Aranara Flower Garden
Galatanu Marco
1207B

1 pct oficiu pentru aspect, respectare cerințe

Povestea jocului:(1 pct)

"Aranara Flower Garden" este un joc de tip puzzle, creat special pentru a testa abilitatile jucatorilor de a găsi cele mai bune soluții pentru probleme complexe. Personajul principal al jocului este Traveler, un calator curajos, care ajunge pe planetă Teyvat și se intalneste cu un trunchi de copac fermecat numit Todd.

Todd ii da lui Traveler o sarcina importanta: să planteze semintele unei flori vitale pentru populația planetei Teyvat. Dar, exista o problema: zona în care trebuie sa fie plantate semintele este foarte mare, iar Traveler trebuie sa găsească o soluție pentru a acoperi întreaga zona fără a trece de doua ori prin același pătrați.

Prezentare joc: (1 pct)

"Aranara Flower Garden" este un joc distractiv și educativ, care pune la încercare abilitățile de rezolvare a puzzle-urilor ale jucatorilor, dar și creativitatea si imaginatia lor in crearea unor gradini virtuale unice.

Reguli joc:(1 pct)

1. Scopul jocului este de a planta toate semintele din zona de plantare fără a trece de două ori prin aceeași zonă.
2. Jucătorul începe într-un colț al zonei de plantare și trebuie să exploreze împrejurimile pentru a găsi locurile unde trebuie plantate semintele.
3. Jucătorul poate merge în toate direcțiile, cu excepția zonei prin care a trecut deja.
4. Fiecare zonă de plantare are o anumită formă și dimensiune, iar jucătorul trebuie să găsească calea corectă pentru a ajunge la fiecare zonă.
5. Dacă jucătorul se blochează și nu mai poate merge înainte, poate apăsa un buton de resetare pentru a începe din nou din colțul de pornire, dar pierde un punct de HP.
6. Jocul este câștigat atunci când toate semințele sunt plantate în toate zonele de plantare.

Personajele jocului:(1 pct)

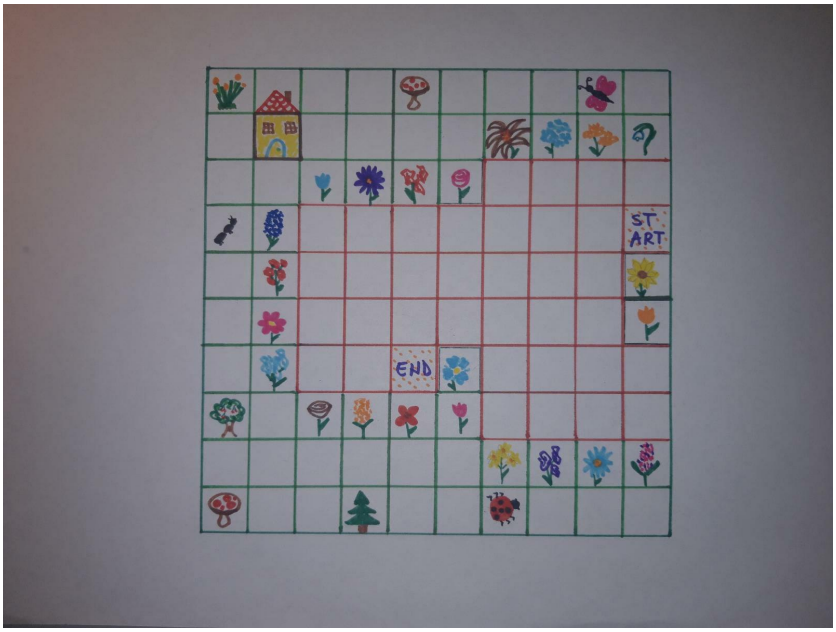
-Traveler este personajul principal. Scopul lui este de a îndeplini sarcinile date de creaturile magice din Teyvat (planeta pe care acesta a ajuns).

Tabla de joc (2pct):

Tabla de joc este formată din dale de dimensiuni - 48x48 pixeli dispuse într-un model pătratic de dimensiune 40x40.

Componentele pasive ale tablei sunt reprezentate de texturile de iarbă, sol, pietre și copaci care definesc aspectul general al nivelului de joc.

Componentele active ale tablei includ terenurile speciale unde se pot planta semințe și unde se pot crește și îngriji plantele.


START
END

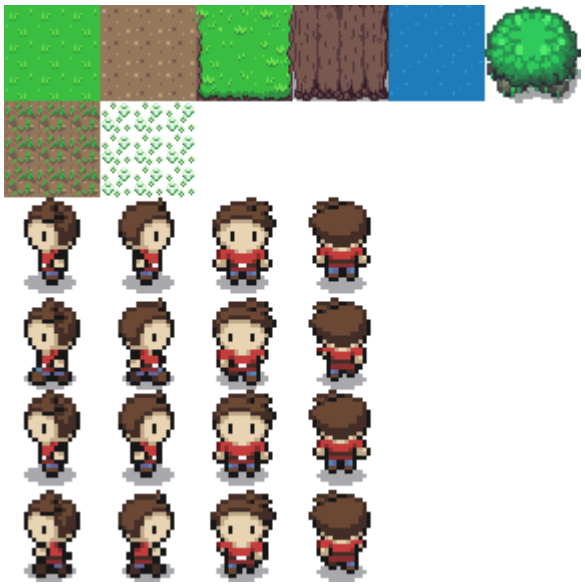
Mecanica jocului (1 pct)

În unul dintre colțurile ecranului vor fi prezente diferite informații, cum ar fi timpul care s-a scurs, timpul în care trebuie sa se incadreze sau timpul cu care a început nivelul.

Unele zone ale tablei pot fi blocate inițial și pot fi accesate doar după ce jucătorul a colectat plantele speciale necesare pentru a debloca aceste zone. Aceste zone noi pot conține alte tipuri de plante speciale, care adaugă noi puzzle-uri pentru jucător.

Jucătorul trebuie să planteze semințele pe terenurile active fără a trece de două ori prin aceeași zonă a tablei. Acesta trebuie să planifice cu atenție mișcările și să găsească calea corectă pentru a planta și îngriji plantele în timp ce explorează și deblochează noi zone ale tablei de joc.

Game sprite (1 pct)



1 pct pentru:
Descriere fiecare nivel

Nivelul 1:

In acest nivel, jucătorii trebuie sa planteze semintele florilor în jurul unei case pitorești aflate la marginea pădurii. Zona este limitata de un gard în jurul casei și un munte în spatele acesteia. Resursele grafice disponibile sunt iarba, gardul, casa și muntele.

