Alexey Khokhlov Michel Witt Ayaz Khudhur

Zyklus 2: Konzept

Inhalt

Freundlicher NPC: Geist (5p)

Beschreibung der Aufgabe

- Welche Komponenten braucht der Geist
- Geist Textur suchen
- Grabstein Textur suchen
- Wechsel Sichtbar/unsichtbar
- Verhalten: verfolgen/zufällig (wechselhaft)
- Beim Auffinden des Grabsteins belohnen/bestrafen

Beschreibung der Lösung

Realisierung eines Geistes und eines Grabsteins, der Geist soll sich zufällig sichtbar und unsichtbar durch den Dungeon bewegen oder den Spieler verfolgen, beim Auffinden des Grabsteins.

 Ein Geist der sich im Dungeon bewegt.

 Grab des Geistes.

Methoden und Techniken

Der Code wird mit Javadoc ggf. mit normalen Kommentaren (private) dokumentiert.

Für die Realisierung der NPCs wird das Type Object Pattern verwendet => Übergeordnete Muster Klasse.

Ansatz und Modellierung

Es wird eine abstrakte NPC-Klasse erstellt, von der die anderen NPC-Arten erben.
Die abstrakte NPC-Klasse enthält einige Methoden und Attribute, die alle NPCs haben, die speziellen Eigenschaften jedes NPC-Typs werden in eigenen Klassen realisiert, das gleiche gilt für Statische Elemente.

Grundeigenschaften

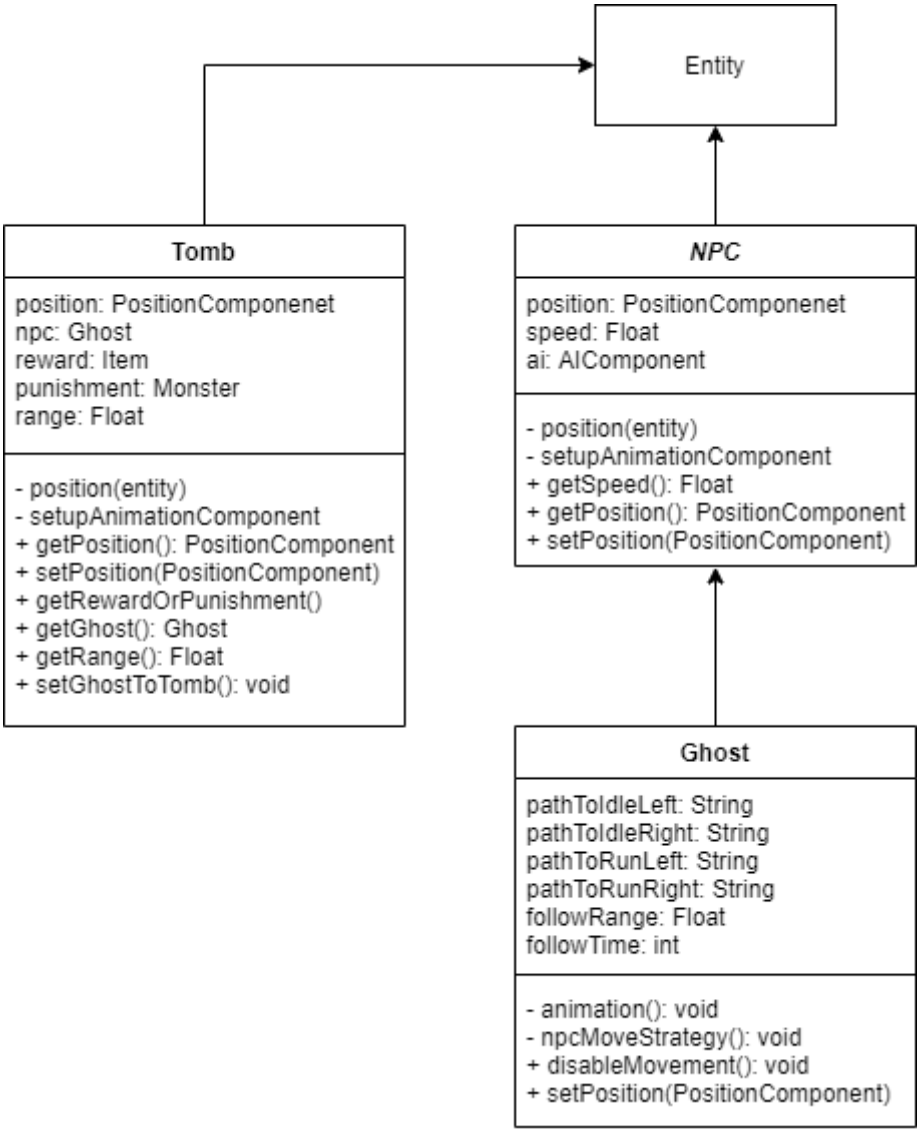
Alle NPCs:
- PositionComponent position // Position der NPCs
- Animation worldAnimation // Darstellung im Level

