul de la început va cuprinde un buton de “START”, prin care se va începe un nou joc, butonul de “HELP”, care va cuprinde comenzi jocului, și butonul de “QUIT”, prin care se va ieși din joc.





GAME CONTROLS



MOVE LEFT/RIGHT



RUN



ATTACK

SPACE

JUMP

Back

Meniu Game Over:

Acesta va putea fi accesat când personajul își va pierde toate viețile.

Meniul va cuprinde un buton de “RESTART”, prin care se va reîncepe jocul, butonul de “MAIN MENU”, prin care se va accesa meniul principal, și butonul de “QUIT”, prin care se va ieși din joc.



Meniu Finish:

Acesta va putea fi accesat la finalul jocului.

Meniul va cuprinde un buton de "RESTART", prin care se va reîncepe jocul, butonul de "MAIN MENU", prin care se va accesa meniul principal, și butonul de "QUIT", prin care se va ieși din joc.

Se vor afișa scorul obținut în fiecare nivel, numărul de vieți rămase și numărul de inamici uciși. Se vor acorda puncte bonus în funcție de vietile rămase, numărul de inamici uciși și pentru învingerea boss-ului final, iar la final se va regăsi scorul total.



ANEXA 2

Structura Bazei de Date

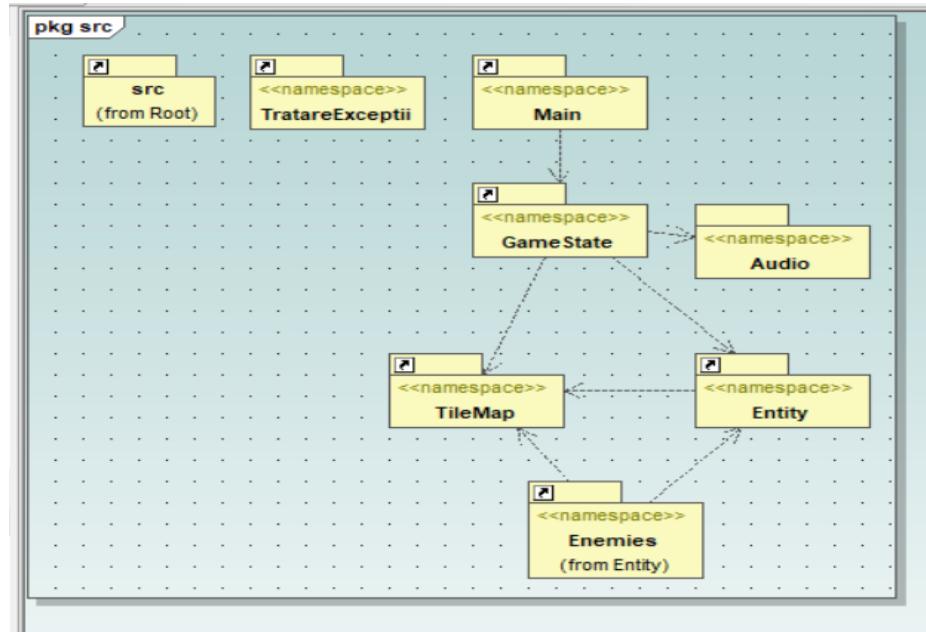
Am creat o Tabelă în care voi salva punctajele obținute pentru fiecare nivel, numărul de viați rămase după terminarea fiecărui nivel și numărul de inamici uciși în fiecare nivel. Se va crea o nouă tabelă pentru fiecare nou joc început.

The screenshot shows the DB Browser for SQLite interface with a database named 'ScoreDB.db'. A table named 'PLAYER' is selected, displaying the following data:

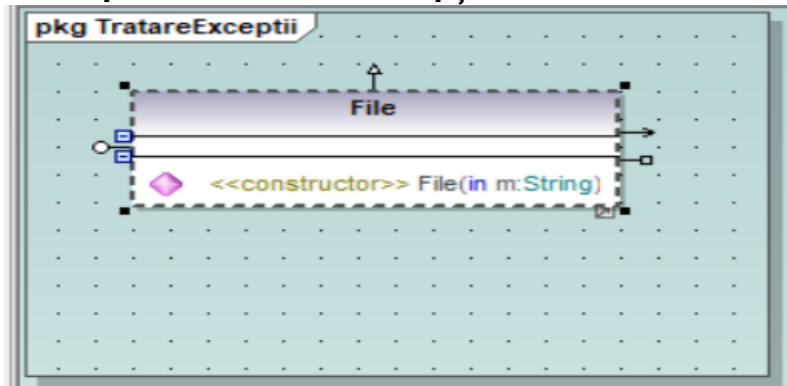
	Score	Lives	Kills
1	225	3	6
2	425	4	6
3	0	4	0

