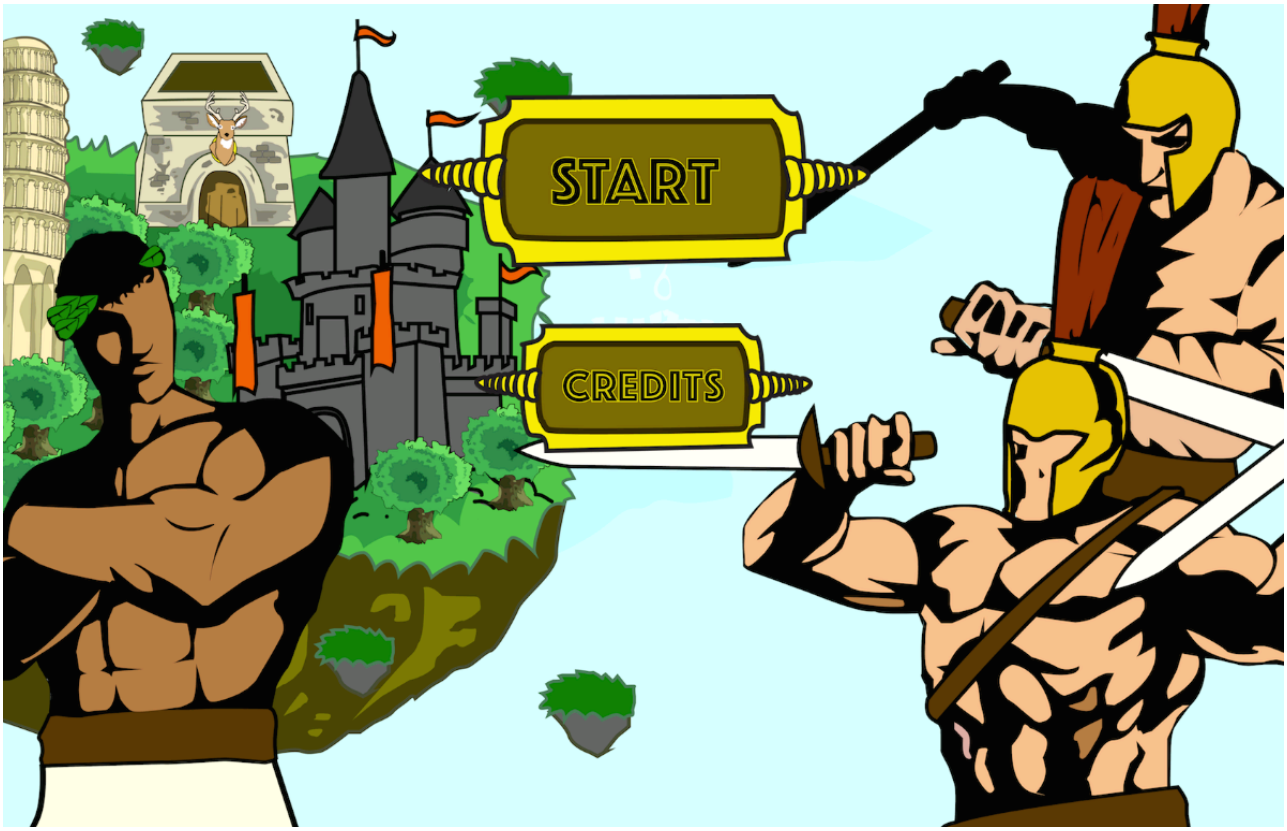
TowerDef Game Engine



PROIECT REALIZAT DE ELEVII FLOREA MĂLINA ȘI CAȘUNEANU ANDREI

PROFESOR COORDONATOR EMANUELA CERCHEZ

COLEGIUL NAȚIONAL EMIL RACOVIȚĂ IAȘI

Cuprins

I. Introducere

Ideea de a face un Game Engine pentru jocurile de tip Tower Defense a apărut în momentul în care ne-am dat seama cât de ușor poate fi realizat un joc în momentul în care ai o bază solidă și ușor de utilizat. Am mai implementat jocuri, în schimb, erau greu upgradabile din cauza claselor făcute special pentru jocul respectiv. Astfel ne-am gândit să facem un game engine pentru jocurile de tip Tower Defense pentru Android.

Jocurile de tip Tower Defense sunt o subramură a jocurilor video de strategie al cărui scop este de a opri inamicii din a ajunge într-un anumit punct de pe hartă. Inamicii sunt opriți de diferitele turnuri construite de jucător, care trag in inamici în timp ce aceștia trec pe lângă ele. Inamicii și turnurile au abilități, costuri și upgrade-uri variate, în funcție de nivel.

