Alexey Khokhlov Michel Witt Ayaz Khudhur

Zyklus 3: Konzept

Inhalt

Fähigkeiten (10p)

Aufgabe

- Levelaufstieg
- Fähigkeiten
- Codeanalyse

Codeanalyse

Skill-System:
Der SkillSystem erbt von ECS_System, in dem die Methode update() bei jedem Frame aufgerufen wird. Die Methode wird überschrieben und iteriert nur über die Entities, die ein SkillComponent besitzen. Außerdem ruft die Methode reduceAllCooldowns() auf und reduziert die Abklingzeit jeder Fähigkeit um 1 Frame.

Feuerball-Skill
Der FireballSkill erbt von DamageProjectileSkill, und im Konstruktor werden die notwendigen Informationen übergeben. Es wird eine ITargetSelection-Referenz übergeben, z. B. die Mausposition (Point). Ein Texturpfad für den jeweiligen Skill, die Geschwindigkeit für das Projektil, ein neues Damage-Objekt (Anzahl der zuziehenden Lebenspunkte, ein DamageTyp und ein Entity, die den Schaden verursacht, zum Beispiel der Spieler; kann auch null sein), die Projektile (Point), die ITargetSelection (wo es hin geht) und die Projektilreichweite.

XP-System
Der XPSystem erbt ebenso von ECS_System und ruft die Methode permanent update() auf. Es iteriert über alle Entities die ein XPComponent bestizen, im Zeile 14 wird ein Objekt von XPComponent erstellt, ein Attribut von typ long und ein While-Schleife. In der While-Schleife ist die Bedienung ob, xpLeft(bekommt die Werte von xp für nächsten Level) kleiner-gleich 0 ist), solang es wahr ist, wird die private methode performLevelUp() aufgerufen. Die aktuelle Stufe wird erhöht und die aktuellen EP zurückgesetzt.

Lösung

Der Spieler verfügt über drei Fähigkeiten, die ihm beim Fortschritt von Quests und Dungeons helfen sollen. Der Blitzschlag dient der Selbstverteidigung und beschwört einen Blitz, der Schaden an den Monstern verursacht. Die Verwandlungsfähigkeit unterstützt den Spieler bei Boss Monstern oder Monsterhorden. Die Verwandlung erfolgt in zwei Phasen: In der ersten Phase kann sich der Spieler in normale Monster verwandeln, während er in der zweiten Phase die Möglichkeit hat, sich in ein Boss Monster zu verwandeln. Die Gedankenkontrolle ist nützlich bei schwierigen Quests oder Bossen und liefert dem Spieler wichtige Informationen über das Ziel.

Beschreibung:

Blitzschlag:

- Aktivierung ab Level 3
    - Abklingzeit zwischen 5-10 Sekunden
    - Zieht 2 Ausdauerpunkte ab
    - Macht 5 Schadenpunkte
- Verwandlung:
    - Aktivierung ab Level 10 und 20
    - Level 10
        - In Zufälliges Monster verwandeln
        - Kostet 5 Ausdauerpunkt
    - Level 20
        - Zufälliges BossMonster
            - Zufall zwischen
                - bigOrk
                - bigDemon
                - bigZombie
        - 1.5f * Level – Ausdauerpunkte
        - 0.25f * Lebenspunkte – Zieht von Lebenspunkte zusätzlich und einmalig ab. (Nach jedem Aufruf)
- Gedankenkontrolle
    - Aktivierung ab Level 15
    - Voraussetzung
        - Kampf zwischen BossMonster und Spieler
    - Ausführung
        - Der BossMonster verrät welche Skills und Fähigkeiten es hat.
    - Verbrauch
        - 2f * Level – Ausdauerpunkte
        - 0.2f * Lebenspunkte – Zieht von Lebenspunkte zusätzlich und einmalig ab. (Nach jedem Aufruf)
- Wenn Ausdauerpunkte kleiner oder gleich 0 sind, wird von Lebenspunkte abgezogen. Bis der Spieler stirbt oder die Fähigkeit nicht mehr benutzt wird
- Die Ausdauerpunkte regenerieren sich nach einer gewissen Zeit oder durch Killen von BossMonstern oder Leveln.
- Bei Levelaufstieg erhält der Spieler 20% der Ausdauerpunkte

Blitzschlag:

Verwandlung:

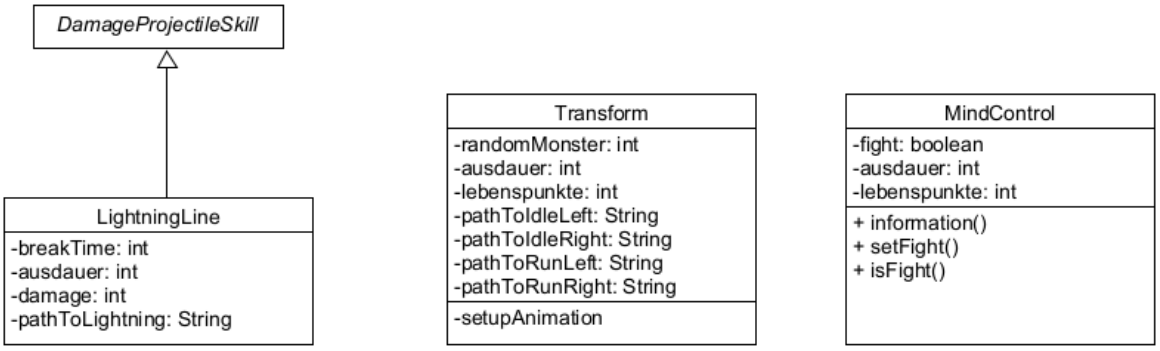
Grundeigenschaften

Blitzschlag
- breakTime
- ausdauer
- damage
- pathToLightning

Verwandlung
- randomMonster
- ausdauer
- lebenspunkte
- pathToIdleLeft
- pathToIdleRight
- pathTorunLeft
- pathTorunRight

Gedankenkontrolle
- isBoss

UML



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 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.

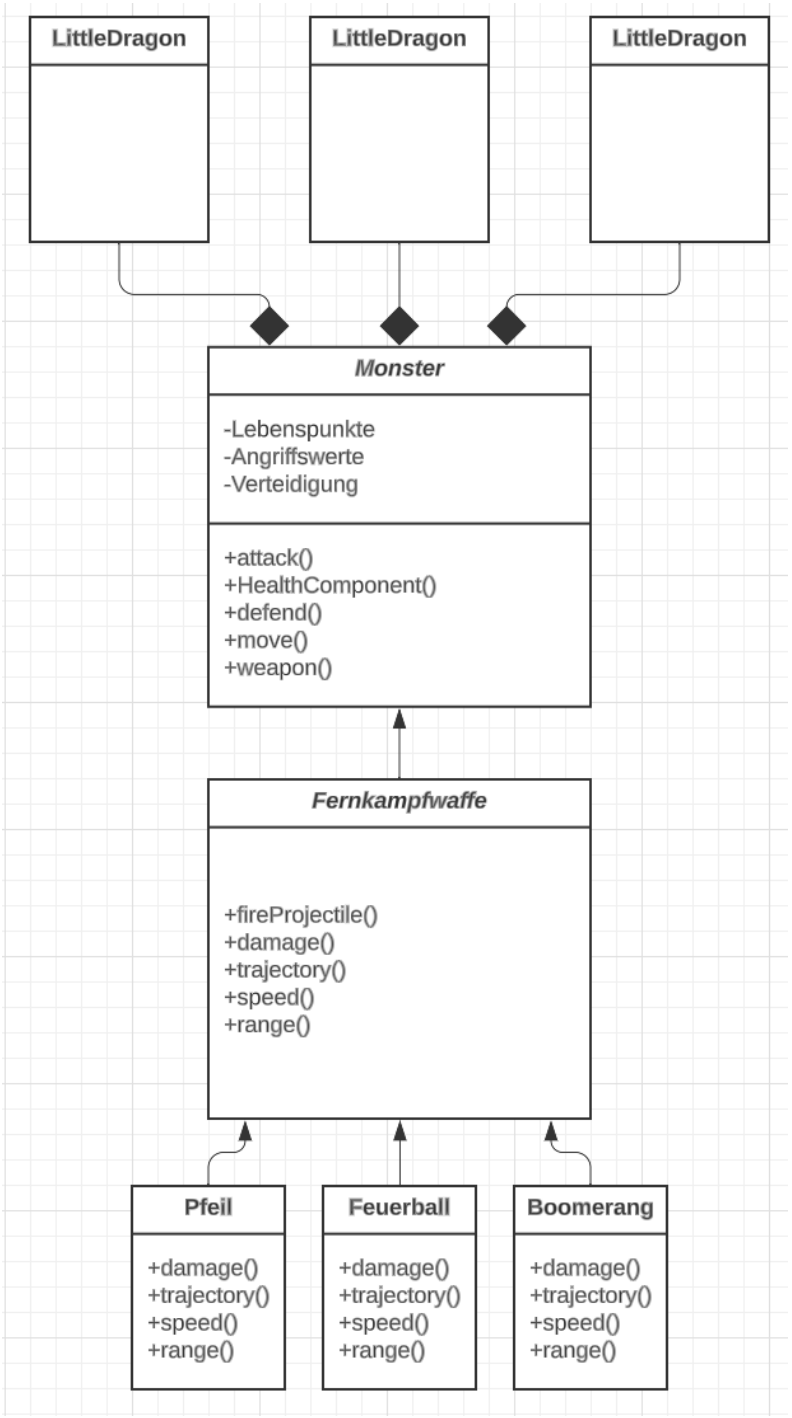
Beim Treffen eines Projektils, wird der Held entgegnend seiner Blickrichtung ein kleines Stück nach hinten versetzt.