Nivelul 2:

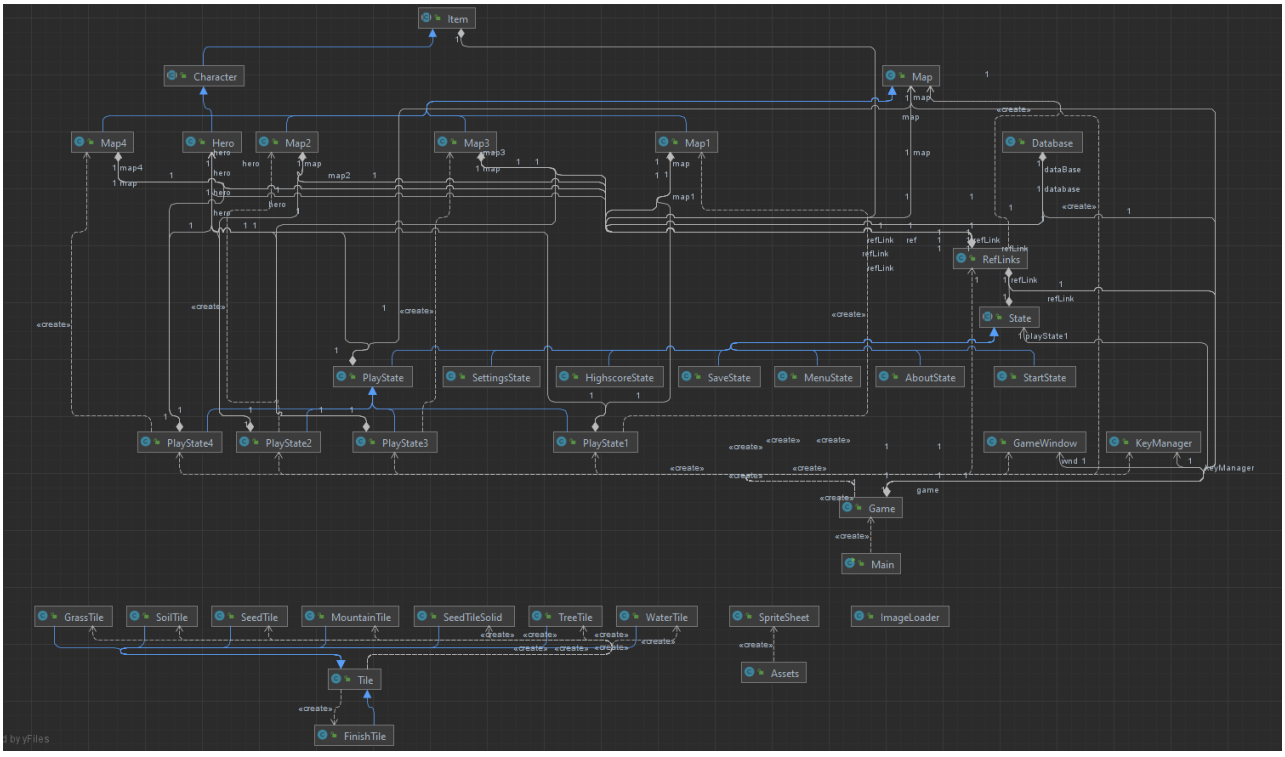
In acest nivel, jucătorii se vor concentra pe plantarea florilor în jurul unei frumoase fantani din mijlocul unei grădini în aer liber. Acest nivel este mult mai mare decat primul nivel și are mai multe obstacole, cum ar fi copacii și un castel în apropiere. Jucătorii trebuie sa-si folosească abilitățile de gandire logica și creativitatea pentru a găsi cea mai buna soluție. Resursele grafice disponibile sunt apa, iarba, copacii, fantana si castelul.

Nivelul 3:

In acest nivel, jucătorii trebuie sa planteze semintele florilor în jurul unui lac mare situat la baza unui munte înalt. Zona este limitata de apă și de munte, ceea ce face dificila plantarea florilor in unele zone. Resursele grafice disponibile sunt apa, iarba, muntele si copacii, ceea ce face acest nivel unul dintre cele mai complexe și provocatoare nivele din joc.

Etapa 2:

Diagrama UML



Descriere:

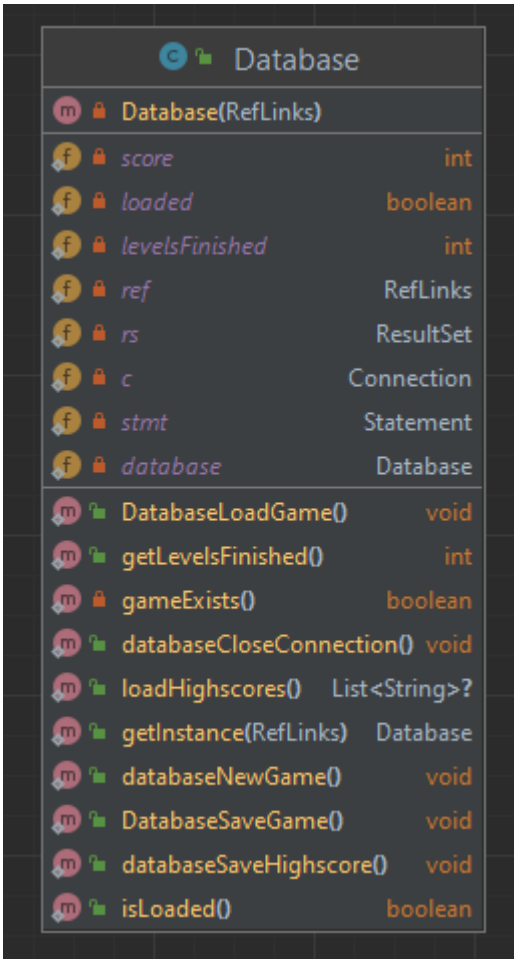
-pachetul Paoo Game conține toate pachetele (Game Window, Graphics, Input, Items, Maps, States, Tiles) dar și clasele:

Main (clasa în care se creeaza obiectul de tip Game și se apelează metoda StartGame()).

Game (clasa principala a întregului proiect. Implementează Game Loop (Update -> Draw))

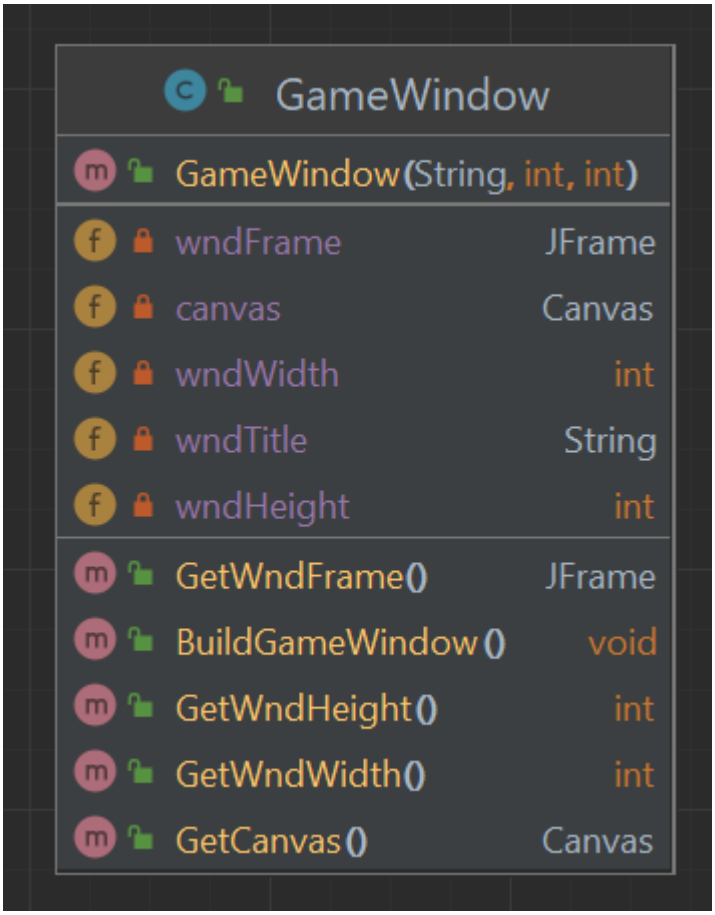
Reflinks (reține o serie de referințe ale unor elemente pentru a fi ușor accesibile).