Game Engine-ul se adresează programatorilor cu experiența în programarea jocurilor de acest tip, întrucât are la bază programare orientată pe obiecte și este scris in Java.

Am gândit design-ul claselor astfel încât fiecare să se ocupe strict de acțiunile ei. Acest mod de lucru a dus la o aplicație foarte modularizată și ușor de modificat. Deasemenea datorită comportamentului strict al claselor, sursa unui eventual bug este localizată foarte eficient. Clasele au fost concepute cu scopul de a fi cât mai universale.

II. Elemente de implementare

1. Clase pentru nivel

SuportTurn

constructor

```
public SuportTurn(Context context, Bitmap framebuffer, int x, int y, int width,
int height);
```

parametrii constructorului

Context context = informații despre aplicație și mediu, necesare pentru crearea altor elemente
Bitmap framebuffer = Bitmap-ul pe care se vor desena obiectele clasei
int x = poziția pe axa OX
int y = poziția pe axa OY
int width = lățimea
int height = înălțimea

detectarea click-urilor

```
public int click(MotionEvent event);
```
Este apelat când apare un eveniment din partea utilizatorului.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

MotionEvent event = acțiunea utilizatorului
int rezultat = codul acțiunii făcutedacă este cazul (este din clasa EventTypes)

```
protected int onTouch(MotionEvent event);
```
MotionEvent event = acțiunea utilizatorului
int rezultat = codul acțiunii făcutedacă este cazul (Este din clasa EventTypes)

Este apelată de fiecare data de către funcția click. Interpretează event-ul și alege acțiunea potrivita evenimentului primit ca parametru. Este recomandat să fie supraîncărcată funcția în clasele derivate pentru a corespunde cu acțiunile respectivei clase.

desenarea pe framebuffer

```
public void paint();
```
Este apelată când este necesară desenarea obiectului și a componentelor acestuia.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
protected void onDraw();
```
Desenează pe framebuffer obiectele și componentele acestora.
Este recomandat să fie supraîncărcată funcția în clasele derivate pentru a corespunde obiectele ce trebuie să fie desenate cu clasa respectiva.

updatarea

```
public void update();
```
Este apelată când este necesară updatarea obiectului și a componentelor acestuia. Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
protected void onUpdate();
```
Updatează obiectul și a componentele acestuia. Este recomandat să fie supraîncărcată funcția în clasele derivate pentru a corespunde obiectele ce trebuie să fie desenate cu clasa respectiva.

funcții

```
public boolean push(int x, int y);
```
Aflădacă un click (pe poziția x și y) se intersectează cu suportul de turn.

```
public void deschide();
```
Deschide butonul pentru selectarea turnului.

```
public void inchide();
```
Închide butonul pentru selectarea turnului.

Turn (clasă abstractă)

funcții

```
public void paint();
```
Este apelată când este necesară desenarea obiectului și a componentelor acestuia.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
public int click(MotionEvent event);
```
Este apelat când apare un eveniment din partea utilizatorului.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
public void update();
```
Este apelată când este necesară updatarea obiectului și a componentelor acestuia.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
protected abstract void onDraw();
```
Trebuie definită în clasele derivate.

```
protected abstract int onTouch(MotionEvent event);
```
Trebuie definită în clasele derivate.

```
protected abstract void onUpdate();
```
Trebuie definită în clasele derivate.

TurnArcasi, TurnSulitasi, TurnTun

(clase derivate din clasa Turn)

include funcțiile paint(), update(), onDraw(), onUpdate(), click(event) și onTouch(event)

TurnArcasi – conține Arcasi
TurnSulitasi – conține Sulitasi
TurnTun – conține Tun

constructor

```
TurnArcasi(Bitmap framebuffer, Context context, int xbaza, int ybaza);
TurnSulitasi(Bitmap framebuffer, Context context, int xbaza, int ybaza);
TurnTun(Bitmap framebuffer, Context context, int xbaza, int ybaza);
```

parametrii constructorului

Context context = informații despre aplicație și mediu, necesare pentru crearea altor elemente
Bitmap framebuffer = Bitmap-ul pe care se vor desena obiectele claselei
int xbaza = poziția bazei turnului pe axa OX
int ybaza = poziția bazei turnului pe axa OY

Arcas, Sulitas, Tun (implementeaza interfața Collidable)

Include funcțiile paint(), update(), onDraw() și onUpdate()
Collidable = se poate lovi de alte elemente de tip Collidable

Arcas – conține Arc
Sulitas – conține Sulita
Tun – conține Ghiulea

constructor

