

Titlu proiect : Tom's adventure
Autor : Boaca Madalina-Elena
Grupa : 1207A

Povestea jocului :

Motanul Tom asculta mereu poveștile bunicului sau care în tineretea lui fusese marinări. În una dintre povesti aflată ca acum mulți ani, în urma naufragiului pe o insulă din oceanul Atlantic a găsit la capătul unui tunel o comoară, dar pe care nu a reușit să o aducă de acolo, presat de oamenii acelor locuri care l-ar fi jefuit și faptului că avea să plece de acolo cu o simplă barcă. Tom hotărăște să plece în căutarea comorii, astfel își începe marea lui aventură. Debarcă la portul insulei și de aici își începe drumul spre peșteră. Va reuși Tom să găsească comoara? Dar va rămâne el în viață?

Prezentarea jocului :

Jocul este single-player, în acesta jucătorul trebuie să învingă dușmanii pentru a avansa între cele trei nivele ale jocului: nivelul portului, nivelul pădurii și cel al tunelului, la capătul căruia se află comoara descoperită de bunicul său.

Reguli joc :

Jocul implică deplasarea motanului spre trecerea de la un nivel la altul și la ultimul spre comoară. El are capacitatea de a-și împușca dușmanii, însă trebuie să aibă grija că o dată ce a intrat în contact cu ei poate fi la rândul său ucis, totodată în nivelul 2 nu trebuie să ia contact cu focul (dacă se întampla acest lucru moare direct). Jucătorul se va deplasa înainte și înapoi (pentru a evita situațiile când ajunge la un blocaj) și va putea sări pe anumite blocuri situate la înălțime, cu scopul de a colecta bani sau diamante și a evita contactul cu dușmanii. Castigarea jocului presupune ca jucătorul să ajungă la comoară, iar infrangerea are loc atunci când personajul își pierde viață (el are inițial trei vieți și pierde jumătate de fiecare dată când se atinge de sarpe, una când se atinge de Rino și 3 când se atinge de Mogly). Motanul merge în patru labe și sare tot în acest mod dar pentru a trage în dușmani este necesar să se ridice în picioare. La atingerea finish se va putea trece în următorul nivel prin teleportare.

Personajele jocului :

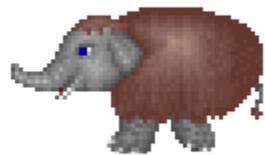
- **Motanul Tom** este protagonistul și jucătorul personaj. El este descris ca o pisică curajoasă și cu un mare simț al aventuri. Totuși principala sa caracteristică este de a evita pe cat posibil contactul cu adversari în loc să îi înfrunte în mod direct.



- **Şerpii din familia Snake** sunt principalii lui adversari care fac tot posibilul ca Tom să nu ajungă la comoara sub care este casa lor .



- **Mamutul Rino** este tulburat de prezența tot mai deasă a căutătorilor de comori din pădure și din acest motiv îi vânează pe toți cei ce intră în pădure cu acest scop. Nici Tom nu v-a scăpa de agresivitatea lui. Alături de șerpi și mamutul Rino îl va urmări în nivelul 2 și 3.



- **Bastinasul Mogly** este cel ce păzește comoara și este cel mai periculos dintre toți . Apariția lui va avea loc la intrarea în peșteră.

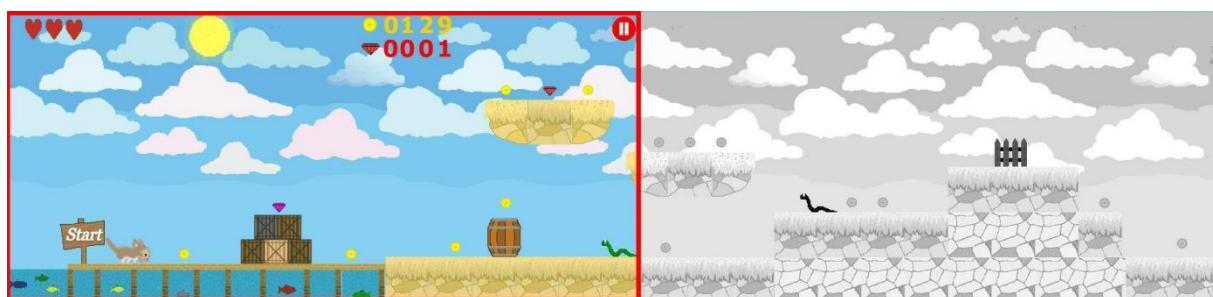


Tabla de joc :

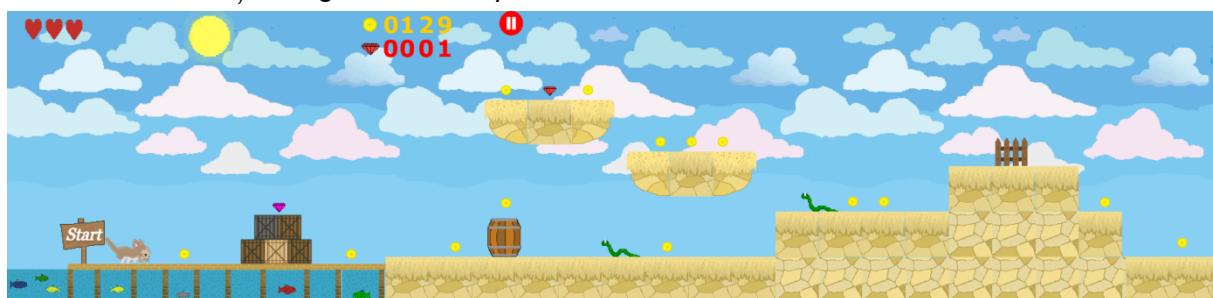
- ❖ **Componente pasive:** *sol(pamant, iarba, nisip, piatra), lemn, cutii lemn, gard, butoaie, pancarta(start/finish), soare, apa* etc. Componentele pasive au ca proprietate faptul ca personajele nu pot trece prin ele .
- ❖ **Componente active :** *bani, diamantele, cufarul, pesti, gloantele, foc.* Banii și diamantele dispar la contactul cu protagonistul, odată cu atingerea cufarului se va afișa mesajul pentru joc castigat. Contactul cu focul determina moartea lui Tom.
- ❖ **Structura tablei de joc :**
 - *Nivelul 1* : Tabla este construită prin adaugarea peste background(cer albastru care se va repeta de mai multe ori pe parcursul nivelului) a unei parti din elementele pasive și active(nisip, lemn, cutii de lemn, butoaie, banuti, diamante etc). Elementele sunt dispuse în principal în partea inferioară pentru a alcătui oceanul și solul , dar există blocuri de nisip și la înălțime.
 - *Nivelul 2* : Tabla este construită prin adăugarea peste background(padure) a unei parti din elementele pasive și active(pamant, iarba, cutii de lemn, butoaie, banuti, diamante, copaci, tufisuri etc) . Elementele sunt dispuse în principal în partea inferioară pentru a alcătui solul, dar există blocuri de pamant/iarba și la înălțime.
 - *Nivelul 3* : Tabla este construită prin adaugarea peste background(simularea unei pesteri) a unei părți din elementele pasive și active (piatra, cutii de lemn, butoaie, banuti, diamante etc).
- ❖ **Nivelul 1(Stadiul de proiectare etapa 1):**
Background :



Model nivel cu implementare cameră pe partea vizibilă efectiv pe ecran la un moment dat :



Același imagine fără a implementa camera :



❖ Nivelul 1(Stadiu final - jocul efectiv):

Background :



Screenshot-uri:



❖ **Nivelul 2:**

Background:

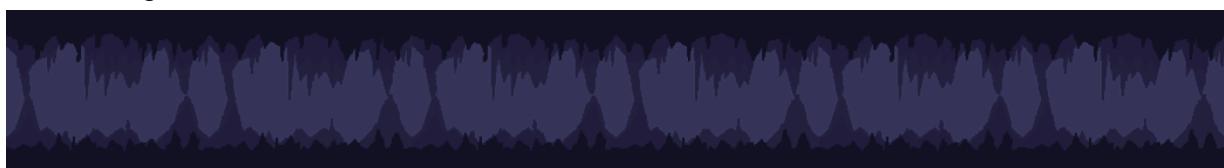


Screenshot-uri:



❖ **Nivelul 3:**

Background:



Screenshot-uri:



Mecanica jocului :

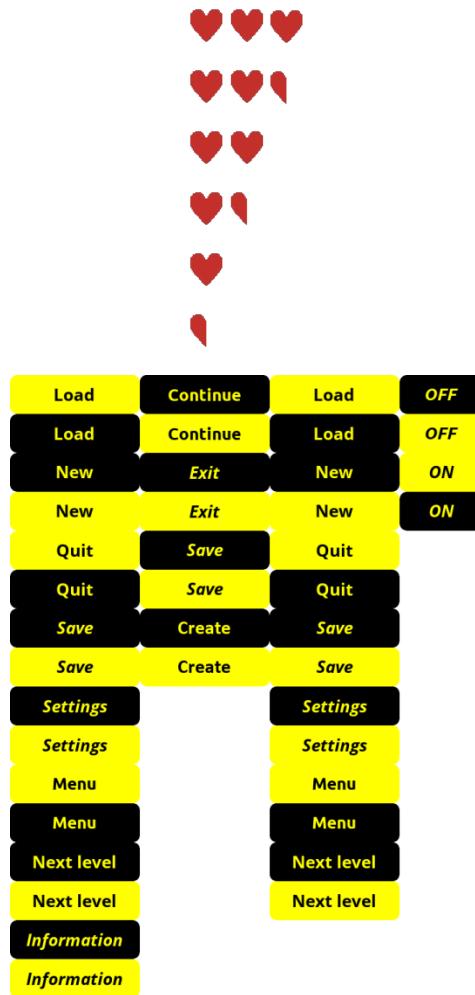
Motanul Tom se va deplasa stanga-dreapta și va putea efectua sărituri pentru a ajunge la blocuri situate mai sus de poziția lui, de unde va putea reveni prin cădere, deasemenea el va putea să tragă în adversari(realizând pentru asta o ridicare în două picioare). La contactul protagonistului cu bănuți și diamante ele vor dispărea și vor fi contorizate și afișate în partea superioară.

