KIDS SEE GHOSTS

Chiteala Tudor Matei

Gameplay: Jocul este single player si campania consta in eliminarea inamicilor pentru a putea trece la nivelul urmator.Eroul poate pe parcursul jocului sa elimine inamicii sau sa-i evite, inamicii putand in acelasi timp sa omoare eroul daca acesta este in zona lor de atac.Eroul pe parcursul campaniei are doar 2 vieti si fiecare inamic doar o viata.

Plot: Jocul este setat in dimensiunea “Dune” cunoscuta si ca “GHOSTTOWN”.Eroul pe nume “Yandi” care este un om-schelet din dimensiunea “Sun kill moon” a patruns in alta dimensiune printr-o piatra filozofala.Yandi,eroul, trebuie sa ajunga inapoi in dimensiunea lui pana cand portatul nu se inchide si gardienii dimensiunii Dune nu-l prind.

Characters:

1.***Yandi*** este protagonistul care face parte din dimensiunea “Sun kill moon” si el este un om-schelet.

2.***Ghost gardiens*** sunt protectorii dimensiunii Dune(GHOSTTOWN).

Mechanics:

User Interface :

Menu :

Buton play – porneste jocul

Buton help – setari sonor

Buton quit – iesire din joc

PlayState:

Eroul controlat de user poate:

sa sara

sa se deplaseze pe harta

sa urce scari

sa elimine inamicii.

Taste: W –(up) :doar atunci cand intalneste un obiect catarabil(ex:scara);

A –(left): deplasare stanga

D-(right):deplasare dreapta

S-(down):coborare

K-(attack):atac inamicii

T-(reverse):pentru a putea iesi din diferite blocaje

Enemy:

Inamicul poate sa atace eroul atunci cand este in perimetrul lui

Game Points:

In PlayState dreapta sus userul poate verifica cate vieti are.Scopul jocului ca acesta sa ramana cel putin cu o viata.

Victory:User-ul castiga atunci cand el parcurge toate cele 7 nivele ce cel putin o viata.

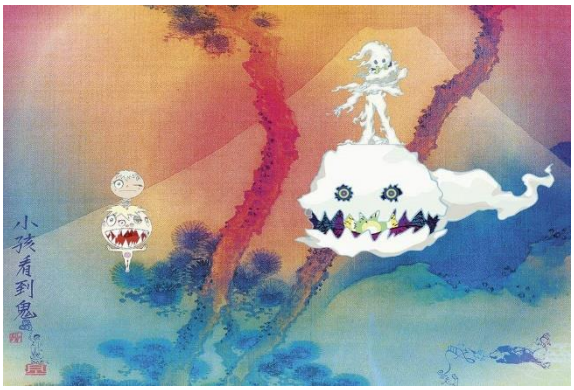
Defeat:User-ul pierde atunci cand ramane fara vieti si atunci este obligat sa reia nivelul.

SpriteSheets:

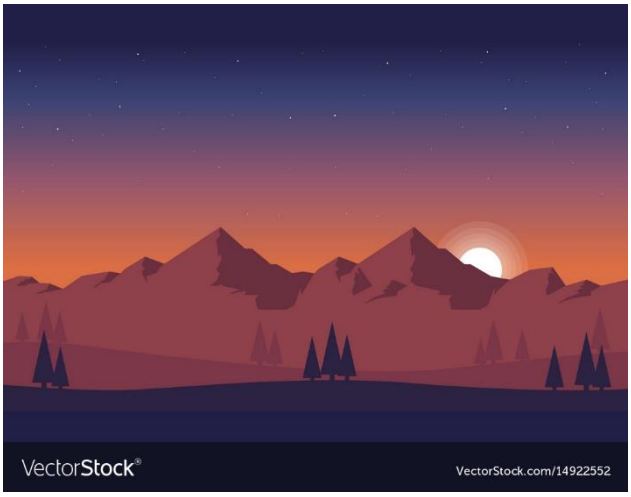
Hero:

Ghost:

BackGround1:



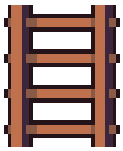
BackGround2:



Tree:

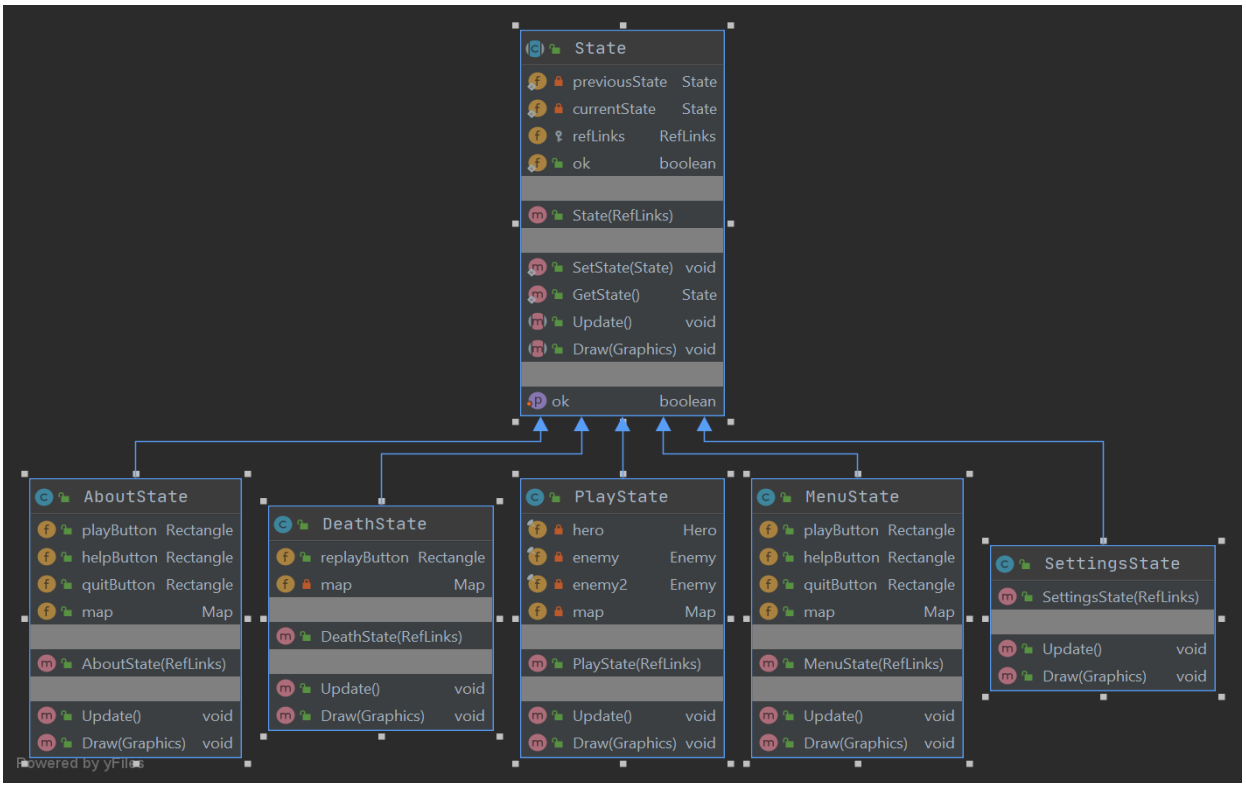
Rock:

Ladder:



Stone:

Diagrame de clase:



Class State:

Este o clasa abstracta folosita pentru starile jocului(AboutState,DeathState,PlayState,MenuState si SettingsState).

previousState : Starea precedenta a jocului

currentState : Starea curenta a jocului

refLinks : Referinta la clasa RefLinks

ok : Parametrul va arata in cadrul functiei Update() daca inamicii trebuie create sau au fost deja create

Class PlayState:

Specifica starea principala a jocului

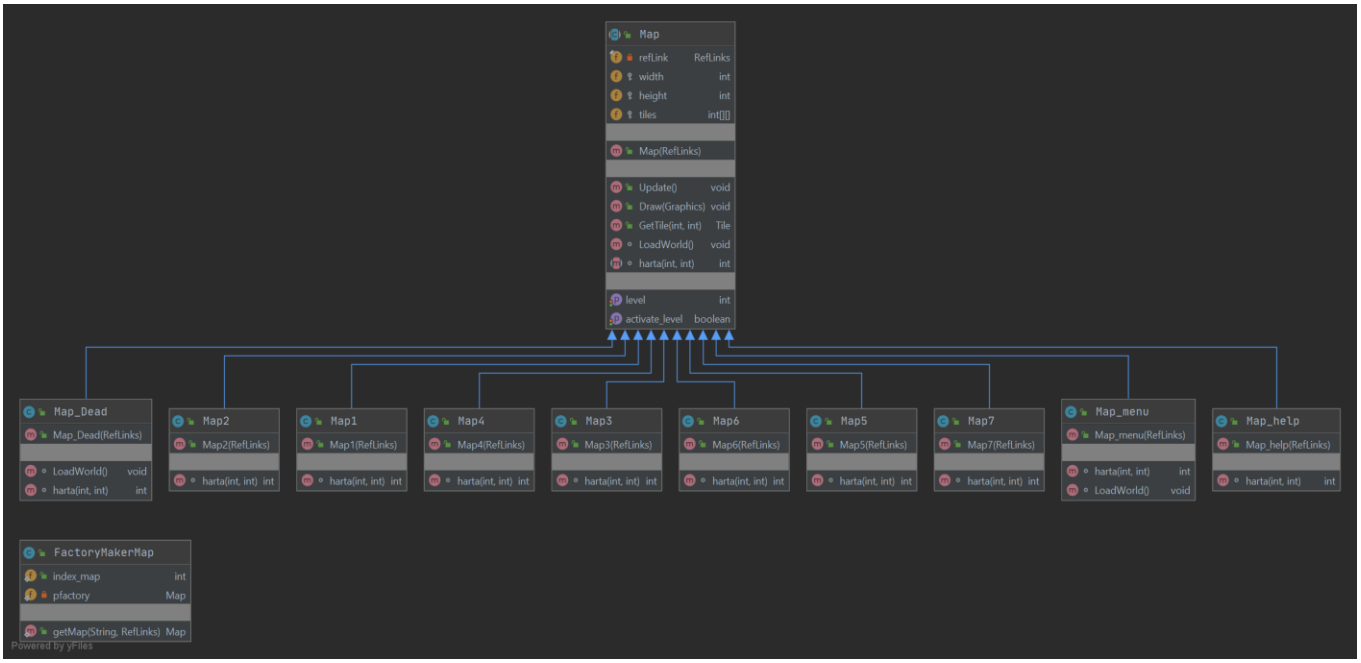
void Update() – functia va actualiza starea curenta a jocului in functie de nivel. Va selecta