Diagrama de Clase

Packages dependencies

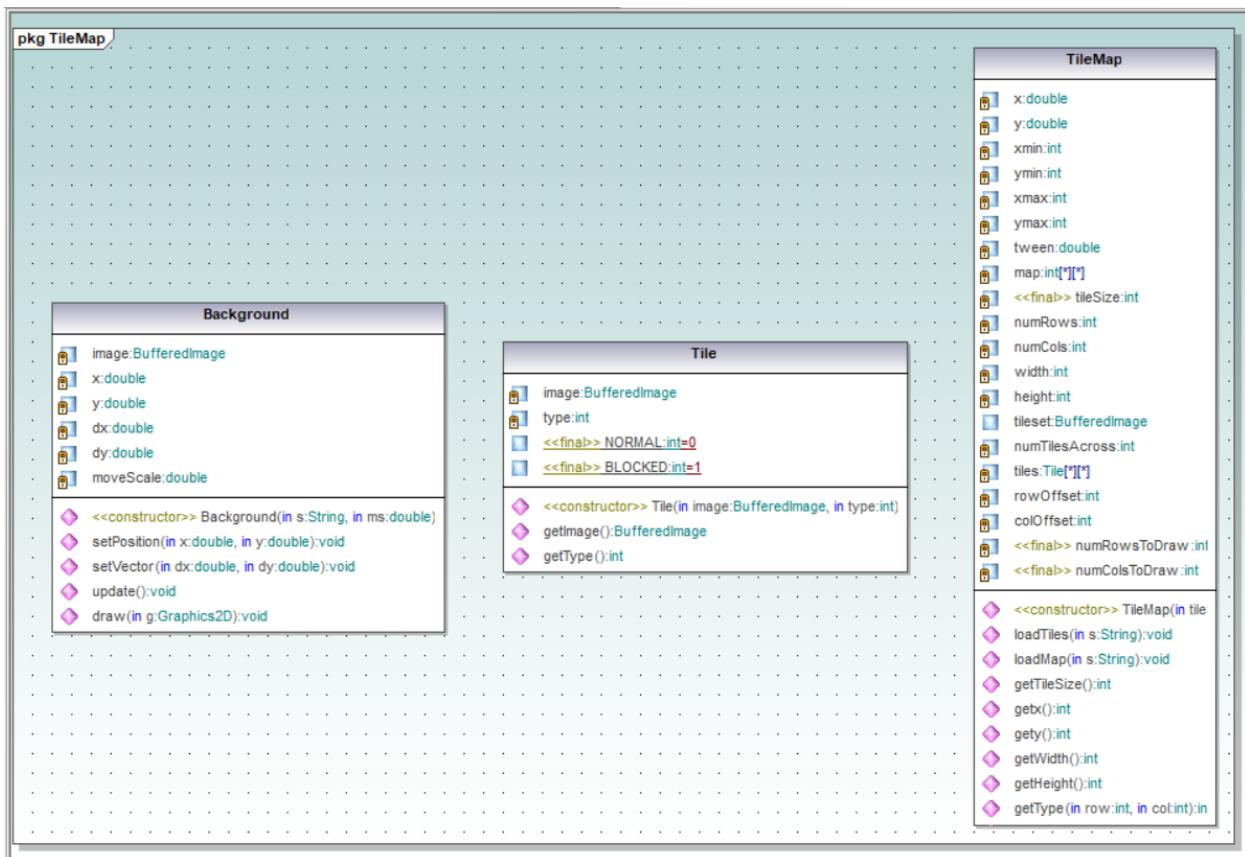


Clasa pentru tratarea exceptiilor



Clasa **File** am folosit-o pentru aruncarea exceptiilor în care am încărcat o imagine ce nu există în package-ul Resources.