Inamicii lui Tom vor avea doar capacitatea de a se deplasa stanga-dreapta .

La moartea unui inamic el va dispărea pur și simplu.

Game sprite-urile utilizate :





Descriere fiecare nivel :

- ★ **Nivelul 1 :** Acțiunea la acest nivel începe odată cu ajungerea lui Tom la port și se continuă cu parcurgerea unei zone de nisip pe care are diferite obstacole(blocuri de nisip foarte înalte, butoaie, cutii de lemn, garduri), dar și recompense, banuti și diamante. În acest nivel el nu are ca adversari decât șerpii de care o data atins își pierde jumătate dintr-o viata, la un contact de 5-10 secunde. El își poate ucide adversarii tragand cu pistolul. Scopul lui pe parcursul acestui nivel este de a ajunge la placarda Finish de unde va putea trece la nivelul 2 .
- ★ **Nivelul 2 :** Acest nou nivel își desfășoară acțiunea în pădure, unde spre deosebire de cel precedent lumea este compusă din blocuri de pamant și iarba . Are aceleași obstacole pe care trebuie să le depășească. Diferența față de cel precedent va consta în înmulțirea șerpilor și apariția fiorosului Rino care îi va cauza probleme . Totodată la acest nivel cresc numărul recompenselor mai ales al diamantelor .
- ★ **Nivelul 3 :** La capătul acestui nivel Tom va găsi comoara și din acest motiv dificultatea nivelului este mai mare. Acțiunea se va muta preponderent la înălțime intrucât

recompensele vor fi la nivel înalt iar jos va crește numărul șerpilor . Cel mai periculos inamic își face aparitia avand capacitatea de ai lăua cate o viață la fiecare 5 secunde de contact, și pe acesta îl poate ucide cu pistolul. Structura fundamentală a nivelului este dată de blocurile de piatră și obstacolele amintite la celelalte nivale.

Descriere meniului :

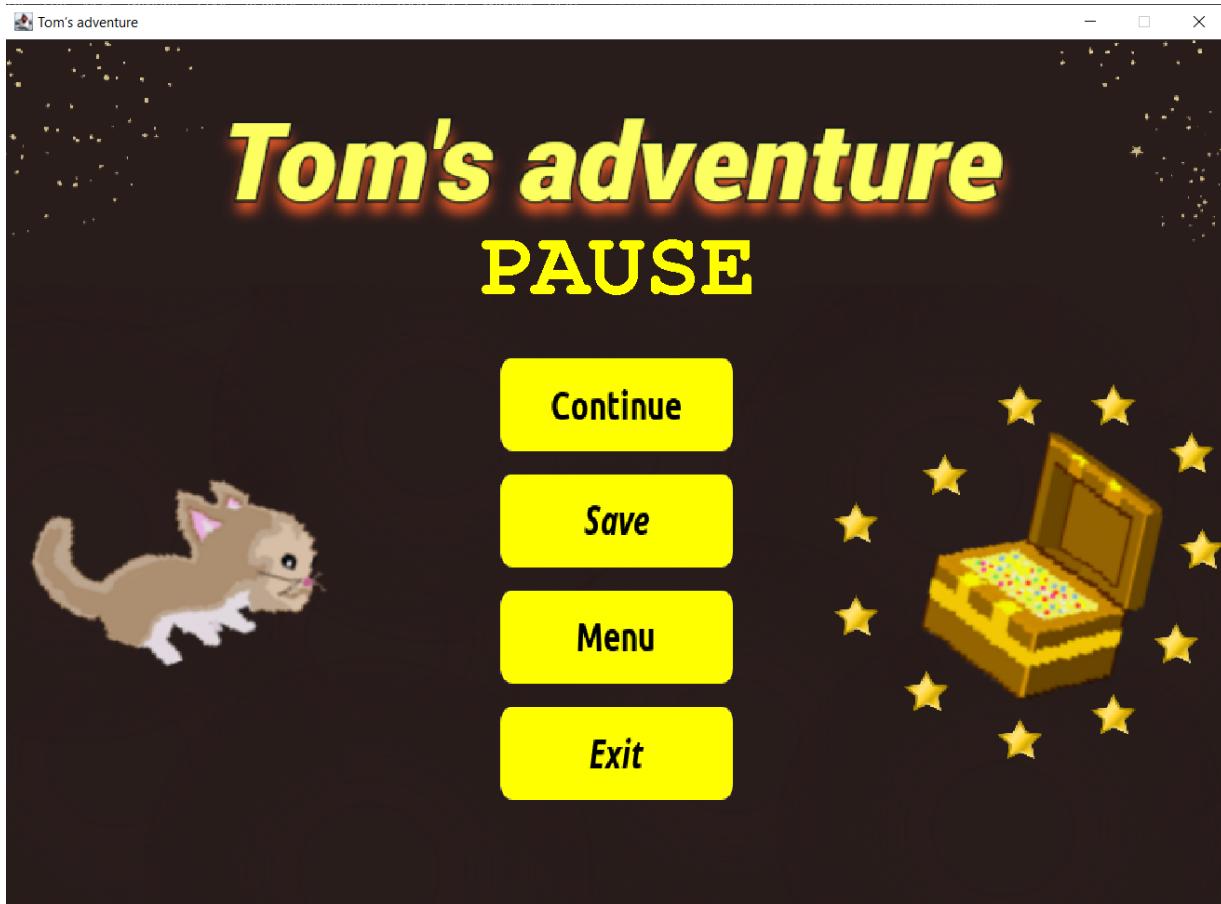
Meniul principal va cuprinde 4 butoane: new, load, information, quit ce vor fi selectate cu mouse-ul.

Butonul new va deschide un joc nou, load va încărca un joc salvat,information va afișa informații despre joc. La apăsarea butonului quit se va părăsi jocul .

Meniul principal are forma :



Dacă în timpul jocului se va apăsa butonul de pauza va apărea un meniu secundar (cu posibilitatea continuari jocului, întreruperi, salvari sesiuni actuale, revenirii la meniul principal) de forma:



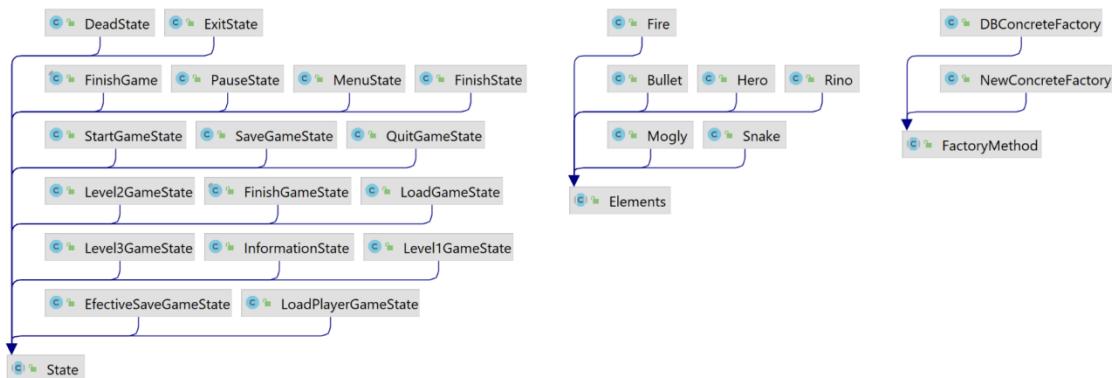
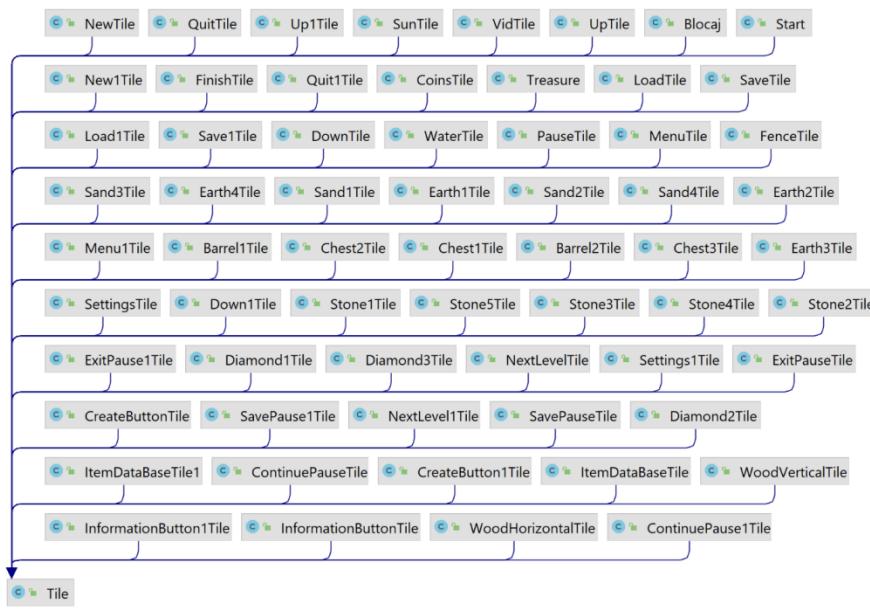
Diagrame de clase si descriere clase:

Obs : Diagramele se găsesc în format png(din cauza numărului mare rezolutia din document lasa de dorit) și în arhiva cu jocul. Din cauza numărului mare de clase(112) utilizate în joc se prezinta atât diagrama generală cât și diagrame la nivel de pachet. Descrierea claselor să făcă la prezentarea pachetului.