ce inamici vor fi create in functie de index_map din clasa FactoryMakerMap

Class MenuState:

Specifica starea de meniu a jocului de unde user-ul va putea da start campaniei,schimba sonorizarea sau inchide jocul.

void Update() – Va decide in ce stare ajunge jocul in functie de inputul user-ului.Va verifica ce buton a fost apasat(Play,Help,Quit).



Class Map:

Este o clasa abstracta folosita pentru a incarca harta jocului

Membrii:

refLink - o referinta catre obiectul RefLink

width – latimea in numar de dale

height – inaltimea in numar de dale

tiles – referinta catre o matrice cu codurile de dale

level – specifica la ce nivel suntem

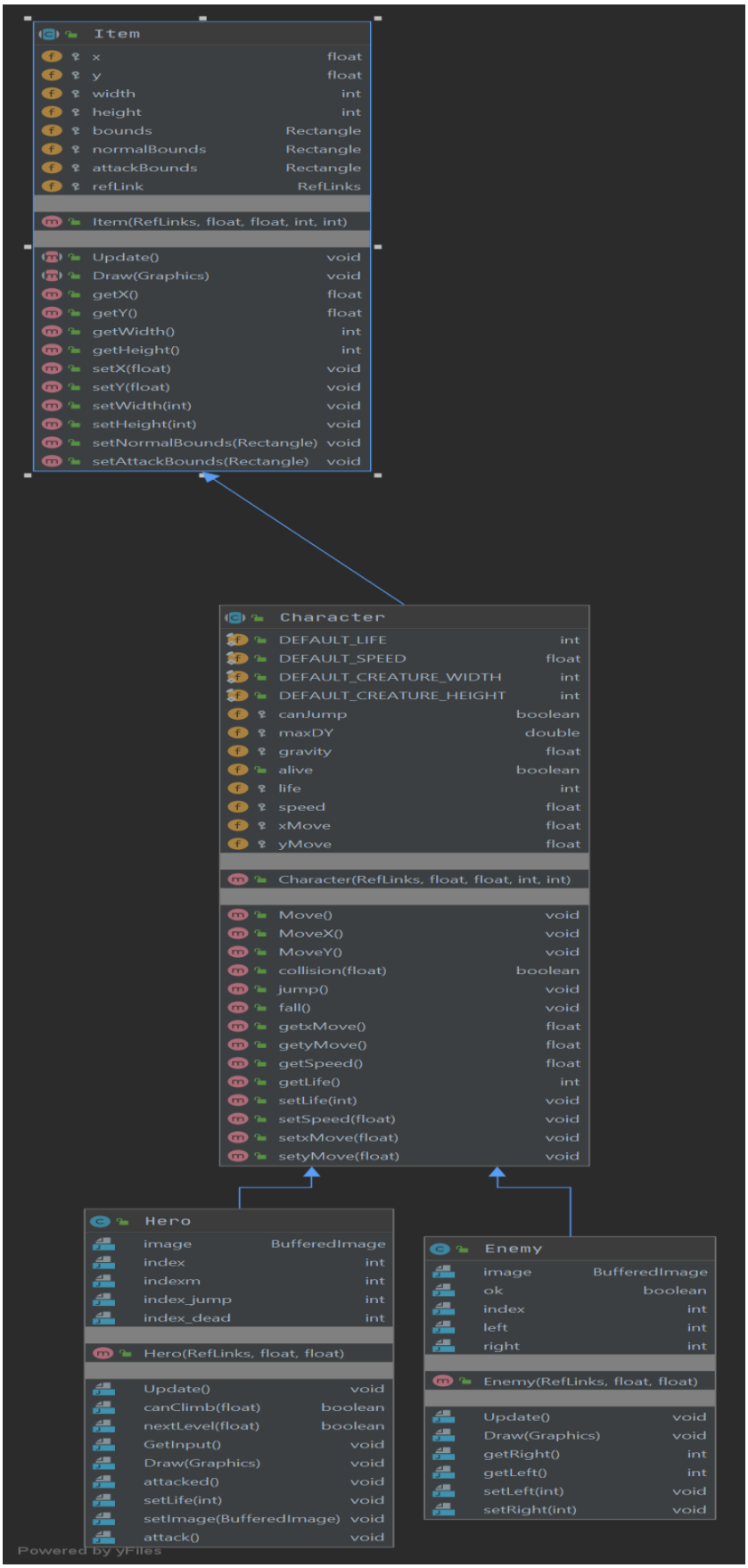
activate_level – specifica daca trebuie sa schimbam harta