TileMap Package

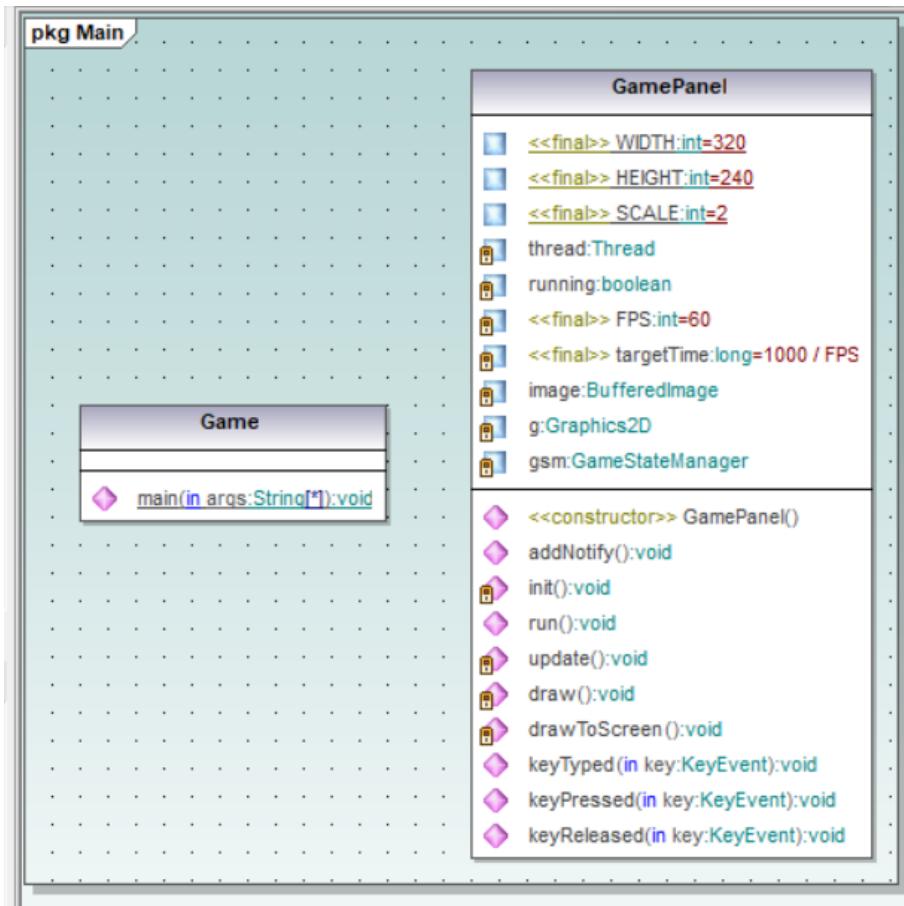


Clasa **Background** va oferi fundalul jocului. Prin intermediul constructorului vom încărca imaginea. Metodele `setPosition()` și `setVector()` vor realiza o trecere lină a camerei asupra fundalului.

Clasa **Tile** va încărca o platformă. Aceasta poate să fie **BLOCKED**, ce nu permite trecerea, sau **NORMAL**, prin care ne putem deplasa.

Clasa **TileMap** are rolul de a încărca harta și platformele aferente. Constructorul clasei va încărca platformele(tiles). Metoda loadTiles va citi imaginea tiles-urilor și stabilește dacă acestea permit sau nu trecerea personajului. Metoda loadMap permite parcurgerea hărtii. Prin metoda getType verificăm tipul platformei.

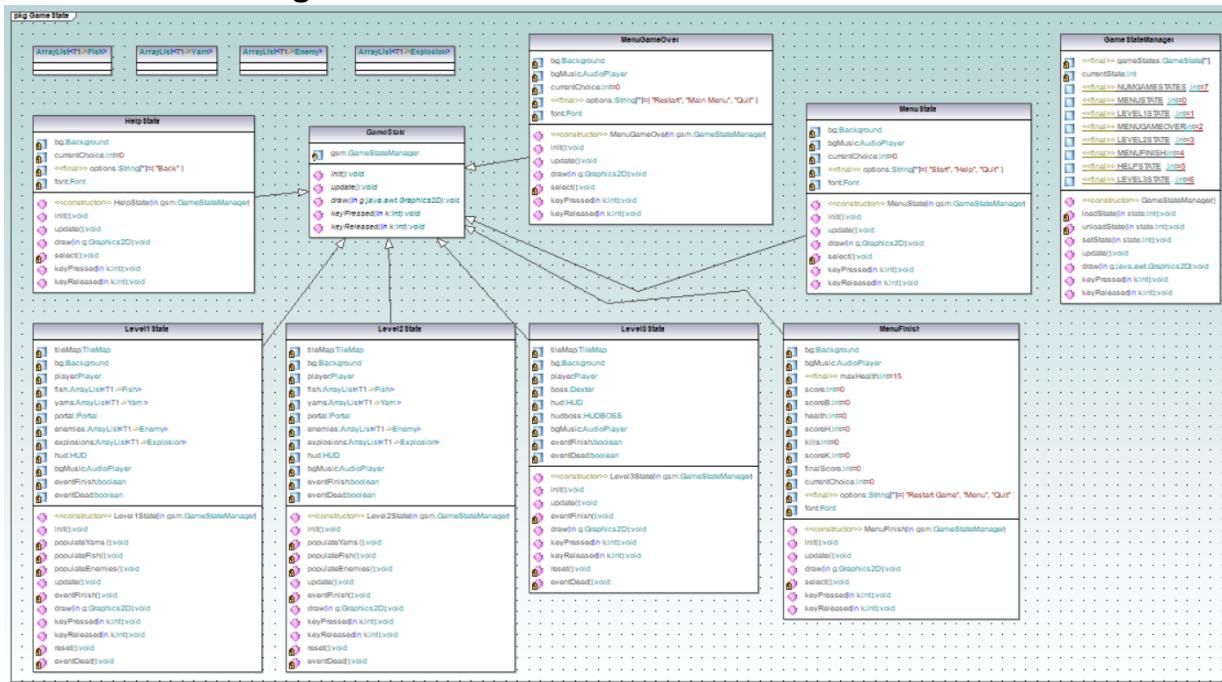
Main Package



Clasa **Game** realizează și afișează fereastra jocului.

Clasa **GamePanel** moștenește clasa JPanel și implementează interfața Runnable, pentru a avea un thread, și KeyListener. Apelul metodei run() initializează un obiect de tipul Thread și va crea o instanță GamePanel. Cu ajutorul metodelor draw() și drawToScreen() se va realiza interfața grafică, iar metoda update() va actualiza starea jocului. Metodele keyTyped(), keyPressed(), keyReleased() vor face posibilă deplasarea personajului pe hartă sau navigarea în diversele meniuri.

GameState Package



Clasa GameState este o clasă abstractă ce definește metode pe care celelalte clase le vor deriva din ea.

Clasa MenuState cuprinde opțiunile unui meniu principal (începerea jocului, comenzi jocului, ieșirea din joc). În constructorul clasei vom încărca fundalul acesteia, vom stabili fontul, culoarea și dimensiunea textului afișat. Prin metoda `select()` vom selecta o opțiune a meniului, iar metoda `keyPressed` ne va ajuta în navigarea prin meniu.

Clasa HelpState este folosită ca o opțiune din MenuState.