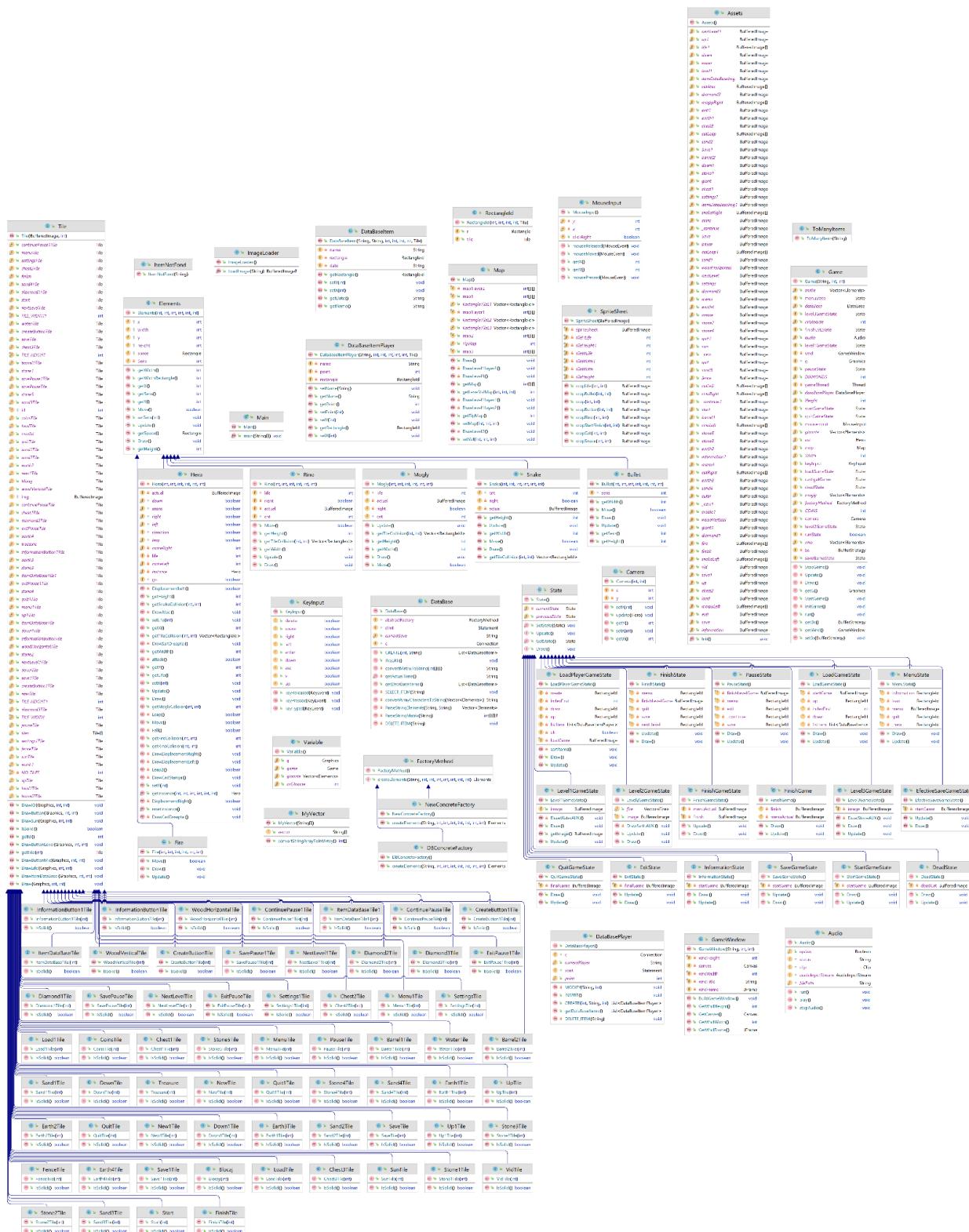
Diagrame generate cu IntelliJ Ultimate

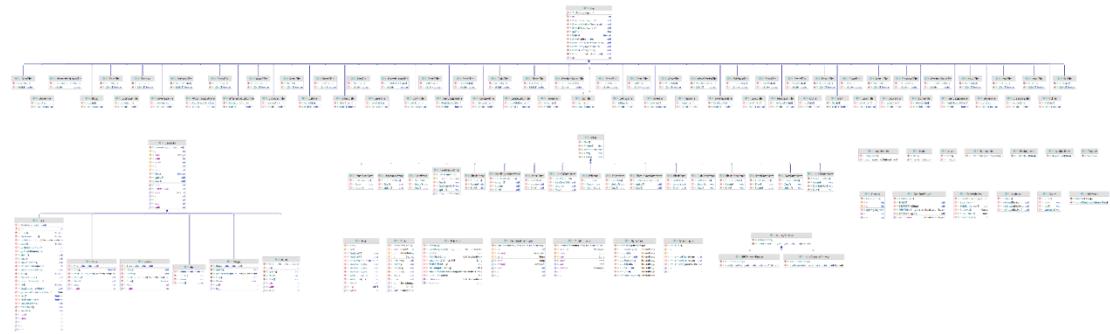
Diagrama generală

1. Diagrama de clase fără membri și metode([UMLToateFaraMembri.png](#)).

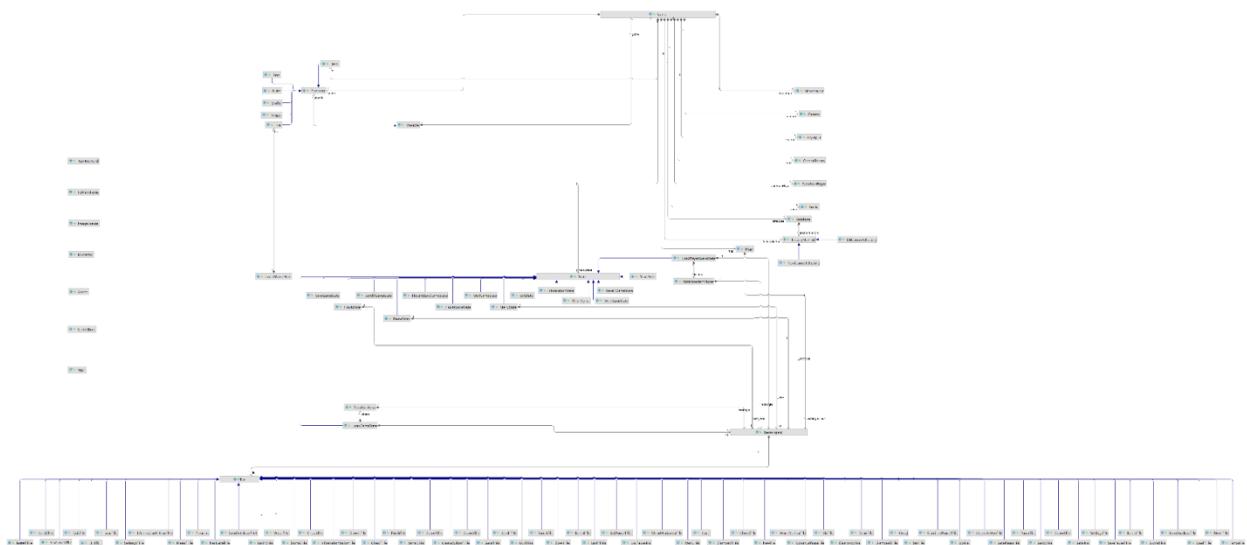


2. Diagrama UML generala cu membri si metode(UMLTota.png, UMLLiniar.png).

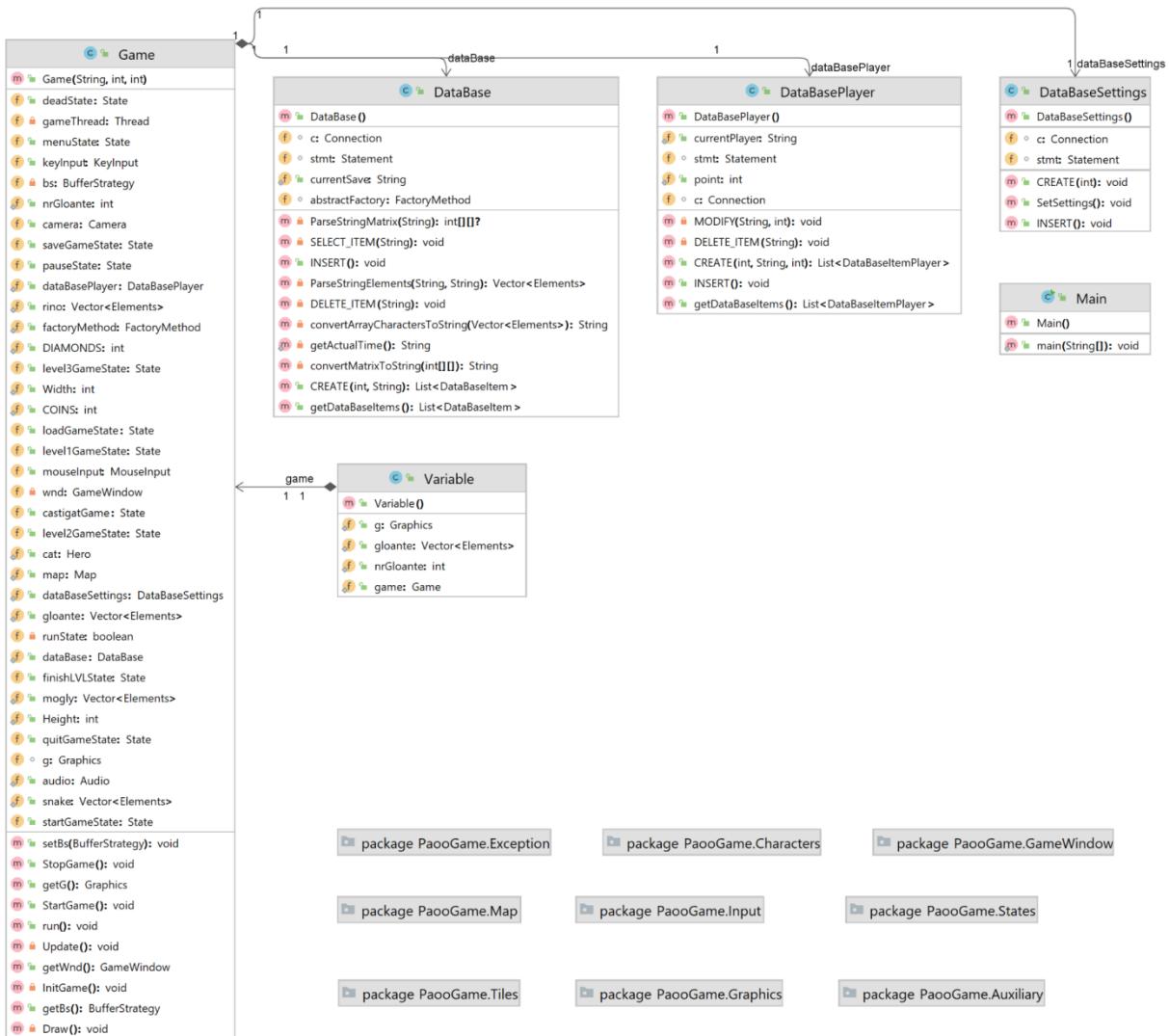




3. UML cu toate clasele cu dependențe ,fata membri([UMLNouCuDependente.png](#)) :



4.Diagrama UML cu pachete(packagePaoGame.png)