Metode:

void LoadWorld() – functia va incarca in matricea tiles codurile dalelor primite de la functia int harta(int x, int y).

Tile GetTile(int x, int y) – functia va scoate dala ceruta din matricea tiles[][]

Void Draw(Graphics g) – functia va desena harta unde “g” este locul unde se realizeaza desenarea


Item
x float
y float
width int
height int
bounds Rectangle
normalBounds Rectangle
attackBounds Rectangle
refLink RefLinks
Item(RefLinks, float, float, int, int)
Update() void
Draw(Graphics) void
getX() float
getY() float
getWidth() int
getHeight() int
setX(float) void
setY(float) void
setWidth(int) void
setHeight(int) void
setNormalBounds(Rectangle) void
setAttackBounds(Rectangle) void
Character
DEFAULT_LIFE int
DEFAULT_SPEED float
DEFAULT_CREATURE_WIDTH int
DEFAULT_CREATURE_HEIGHT int
canJump boolean
maxDY double
gravity float
alive boolean
life int
speed float
xMove float
yMove float
Character(RefLinks, float, float, int, int)
Move() void
MoveX() void
MoveY() void
collision(float) boolean
jump() void
fall() void
getxMove() float
getyMove() float
getSpeed() float
getLife() int
setLife(int) void
setSpeed(float) void
setxMove(float) void
setyMove(float) void
Hero
image BufferedImage
index int
indexm int
index_jump int
index_dead int
Hero(RefLinks, float, float)
Update() void
canClimb(float) boolean
nextLevel(float) boolean
GetInput() void
Draw(Graphics) void
attacked() void
setLife(int) void
setImage(BufferedImage) void
attack() void
Enemy
image BufferedImage
ok boolean
index int
left int
right int
Enemy(RefLinks, float, float)
Update() void
Draw(Graphics) void
getRight() int
getLeft() int
setLeft(int) void
setRight(int) void
Powered by yFiles

Class Hero:

Este clasa care descrie personajul principal si caracterul folosit de user.

Extinde clasa Character

Membrii:

Image – referinta catre imaginea curenta a eroului

index – este index pentru vectorul Assets.runs

indexm – este index pentru vectorul Assets.runsm

index_dead -este index pentru vectorul Assets.dead

index_jump-este index pentru vectorul Assets.jumps

Metode:

void Update() – functia va actualiza pozitia si imaginea eroului in functie de evenimentele de la tastatura si obiectele intalnite in joc.

boolean canClimb(float s) – functia va verifica daca obiectul intalnit este un obiect catarabil.Aceasta va verifica colturile tile-ului care este un patrat si va vedea daca metoda climb returneaza “true”/”false”.

boolean nextLevel(float s) – functia va verifica identic cu functia canClimb doar ca va verifica pentru tile functia IsNext() pentru vedea daca obiectul indica trecerea la urmatorul nivel.

void GetInput() – functia va verifica ce taste au fost apasate si in ce pozitie va trebui eroul mutat. Va testa fiecare tasta de la GetKeyManager si daca directia in care eroul vrea sa mearga nu intalneste o coliziune.

void attacked() – functia va verifica intersectia cu fiecare inamic din vectorul refLink.getEnemies si daca eroul a intersectat inamicul acesta va pierde o viata.

void attack() – functia verifica daca dupa apasarea tastei “k” de atac acesta a intersectat un inamic.Daca a intersectat inamicul va muri.

Class Enemy:

Este clasa care descire inamicul eroului.

Extinde clasa Character.