-pachetul Database conține clasa:



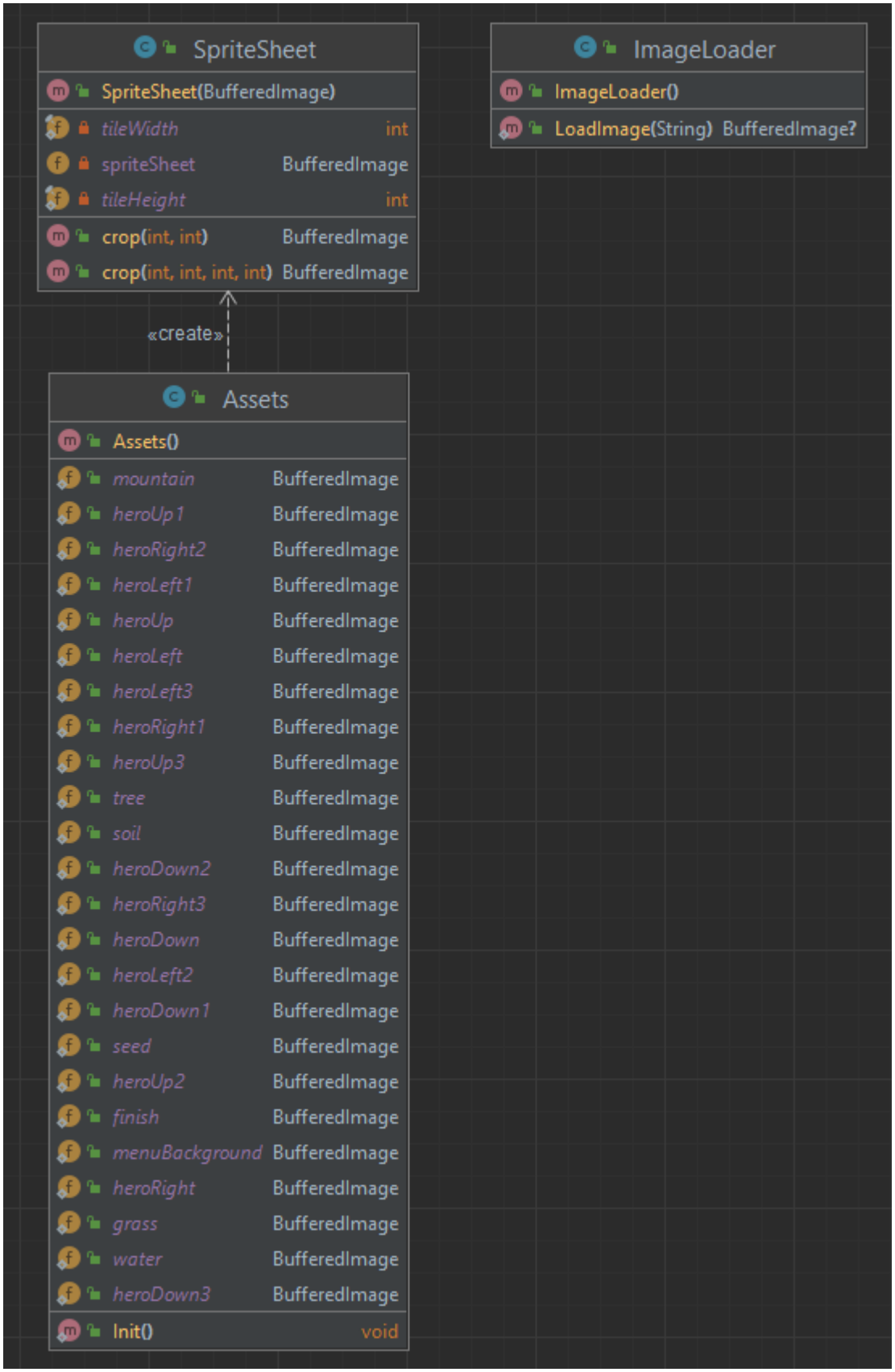
Database: Se ocupă de salvarea stării în care se afla jocul

-pachetul GameWindow conține clasa:



GameWindow: implementează noțiunea de fereastra a jocului. Membrul wndFrame este un obiect de tip JFrame care va avea utilitatea unei ferestre grafice și totodată și cea a unui container (toate elementele grafice vor fi conținute de fereastra).

-pachetul Graphics contine clasele:



Assets: (Clasa ce încarcă fiecare element grafic necesar jocului)

- Assets include tot ce este folosit într-un joc: imagini, sunete, hărți etc.
- Metoda Init() inițializează referințele către elementele grafice utilizate.

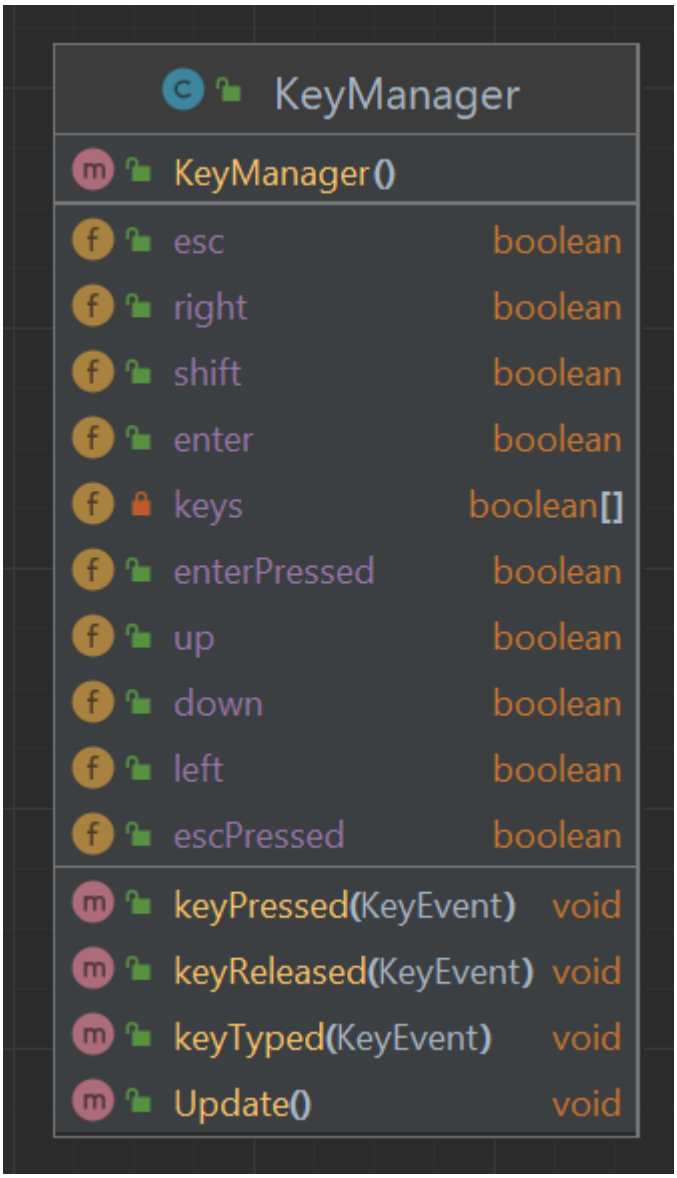
ImageLoader: (contine o metoda statică pentru încărcarea unei imagini in memorie)

- Metoda LoadImage() încarcă o imagine intr-un obiect BufferedImage si returnează o referință către acesta.