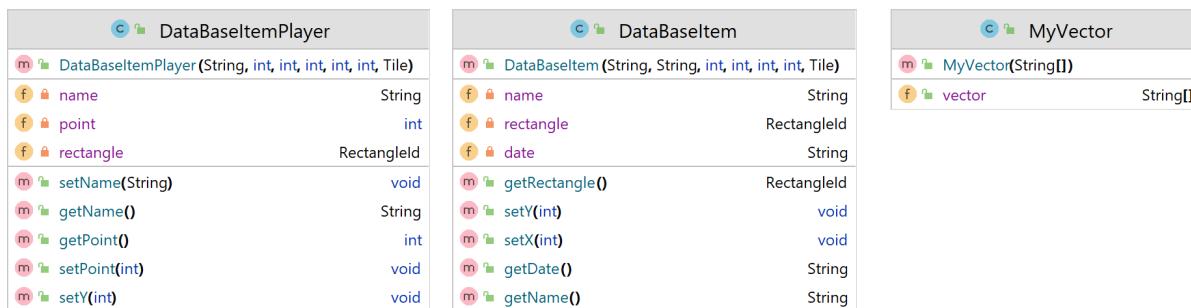
- **DataBase** - clasa va gestiona baza de date pentru salvarea unui nivel. Membrul `abstractFactory` va fi o referinta catre o fabrica concreta prin care se vor crea elemente active cu date extrase din baza de date. Referintele `c` și `stmt` sunt utilizate pentru conectarea la baza de date ce va fi salvata in memorie. Toate interacțiunile cu baza de date sunt făcute cu metoda `CREATE()` care va primi ca parametri un int ce da codul operației de făcut și un String pentru extragerea din baza de date a unei valori. Metodele private `INSERT()`, `SELECT_ITEM()`, `DELETE_ITEM()` vor fi apelate din funcția `CREATE()`. Celelalte metode vor implementa prelucrări asupra datelor.
- **DataBasePlayer** - va gestiona baza de date pentru salvarea jucatorilor. Membri `currentPlayer` și `point` rețin cu ce jucător lucrez. Semnificația celorlalte funcții/membri este aceeași cu clasa `DataBase` cu adăugarea funcției `Modify()`

care va actualiza pentru jucătorul curent numărul de puncte dacă el este mai mare decat cea ce am înregistrat în baza de date.

- **DataBaseSettings** - clasa de date ce va realiza salvarea setărilor în baza de date și restaurarea lor la deschiderea jocului.
- **Game** - clasa principală a jocului va conține referințe către setările jocului(pauseState,menuState,level2GameState etc), referinte catre dusmani(snake,reno,mogly), referinta catre pisica, date despre joc cum ar fi numărul de bani,diamante colectate. Alte referinte utile catre camera, fabrica de obiect, bazele de date, context grafic, dimensiuni fereastra, thread-ul în care se va face randarea.Intrucat clasa va implementa interfața Runnable se vor defini metodele StopGame() - oprire joc, Update() - se va apela update pentru starea curentă, Draw() se va apela tot pentru fiecare stare în parte. Metoda StartGame() va porni jocul și va fi apelata din main(). Metoda InitGame() va initializa un joc nou.
- **Main()** va crea o variabilă de tip Game și va porni jocul prin apelul metodei StartGame().
- **Variable** va reține referințe către cele mai utilizate obiecte din joc(mai ales referinta game către jocul curent).

Diagrame la nivel de pachet

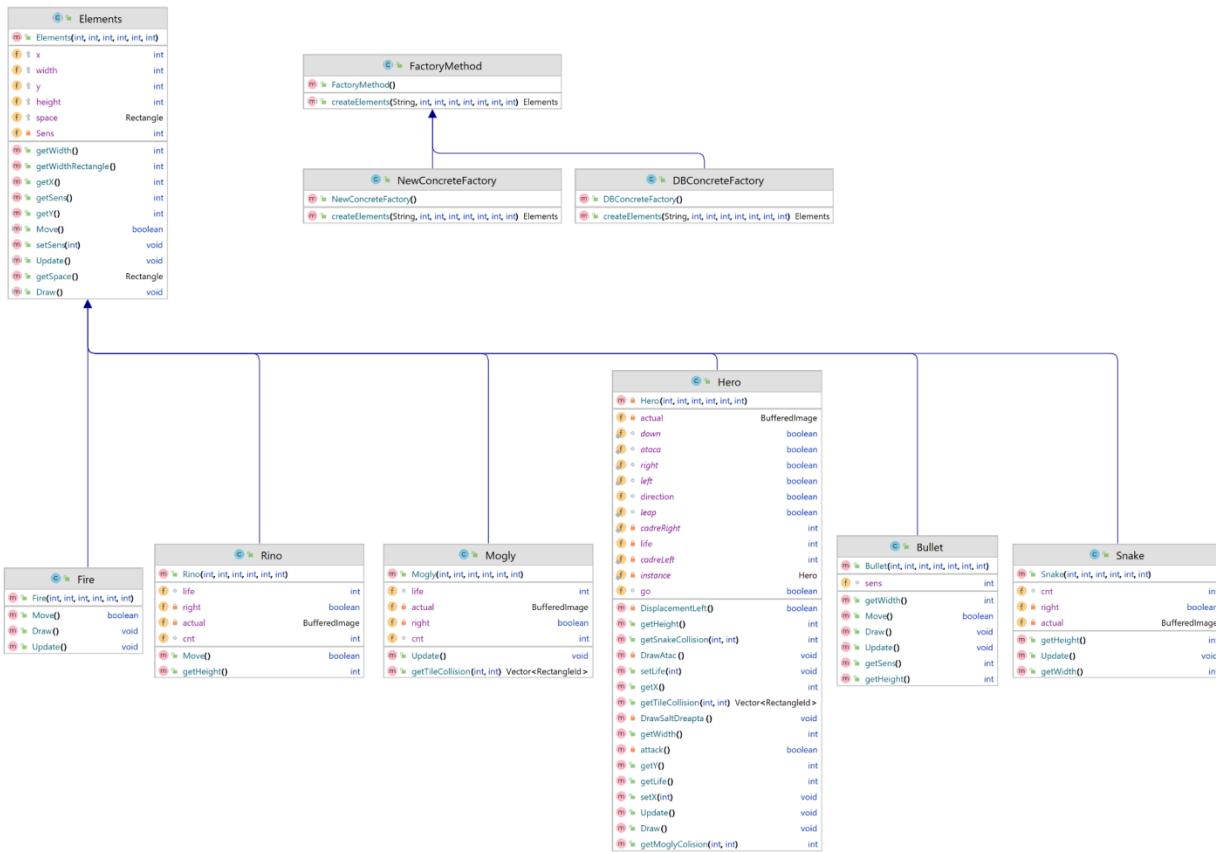
Pachetul Auxiliary([packageAuxiliary.png](#)):



- **Clasa DataBaseItemPlayer** reține datele necesare afisari(la selecția protagonistului) a unui jucător din : DataBasePlayer. Rectangle - coordonate pentru afișarea pe ecran a unui element , point - numărul de puncte(se afișează sortati jucători după acest camp), name(nume unic pentru jucător).
- **DataBaseItem** reține date necesare pentru afișarea unui nivel salvat. În afișare se vor preciza numele și data salvari. Rectangle reține poziția, dimensiunile și dacă este sau nu selectată o opțiune.

- **MyVector** are rolul de a converti vectori de string la int(salvarile în baza de date a vectorilor se fac ca string, astfel pentru restaurare este necesara conversia la int).

Pachetul Characters(packageCharacters.png):



- **Elements** - clasa abstractă ce va fi extinsă pentru a crea personajele(pisica,serpi,mamuti,bastinasi) și elementele active(foc,gloante). Contine membri de stare(poziții , dimensiuni). Cele mai importante metode : Update,Draw,Move vor fi suprascrisse în clasele derivate.
- **Hero** - clasă ce va defini pisica. În ea se definesc acțiunile pisici : mers, salt, atac. Totodată se definesc metodele de desenare și de determinare a coliziunilor. Pierderea jocului, castigarea unui nivel, colectarea de bani și diamante își găsesc implementarea aici odată cu mutarea personajului.
- **Snake,Rino,Mogly** - suprascriu metodele din clasa de bază definind miscările lor(stanga, dreapta), coliziunile care le împiedică deplasarea și totodată moartea inamicilor atunci cand numărul de veți ajunge la 0 .
- **Bullet** - definește conceptul de glonț(deplasarea glontului și dispariția lui la contactul cu elemente solide).
- **Fire** - definește conceptul de foc(se afișează imagini succesive pentru a da impresia de ardere,nu există mutări ale acestuia).

- **FactoryMethod** clasa abstracta ce va fi extinsa de **DBConcreteFactory** și **NewConcreteFactory** vor crea obiecte prin apelul constructorilor pentru snake,Rino,Mogli,Bullet etc. Metodele din clasele derivate vor primi ca parametru un string ce va identifica tipul obiectului de generat și parametri pentru constructor.

Pachetul Exception([packageException.png](#)):



- **ItemNotFound** - excepție pentru negasirea unui element în baza de date.
- **ToManyItems** - excepție pentru depășirea capacitatei de înregistrare în baza de date(la nivelul elementelor).
- **TypeNotFound** - excepție aruncată de fabrica de obiecte atunci când primește un tip de date ce nu are un obiect corespondent.