Clasa Level1State va crea nivelul 1 al jocului. Metoda `init()` va încărca tiles-urile, harta, fundalul pentru acest nivel, va crea personajul, poziționându-l într-un anumit punct inițial, se vor popula inamicii și obiectele de colectat, prin intermediul metodelor `populateEnemies()`, `populateFish()` și `populateYarns()`, acestea având rolul de a crea un vector de obiecte și de a stabili unde vor fi poziționate, și se va stabili poziția portalului. În metoda `update()` se va verifica starea jucătorului, atacul asupra inamicilor, colectarea obiectelor și se vor actualiza viețile și scorul când este necesar.

Metoda `draw` va desena player-ul, obiectele, tiles-urile, fundalul, hud-ul și portalul de trecere la nivelul următor.

Metoda `eventDead` va porni **MenuGameOver** când personajul rămâne fără vieți.

Metoda `eventFinish` face trecerea la nivelul următor și va salva în baza de date punctajul obținut în acest nivel, viețile rămase la terminarea nivelului și numărul de inamici uciși pe parcursul nivelului.

Metoda `keyPressed` ne va ajuta să mișcăm personajul.

Clasa Level2State va crea nivelul 2 al jocului. Aceasta clasa are la bază ideile din **Level1State**. Diferă harta, tiles-urile și unde sunt poziționate obiectele de colectat și inamicii.

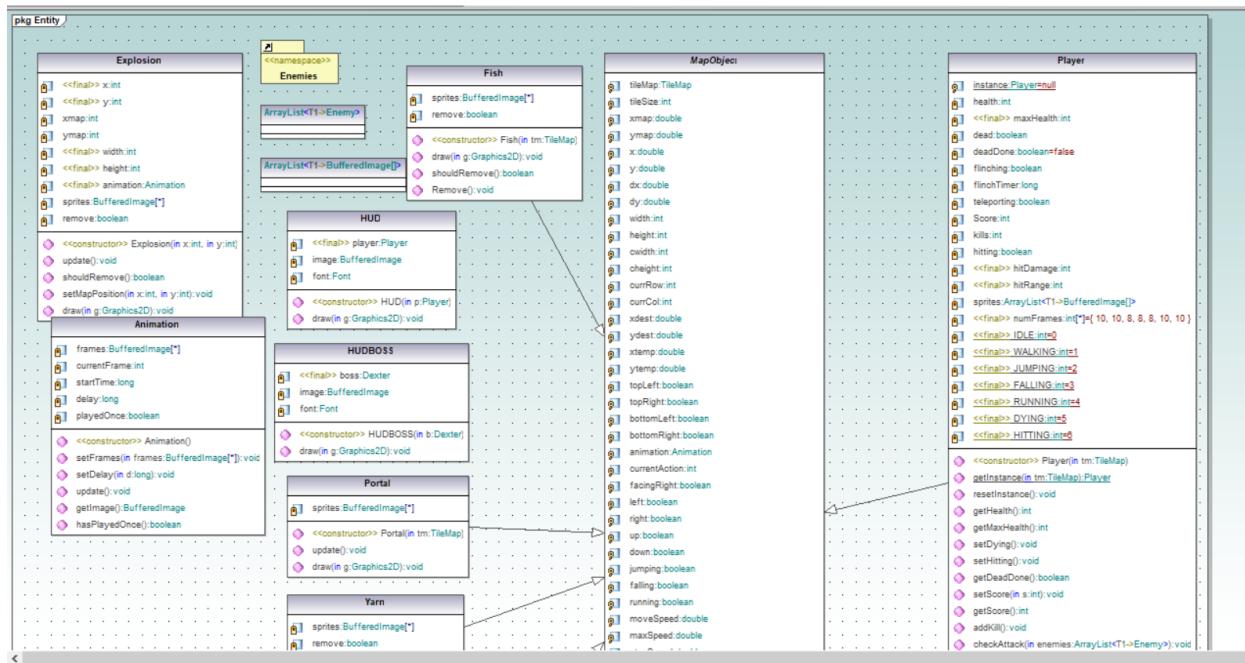
Clasa **Level3State** va crea nivelul 3 al jocului. În aceasta nu vor mai exista inamici sau obiecte de colectat, ci va fi un inamic final, boss-ul. Acesta va avea mișcări asemănătoare personajului principal, îl va urmări și ataca. Odată învins inamicul, se va trece la MenuFinish.