SpriteSheet: (reține o referința către o imagine formată din dale)

- Metoda crop() returnează o dala de dimensiuni fixe (o subimagine) din sprite sheet de la adresa (x * latimeDala, y * inaltimeDala)

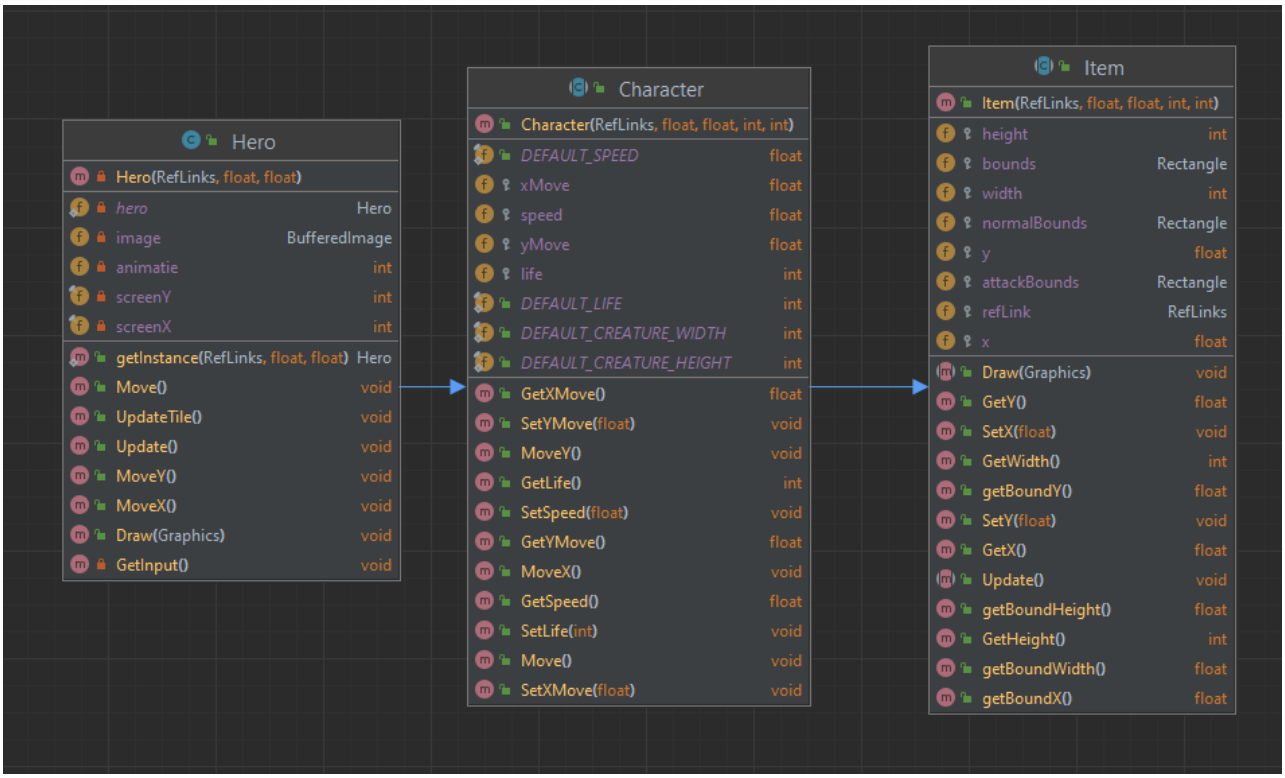
-pachetul Input conține clasa:



KeyManager: (gestionează input-ul de tastatura.)

- Clasa citește dacă au fost apasata o tasta, stabilitate ce tasta a fost actionata si seteaza corespunzator un flag.In program trebuie sa se tina cont de flagul aferent tastei de interes. Dacă flagul respectiv este true înseamnă ca tasta respectivă a fost apasata și false nu a fost apasata.
- Metoda Update() este folosită în metoda Update() din clasa **Game** pentru a determina starea tastelor

-pachetul Items contine clasele:



Item: (Implementează noțiunea abstractă de entitate activa din joc, "element cu care se poate interactiona: monstru, turn etc.".)

- Constructorul inițializează obiectul cu anumite caracteristici (creeaza dreptunghi de coliziune pentru modul normal, creeaza dreptunghi de coliziune pentru modul de atac, iar dreptunghiul de coliziune implicit este setat ca fiind cel normal).
- Metoda abstractă Update() este destinata utilizarii stării curente.
- Metoda abstractă Draw() este destinata desenării stării curente.
- Clasa mai contine si diferite gettere si settere pentru poziție(x,y), caracteristici ale bounding box-ului, dar și al stării itemului.

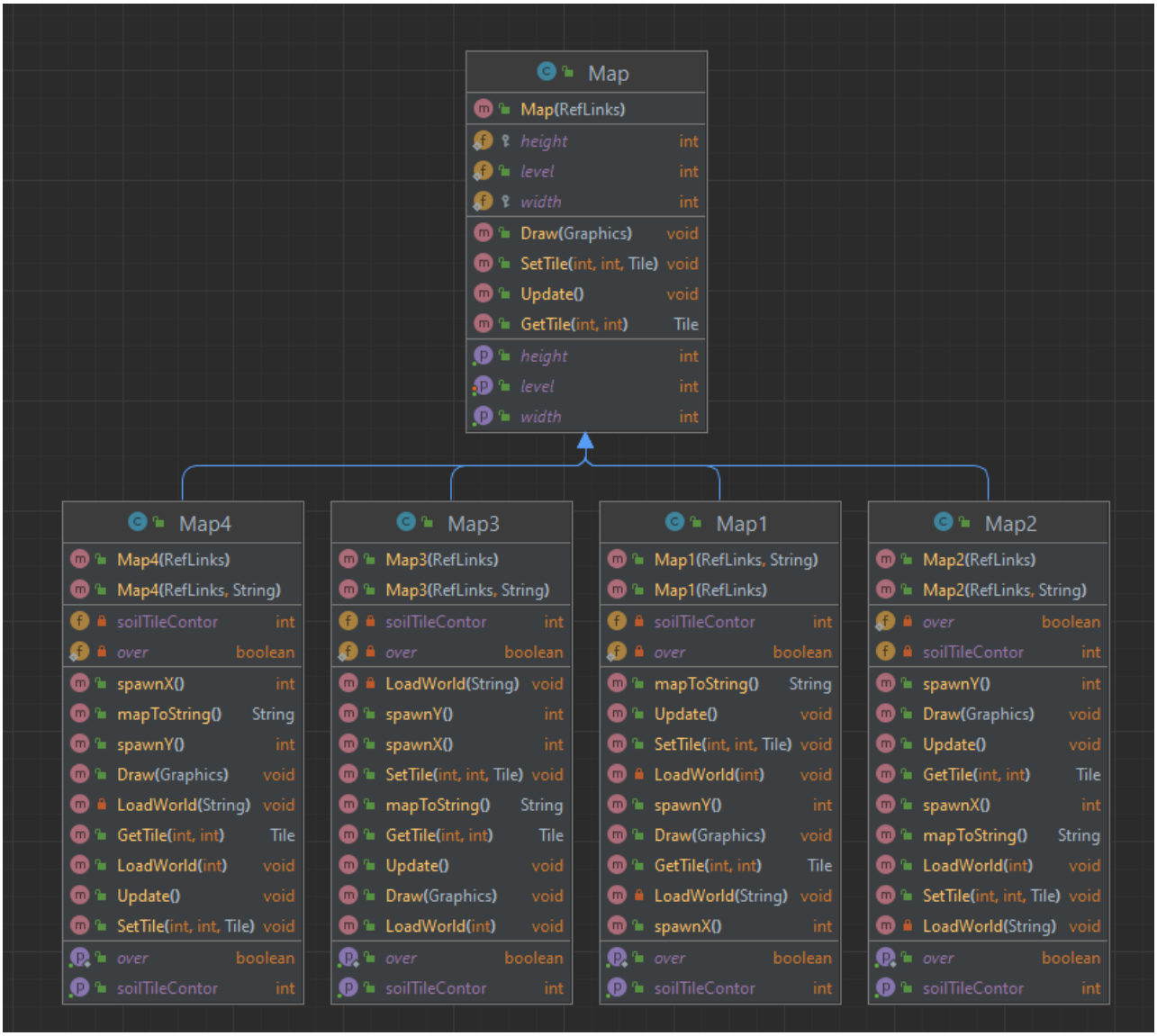
Character: (Defineste notiunea abstractă de caracter/individ/ființă din joc.)

