Save the Inmate

Varvara Ioan, grupa 1207A

Facultatea de Automatica si Calculatoare, Iasi

Email: ioan.varvara@student.tuiasi.ro

Proiect PAOO

Contextul jocului:

Acțiunea jocului se situează în laboratorul doctorului Aldini, care a răpit un copil și l-a închis într-o cameră secretă a laboratorului său. Detectivul Holmes, care este și personajul principal al acestui joc, preia acest caz, iar după câteva săptămâni de încercări, echipa sa nu reușește în niciun fel să-l salveze pe copil, iar astfel pleacă singur în laboratorul doctorului pentru a-l salva. Ajuns în laborator, el trebuie să găsească copilul, iar pentru a realiza acest lucru trebuie să treacă de paznicii laboratorului care au diferite puteri.

Sistemul jocului:

Acesta este un joc tile-based, cu vedere de sus, în care jucătorul parcurge laboratorul pe parcursul a 3 niveluri de dificultate diferită, ajungând în finalul ultimului nivel să salveze deținutul, acesta fiind și scopul principal al jocului. În fiecare nivel vor fi inamici de care eroul trebuie să treacă pentru a-și îndeplini obiectivul. Mișcarea eroului se va face folosind tastele W(sus), A(stânga), S(jos), D(dreapta), de la tastatură. Pe parcursul deplasării sale, eroul va găsi de-a lungul hărții anumite obiecte de care se va putea folosi pentru a trece de inamici; de exemplu, pe parcursul hărții se vor găsi săgeți, pe care jucătorul le va putea utiliza folosind tasta F, iar lovind inamicii cu săgeți, îi va ameti pentru câteva secunde, moment în care va putea trece pe lângă ei fără a fi detectat. Dacă se află la o anumită distanță față de inamic, eroul va putea fi detectat, iar inamicul va începe să-l urmărească până când eroul va fi la o distanță destul de mare să nu mai fie văzut. La nivelele superioare, vor exista inamici care vor putea trimite un glob către erou. Dacă

eroul se va afla în coliziune cu unul dintre inamici, jocul va fi pierdut, iar jucătorul va trebui să reia nivelul de la început prin apăsarea tastei R, iar daca eroul va fi lovit de globul inamicului, acesta va ameti pentru cateva secunde moment in care poate fi prins de inamici daca acestia se afla in apopiere. Apăsând tasta SPACE eroul va putea efectua un "dash" (se va mișca mult mai rapid într-o direcție anume), lucru ce îl va ajuta să depășească mai ușor inamicii, însă va fi nevoie de o anumită perioadă de timp înainte să folosească "dash"-ul din nou. Pe parcursul nivelelor, în anumite locuri din hartă vor fi computere care vor afișa mesaje indicatoare pentru jucător, pentru a ști ce să facă mai departe. Pentru a câștiga nivelul, jucătorul trebuie să ajungă la portalul spre nivelul următor care se află undeva pe hartă, acestea trebuie să fie găsite. Jucatorul va putea castiga puncte astfel: 50 de puncta pentru fiecare sageata luata de pe jos, 100 de puncta pentru fiecare lovitura cu sageata in inamici, iar va pierde cate 10 puncte de fiecare data cand se va lovi de un laser sau va fi lovit de globul inamicului alb.

Conținutul jocului:

Personajul principal al jocului este detectivul Holmes. Pe parcursul misiunii sale acesta se va înfrunta cu următorii inamici:

- Inamicul galben: acesta nu are nicio putere, însă dacă eroul se va afla la o anumită distanță față de acesta, inamicul va intra într-o stare de alertă și va alerga după erou, până când acesta din urmă va reuși să se afle la o distanță suficient de mare pentru a nu mai fi detectat.

- Inamicul alb: acesta este capabil să arunce cu un glob după erou, glob care îl va ameți și va fi foarte vulnerabil și ușor de capturat de către inamicul galben; inamicul alb nu se poate mișca pe hartă.