```
Arcas(Bitmap framebuffer, Context context, int xbaza, int ybaza);
Sulitas(Bitmap framebuffer, Context context, int xbaza, int ybaza);
Tun(Bitmap framebuffer, Context context, int xbaza, int ybaza);
```

parametrii constructorului

Bitmap framebuffer = Bitmap-ul pe care se vor desena obiectele claselei
int xbaza = poziția bazei turnului pe axa OX
int ybaza = poziția bazei turnului pe axa OY

funcții

```
protected boolean canAttack();
```
Verificădacă dacă arcașul/sulițașul/tunul poate ataca
(a trecut destul timp de la ultimul atac astfel încât să mai atace o data un inamic)

```
protected void schimbaAnim(Sageata sageata);
```
Schimbă animația arcașului în funcție de direcția în care e orientat când trebuie să lanseze o săgeată (se orienteaza în direcția săgeții)

```
public void colidedWith(Collidable object);
```
Funcția definită din interfața Collidable care include acțiunile necesare în cazul coliziunii cu un obiect primit ca parametru.

funcții de tip get/set sau interogare a stărilor

```
public int getX();
public int getY();
public int getW();
public int getH();
public int getRadius();
public voit setRadius(int radius);
public int getDamage()
public int getAttackSpeed();
public boolean isMoving();
```

Proiectil (clasă abstractă)

Include funcțiile paint() și update() și onDraw() și onUpdate()

```
public abstract int getAngle();
```
funcția care returneaza unghiul la care este proiectilul

SagetaDreapta, Sulita, Ghiulea

(derivate din clasa abstractă Proiectil)

Include funcțiile paint() și update() și onDraw() și onUpdate()

constructor

```
public SageataDreapta(Bitmap framebuffer, Context context, int x, int y, Inamic inamic);
```

int x, y = x-ul și y-ul de început
inamic = obiectul de tip Inamic care reprezintă ținta la care trebuie să ajung

funcții de tip get/set/interogare

```
public void setDamage(int damage);
public int getDamage();
public int getAngle();
public void setVisible();
public boolean isVisible();
```

Pluton

Include funcțiile paint(), onDraw(), update(), onUpdate(), click(),
Reprezintă un număr de inamici care vin toți pe acelasi drum specific plutonului dupa inceperea nivelului (fiecare avand un anumit delay)

constructor

```
public Pluton(int nr_inamici);
```
creează un vector de inamici de dimensiune nr_inamici

funcții

```
public void start();
```
Incepe acțiunea plutonului: de obicei se apelează la începutul nivelului

```
public void setInamicPoz(Inamic inamic, int poz);
```
Pune inamic în vectorul de inamici pe poziția poz

funcții de tip get/set/interogare

```
public void setNrInamici(int nr_inamici);
public int getNrInamici();
public void setDelay();
public int getDelay();
public void setDrum(Drum drum);
public Drum getDrum();
public Inamic getInamicPoz(int poz);
```

Drum

constructor

```
public Drum(Context context, String nume);
```
Citește un fișier cu numele nume în formatul: pe prima linie un număr, n; pe celelalte fișiere câte 2 numere reprezentand coordonatele x și y

```
public Drum(Context context, int lg);
```
Creează un vector gol de puncte de lungime lg în care se pot adaugă separat mai târziu.

funcții

```
public void addPoint(Point point, int poz);
```
Adaugă un punct point pe poziția poz în vector (daca poz este -1, adaugă pe ultima poziție disponibilă)

```
public int getLg();
```
public Point getPoint (int ind);

Inamic (clasă abstractă)

Include funcțiile paint(), onDraw(), update(), onUpdate()

funcții

```
public void hitYa(int damage);
```
Funcția care seteaza comportamentul inamicului în caz ca este lovit cu un anumit damage

funcții de tip get/set/interogare

```
public void setX(int x);
public void setY();
public void setXbaza(int x);
public void setYbaza(int y);
public void setViata(int viata);
public void setAtac(int atac);
public void setArmour(int armour);
public void setDrum(Drum drum);
public void setViteza(int viteza);
public void setDelay(int delay);

public int getX();
public int getY();
public int getViata();
public int getAtac();
public int getArmour();
public Drum getDrum();
public int getViteza();
public int getDelay();
public int getW();
public int getH();
public boolean isDead();
```

2. Clase pentru meniu

Screen (clasă abstractă)

include funcțiile paint(), update(), onDraw(), onUpdate(), click(event) și onTouch(event)
Clasă abstractă care conține grafica ecranului.

Meniu, Start, Credits, Levels, Options, Help

(derivate din Screen)

include funcțiile paint(), update(), onDraw(), onUpdate(), click(event) și onTouch(event)
conțin grafica și elementele necesare ecranului (background și butoane)

constructor