Membrii:

image- referinta catre imaginea curenta a inamicului

ok – folosit pentru drumul inamicului

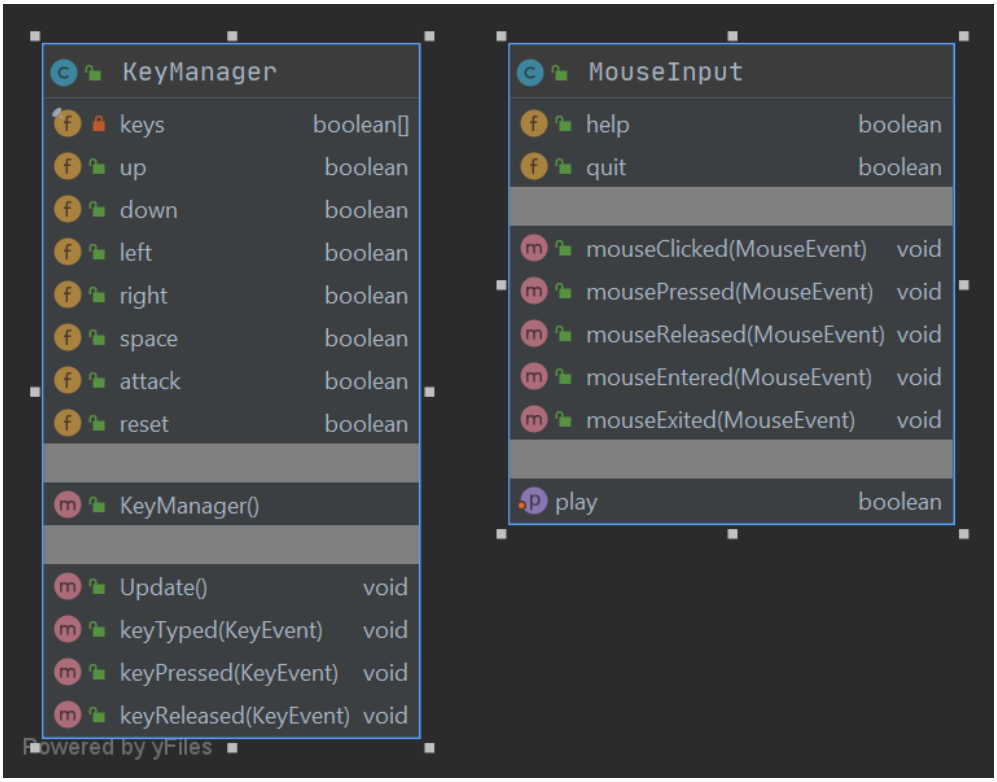
index- index pentru vectorul Assets.enemy

left- distanata maxima parcusa pe axa Ox in stanga

right – distanta maxima parcursa pe axa Ox in dreapta

Metode:

void Update() – functia actualizeaza pozitia si imaginea inamicului.Va ferifica in functie de pozitie ce animatie inamicul va folosi.



Class KeyManger:

Clasa verfica si contine informatii despre tastele apasate de user.

Membrii:

Keys – vector de flaguri pentru toate tastele[0-255]

up, down, left, right, space, attack, resert – parametrii disponibili user-ului pentru a interactiona cu jocul

void Update() – verifica daca tastele au fost apasate

void keyPressed(KeyEvent e) – daca a fost detectat un eveniment tasta respectiva va fi modificata(va primi “true”)

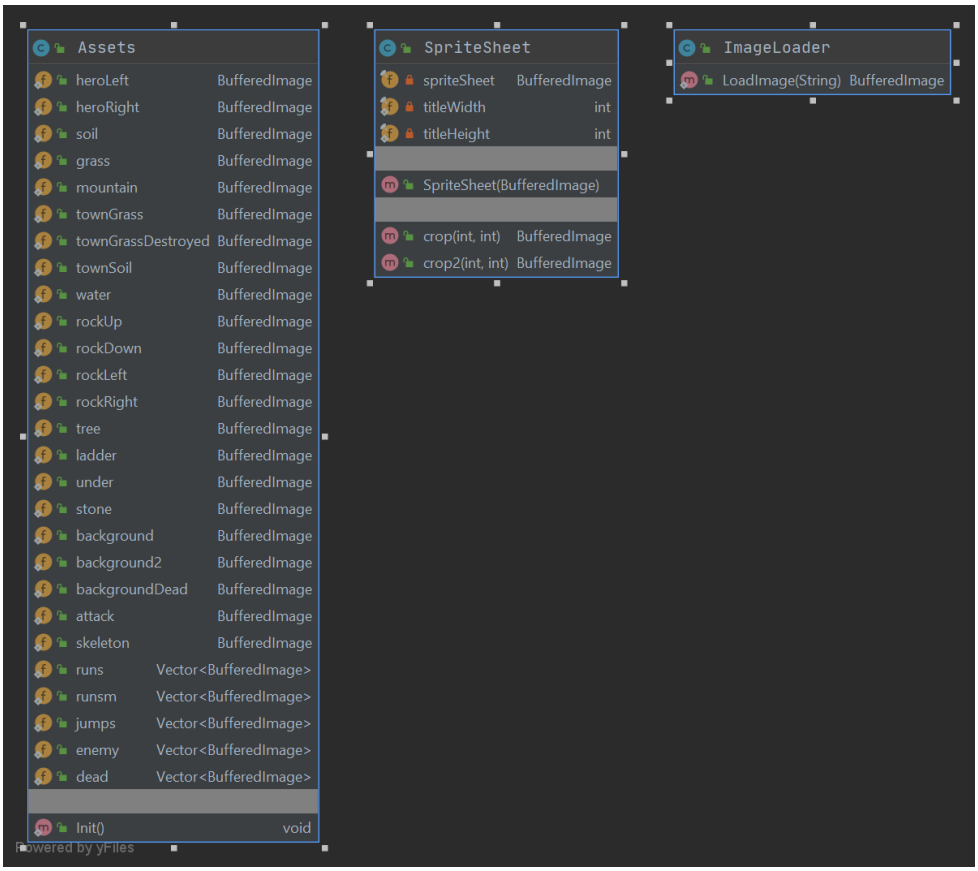
void keyReleased(KeyEvent e) – daca tasta nu mai este apasata aceasta va primi “false”