- Inamicul rosu: acesta este inamicul care putea sa urmareasca eroul doar in momentul cand este in aria sa vizuala, in restul timpului nu va putea sa se miste pe harta.

Jocul este structurat în 3 nivele de dificultate diferită.

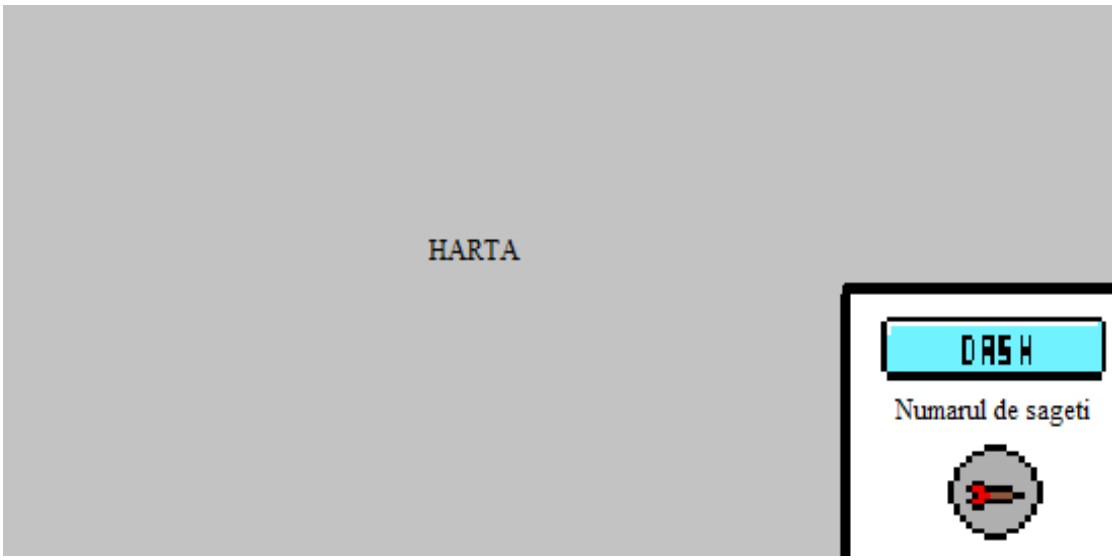
- Nivelul 1: în acest nivel vor exista numai inamici galbeni, într-un număr relativ mic; este un nivel care va avea și rolul de a-l face pe jucător să înțeleagă ideea jocului.

- Nivelul 2: în acest nivel, pe lângă inamicul galben, își va face prezența și inamicul alb; numărul inamicilor galbeni va crește, făcând misiunea eroului mai grea.

- Nivelul 3: acest nivel este asemănător nivelului 2, însă inamicii sunt mult mai abundenti, iar vizibilitatea jucatorului va fi redusa, fiind capabil sa vada doar cateva tile-uri, in jurul lui creandu-se practic un cerc luminos.

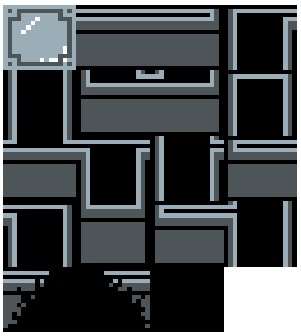
Dupa incheierea celui de al 3-lea nivel, jucatorul va fi trimis prin teleporter catre etaul unde se afla prizonierul pe care trebuie sa il salveze, si astfel castiga jocul.

Interfața jocului va fi una simplă, va informa jucătorul când va fi posibilă utilizarea "dash"-ului, și de asemenea numărul de săgeți pe care îl are la dispoziție pentru a le folosi, după ce a reușit să le adune de pe hartă.

