**Project tile (MAZE RUNNER)**

**Author (Mihalache Nicoleta-Ecaterina, grupa 1211B)**

**Gameplay (rules):** Single player campaigns in which he must avoid as much as possible his intersection with monsters that appear randomly through the maze to find and reach the exit of the maze. The game involves collecting several pennies by the player who are also randomly placed in the maze, as well as finding the exit as quickly as possible. The player must avoid as much as possible the intersection with the monsters in the maze so as not to die. He is killed if he touches a monster and the game ends.

**Plot (game story):** The game is set in a place called Poiana, on Earth, the home planet of Thomas. He wakes up in Poiana without any memory, this place being surrounded by four large and thick metal walls and outside them is the Labyrinth, a labyrinth of high concrete walls covered with ivy that hosts strange, lethal creatures, known as Grievers. Monsters are amorphous from metal and meat. Thomas enters the labyrinth in the hope of finding the exit from the Labyrinth as soon as possible, while collecting coins and especially hiding from lethal monsters.

**Characters:**

1. Thomas is the protagonist and the character-player. He is usually portrayed as a heroic but cheerful guy who hopes to find his way out of the Labyrinth before intersecting with monsters and dying.
2. Monsters are ugly and lethal creatures of flesh and metal called Grievers.

**Mechanics:** turns, user interaction, keys, game points, interaction of many elements of the game for example: collision of the character with the Labyrinth (he can only walk on light green tiles), collision of the character with pennies (when he touches a penny he disappears and the player's score increases) , the player's collision with the monsters (when he intersects with a monster he dies and the game ends).

**SpriteSheet**

1. Harta jocului meu ofera o perspectiva ortografica fiind construita din dale (tile) care ocupa o anumita parte a suprafetei ferestrei. Fereastra mea are in total 400 de dale (20 de dale pe lungime si 20 de dale pe latime) avand dimensiunea in pixeli de 960 de pixeli lungime si 960 pixeli latime. Fiecare nivel este definit printr-o matrice generata in interiorul programului prin sablonul de proiectare Factory Method.

1. Personajul meu se poate misca in toate directiile (sus, joc , stanga, dreapta) si pentru acest lucru am pentru fiecare stare cate un spritesheet:

1. Monstrii mei se misca doar din dreapta hartii in stanga hartii si au urmatorul spritesheet:



1. Gaming points(scorul) este realizat prin cresterea punctajului obtinut pentru fiecare banut strans. Banutii sunt generati random pe harta si au urmatorul spritesheet:



1. Pentru butonul de start am doua spritesheet-uri (neapasat si apasat):

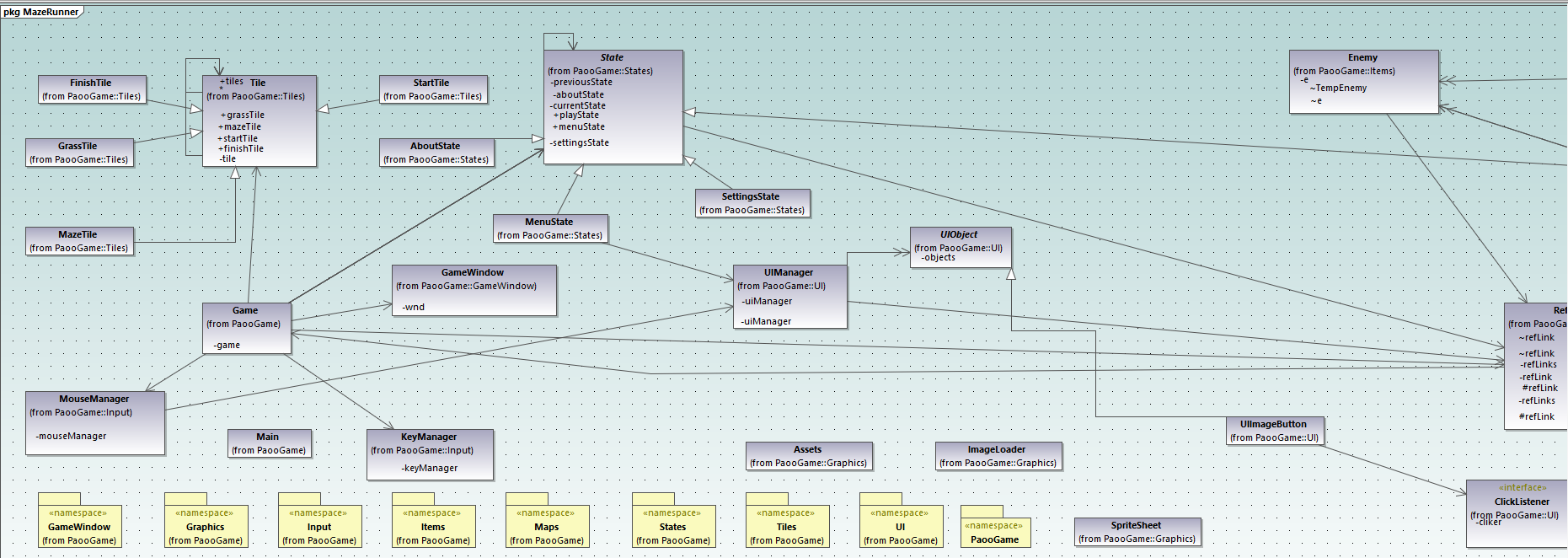
1. Interfata cu utilizatorul este una simpla. Pagina de meniu peste care este desenat butonul de start :