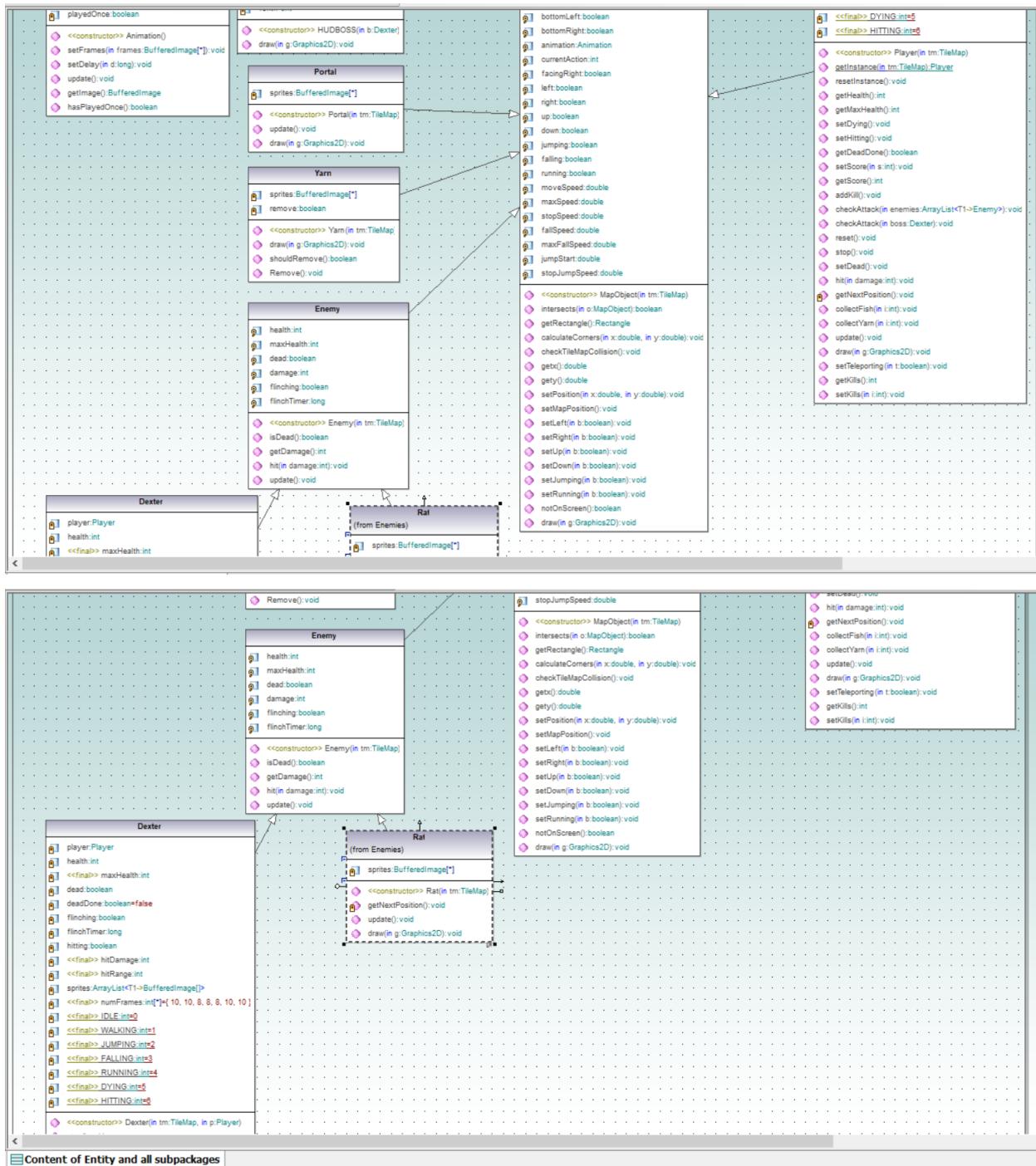
Clasa **MenuGameOver** va fi apelată dacă personajul va rămâne fără vieți. Este asemănătoare cu MenuState. Diferența constă în opțiunile meniului. Prin MenuGameOver putem reîncepe jocul (RESTART) și se va crea o nouă tabelă în baza de date, putem să ne întoarcem în meniu principal (MAIN MENU) sau putem ieși din joc (QUIT).

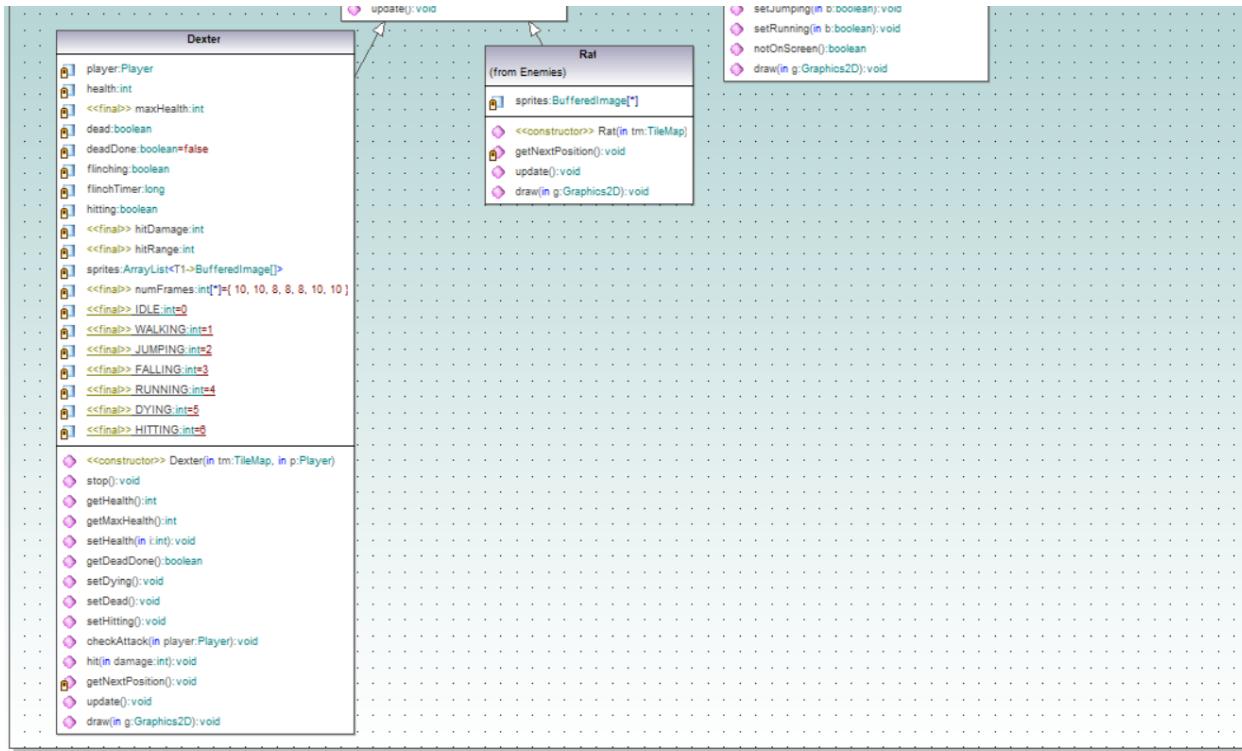
Clasa **MenuFinish** va fi apelată după ce jucătorul va termina jocul. Aceasta este asemănătoare cu MenuGameOver, dar față de cea precedentă, vom lua din baza de date elementele salvate pentru a le afișa pe ecran și se vor adăuga puncte bonus unde este cazul.