Class MouseInput:

Clasa verifica si retine informatii venite de la mouse.

Metode:

void mousePressed(MouseEvent e) – Verifica daca mouse-ul a fost apasat si in ce pozitie pe ecran.Functia testeaza in functie de dimensiunile butoanelor si pozitia mouse-ului ce tasta a fost apasata.

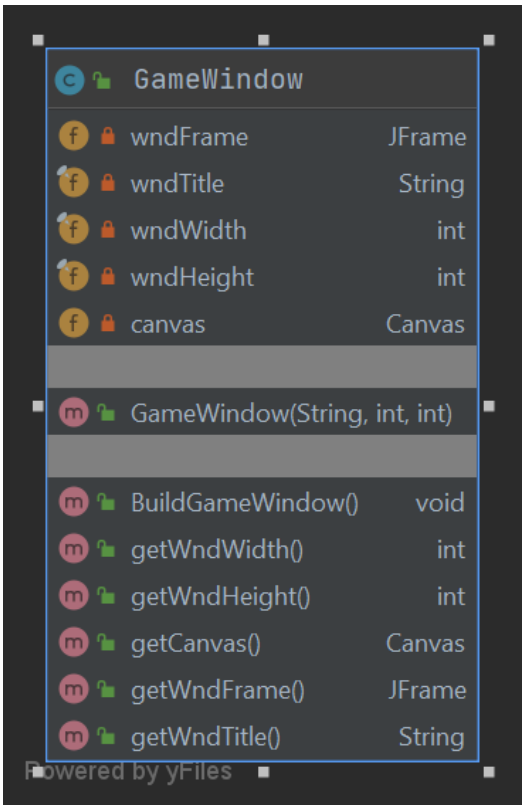


Class Assets:

Pastreaza imaginile pentru joc

Metode:

Static void Init() – functia initializeaza si creaza SpriteSheet-urile pentru joc



GameWindow:

Este clasa care creeaza fereastra jocului in care:

Membri:

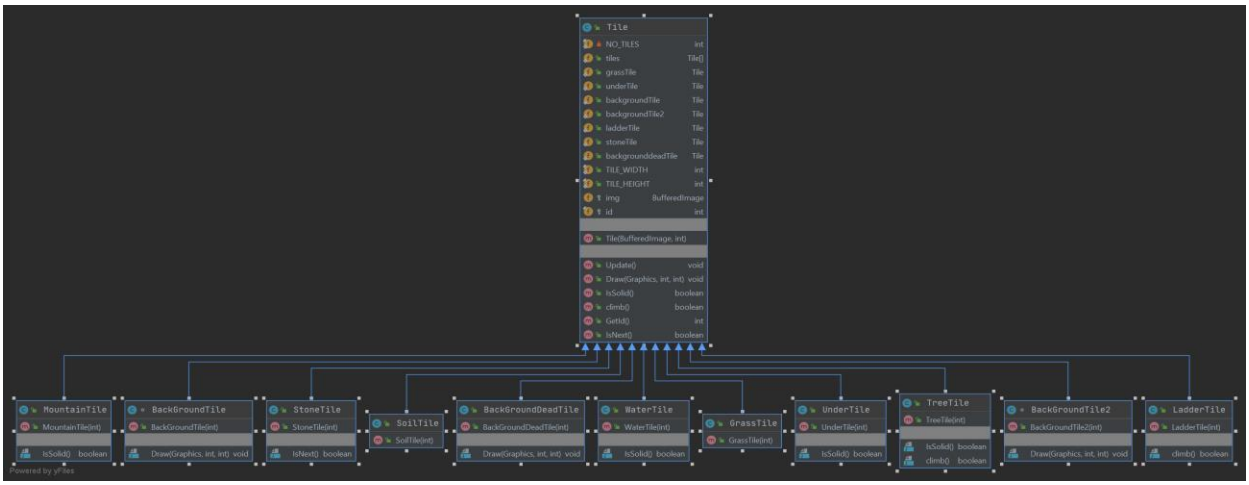
-*wndFrame* – fereastra jocului

-*wndtTitle* – titlul ferestrei

-*wndWidth* – latimea ferestrei

-*wndHeight* – inaltimea ferestrei

-*canvas* – tabloul unde se poate desena



Class Tile:

Are rolul de a retine toate dalele intr-un vector si gasirea acestora dupa un id

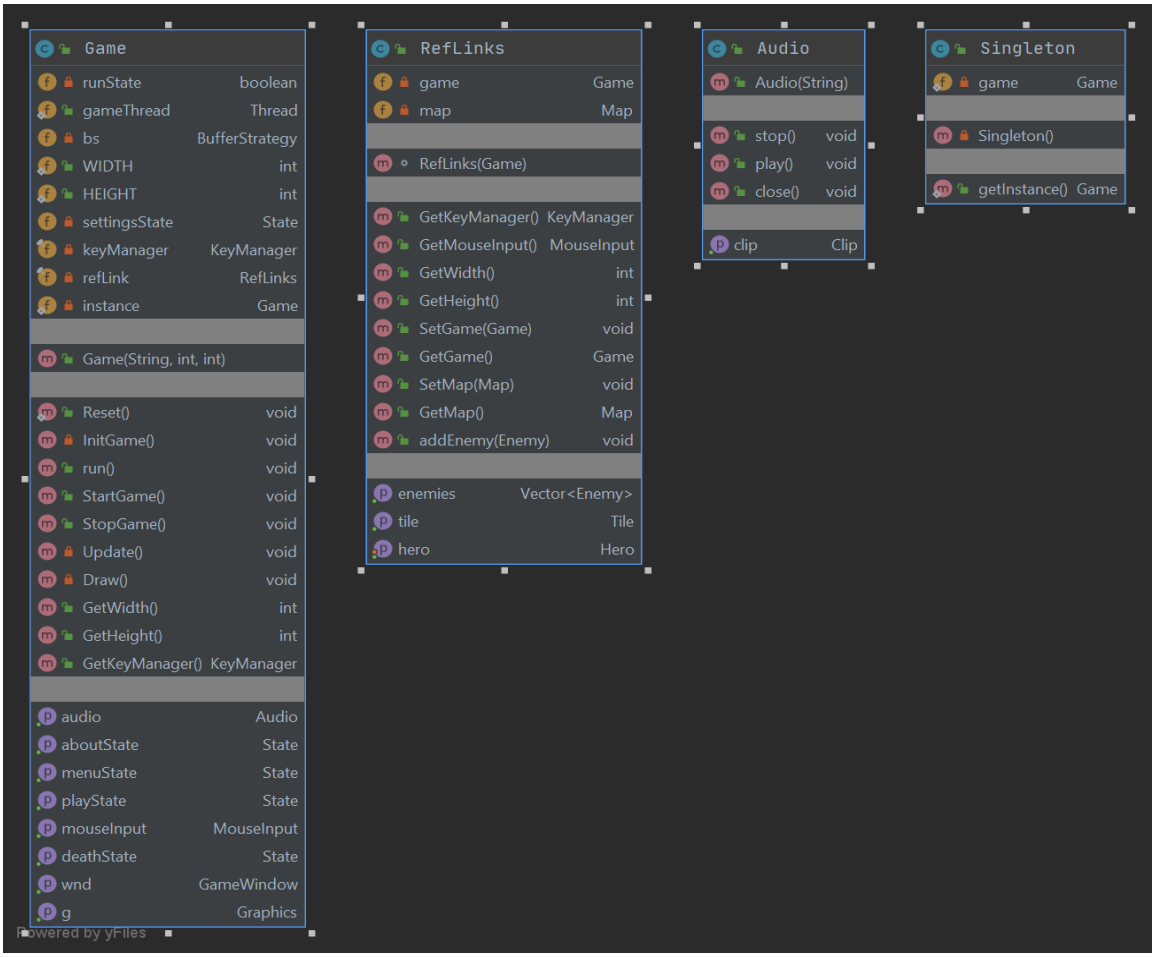
Tile[] tiles – vector pentru tipuri de dale

void Draw(Graphics g, int x, int y) – deseneaza in fereastra dala respective in functie de x si y

boolean IsSolid() – daca dala este solida sau nu

boolean climb() – daca dala este catarabila sau nu

boolean IsNext() – daca dala indica trecerea la nivelul urmator



Class Game:

Creeaza jocul,starile,threadul pentru joc

Membrii:

Wnd- fereastra in care se va desena jocul

runState- flag cu starea firului de executie

gameThread- referinta catre thread-ul de update si draw al ferestrei

WIDTH- latime joc

HEIGHT-inaltime joc

G – contextual graphic

void InitGame() – functia construieste fereastra jocului si starile. Va selecta starea cu care jocul incepe

synchronized void StartGame() – creeaza firul de executie si porneste jocul

void Update() – functia verifica starea tasteleor si apeleaza functia Update() pentru starea curenta

void Draw() -functia reprezinta operatia de desenare a jocului

Class RefLinks:

Clasa are rolul de a retine referintele celorlalte obiecte pentru a putea fi mai usor de accesat de diferite clase.