Pachetul GameWindow([packageGameWindow.png](#)):

Camera	
m	Camera(int, int)
f	x int
f	y int
m	setY(int) void
m	update(Hero) void
m	getY() int
m	setX(int) void
m	getX() int

GameWindow	
m	GameWindow(String, int, int)
f	wndHeight int
f	canvas Canvas
f	wndWidth int
f	wndTitle String
f	wndFrame JFrame
m	BuildGameWindow() void
m	GetWndHeight() int
m	GetCanvas() Canvas
m	GetWndWidth() int
m	GetWndFrame() JFrame

- **Camera** memorează datele necesare pentru conceptul de camera . Mutarea efectiva se va face prin apelarea metodei translate definită în [java.awt](#).
- **GameWindow** - creaza fereastra grafica.

Pachetul Graphics(packageGraphics.png):



- **SpriteSheet** - realizeaza decuparea din spriteSheet după mai multe dimensiuni pentru a avea imagini clare cele mai detaliate au o rezolutie mai crescută, sau cazuri particulare pentru obiecte nepatratice.

- **Assets** - incarca elementele grafice necesare jocului(fara background) se vor defini referințe spre imaginile ce vor fi tăiate și se va apela pentru decupare la metodele din SpriteSheet.
- **ImageLoader** - returneaza BufferedImage pentru imaginile date prin parth-uri.

Pachetul Input([packageInput.png](#)):

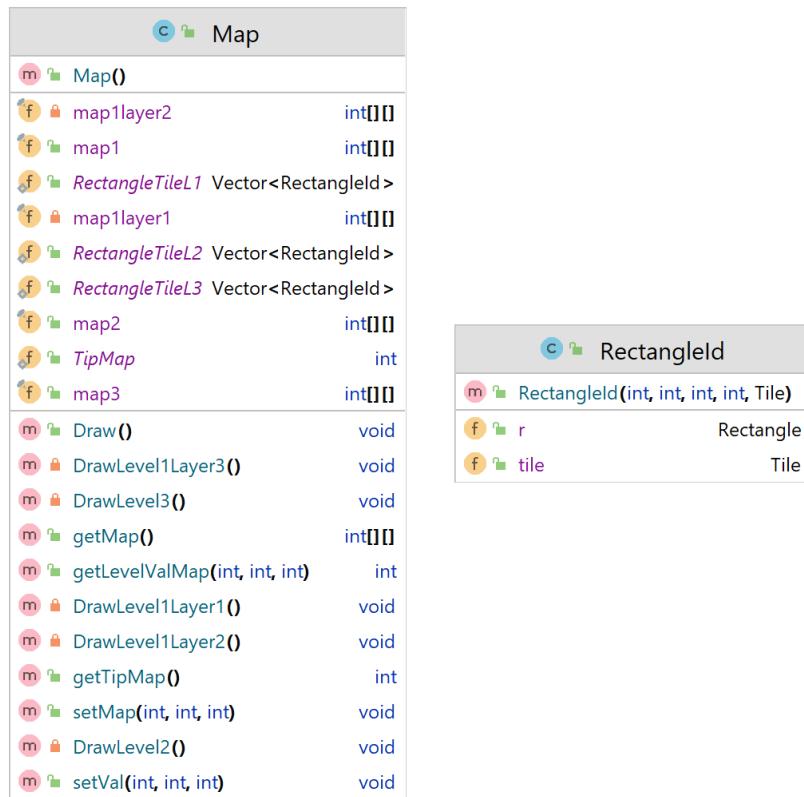
MouseInput		
m	MouseInput()	
f	y	int
f	x	int
f	clickRight	boolean
m	mouseReleased(MouseEvent)	void
m	mouseMoved(MouseEvent)	void
m	getX()	int
m	getY()	int
m	mousePressed(MouseEvent)	void

KeyInput		
m	KeyInput()	
f	delete	boolean
f	space	boolean
f	right	boolean
f	left	boolean
f	enter	boolean
f	down	boolean
f	esc	boolean
f	v	boolean
f	up	boolean
m	keyReleased(KeyEvent)	void
m	keyPressed(KeyEvent)	void
m	keyTyped(KeyEvent)	void

Audio		
m	Audio()	
f	option	Boolean
f	status	String
f	clip	Clip
f	audioInputStream	AudioInputStream
f	filePath	String
m	run()	void
m	play()	void
m	stopAudio()	void

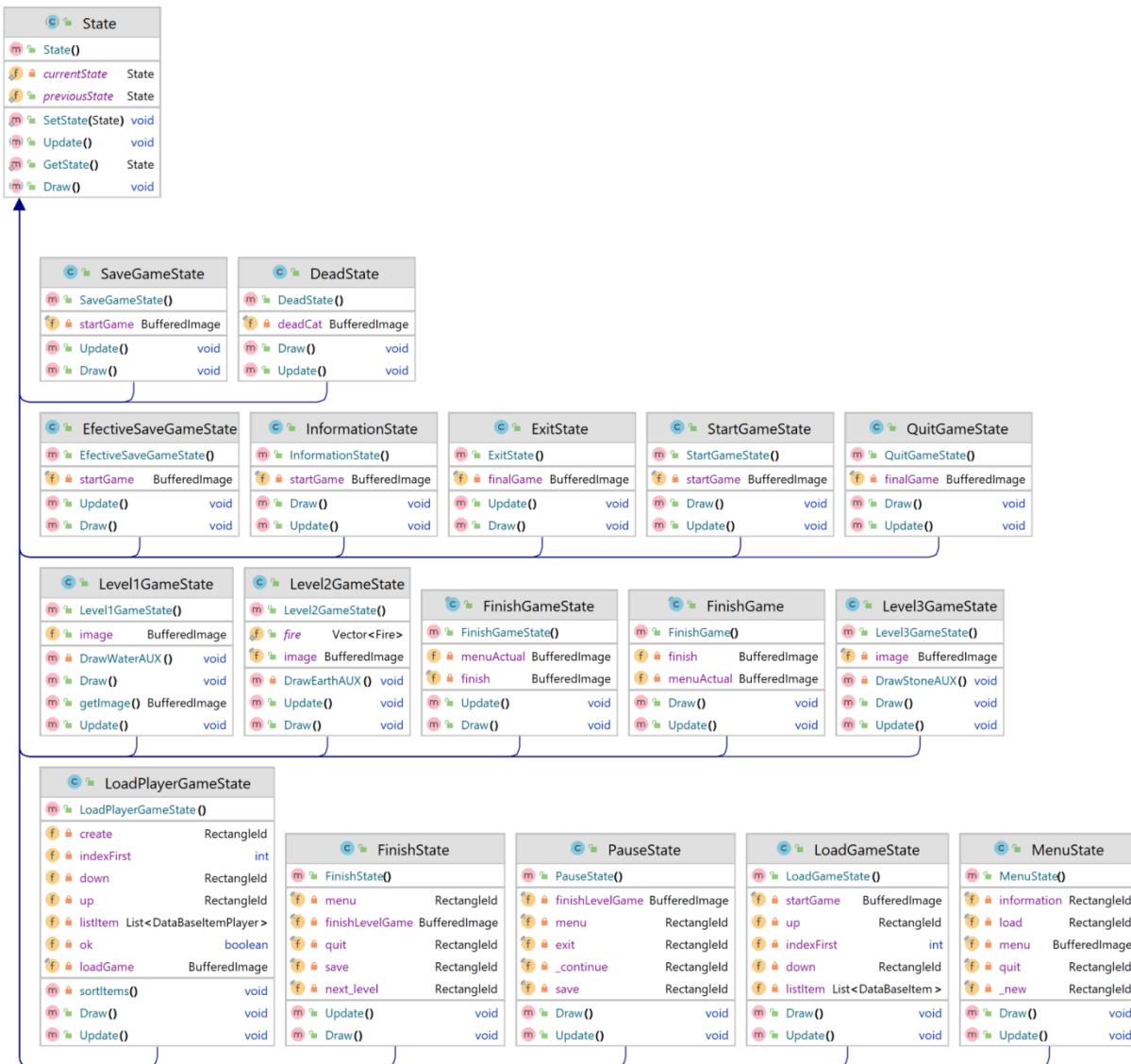
- **MouseInput** - se ocupă de gestionarea mouse-ului. Primește datele de la sistem despre evenimentele petrecute și memorează poziția și dacă a fost apelat click stanga. Metode de remarcat : mouseMoved(tratează mutarea mouse-ului), mousePressed si keyReleased(tratează apasarea click).
- **KeyInput** - reține apasarea tastelor utilizate în joc(enter, sageata stanga, sageata dreapta, săgeata sus, săgeata jos, space, v, delete).
Metode de remarcat : keyPressed(tratează apasare tasta - se setează flag-ul pentru tasta apasată), keyReleased(tratează eliberare tasta - se resetează flag-ul pentru tasta eliberată).
- **Audio** - clasa ce va gestiona sunetul de fundal al jocului. Membrul clip este o referință către clipul audio ce va fi pornit. Metodele play() si stopAudio() gestionează pornirea și oprirea melodiei.

Pachetul Map(packageMap.png):



- **Map** - reține matricele pentru hartii(nivelul 1 are 3 straturi), conține metode de gestionare a relațiilor cu harta : modificare harta, desenare harta. Totodată aceasta clasa crează și vectori pentru coliziuni(RectangleTileL1, RectangleTileL2, RectangleTileL3).
- **RectangleId** - clasa auxiliara folosită de map pentru a reține elemente din harta(lungime, latime, coordonata x, coordonata y și dala folosita: trebuie să stiu la coliziune ce dala am). Clasa a mai fost folosită și la meniuri(definire de butoane).