Statische Elemente:
- PositionComponent position // Position der Gegenstände
- Animation worldAnimation // Darstellung im Level

Geist:
- heroPosition // Held Position
- visible // sichtbar/unsichtbar
- IdleAI // Laufverhalten im Dungeon
- tomb // Grabstein

Grabstein:
- range // Wirkungsradius
- reward // Belohnung
- punishment // Strafe

UML



Beschreibung

Geist:

- speed 0.3f // (Hero speed)
- position // Random Dungeon Tile
- followRange // Random 0.01f – 0.1f
- followTime // Random: Verfolgungszeit des Helden

Grabstein:

- reward // Random Item
- punishment // Random Monster

Grabstein und Geist werden zufällig im Dungeon verteilt (1x Geist und 1x Grabstein pro Level).

Boss-Monster: Zombie (5p)

Beschreibung der Aufgabe

In dieser Aufgabe wird ein einzigartiges Boss-Monster implementiert: Ein Zombie.

Beschreibung der Lösung

Der Zombie-Boss wird gut erkennbar sein, um ihn herum werden 3-5 Zombies positioniert sein. Sobald der Spieler in seine Nähe kommt oder zu nahekommt, greift der Zombie-Boss an. Der Zombie-Boss wird mit zwei Fähigkeiten ausgestattet sein: Fähigkeit 1 verursacht Flächenschaden in Form von Giftschleim um den Boss herum. Wenn der Spieler sich im Schleim befindet, werden ihm Lebenspunkte abgezogen. Fähigkeit 2 bewirkt das Erscheinen von zufälligen Zombies, die den Spieler angreifen.

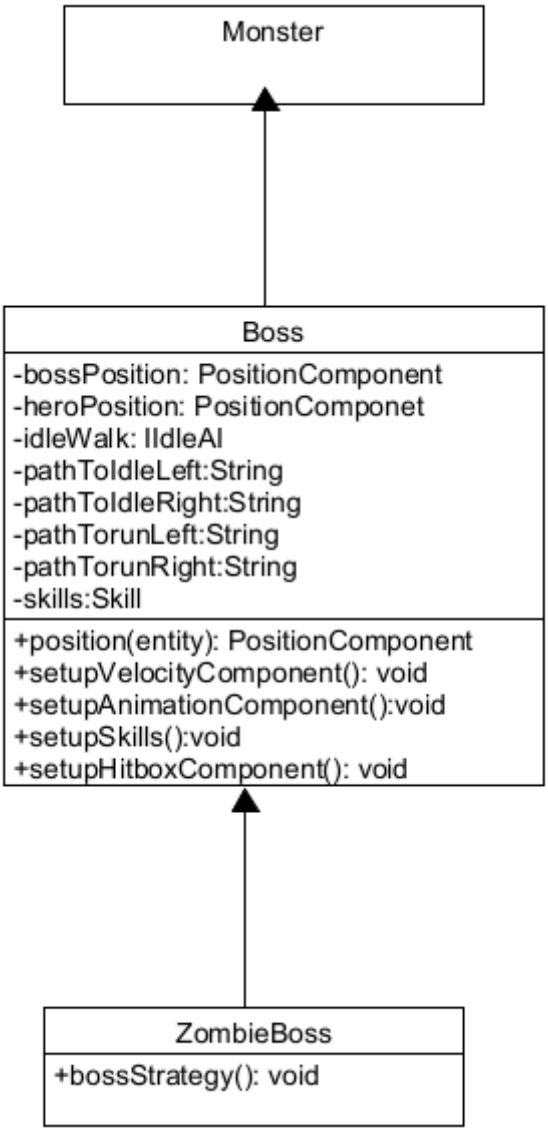
 Boss Monster

 Fähigkeit: Schleim

Grundeigenschaften

- bossPosition //Boss Position
- heroPosition // Held Position
- idle //Laufverhalten
- pathToIdleLeft
- pathToIdleRight
- pathTorunLeft
- pathTorunRight
- skill //Skills
- bossStrategy //Verhaltensmuster

UML


Monster
Boss
-bossPosition: PositionComponent
-heroPosition: PositionComponet
-idleWalk: IIdleAI
-pathToIdleLeft:String
-pathToIdleRight:String
-pathTorunLeft:String
-pathTorunRight:String
-skills:Skill
+position(entity): PositionComponent
+setupVelocityComponent(): void
+setupAnimationComponent():void
+setupSkills():void
+setupHitboxComponent(): void
ZombieBoss
+bossStrategy(): void

Beschreibung

Zombie Boss:

- 45hp * lvlCounter * 1.0 // Der Multiplikator wird um 0,5 erhöht
- 6dmg * 1.5 // Der Multiplikator wird um 0,5 erhöht
- Schleim dmg: (normal dmg * 3dmg) * 0,5 (für 4 Sekunden) // Der Multiplikator wird um 0,5 erhöht
- Zufälligen Zombies (Zombie dmg * 0,5) * lvlCounter (für 5 Sekunden) // Der Multiplikator wird um 0,5 erhöht
- 40xp * lvlCounter * 1.0 // Der Multiplikator wird um 0,5 erhöht

Zweite Phase (50% seiner Lebenspunkte):

- Alle Standardangriffe
    - werden mit 2 multipliziert
- Die Zeit für die einzeln Fähigkeiten sind deaktiviert.
- Ist der Spieler im Fernkampf, wird die zweite Fähigkeit als Schutz dienen.