- Clasa este mostenita din clasa Item și adaugă aspecte în plus cum ar fi: viața, viteza de deplasare și distanță cu care trebuie sa se miste/deplaseze în urma calculelor (în toate direcțiile trebuie sa se miste cu aceeași viteza)
- Metoda Move() modifica poziția caracterului folosindu-se de alte 2 metode MoveX() și MoveY() ce îi modifica poziția pe axele respective.
- Ca și clasa Item, aceasta clasa conține și ea diferite settere si gettere pentru proprietățile noi adaugate

Hero: (Implementează noțiunea de erou/player (caracterul controlat de jucător))

- Clasa este mostenita din clasa Character și adaugă aspecte în plus cum ar fi: imaginea, deplasarea și dreptunghiul de coliziune.
- Implementează un design pattern de Singleton, adică o singura instanță a eroului poate exista, acest design pattern se caracterizează prin constructorul privat, metoda statică getInstance ce returnează referinta către eroul deja creat, sau către cel creat dacă acesta era null.
- Metoda Move() suprascrie metoda Move() din clasa de baza, adaugand aspectul coliziunilor cu tile-uri si cu window-ul in care se afla eroul.
- Metoda UpdateTile() modifica tile-ul pe care calcă eroul, asigurandu-se astfel ca marchează pe unde a trecut, pentru a nu se mai întoarce pe același drum.
- Metoda Update() verifica daca a fost apasata o tastă, actualizează pozitia, actualizează harta și apoi actualizează imaginea jucătorului.
- Metoda Draw() desenează eroul in nouă poziție.

-pachetul Maps contine clasele:



Map: (Implementează noțiunea de hartă a jocului.)

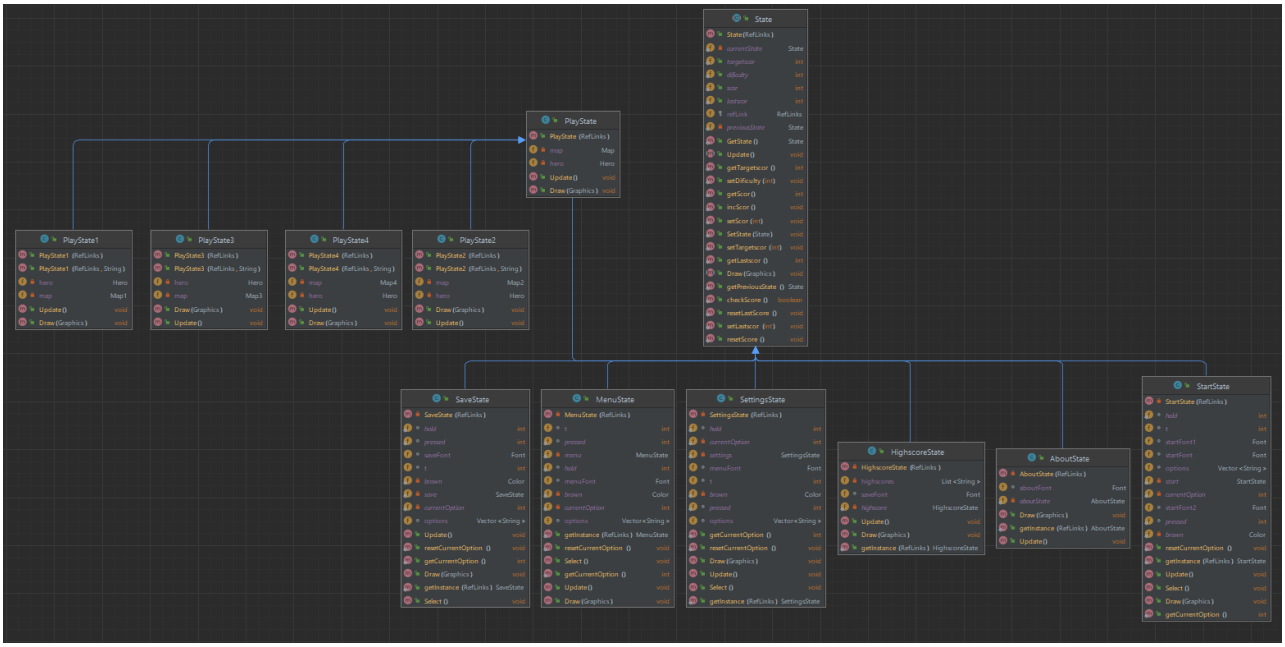
- Constructorul Map() incarca harta de start folosindu-se de metoda LoadWorld()
- Metoda LoadWorld() încarcă din fișierul Map.txt hartă, citind de pe prima linie a fișierului cate linii si cate coloane are hartă, apoi cu un for citește în tiles[x][y] valorile din fișier.

Map [n]: (Implementează noțiunea de hartă a jocului.)

- Fiecare clasa Map [n] are 2 constructori, unul de baza ce primește doar reflink pentru utilizarea ușoară a “scurtaturilor”, iar celalalt pentru initializarea stării mapei în cazul în care starea salvată ultima data era pe mapa n
- Fiecare clasa Map [n] are 2 metode LoadWorld(), primul este pentru incarcarea mapei default din fișier, iar celălalt este pentru incarcarea

fisierului, creat si salvat de baza de date, ce reprezinta starea hartii atunci cand a fost salvat jocul.

-pachetul States contine clasele:



State: (Implementează noțiunea abstractă de stare a jocului/programului.)

- Din ea se vor deriva toate starile urmatoare: AboutState, MenuState, PlayState, SettingsState
- Metoda SetState() seteaza starea curentă a jocului, va fi folosită pentru tranziția dintr-o stare în alta (Meniu -> Setari, Meniu -> About, Meniu -> Play)
- Metoda GetState() returnează starea curentă a jocului, pentru a putea actualiza starea curentă din clasa Game.

AboutState: (Implementează noțiunea de credentiale.)

- Momentan este o clasa ce urmează a fi implementată
- Metoda Update() va actualiza poziția credentialelor.
- Metoda Draw() va afișa pe ecran credentialele la nouă lor poziție.

MenuState: (Implementează noțiunea de meniu pentru joc.)

- Momentan este o clasa ce urmează a fi implementată
- Metoda Update() va actualiza starea în cazul în care este selectata vreo opțiune din meniu
- Metoda Draw() va afișa meniul, iar dacă vreo opțiune este pe cale de a fi selectata, aceasta va fi afișată cu alta culoare.

