Aufgabe: Fähigkeiten

Antonuccio Matteo, Moritz Lützendorf, Mazlum Cin

Codeanalsyse: Skill-System

Das SkillSystem ist für die Verwaltung von Fähigkeiten zuständig. In der Methode update() wird die Abklingzeit für alle Fähigkeiten reduziert, die von den aktiven Entitäten verwendet werden. Es werden nur Entitäten berücksichtigt, die eine SkillComponent haben. Die Methode wird in regelmäßigen Abständen aufgerufen, um sicherzustellen, dass die Fähigkeiten bereit sind, wenn der Spieler sie erneut verwenden möchte.

Codeanalyse: FireballSkill

Die Klasse FireballSkill ist eine Unterklasse der Klasse DamageProjectileSkill und repräsentiert die Implementierung des Feuerball-Skills. Der Feuerball-Skill wird mit einem Bildpfad, einer Geschwindigkeit, einem Schadensobjekt, einem Startpunkt, einer Zielauswahl und einer Abklingzeit initialisiert. Wenn der Feuerball auf ein Ziel trifft, wird der Schaden verursacht und das Ziel erhält entsprechende Schadenspunkte in der Kategorie Feuer.

Codeanalyse: XP-System

Die Klasse XPSystem implementiert ein XP-System, das für das Sammeln von Erfahrungspunkten und den Levelaufstieg des Spielers zuständig ist. Die update()-Methode durchläuft alle Entitäten, die eine XPComponent haben, um die Erfahrungspunkte zu aktualisieren. Wenn ein Spieler genug Erfahrungspunkte gesammelt hat, wird die performLevelUp()-Methode aufgerufen, um den Spieler auf das nächste Level zu bringen.

Beispielfähigkeit: Gedankenkontrolle

```
public class MindControlSkill extends MagicSkill {

 private static final float MANA_COST = 15f;

private static final int LEVEL_REQUIREMENT = 10;

 private static final float CAST_TIME = 2.5f;

public MindControlSkill() {

 super( "Gedankenkontrolle", "mind_control_icon.png", MANA_COST, LEVEL_REQUIREMENT,
CAST_TIME );

 }

public void cast(Point target) {

targetEntity = getEntityAt(target);
```

```
if (targetEntity != null && targetEntity.isMindControllable()) {
targetEntity.setControlledBy(getCaster()); } }

}

}
```

Beispielfähigkeit : Telekinse

```
public class TelekinesisSkill extends MagicSkill {

private static final float MANA_COST = 20f;

private static final int LEVEL_REQUIREMENT = 15;

private static final float CAST_TIME = 3.0f;


public TelekinesisSkill() {

   super(

         "Telekinese",

         "telekinesis_icon.png",

         MANA_COST,

         LEVEL_REQUIREMENT,

         CAST_TIME

   );

}


public void cast(Point target) {

   // Code zum Ausführen der Telekinese

   Entity targetEntity = getEntityAt(target);

   if (targetEntity != null && targetEntity.isTelekinetic()) {

      targetEntity.move(getRandomDirection());
```

```
    }
}

private Direction getRandomDirection() {
    // Code zum Zurückgeben einer zufälligen Richtung
}
```

Abstrakte Klasse:

```
public MagicSkill (String name, String iconPath, float manaCost, int levelRequirement, float castTime)
{
    this.name = name;
    this.iconPath = iconPath;
    this.manaCost = manaCost;
    this.levelRequirement = levelRequirement;
    this.castTime = castTime;
}

public String getName() {
    return name;
}

public String getIconPath() {
    return iconPath;
}

public float getManaCost() {
```

```
    return manaCost;
}

public int getLevelRequirement() {
    return levelRequirement;
}

public float getCastTime() {
    return castTime;
}

public abstract void cast(Point target);
```

HealingSkill
-healPercentage:float +HealingSkill(int,float,int,float)
+use():void

Sprintkill

-damagePerTick(): int

-projecttileSpeed:int

-projecttileRange:int

+SprintSkill(int,float,int,int,float)

+use():void

MagicSkill

~manaCost:int

~dmg:float

~level: int

+MagicSkill(int,float,int)

+use():void {abstract}

Aufgabe: Fallen

Beschreibung der Aufgabe

Dem Spiel werden mindestens drei Fall-Objekte hinzugefügt.

Beschreibung der Lösung

Unser Ziel ist es, Fall-Objekte zu implementieren, die automatisch aktiviert werden, wenn der Spieler sie berührt. Die Positionen werden zufällig bei jedem Levelwechsel festgelegt.

Es gibt drei Arten von Fallen: Giftfallen, Mausefallen und Lavafallen, diese beinhalten auch eine Textur. Bei den ausgelösten Fallen wird diese Farbe der Textur geändert, damit der Spieler weiß, dass diese schon ausgelöst wurde.

Die Giftfalle fügt dem Spieler Schaden zu, während die Mausefalle den Spieler für eine gewisse Zeit bewegungsunfähig macht. Bei der Lavafalle erleidet der Spieler ebenfalls für eine bestimmte Zeit Schaden.