Monster: Fernkampf (5p)

Beschreibung der Aufgabe

Die Aufgabe besteht darin, ein System zu entwickeln, das es ermöglicht, den bereits im Spiel implementierten Monstern ein Fernkampfverhalten zu geben. Der Fernkampf soll vorerst aus drei Projektilen bestehen, welche unterschiedliche Flugbahnen haben. Außerdem soll der Held nach einem Treffer ein wenig zurückgeschleudert werden.

Beschreibung der Lösung

Die Realisierung der der Aufgabe sieht vor, jedem der drei vorhandenen Monster, ein Fernkampf verhalten zu geben. Diese sollen sich aber unterscheiden.

- Kleiner Drache soll das Projektil „Feuerball“ bekommen
- Beißer soll das Projektil „Pfeil“ bekommen
- Zombie soll das Projektil „Boomerang“ bekommen

Dabei soll der Feuerball einfach geradeaus fliegen, bis dieser auf ein Hindernis oder den Helden trifft. Der Pfeil soll nur eine gewisse Reichweite zurücklegen können und wird auch „zerstört“, sobald dieser auf ein Hindernis oder auf den Helden trifft.

Boomerang soll auch nur eine gewisse Reichweite zurücklegen könne soll aber zum Monster zurückkehren, sprich der Boomerang könnte den Helden theoretisch auch zweimal treffen.

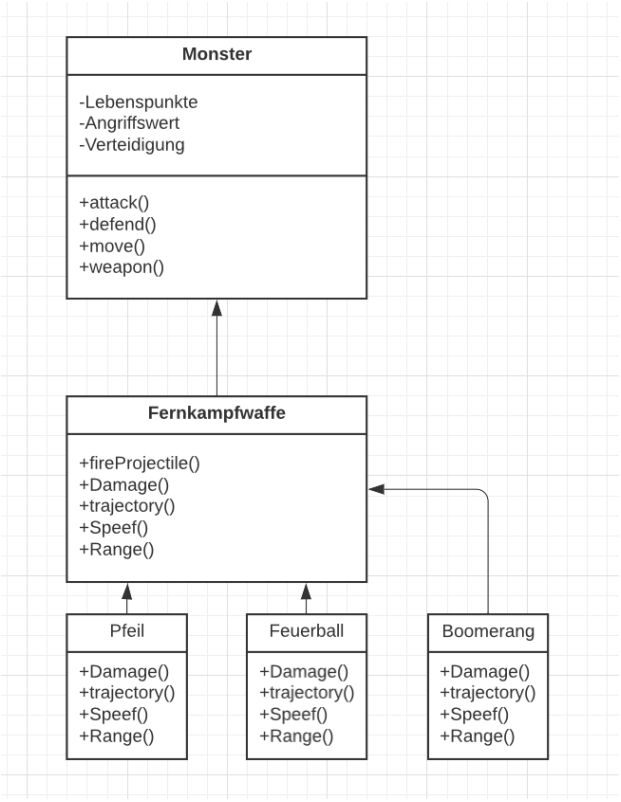
Beim Treffen eines Projektils, wird der Held entgegnend seiner Blickrichtung ein kleines Stück nach hinten versetzt.

Ansatz und Modellierung

Die Modellierung der verschiedenen Projektile soll auf der bereits bestehenden Klasse „Feuerball“ aufbauen.

- Grundeigenschaften
- Damage // Schadenswert des Projektils
- Trajectory // Die Flugbahn des Projektils
- Speed // Wie schnell das Projektil fliegt
- Range // Die Entfernung, die ein Projektil zurücklegen kann

UML



Beschreibung

Feuerball:
- Schaden 1.0f
- Geschwindigkeit von 0.8f
- Reichweite bis Kontakt mit Helden oder Wand
- Rückstoß bei Treffer 0.2f

Pfeil:
- Schaden 1.5f
- Geschwindigkeit von 0.6f
- Reichweite von 0.8f
- Rückstoß bei Treffer 0.1f

Boomerang:
- Schaden von 1.2f
- Geschwindigkeit von 1.0f
- Reichweite von 0.9f, danach soll der Boomerang in einer geraden Linie zurück zum Monster kehren
- Rückstoß bei Treffer 0.1f