Clasa **GameStateManager** are scopul de a accesa toate state-urile jocului (nivele și meniuri). Totodată, în cazul în care există deja o tabelă în baza de date, clasa o va șterge și va crea una nouă prin care să salvăm noile date.

Entity Package







Clasa **MapObject** are rolul de a realiza coliziunile cu harta. Metoda intersect verifică intersectarea personajului cu inamicii, obiectele de colectat sau portalul. Metoda checkTileMapCollision() determină deplasarea obiectului și coliziunile acestuia cu tiles-urile. Metodele setPosition și setMapPosition() stabilesc poziția obiectului.

Metoda onScreen verifică dacă personajul este în fereastra de joc, iar metoda draw desenează personajul orientat spre stânga sau dreapta.

Clasa **Animation** redă animațiile sprite-urilor.

Clasa **Dexter** creează boss-ul final. Aceasta va încărca sprite-urile boss-ului, va stabili însușirile lui (health, hitDamage, etc). Metoda checkAttack verifică dacă playerul ataca inamicul. Metoda getNextPosition stabilește următoarea poziție a boss-ului, în funcție de poziția player-ului.

Clasa **Enemy** moștenește clasa MapObject. Aceasta verifică viața inamicilor.

Clasa **Rat** creează inamicii jocului, realizând și coliziunile acestora cu player-ul și harta.

Clasa **Explosion** realizează animația de dispariție a unui inamic ce tocmai a fost ucis.

Clasele **Fish** și **Yarn** moștenesc clasa MapObject. Acestea vor încărca imaginile obiectelor ce vor fi colectate pe parcursul jocului.

Clasele **HUD** și **HUBOSS** vor desena bara de stare a personajului și a boss-ului. Acestea se vor mișca în același timp cu personajul principal. HUD-ul player-ului va ține evidența vieților acestuia și a scorului obținut, pentru fiecare nivel, iar HUD-ul boss-ului va ține evidența vieților lui.

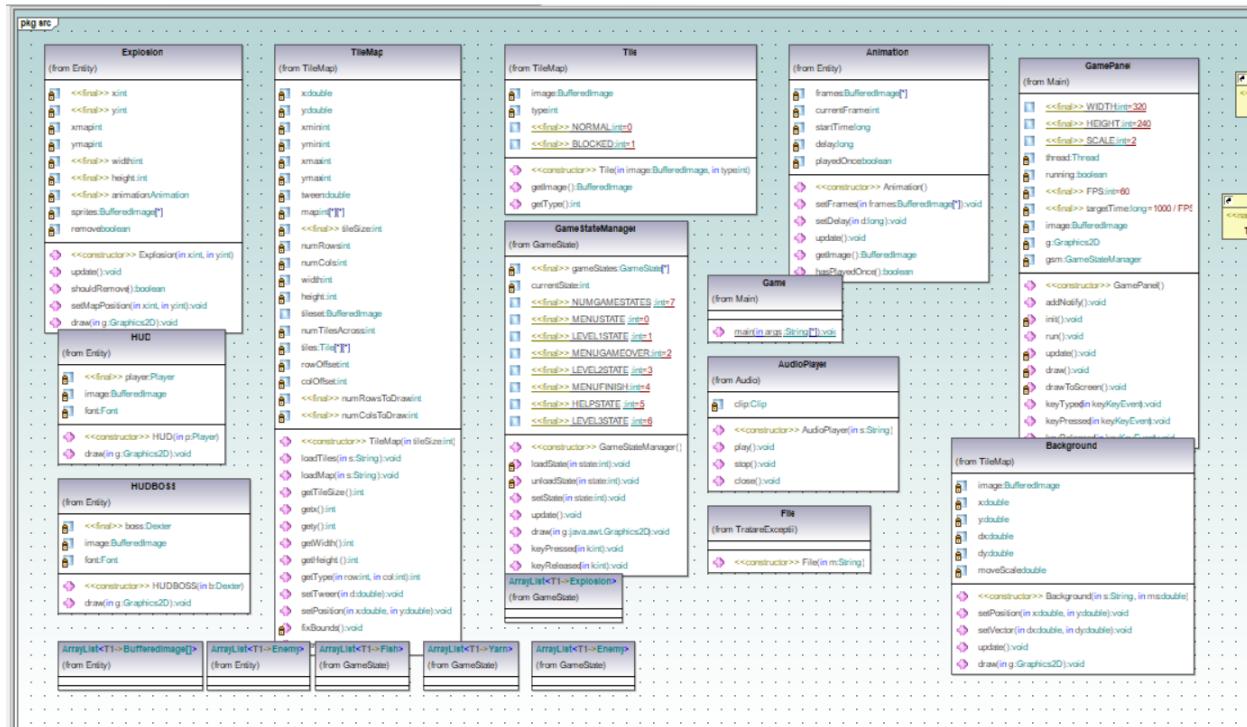
Clasa **Player** creează personajul principal. Aceasta va încărca sprite-urile player-ului, va stabili însușirile lui (health, hitDamage, etc). Metoda checkAttack verifică dacă