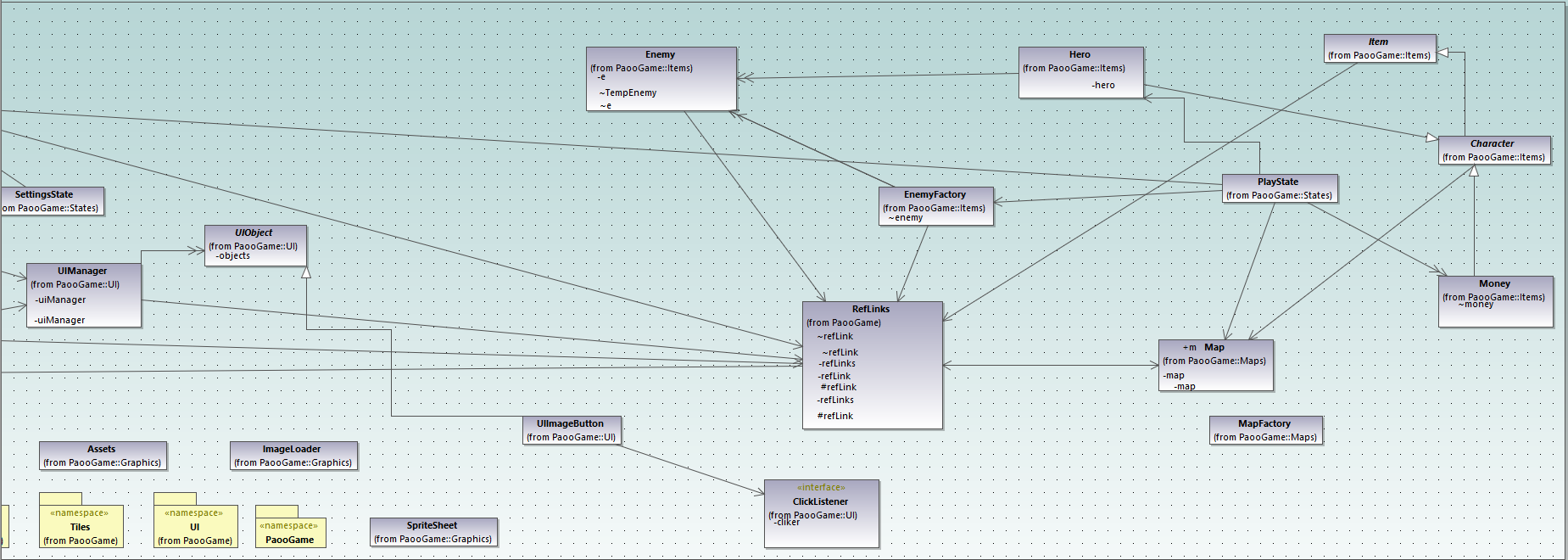


1. Pentru pagina de final atunci cand personajul a castigat sau a murit:

Documentatia tehnica a proiectului atat descriptive, cat si sub forma unei diagrame de arhitectura:





Descrierea celor mai importante clase din joc:

1. **Clasa Game:** cea mai importanta clasa a jocului ce implementeaza interfata Runneble, utilizata pentru a crea un fir de executie avand ca argument clasa Game. Respectiva clasa initializeaza jocul, actualizeaza elementele jocului (prin metoda Update), deseneaza totul pe tabla de joc (prin metoda Draw), porneste si opreste jocul prin itermediul metodelor existente in aceasta.
2. **Clasa PlayState:** este clasa care controleaza intreg jocul (instantiaza player-ul, harta, monedele, monstrii), deseneaza si actualizeaza starea curenta a jocului.
3. **Clasa Item:** implemtenteaza notiunea abstracta de entitate active din joc, un element cu care poti interactiona.
4. **Clasa EnemyFactory:** implementeaza o fabrica de monstri de tip clasei Enemy random pe harta, ii deseneaza si actualizeaza pozitia si imaginea inamicilor.
5. **Clasa Money:** implementeaza notiunea de moneda random pe harta; atunci cand player-ul atinge o moneda scorul jocului creste.
6. **Clasa Map:** implementeaza notiunea de harta a jocului. Harta este generata printr-o matrice de 20 pe 20 (unde este 0 player-ul poate merge, unde este 1 player-ul nu poate merge deoarece reprezinta peretele Labirintului, unde este 3 am dala de inceput de joc iar unde am 2 este dala de final de joc).
7. **Clasa MenuState:** implementeaza notiunea de meniu al jocului (la inceputul jocului ne apare o interfata pe care exista un buton de Start- apasand ope buton intram in joc), metodele Draw() si Update() deseneaza si actualizeaza pe ecran starea curenta a meniului.
8. **Clasa Tile:** retine toate dalele intr-un vector de dale si ofera posibilitatea regasirii lor dupa un id.
9. **Clasa RefLinks:** clasa ce retine o serie de referinte ale unor elemente pentru a fi usor accesibile, ajuta la imbunatatirea programarii.