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



- Fernkampfwaffe ist eine abstrakte Klasse, die die Grundstruktur eines Projektils definiert. Sie enthält die Eigenschaften damage, trajectory, speed und range. Diese Eigenschaften sind als private Attribute gekennzeichnet, um den direkten Zugriff von außen zu verhindern. Die genauen Werte für diese Attribute können in den abgeleiteten Klassen festgelegt werden.

- Monster repräsentiert ein Monster, das ein bestimmtes Projektil verwendet. Es hat eine Assoziation zu einem Projectile-Objekt, das das tatsächliche Projektil darstellt, das vom Monster verwendet wird.

- LittleDragon, Biter und Zombie sind abgeleitete Klassen von Monster, die jeweils ihr eigenes spezifisches Projektil haben. Diese Klassen erben die Beziehung zu Projectile von der Basisklasse Monster.

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

Zyklus 4: Konzept (ggf. Zyklus 3)

Monster und Held: Nahkampf (5p)

Beschreibung der Aufgabe

- Held/Monster sollen im Nahkampf Schaden verursachen
- Konzept Vorgaben erweitern
- Rückschlag bei Treffer

Beschreibung der Lösung

Realisierung eines Nahkampfsystems, der Held soll automatisch bei Kollision mit einem Monster angreifen, das gleiche gilt für das Monster, wenn es mit dem Helden kollidiert.

Wenn das Monster stirbt, soll der Held XP erhalten und ein zufälliges Item wird vom Monster gedroppt.

Methoden und Techniken

Der Code wird mit JavaDoc ggf. mit normalen Kommentaren (private) dokumentiert. Logger wird an sinnvollen Stellen gesetzt (Kollision, Schaden, Sterben, XP).

Es werden keine weiteren Klassen erzeugt, sondern nur Methoden implementiert.

Ansatz und Modellierung

Es wird ein Melee System eingebaut, das auf Kollisionen reagiert und automatische angriffe auslöst, es werden bei dem Helden und bei den Monstern nur neue Methoden hinzugefügt, die auf den Helden/Monster Werten von Zyklus 1 aufbauen.

In der update() Methode, die bei jedem Frame aufgerufen wird findet das eigentliche „Kampfsystem“ statt das mit hitEnemy() aufgerufen wird, wenn die jeweilige Hitbox (Hero oder Monster) eine Kollision wahrnimmt mit gegnerischem Part Held => Monster und Monster => Held wird ein Boolean Wert auf true gesetzt und der/das Held/Monster erhält schaden über Zeit, solange die Kollision besteht, wird die Kollision verlassen wird der Boolean Wert wieder auf false gesetzt.

Wenn ein Monster stirbt dropt es ein zufälliges Item (Zufälliges Item in Zyklus 1 schon implementiert) mit der onDeath() Methode, zusätzlich soll hier auch die XP für den Helden angerechnet werden.