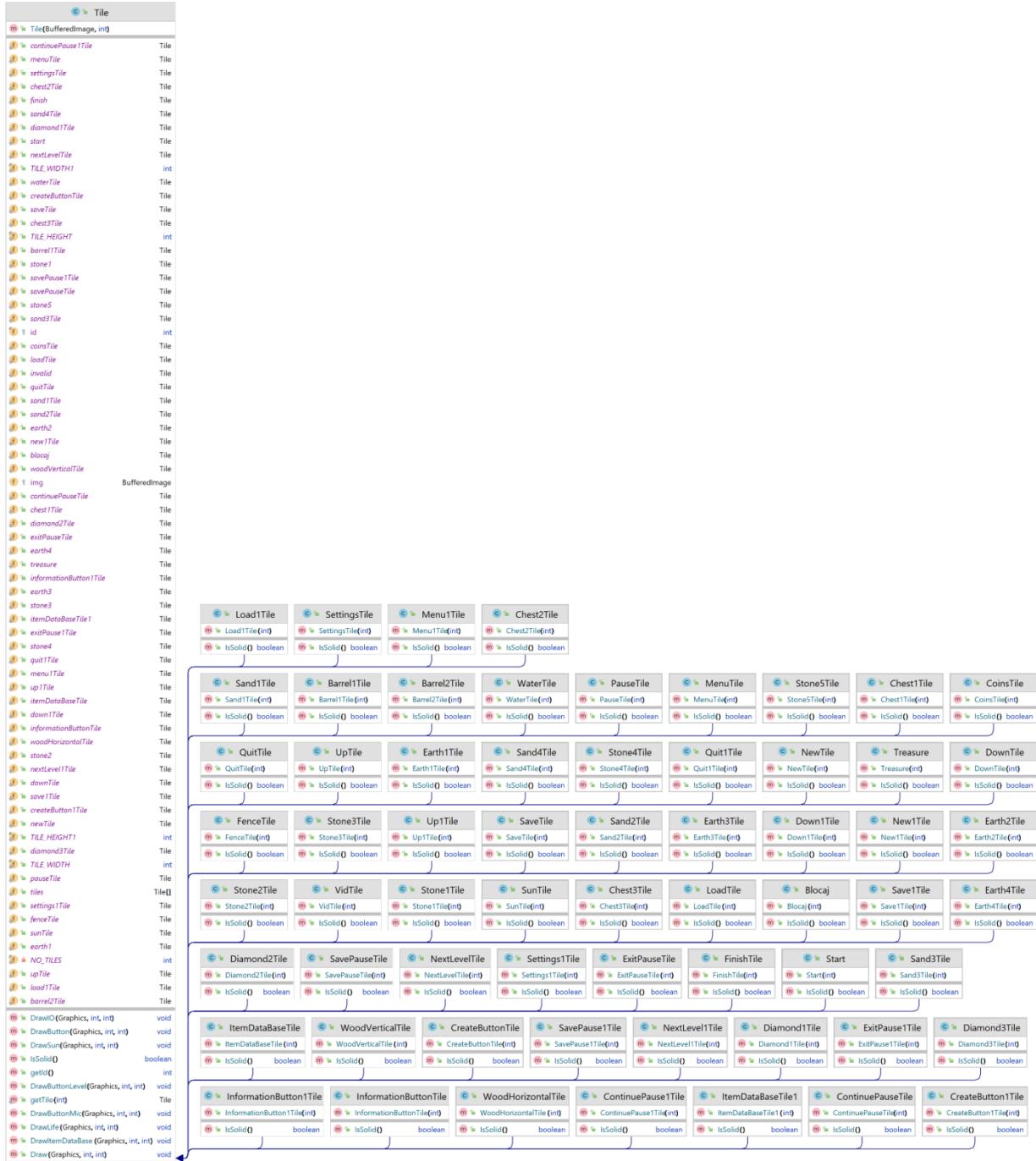
Pachetul States([packageStates.png](#)):



- **State** - clasa abstractă ce va fi extinsă de celelalte clase pentru stări. Are metode abstracte de `Update`(se schimba lucruri în majoritatea stărilor), `Draw`(se desenează starea curentă fiecare stare are altceva de desenat pentru ea).
- **SaveGameState** - în aceasta clasa se defineste stare în care se va sta după salvare așteptând ca utilizatorul să revină la joc. Metoda update nu va avea decat scopul tratari reveniri la joc, se va desena doar un fundal simplu în care se va afișa numele salvari.
- **DeadState** - metoda `Update` tratează doar apăsarea unui buton, iar `Draw` afișează o simplă imagine.

- **EfectiveSaveGameState** - Update apelează funcția pentru scrierea în baza de date ce va salva contextul actual al jocului și setarea variabilei ce da starea curentă necondiționat la SaveGameState.
- **InformationState** - pentru starea în care utilizatorul citește informații.
- **ExitState, QuitGameState** - definesc parasirea jocului.
- **FinishGameState** - gestionează starea de castigare a unui nivel.
- **FinishGame** - definește starea de castigare a jocului.
- **LoadPlayerGameState** - va implementa salvarea jucatorilor, aceasta clasa gestionează meniul de unde se poate alege sau crea un jucător. Metoda Update se ocupă de actualizarea la navigarea cu mouse-ul pe butoanele pentru stări, apăsarea butoanelor pentru derularea și apasarea delete pentru ștergerea unei înregistrări. Draw va desena în funcție de membrul RectangleId din DataBasePlayerItem. Membrul de remarcat este listItem ce contine o lista cu cele mai importante campuri ale elementelor din baza de date. Metoda sortItem() va realiza sortarea descrescătoare după scor a jucatorilor.
- **PauseState** - va gestiona starea de pauza. Metoda Update() va actualiza butoanele la mișcarea cu mouse-ul și va efectua tranziții la apăsarea butoanelor. Metoda Draw va desena fereastra.
- **LoadGameState** - va implementa starea de alegere a unei salvări ce va fi restaurată. Metoda Update() va reacționa la comenziile utilizatorului de alegere a unei salvări, de navigare printre salvări atunci când sunt mai multe și de ștergere. Metoda Draw() va desena grafica pentru efectuarea operațiilor enunțate mai sus.
- **MenuState** - va implementa starea de meniu. Metoda Update() va determina pe ce buton sunt eu cu mouse-ul și în cazul apăsării click dreapta pe acel buton va face acțiunea specifică (deschidere joc nou, încarcare nivel, afișare informații despre joc, parasire joc)
- **Level1GameState, Level2GameState, Level3GameState** - vor implementa cele trei niveluri ale jocului. Metoda Update() va apela funcțiile de Update pentru elementele active (pisica, serpi, mamuti, bastinas, focuri) și va gestiona apasarea tastei esc (caz în care se trece la meniul de pauza). Metoda Draw() va realiza desenarea nivelului curent prin desenarea background-ului, a numărului de banilor colectați, diamantelor, numărului de vieți, numărului de gloanțe și apelul funcțiilor Draw() pentru elementele active.

Pachetul Tile(packageTiles.png):



- Tile** clasa va reține obiecte statice ale claselor derivate ce vor fi utilizate pentru desenare. Fiecare tila este definită de un identificator unic atunci cand este necesar sau de unul general cand nu este relevant. Sunt mai multe metode draw...() care vor desena obiecte de mai multe dimensiuni.
- Celelalte clase** extind Tile și vor avea particular o alta imagine și o alta stare pentru coliziune(true sau false). Utilizarea lor poate fi dedusa din denumire.

Sabloane(modele) de proiectare utilizate:

- A. **Factory Method** - sa folositi o fabrica de obiecte pentru a crea inamici și gloanțele în două forme una pentru crearea la lansarea unui nivel nou și una pentru crearea la încărcarea din baza de date. În acest scop au fost definite clasele **DBConcreteFactory**, **NewConcreteFactory** și clasa abstractă **FactoryMethod**. Obiectele create au fost din clasele Snake, Bullet, Rino și Mogly ce extind clasa abstractă Elements.
- B. **State** - folosit pentru implementarea legăturilor între ferestre. În acest scop a fost definită clasa abstractă State și extinsă de următoarele clase:
 - DeadState - pentru pierderea jocului(pisica moare),
 - EfectiveSaveGameState - stare în care apare numele savari,
 - ExitState - stare în care se sta 2 secunde înainte de parasirea jocului,
 - FinishGameState - stare în care se ajunge cand se castiga jocul,
 - FinishState - starea de castigare a unui nivel,
 - InformationState - starea pentru informații,
 - Level1GameState - starea pentru nivelul 1,
 - Level2GameState - starea pentru nivelul 2,
 - Level3GameState - starea pentru nivelul 3,
 - LoadGameState - starea de încărcare a unei salvari,
 - MenuState - starea de meniu,
 - PauseState - starea de pauza,
 - QuitGameState - starea de ieșire din joc,
 - SaveGameState - starea de salvare a jocului,
 - StartGameState - starea de început a jocului.
- C. **Singleton** - pisica trebuie să fie unică dar crearea ei nu se face în clasa Game, deoarece țin de conceptul de nivel. Nivelele nu au o ordine de încărcare(se poate incarca din baza de date) deci voi crea doar o pisica în joc, aceeași pentru toate nivelurile folosind Singleton(clasa Hero are constructor privat, o referinta la un element erou și metoda getInstance).

Structura și implementarea bazelor de date:

- A. **Baza de date pentru jucători** are doi membri numele jucătorului(**NAME**) și numărul de puncte obținute de el (**POINT**) în aceasta baza de date se vor salva pentru fiecare jucător numărul de puncte. Datele vor fi afișate în meniul de selecție al personajului unde vor fi sortate descrescător după numărul de puncte. Pe masura ce respectivul jucător va parcurge nivelele se va actualiza numărul de puncte dacă e mai mare decât cel actual din baza de date.Numărul de puncte = $2 \times$ numărul de diamante + numărul de banuti. De asemenea în ferestrele de load se pot șterge înregistrări prin apasarea delete.