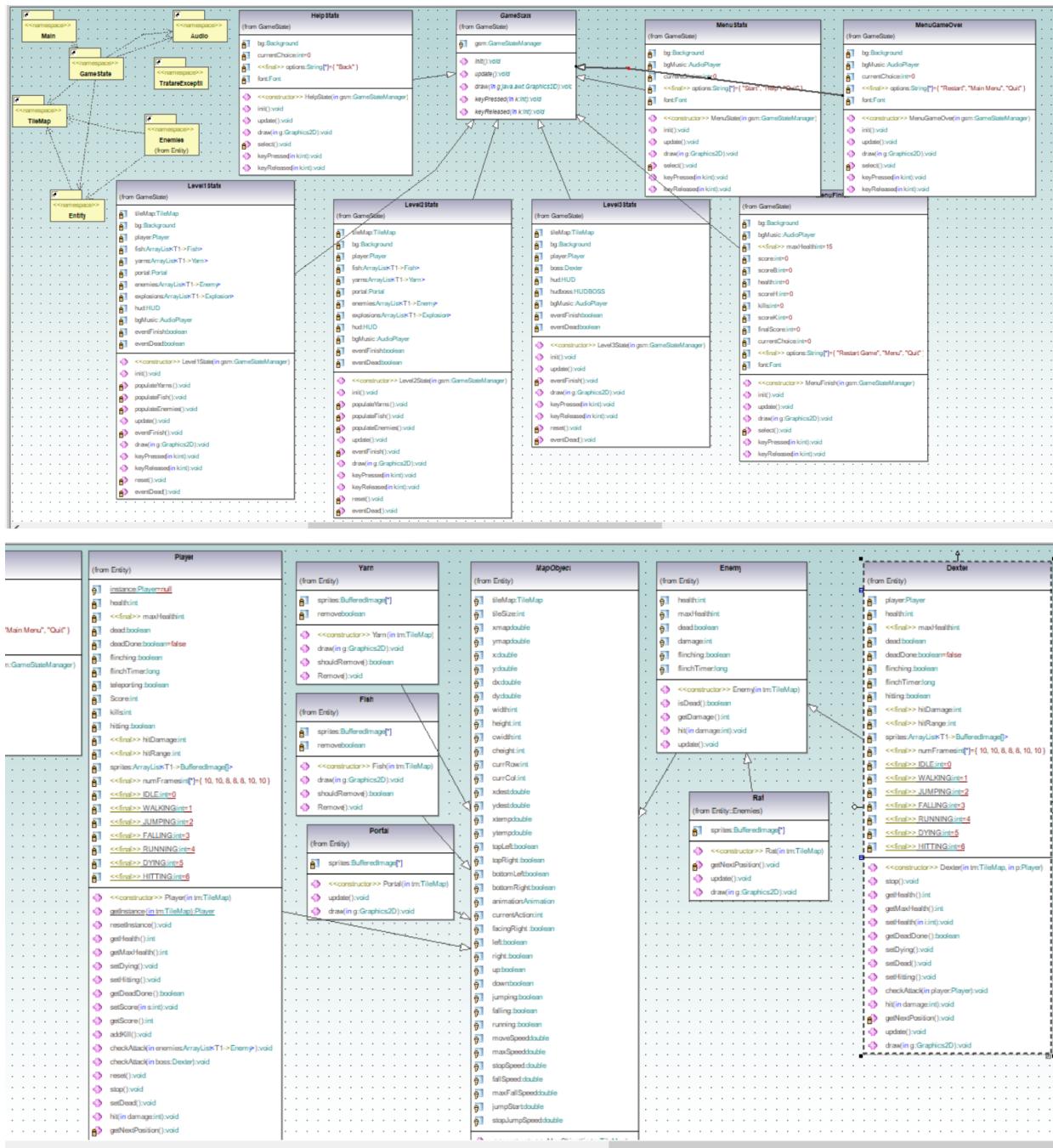
playerul este atacat. Metoda `getNextPosition` va stabili următoarea poziție a player-ului. Metoda `update` va actualiza poziția și mișcările, va verifica coliziunile cu harta. Clasa **Portal** va încărca imaginea ce va face posibilă trecerea la nivelul următor.