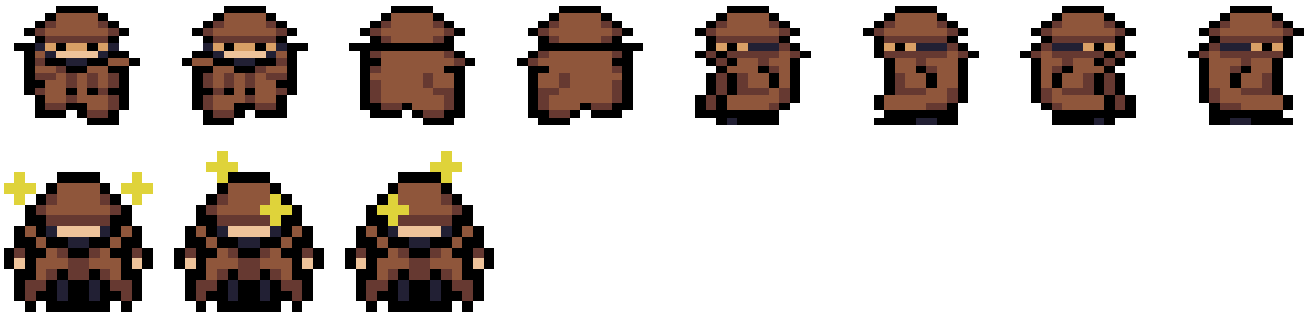


Sprite-uri folosite:

- Sprite pentru crearea hartii:

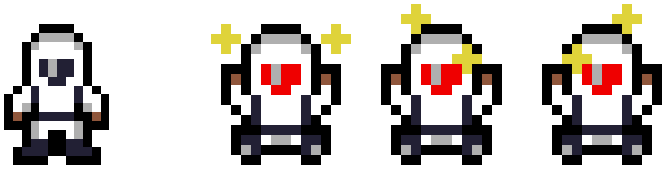


- Sprite folosit pentru personajul principal:

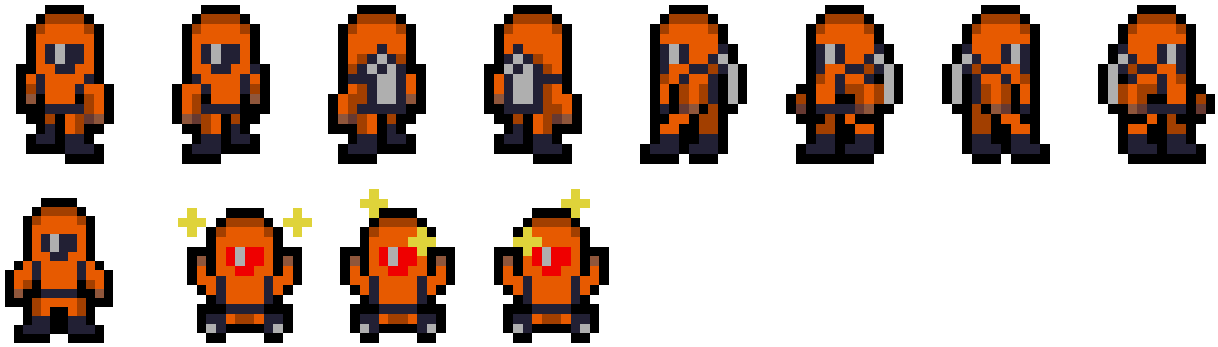


- Sprite folosit pentru inamicul galben:

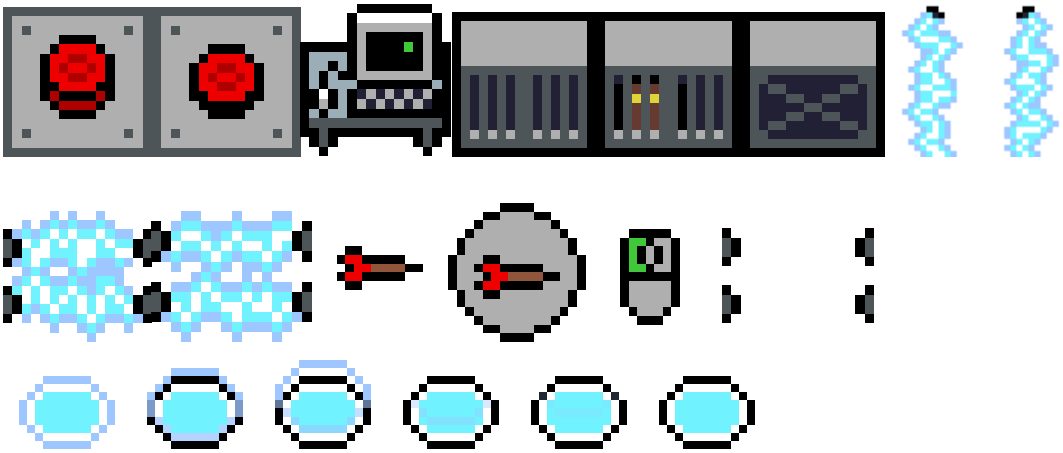
- Sprite folosit pentru inamicul alb:



- Sprite folosit pentru inamicul rosu:

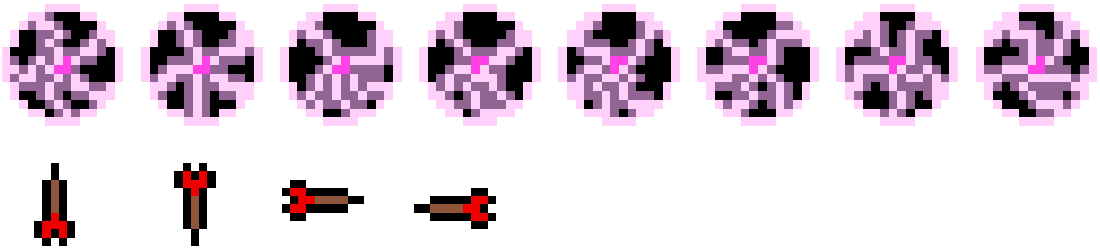


- Sprite-uri folosite pentru obiecte interactive si decoruri pe harta:



- Sprite folosit pentru Meniul Principal:

- Sprite-uri folosite pentru globul inamicului alb si sageata pe care o arunca eroul:



- Sprite folosit pentru prizonierul ce trebuie salvat:

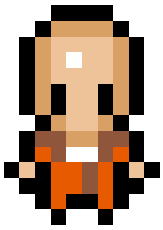


Diagrama UML:

Descriere clase din digrama UML :

Clasa **GamePanel** reprezintă este o subclasă a clasei JPanel și implementează interfața Runnable, ceea ce înseamnă că poate fi executată într-un fir de execuție separat. Iată o explicație pe scurt a acestei clase:

- Clasa face parte din pachetul Main.
- Importurile de clase necesare sunt prezente în partea de sus a codului.
- Clasa extinde clasa JPanel, ceea ce înseamnă că este o componentă grafică care poate fi adăugată într-o interfață utilizator.
- Clasa implementează interfața Runnable, ceea ce înseamnă că are o metodă run() care va fi executată într-un fir de execuție separat. Aici se realizeaza GameLoop-ul.
- Clasa conține o serie de variabile membru care reprezintă setări pentru ecran, harta și alte aspecte ale jocului.
- Există și metode pentru inițializarea jocului, repornirea jocului, setarea modului ecran complet, începerea firului de execuție al jocului și metode pentru actualizarea și desenarea jocului.
- Clasa conține obiecte și liste de obiecte care reprezintă jucătorul, obiectele din joc, inamicii și proiectilele.
- Există o variabilă gameState care stochează starea curentă a jocului (de exemplu, ecranul de titlu, jocul în sine, pauza etc.).
- Metoda update() este responsabilă de actualizarea stării jocului în funcție de starea curentă.
- Metoda drawToTempScreen() desenează elementele jocului pe un obiect BufferedImage, pentru a putea juca jocul in full-screen. Astfel desenam orice obiect din joc intr-un BufferedIamge.
- Metoda drawToScreen() desenează imaginea din BufferedImage pe ecran.