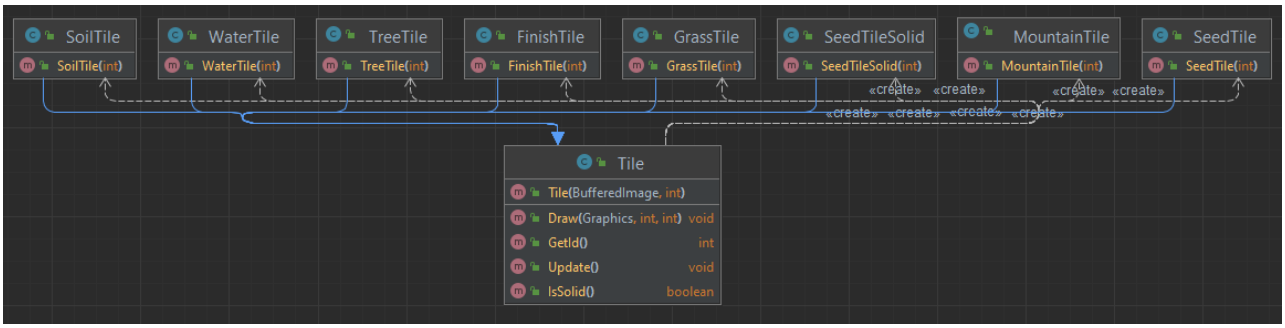
PlayState: (Implementează/controlează jocul și eroul.)

- Metoda Update() actualizează starea curentă a jocului, apeland map.Update() și hero.Update(), actualizând starea mapei și starea eroului.
- Metoda Draw() desenează pe ecran starea curentă a jocului, apeland la fel ca si metoda Update(), mai intai map.Draw() apoi hero.Draw().

SettingsState: (Implementează noțiunea de setări ale jocului)

- Momentan este o clasa ce urmează a fi implementată
- Metoda Update() ca actualiza starea setarii respective in cazul in care este modificata vreo setare
- Metoda Draw() va afișa setarile, iar dacă vreo setare este pe care de a fi modificata, aceasta va fi afișată cu alta culoare pentru a ști pe ce setare ne situăm.

-pachetul Tiles contine clasele:



Tile: (Reține toate dalele într-un vector și oferă posibilitatea regasirii după un id.)

- Metoda IsSolid() returnează în mod implicit
- Metoda Draw() va desena dala pe ecran la coordonatele x și y din matricea de dale, înmulțite cu lățimea și înălțimea unei dale.
- Metoda GetId() returnează id-ul dalei care este folosit în matricea de dale.

GrassTile: (Abstractizeaza noțiunea de dala de tip iarba.)

- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.grass și id-ul respectiv dalei de iarba din matrice.

MountainTile: (Abstractizeaza noțiunea de dala de tip munte sau piatra.)

- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.mountain și id-ul respectiv dalei de munte din matrice.

SoilTile: (Abstractizeaza noțiunea de dala de tip pamant.)

- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.soil și id-ul respectiv dalei de pamant din matrice.

TreeTile: (Abstractizeaza noțiunea de dala de tip copac.)
- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.tree și id-ul respectiv dalei de copac din matrice.

WaterTile: (Abstractizeaza noțiunea de dala de tip apa.)
- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.water și id-ul respectiv dalei de apa din matrice.

FinishTile: (Abstractizeaza noțiunea de dala de tip final.)
- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.finish și id-ul respectiv dalei de apa din matrice.

SeedTile: (Abstractizeaza noțiunea de dala de tip pamant pe care s-a plantat.)
- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.seed și id-ul respectiv dalei de apa din matrice.

SeedTileSolid: (Abstractizeaza noțiunea de dala de tip pamant pe care s-a plantat.)
- Are un constructor ce apelează constructorul clasei de baza Tile cu Assets.seed și id-ul respectiv dalei de apa din matrice, dar face si solid=true, adica este un tile solid.

Design patterns:

Singleton:

Design pattern-ul Singleton este un pattern de creare care se utilizează atunci când dorim să ne asigurăm că o clasă are o singură instanță și să oferim un punct global de acces la această instanță în cadrul întregii aplicații. Acesta garantează că o clasă va avea o singură instanță și oferă un mecanism prin care această instanță poate fi accesată.

Caracteristica principală a pattern-ului Singleton este că constructorul clasei este privat, ceea ce înseamnă că nu poate fi creată o instanță direct prin intermediul operatorului "new". În schimb, clasa oferă o metodă statică prin care se poate obține referința la instanța unică a clasei.

Atunci când este apelată această metodă statică, se verifică dacă instanța deja există. Dacă există, se returnează acea instanță. În caz contrar, se creează o nouă instanță și se

returnează. Astfel, indiferent de câte apeluri sunt făcute la metoda de acces, întotdeauna se va returna aceeași instanță.

```
25 usages   GalatanuM *
public class Hero extends Character
{
    57 usages
    private BufferedImage image;    /*!< Referinta catre imaginea curenta a eroului.*/
    3 usages
    private static Hero hero = null;
    80 usages
    private int animatie;
```

```
1 usage   GalatanuM
private Hero(RefLinks refLink, float x, float y)
{
        ///Apel al constructorului clasei de baza
    super(refLink, x,y, Character.DEFAULT_CREATURE_WIDTH, Character.DEFAULT_CREATURE_HEIGHT);
        ///Seteaza imaginea de start a eroului
    image = Assets.heroDown;
    animatie=0;
        ///Stabilieste pozitia relativa si dimensiunea dreptunghiului de coliziune, starea implicita(normala)
    normalBounds.x = 16;
    normalBounds.y = 16;
    normalBounds.width = 16;
    normalBounds.height = 24;
}

 GalatanuM
public static synchronized Hero getInstance(RefLinks refLink, float x, float y)
{
    if(hero == null)
    {
        hero = new Hero(refLink,x,y);
    }
    return hero;
}
```

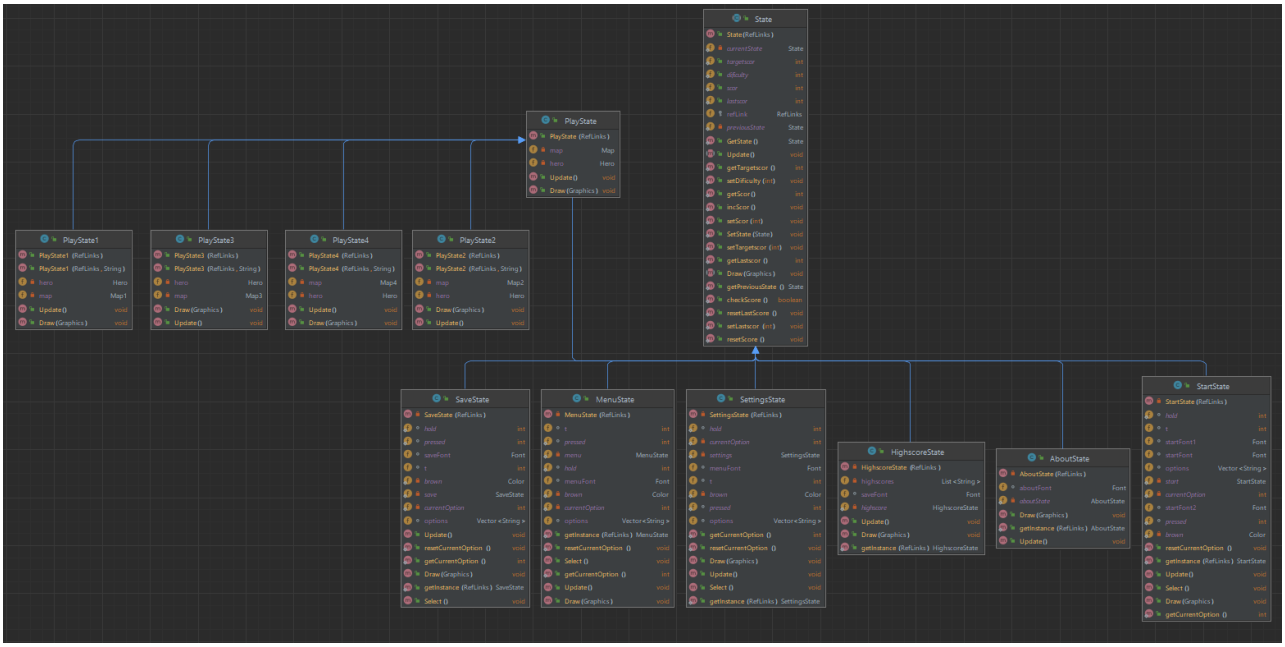
State:

Acest pattern se bazează pe principiul incapsulării, prin care comportamentul asociat unei anumite stări este definit într-o clasă separată numită "stare". Obiectul care utilizează pattern-ul State are o referință către o instanță a clasei Stare și își deleagă responsabilitățile către această instanță.

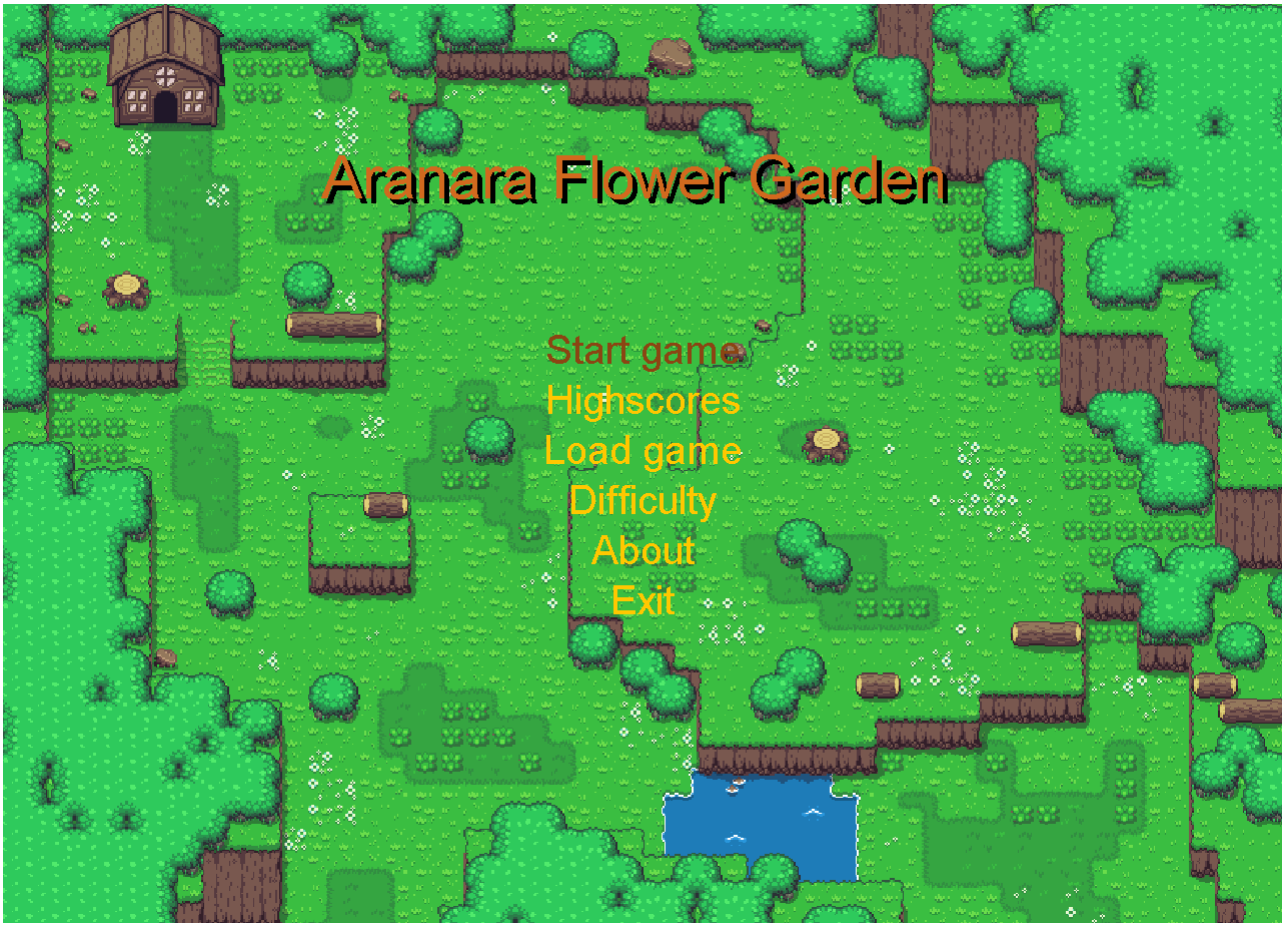
Prin intermediul pattern-ului State, obiectul poate să-și schimbe starea la runtime prin înlocuirea instanței de stare cu o altă instanță corespunzătoare. Astfel, comportamentul obiectului va varia în funcție de starea curentă.

Prin utilizarea acestui pattern, se obține o structură modulară și ușor de întreținut, deoarece fiecare stare este reprezentată de o clasă separată și responsabilitățile sunt distribuite în mod clar între aceste clase. De asemenea, adăugarea de noi stări devine ușoară, deoarece se poate crea o nouă clasă stare și se poate înlocui instanța curentă cu cea nouă.

Pattern-ul State este adesea utilizat în aplicații care implică interacțiunea cu obiecte complexe care pot avea mai multe stări și comportamente diferite, cum ar fi mașini de stări, interfețe grafice sau simulări.