Methoden und Techniken

Um sicherzustellen, dass die Falle nur vom Monster und nicht vom Spieler ausgelöst wird, verwenden wir entweder eine Abfrage des Namens des Spielobjekts oder des Spielobjekttyps. Die Zufällige Position wird mithilfe der Funktion "setRandomPos()" vom Typ Coordinate verteilt.

Die Klasse Trap wird von der Klasse FloorTile abgeleitet, da diese alle notwendigen Eigenschaften und Funktionen besitzt mit dem wir flexibel Texturen, Level etc. hinzufügen können.

Mit den Enums in DamageType bestimmen wir welche Art von Schaden der Spieler durch die Fallen erleidet.

Ansatz und Modellierung(Matze)

Wir erstellen eine Fallen-Klasse sowie eine Fähigkeiten-Klasse.

Die Fallen werden zufällig mit der Methode getRandoPos auf der Map generiert.

Monster haben zusätzlich folgende Grundeigenschaften: (Matze)

float cooldown: Die Falle wird immer reaktiviert.

Fähigkeiten und Fallen haben zusätzlich folgende Grundeigenschaften:

Public Hero(): Zusätzliche Skills hinzufügen

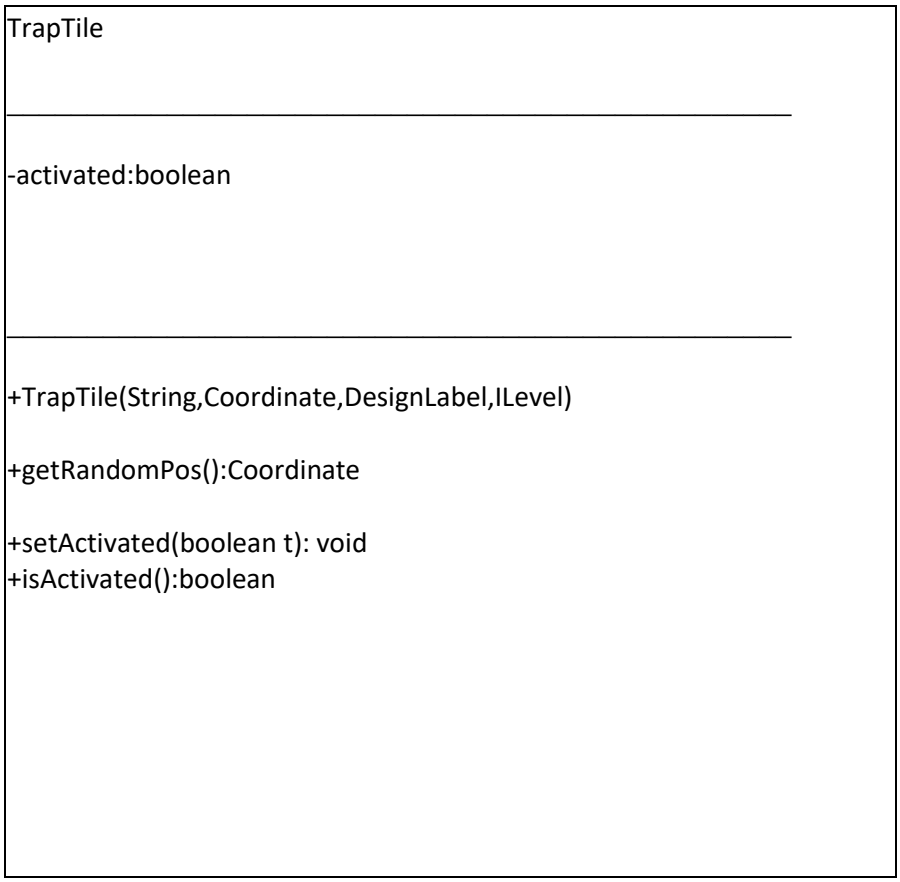
Bei DamageType: Einstellen wie viel der Held Schaden nimmt etwa Physical, Magic, Fire

Fallen haben zusätzlich folgende Grundfunktionen(Moritz)

public class Vector2D{

```
int x,y;
public Vector2D(int x,int y){
this.x = x;
this.y = y;
}
public Vector2D(){
}
}
public Vector2D getRandomPos(){
int mapX = Level.sizeX;
int mapY = Level.sizeY;
return new Vector2D((double)Math.random()*mapX+0,(double)Math.random()*mapY+0);
}
Falle a = new Falle();
a.Vector2D = getRandomPos();
public class Falle{
boolean activated;
int x;
int y;
public Falle(Vector2D vector){
x = vector.x;
y = vector.y;
}
}
```

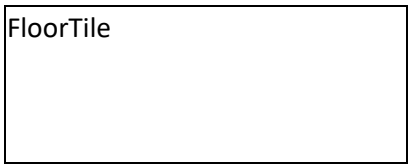
Daraus ergibt sich folgendes UML:



l

l

V



Beschreibung der konkreten Monster(Moritz)

1. Mausefalle

* erzeugt Schaden von 1-5 (Math.random())

* kann nur einmal ausgelöst werden

2. Giftfalle

* erzeugt Schaden von 5, außer bei einem Zauberspruch (Fähigkeiten: Spieler ist immun)

* kann nur einmal ausgelöst werden

Die Fallen werden beim Laden im Dungeon verteilt. Die Funktion getRandomPosition() verteilt die Fallen an einer zufälligen Position.