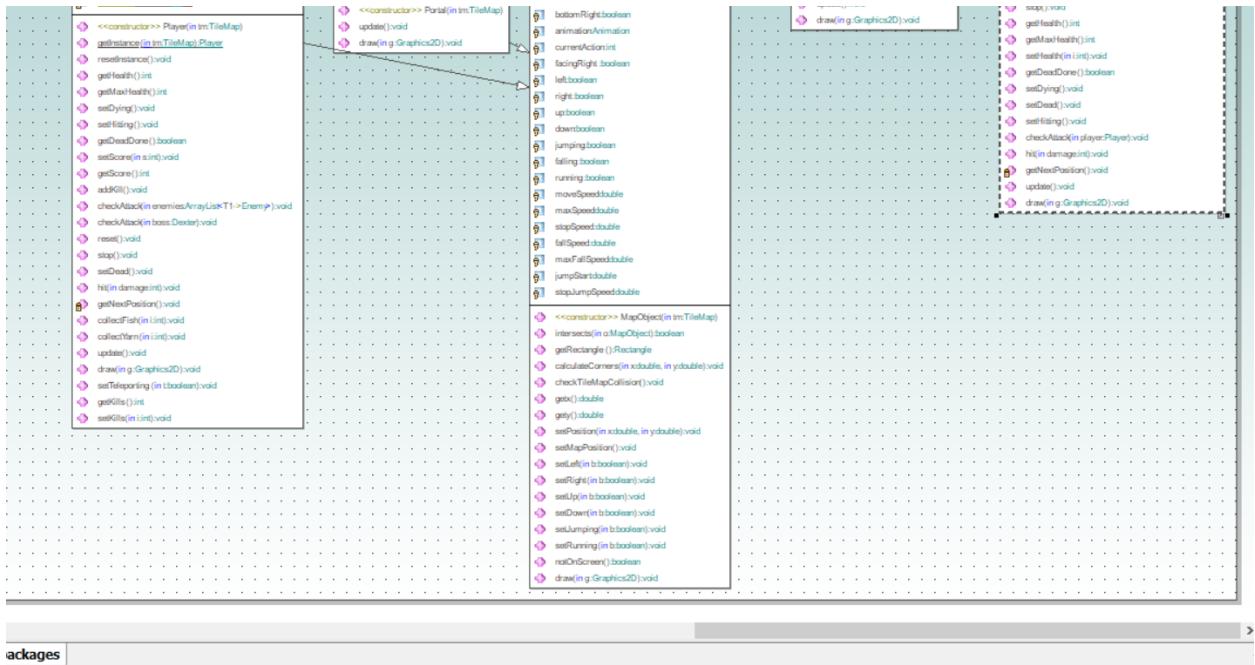
Audio Package



Clasa **AudioPlayer** face posibilă inserarea sunetelor de fundal, metodele `play()` și `close()` pornind și oprind sunetele din fiecare state.







Şabloane de proiectare

- Singleton

Şablonul a fost implementat în clasa Player.

```
protected static Player instance = null;
```

Prin metoda getInstance verificăm dacă nu a fost inițializată instanța. Dacă aceasta nu există, vom apela constructorul clasei Player. Pentru a face o nouă instanțiere o vom reseta prin metoda resetInstance().

```
public static Player getInstance(TileMap tm){
    if(instance == null){
        instance = new Player(tm);
    }
    return instance;
}

public void resetInstance(){ instance = null; }
```

Metoda getInstance a fost apelată la inițializarea player-ului din nivelul 2, iar resetInstance() de fiecare dată când player-ul pierde toate viețile.

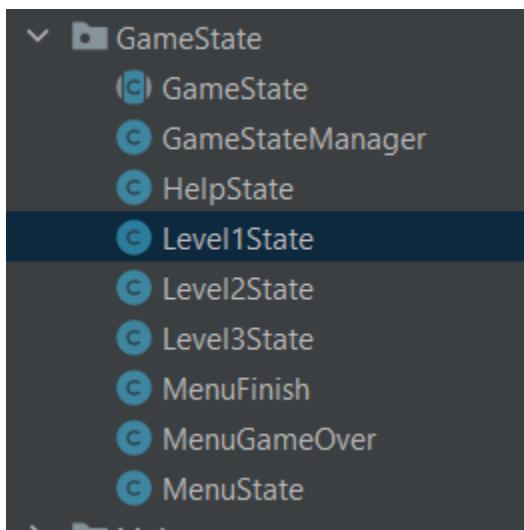
```
bg = new Background(s: "/Backgrounds/CartoonBackground");

player = player.getInstance(tileMap);
//where the player starts level2
player.setPosition(x: 50, y: 150);
player.setScore(0);
player.setKills(0);
```

```
// player has died
private void eventDead() {
    if(player.getHealth()==0){
        player.resetInstance();
        if(player.getDeadDone())
            gsm.setState(GameStateManager.MENUGAMEOVER);
    }
}
```

- State

Şablonul a fost implementat prin clasele din package-ul GameState. Clasele state prezintă un număr de stări prin care putem naviga.



Bibliografie

- <https://opengameart.org/>
- <https://www.fesliyanstudios.com/>