Clasa **KeyHandler** este o clasă care implementează interfața KeyListener și este responsabilă de gestionarea evenimentelor de tastatură în cadrul jocului. Clasa KeyHandler stochează starea tastelor relevante pentru joc și le actualizează în funcție de evenimentele de tastatură.

Clasa **UI** este o clasa responsabila pentru creearea HUD-ului jocului in diferite stagii ale jocului(titleState, playState, dialogueState, etc.). De asemenea, in aceasta clasa se calculeaza si scorul pe care il va vea jucatorul la sfarsitul fiecarui nivel, astfel realizeaza o conexiune cu baza de date SQLite pentru a salva aceste scoruri intr-un table.

Clasa **CollisionChecker** este responsabilă de verificarea coliziunilor într-un joc. Are mai multe metode pentru a verifica dacă obiectele din joc se ciocnesc între ele.

- Metoda **checkTile** verifică dacă un obiect se ciocnește cu dalele hărții. Verifică în ce direcție se mișcă obiectul și verifică dacă dalele în jurul obiectului sunt colizibile.
- Metoda **checkObject** verifică coliziunile dintre un obiect și alte obiecte din joc. Parcurge toate obiectele din harta curentă și verifică dacă obiectul se ciocnește cu unul dintre ele.
- Metoda **checkEntity** verifică coliziunile dintre un obiect și alte entități din joc. Parcurge toate entitățile din harta curentă și verifică dacă obiectul se ciocnește cu o altă entitate.
- Metoda **checkPlayer** verifică dacă un obiect se ciocnește cu jucătorul. Verifică în ce direcție se mișcă obiectul și verifică dacă se ciocnește cu jucătorul.

Pentru fiecare verificare de coliziune, dacă se detectează o coliziune, se setează o variabilă numită **collisionOn** a obiectului la **true**. În cazul metodei **checkObject** cu parametrul **player** setat la **true**, se returnează indexul obiectului cu care s-a produs coliziunea.

Clasa **EventHandler** este responsabilă de gestionarea evenimentelor în joc. Aceasta are câteva variabile membre și metode pentru a trata diferite interacțiuni în joc.

Clasa **AsserSetter** este o clasa din pachetul Main in care populam vectorii de inamici si obiecte, asezandu-i pe harta, in functie de nivel.

Clasa **Config** este o clasa din pachhetul Main in care se salveaza intr-un fisier setarea de Full-Screen din meniul de setari al jocului.

Clasa **Main** este punctul de intrare în aplicație pentru un joc. Implementează un design pattern Singleton pentru a permite existența unei singure instanțe a clasei. Constructorul privat inițializează fereastra jocului și panelul de joc. Metoda statică

"getInstance" returnează instanța existentă sau creează una nouă. Metoda "main" apelează "getInstance" pentru a rula jocul.

Clasa **TileManager** asigura gestionarea și desenarea tile-urilor în cadrul jocului. Constructorul primește un obiect GamePanel și inițializează variabilele necesare pentru gestionarea tile-urilor și hartilor. Metoda "getTileImage" încarcă imaginile tile-urilor dintr-un fișier și le asociază cu obiectele Tile corespunzătoare. Metoda "loadMap" încarcă hărțile din fișiere text și le stochează într-o structură de date. Metoda "draw" desenează tile-urile vizibile pe ecran. Metoda "generateCollisionMap" generează o hartă de coliziuni bazată pe harta curentă a tile-urilor.

Clasa **TileManagerException** reprezinta o clasa de exceptie folosita pentru a prinde si trata exceptii la incarcarea tile-urilor si hartilor in joc.

Clasa **SuperObject** este o clasa abstracta care servește ca bază pentru alte clase de obiecte din joc. Ea conține câmpuri și metode generice pentru desenarea și manipularea obiectelor din joc. Clasele pentru care aceasta clasa este de baza sunt : **Arrow, Button, Computer, DESIGN_Box1, DESIGN_Box2, Laser, LaserButtons, Teleporter.** Clasa **SuperObject** impreuna cu clasa **SuperObjectDesign** si enum-ul **Objects**, realizeaza Factory Design Pattern.