Membrii:

Game – referinta catre obiectul Game

Map – referinta catre harta curenta

Hero – referinta catre erou

Enemies – referinta catre inamici

Class Audio:

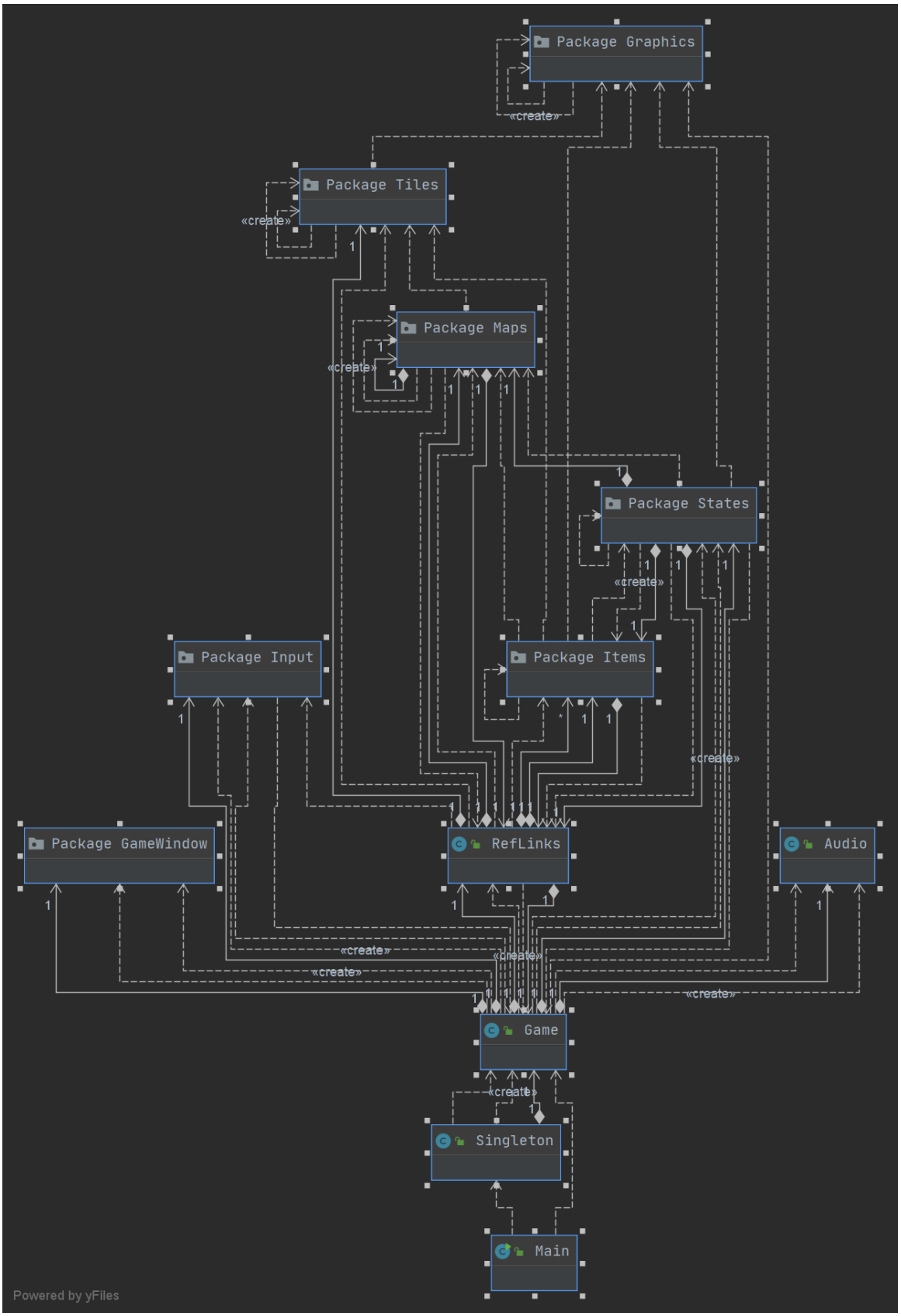
Clasa are rolul de pune sunet pe fundal in timpul jocului

Membrii:

Clip – referinta la clipul musical

Class Singleton:

Pentru a avea o singura instanta a clasei Game


Package Graphics
«create»
Package Tiles
«create»
1
Package Maps
«create»
1
1
1
1
1
1
Package States
«create»
1
1
1
1
1
Package Input
1
Package Items
*
1
1
«create»
Package GameWindow
1
RefLinks
1
1
Audio
1
«create»
«create»
«create»
«create»
1
1 1 1 1 1 1
Game
«create»
1
Singleton
Main
Powered by yFiles