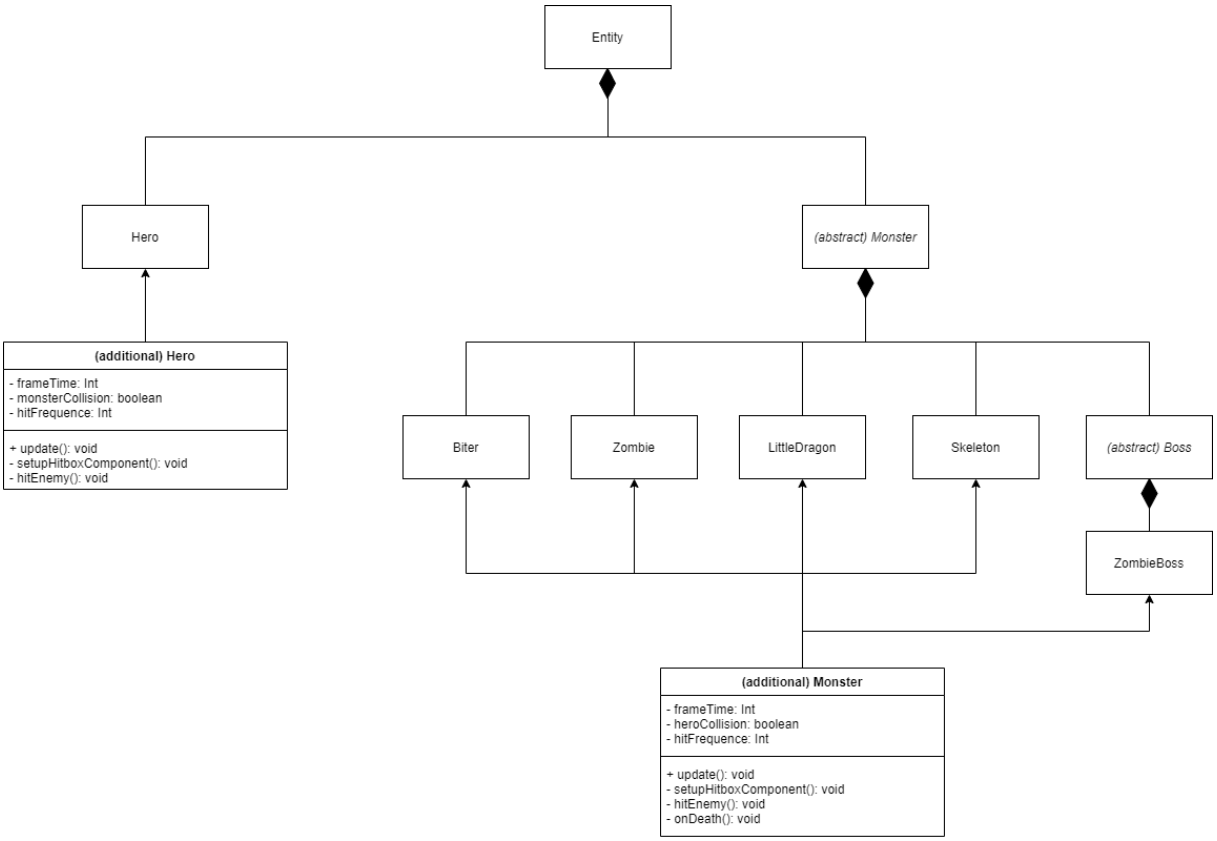
Damit der/das Held/Monster nicht in jedem Frame schaden erhält gibt es Hit Cooldown und Frame Zähler Variablen.

Grundeigenschaften

Held:
- Der in Zyklus 1 gesetzte Schaden wird für die Auto Kampf Funktion genutzt.
- Die in Zyklus 1 gesetzte int HP wird auf das HealthComponent geändert.

Monster:
- Der in Zyklus 1 gesetzte Schaden wird für die Auto Kampf Funktion genutzt.
- Die in Zyklus 1 gesetzte int HP wird auf das HealthComponent geändert.
- Das in Zyklus 1 gesetzte Item wird im Dungeon Level gedroppt.
- Die in Zyklus 1 gesetzte XP wird dem Helden angerechnet, wenn das Monster stirbt.

UML



Beschreibung

Held:
- frameTime // Anzahl der Bilder pro Sekunde
- monsterCollision // Variable für den auto hit zustand => Das auslösen von CollisionEnter setzt diesen Wert auf true (nur bei Monstern), CollisionLeave setzt den Wert auf False
- hitFrequence // Anzahl der Schläge pro Sekunde

Monster:
- frameTime // Anzahl der Bilder pro Sekunde
- heroCollision // Variable für den auto hit zustand => Das auslösen von CollisionEnter setzt diesen Wert auf true (nur beim Helden), CollisionLeave setzt den Wert auf False
- hitFrequence // Anzahl der Schläge pro Sekunde