Pachetul **Environment** contine clasele **EnvironmentManager** si **Lighting,** aceste clase fiind responsabile pentru lumina in joc, de asemenea creeaza cercul necesar pentru nivelul 3, care intuneca etajul si se poate observa doar un cerc luminos in jurul eroului.

Clasa **Entity** este clasa de bază pentru clasele **Player, Projectile** și clasele din pachetul **Enemy**, care extind și se specializează comportamentul entităților în joc. Clasa **Entity** furnizează funcționalități și variabile comune pe care atât clasele **Player, Projectile** cât și clasele din pachetul **Enemy** le vor moșteni și le vor adapta pentru a se potrivi nevoilor specifice. Astfel, aceste clase vor adăuga și vor suprascrie metodele și variabilele din clasa de bază **Entity** pentru a defini comportamentul și caracteristicile specifice ale jucătorului și inamicilor. În acest fel, putem beneficia

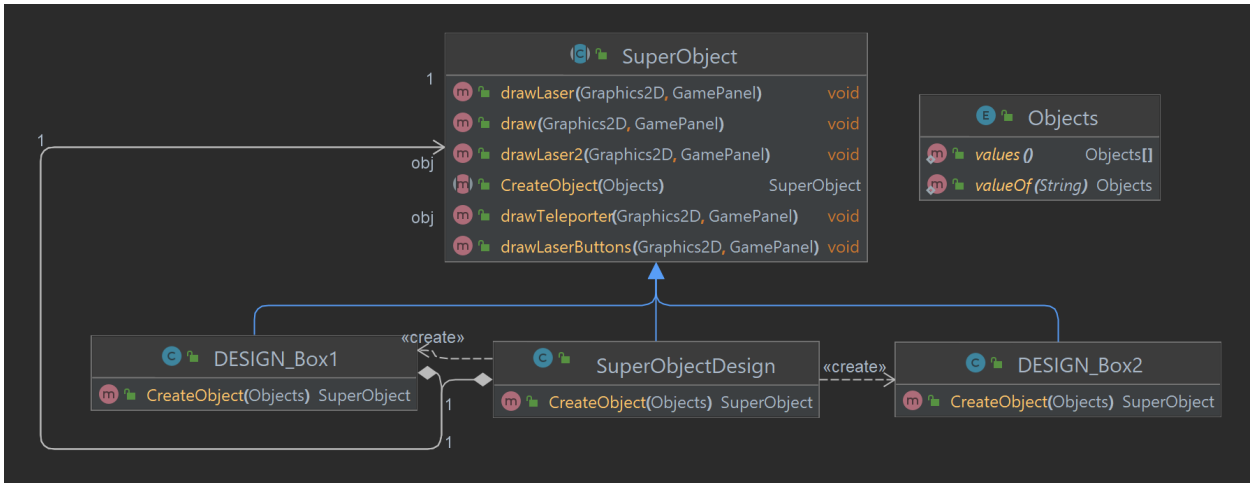
de reutilizarea codului comun în clasa de bază și putem extinde și adapta acest comportament pentru entitățile specifice. Clasa Entity este responsabilă de gestionarea aspectelor generale ale entităților, cum ar fi poziționarea, direcția de deplasare, coliziunile și actualizarea stării lor.

Pachetul **AI** cuprinde doua clase: **Node** si **Pathfinder**, clase responsabile pentru AI-ul inamicilor si a proiectilelor. Implementeaza algoritmul A*, care este un algoritm de cautare a celui mai scurt drum de la inamic la erou; gasind astfel dumul inamicul poate urmari eroul pe drumul respectiv.

Sabloane de proiectare :

A fost implementat sablonul de proiectare Singletone in clasa Main, creandu-se o singura instanta a acesteia.

Implementare sablon de proiectare Factory Pattern :



Bibliografie : https://opengameart.org/content/sci-fi-facility-asset-pack