```
public Meniu(Bitmap framebuffer, Context context);
public Start(Bitmap framebuffer, Context context);
public Credits(Bitmap framebuffer, Context context);
public Levels(Bitmap framebuffer, Context context);
public Options(Bitmap framebuffer, Context context);
public Help(Bitmap framebuffer, Context context);
```

Buton

constructor

```
public Buton(Bitmap framebuffer, String nume, int x, int y, Context context,
boolean fixed);
```
creează un buton cu poza data ca parametru pe poziția x, y

```
public Buton(Bitmap framebuffer, int x, int y, int w, int h, Context context,
boolean fixed);
```
creează un buton invizibil pe poziția x, y cu lățimea w și inaltimea h

funcții

```
public boolean push(int x, int y);
```
testează dacă butonul a fost apasat

```
public void paint();
```
desenează butonul

funcții get/set/interogatoare

```
public int getW();
public int getH();
public void setVisible(boolean visible);
public boolean isVisible();
public void setActive(boolean active);
public boolean isActive();
```

3. Clase aditionale

Renderer

desenează totul pe ecran

funcții

```
public void run();
```
conține loop-ul grafic al jocului

```
public void resume();
public void pause();
public void setRunning(boolean running);
```

Updater

updatează totul pe ecran

funcții

```
public void run();
```
conține loop-ul care updatează jocul

```
public void resume();
public void pause();
public void setRunning(boolean running);
```

Joc

el este activity-ul care conține tot și care intercepteaza event-urile de pe ecran și le transmite mai departe

funcții

```
public void paint();
```
Este apelată când este necesară desenarea obiectului și a componentelor acestuia.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
public int click(MotionEvent event);
```
Este apelat când apare un eveniment din partea utilizatorului.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
public void update();
```
Este apelată când este necesară updatarea obiectului și a componentelor acestuia.
Nu este recomandat să fie supraîncărcată funcția în clasele derivate.

```
protected void onDraw();
```
Trebuie definită în clasele derivate.

```
protected int onTouch(MotionEvent event);
```
Trebuie definită în clasele derivate.

```
protected void onUpdate();
```
Trebuie definită în clasele derivate.

```
public void onResume();
public void onPause();
public void setScreen(Screen screen);
```

Graphics

este folosită pentru desenarea pe framebuffer

constructor

```
public Graphics(Bitmap framebuffer, Context context, boolean fixed)
```
fixed este truedacă grafica isi pastreaza poziția pe ecran orice ar fi

funcții

```
public void drawImage(Image image, int x, int y, int xs, int ys, int ws, int
hs);
```
xs, ys sunt coordonatele sursei

ws, hs sunt mărimile pozei sursa

```
public Image rotateImage(Image image, int angle);
public Image openImage(String nume);
public Bitmap drawText(String text, int textWidth, int textSize);
```

Miscare

este folosită pentru mișcarea obiectelor pe ecran

constructor

```
Mișcare(int xCurr, int yCurr, int xDest, int yDest, int viteza);
```
creează o mișcare noua din xCurr, yCurr în poziția xDest, yDest cu viteza pe secundă viteza

funcții

```
public void update();
protected void onUpdate();
```

funcții get/set/interogatoare

```
public boolean isRunning();
public double getx();
public double gety();
```

Animatie

este folosită pentru mișcarea obiectelor pe ecran

constructor

```
Animatie(Bitmap framebuffer, Context context, String nume, int indW, int indH,
boolean fixed);
```

nume - numele pozei

indW - numărul de poze pe latime

indH - numărul de poze pe înaltime

fixed - este true dacă poziția este fixă și false altfel

funcții

```
public void update();
protected void onUpdate();
```

funcții get/set/interogatoare

```
public void setX(int x);
public void setY(int y);

public void setLooping(boolean looping);
```

seteaza dacă animatia o va lua de la capat dupa ce termina sau nu

```
public void setVisible(boolean visible);
public void setVisibleAfter(boolean visibleAfter);
```

seteaza dacă animatia va ramane vizibila dupa ce termina sau nu

```
public void start();
public boolean isRunning();
public boolean isVisible();

public int getW();
public int getH();
```

```
public void paint();
public void update();
```

Collidable

este o interfața pentru elementele care se pot lovi între ele

CollisionDetector

verifică coliziunea dintre obiectele Collidable

funcții