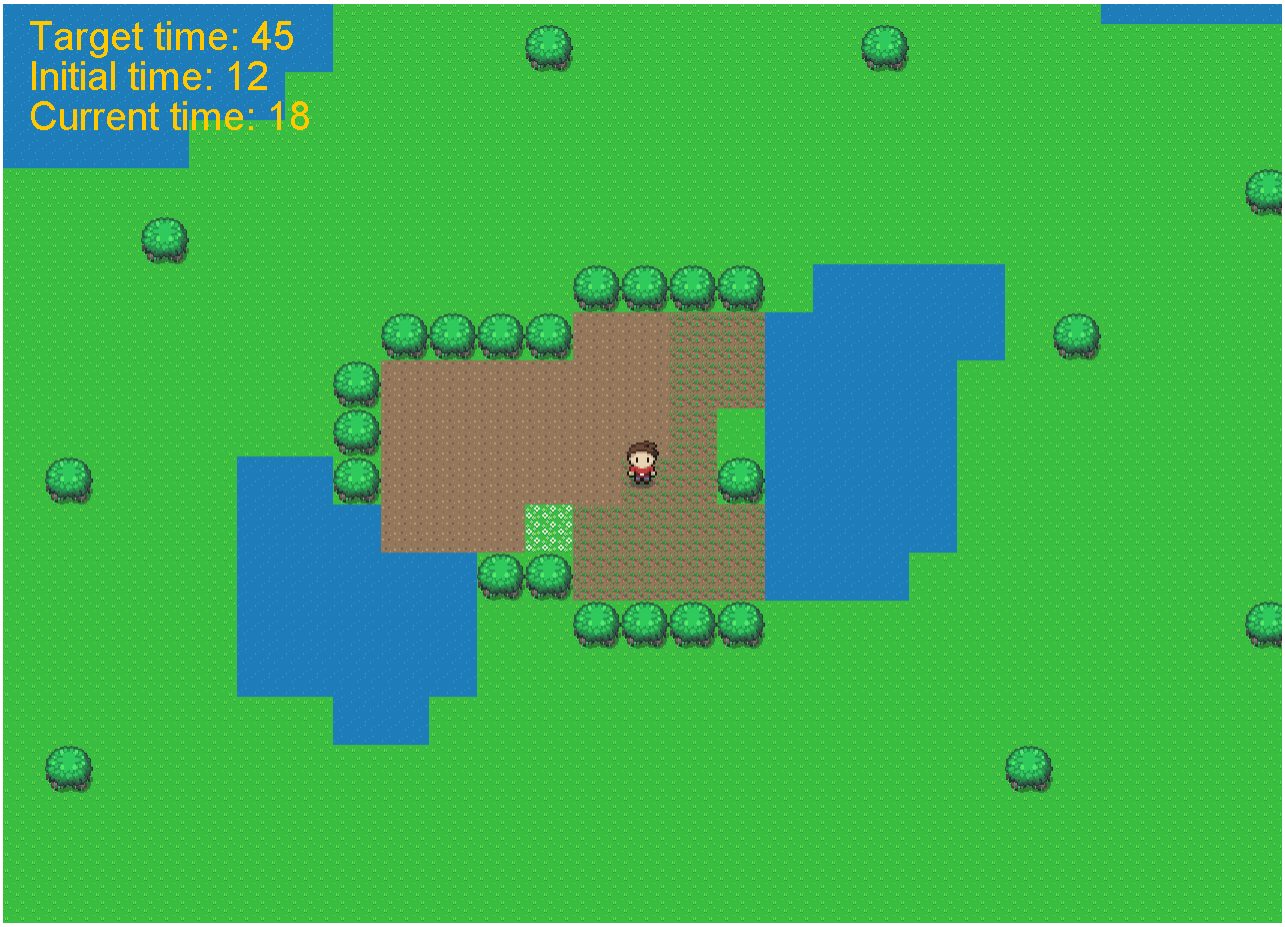


Poze din joc:


Aranara Flower Garden
Start game
Highscores
Load game
Difficulty
About
Exit


Target time: 45
Initial time: 0
Current time: 1


Target time: 45
Initial time: 7
Current time: 9

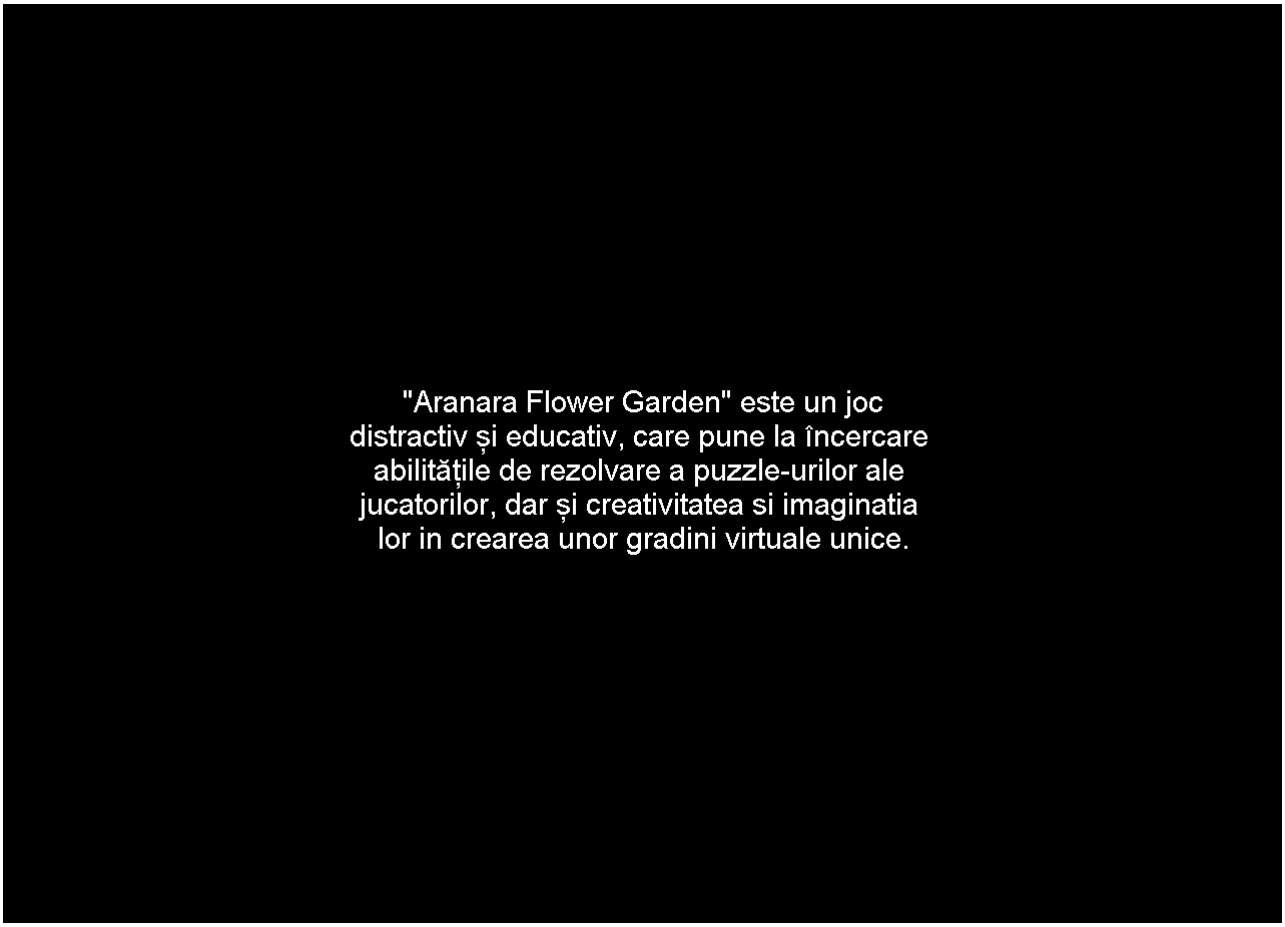

Target time: 45
Initial time: 12
Current time: 18


Target time: 45
Initial time: 21
Current time: 32


Save your score
Exit to main menu

1. 29
2. 31
3. 32
4. 36
5. 40
6. 41
7. 41
8. 42
9. 43
10. 56


Back
Easy
Normal
Hard



Descriere meniu

Start menu:
- Start
- Highscores
- Load game
- Difficulty
- About
- Exit

Pause menu:
- Resume
- Restart level
- Save game state
- Exit to menu

Dificulty menu:
- Back
- Easy
- Normal
- Hard

Bibliografie

https://game8.co/games/Genshin-Impact/archives/387328
https://opengameart.org/content/zelda-like-tilesets-and-sprites