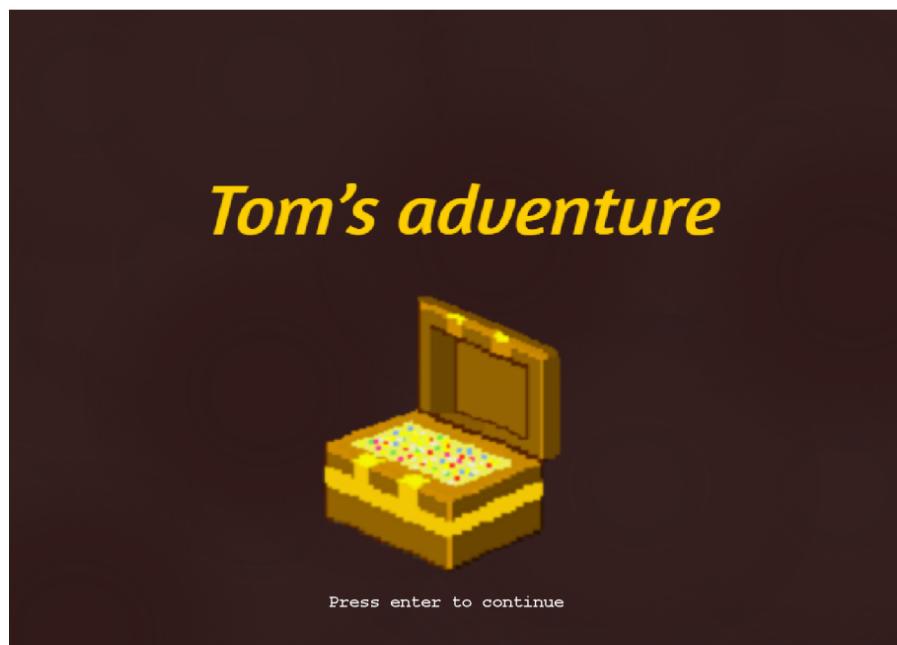
Table: GameDataBasePlayer

	NAME	POINT
	Filter	Filter
1	Player1	57
2	Player7	11114
3	Player8	68
4	Player9	0
5	Player10	0
6	Player11	57
7	Player12	0
8	Player13	0

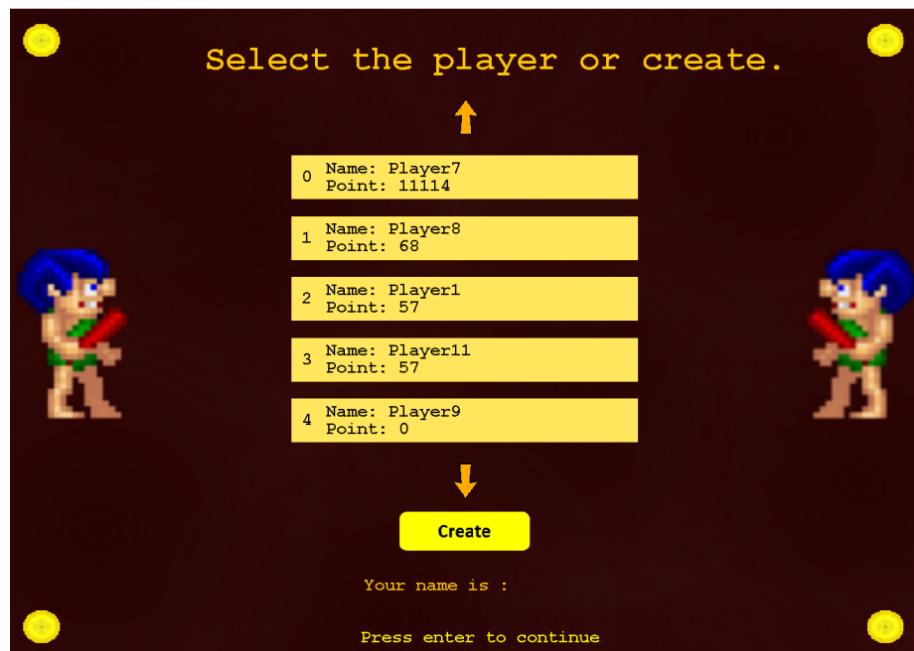
B. **Baza de date pentru salvarea nivelului** gestionează operațiunile de **load** și **save** a sesiunilor de joc. Datele vor fi salvate ca int sau String - convertite din vector , matrice in string și analizate după pentru a le converti înapoi. Baza de date oferă operații de insertie, încărcare a unei înregistrări, afișare a înregistrărilor actuale. Pentru a se putea avea nume unice să utilizat un fișier în care să salvează numărul de jucători ce a fost actualizat când să adăugat unul nou. De asemenea în ferestrele de load se pot șterge înregistrări prin apasarea delete. Membri sunt:

- NAME - numele unei înregistrări,
- TIME - momentul de timp actual ca String(data și ora),
- NUMBER_LEVEL - numarul nivelului,
- CAT_X - coordonata x a pisici,
- CAT_Y - coordonata y a pisici,
- COINS - numărul de banii colectați,
- DIAMOND - numărul de diamante colectate,
- LIFE - numărul de vieți avute,
- BULLET - String ce reține date pentru a reface gloanțele trase(număr,dimensiune,coordonate),
- SNAKE - String ce reține date pentru a reface serpi din joc(număr,dimensiune,coordonate),
- MOGLY - String ce reține date pentru a reface bastinasi din joc(număr,dimensiune,coordonate,număr de vieți),
- RINO - String ce reține date pentru a reface mamuti din joc(număr,dimensiune,coordonate,număr de vieți),
- MAP - String ce retine harta convertita la String(se creaza un String cu '-' separa doua elemente pe coloana si '|' separa doua linii).
- NUMBER_BULLET - reține numărul de gloanțe pe care personajul le mai are

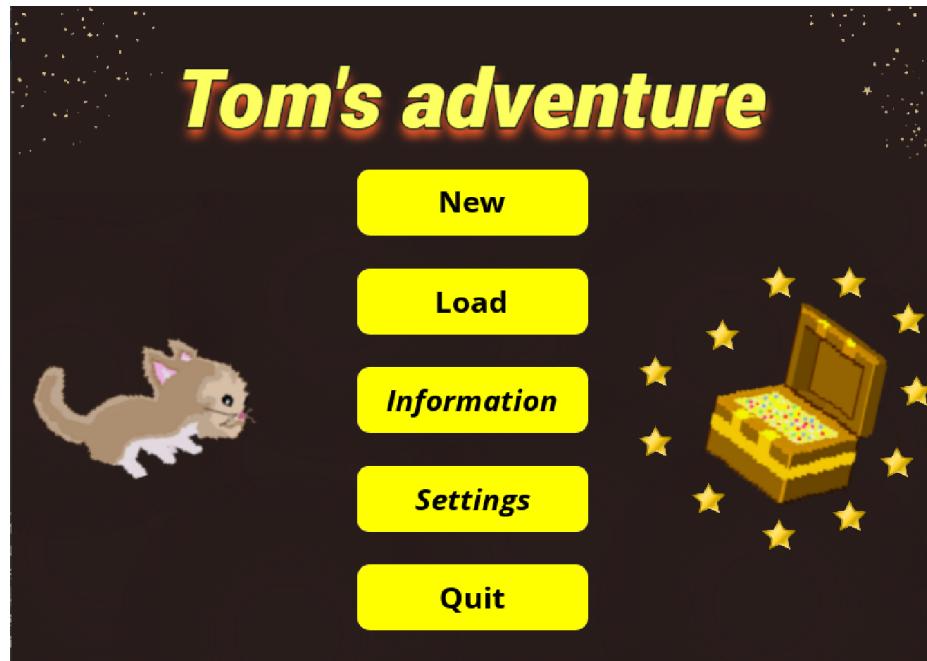
Start:



Selectare jucator:



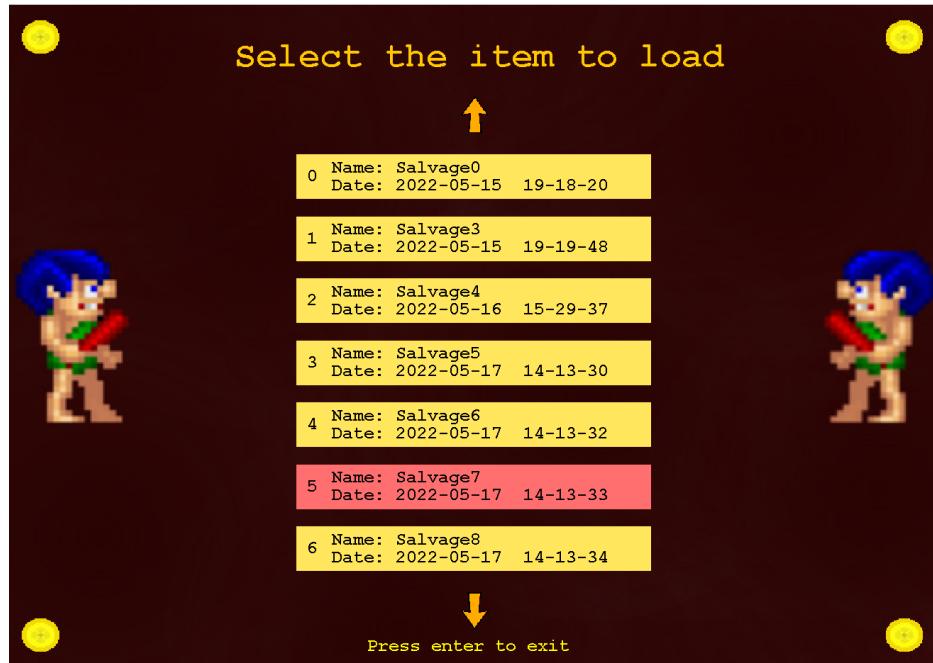
Meniu:



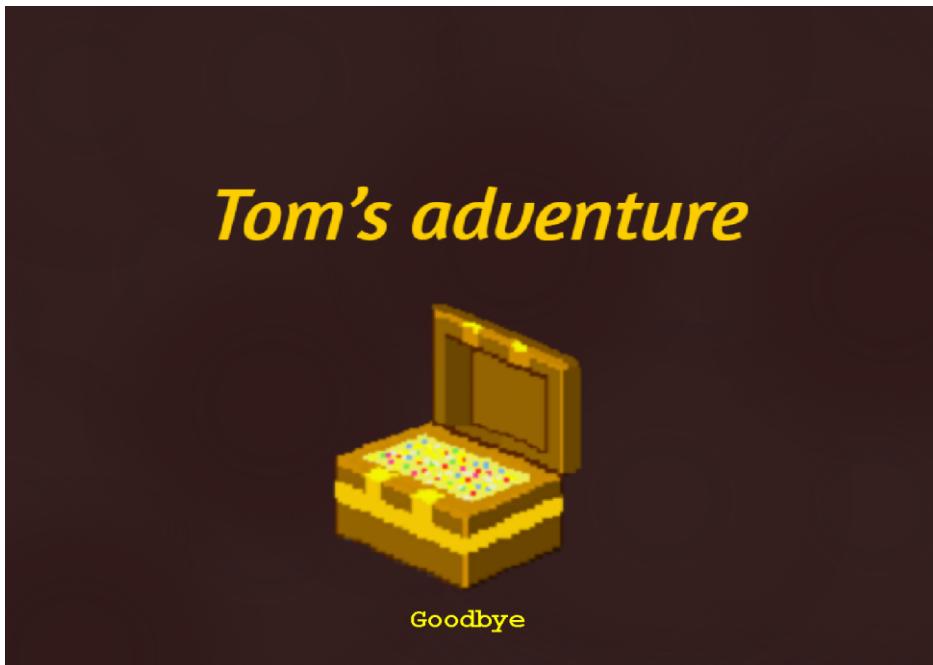
Informatii:



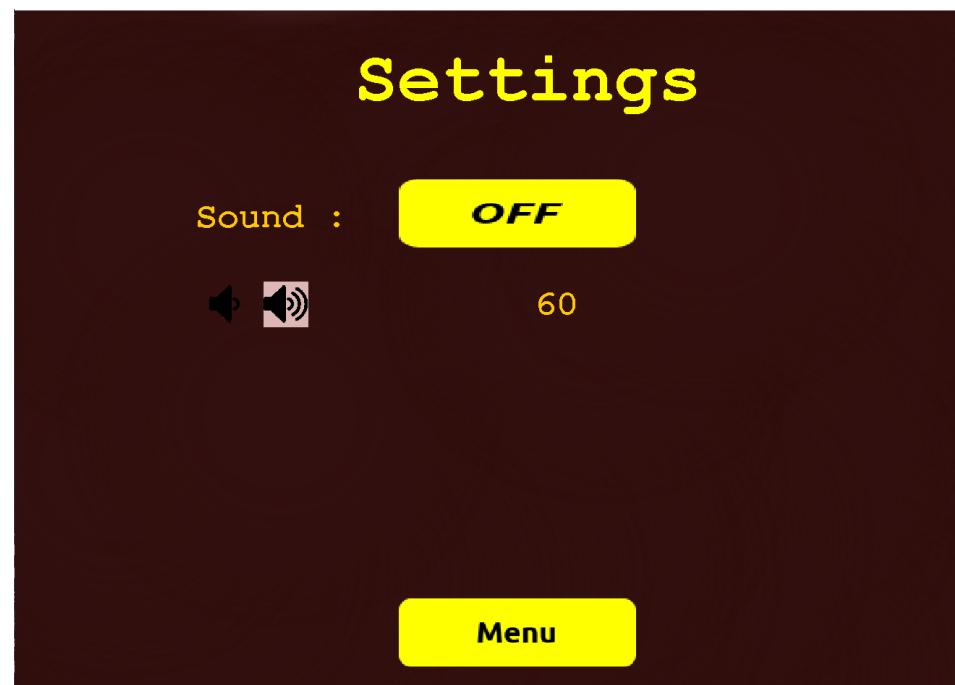
Incarcare salvare:



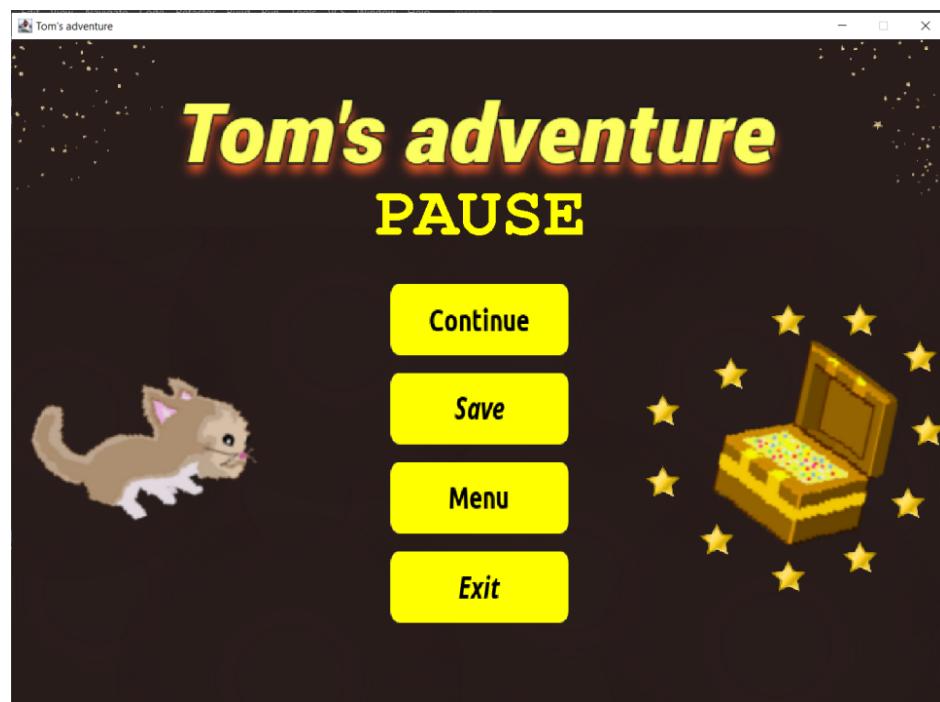
Parasire joc:



Setari :



Pauza:



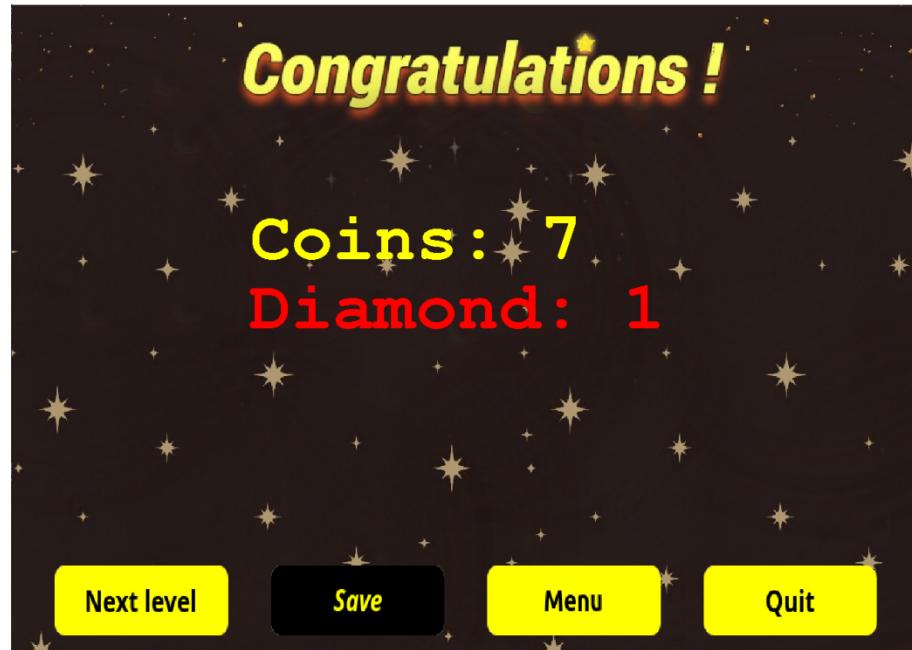
Salvare joc:



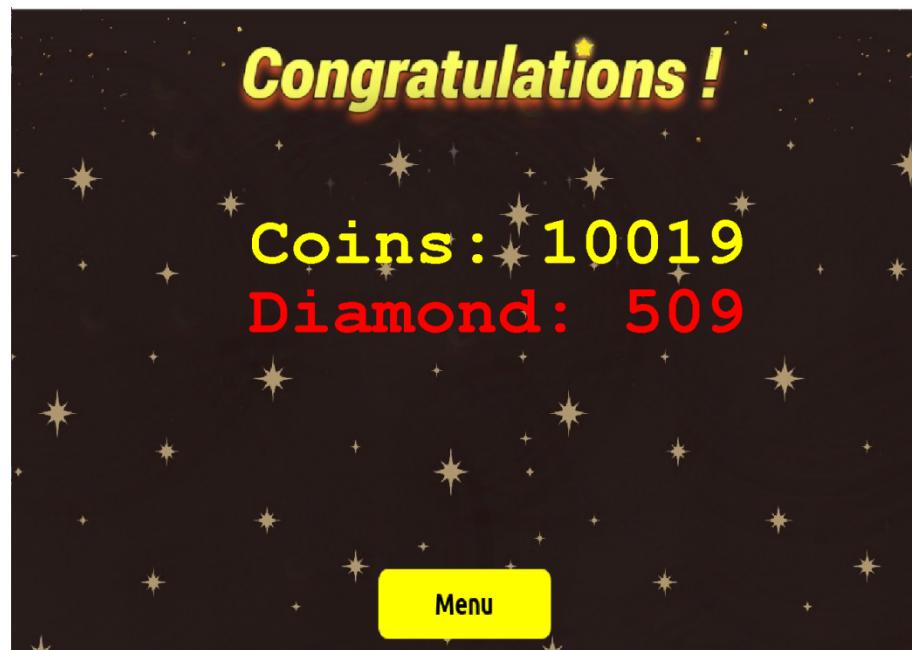
Pierdere joc:



Castigare nivel:



Castigare joc:



Documentatia Doxygen:

Documentația generată cu Doxygen este disponibilă în arhiva cu jocul în fișierul html(se va accesa index.html pentru deschiderea site-ului generat).

Bibliografie :

- Imaginele pentru inamici au fost luate din arhiva disponibilă pe site-ul:
<https://www.widgetworx.com/spritelib/>
- Background-ul pentru nivelul 3 a fost realizat după cel disponibil pe:
<https://lil-cthulhu.itch.io/pixel-art-cave-background>
- Dalele pentru nivelul 3 au fost preluate de
pe:<https://www.shutterstock.com/id/image-vector/2d-game-tileset-platformer-tile-set-1103732150>