```
public static void init();
```

```
public static void checkCollisions();
```
verifică coliziunile dintre obiecte și le anunta ca s-au lovit

```
public static void addObject(Collidable object);
```
adaugă un obiect Colidable la lista de obiecte

ConstructorPluton

verifică coliziunea dintre obiectele Collidable

constructor

```
public ConstructorPluton(Bitmap framebuffer, Context context, String nume,
Drum[] drumuri);
```
interpretează un fișier de plutoane și primeste lista drumurilor posibile

funcții get/set/interogatoare

```
public int getNrPlutoane();
public Pluton getPlutonPoz(int poz);
```

ConstructorPluton

verifică coliziunea dintre obiectele Collidable

constructor

```
public ConstructorPluton(Bitmap framebuffer, Context context, String nume,
Drum[] drumuri);
```
interpretează un fișier de plutoane și primeste lista drumurilor posibile

funcții get/set/interogatoare

```
public int getNrPlutoane();
public Pluton getPlutonPoz(int poz);
```

Ecran

conține dimensiunile ecranului: latime și inaltime

Scaler

este folosit pentru scalarea dimensiunilor jocului pentru ecranul telefonului mobil

funcții

```
public static int scale(int a);
```
returnează dimensiunea scalată a variabilei a

EventTypes

conține tipurile de event-uri posibile

Offset

conține offset-urile ecranului: x, y, scale

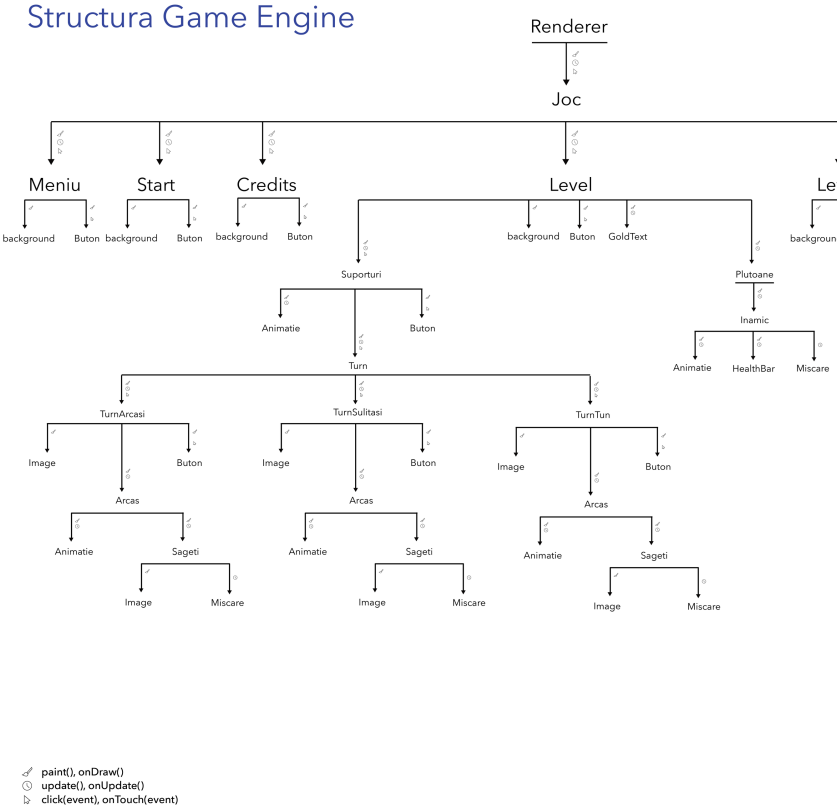
este folosit la scroll și zoom

Player

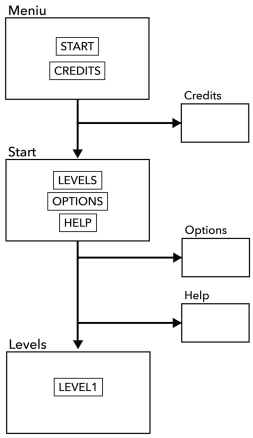
conține date despre player: suma de bani și numărul de vieti

III. Aplicație

Structura Game Engine



Structura Meniu



IV. Resurse

Elemente de implementare a aplicației

Aplicația are la bază principiile programării orientate pe obiecte și este programată în Java. Toate elementele grafice au fost făcute de noi folosind programul Adobe Ilustrator.

Resurse software

Android Studio
Android SDK
Java SDK

Bibliografie

http://developer.android.com/index.html
http://stackoverflow.com