Konzeptskizze für Zyklus 2

Beteiligte Person:

Name	Matrikelnummer
Marlon Cadel	1276264
Sascha Hahn	1320857

Inhaltsverzeichnis:

1 Konzeptskizze für “Basics-Fähigkeiten”

Beschreibung der Aufgabe

Der Hero, soll mit zwei Fähigkeiten erweitert werden. Eines der Fähigkeiten, soll Mana-Punkte verbrauchen und die Fähigkeiten, sollen den Monstern keine Schaden verursachen können.

Beschreibung der Lösung

Der Held soll sich mit der Taste R für 10 Sekunden unsichtbar machen können. Dabei erhält er eine neue halbtransparente Textur und Monster laufen nicht mehr gezielt zu ihm. Auf der Taste F wird der Held 1.5 mal schneller. Diese Fähigkeit verbraucht pro Sekunde einen Mana -Punkt. Alle 4 Sekunden bekommt der Held einen Mana-Punkt. Im Späteren verlauf kann noch hinzugefügt werden, dass der Held eine gewisse Anzahl an Mana-Punkten erhält, wenn er ein Monster besiegt oder bei ähnlichen Errungenschaften.
Wenn der Held in das nächste Level kommt, soll er 40 XP Punkte bekommen. Bei einem LevelUp wird bei dem jeweiligen Entity die levelUp() Methode der SkillComponenten ausgeführt, welche random einen der Werte verbessern.

Methoden und Techniken

Der Code wird entsprechend der Java-Doc Vorgaben Dokumentiert.

Code Analyse

SkillSystem:

Die Skill-System Klasse ist eigentlich nur dafür da die Cooldowns der SkinComponents zu managen. Jeder Frame wird die update() Methode ausgeführt, welche den Cooldown, der Entitäten die eine SkillComponent besitzen , um ein Wert herabsetzt oder bei bei Null bleibt. Es wird ein Stream aus allen Entitäten in der Game Kasse erstellt, und mit flatMap() ein neuer Stream aus SkinComponents, der haltenden Entitäten, erstellt. Abschließend wird mit forEach() für jede SkillComponent die reduvaAllColldowns() Methode aufgerufen.

FireballSkill:

Der FireballSkill, gibt Werte an dene Konstruktor der Überklasse weiter. Darunter sind die Texturen für die animation, die Geschwindigkeit des FireBalls, ein Objekt der Klasse

Damage, welche den Schaden und den Schadenstyp beinhaltet, und die Reichweite des Fireballs. Es wird auch ein Objekt des Typs ITargetSelection übergeben, welches als einziges im Konstruktor der FireBall-Klasse angegeben werden kann. Diese gibt an, in welcher Weise der FireBall abgefeuert werden soll. In der Klasse Hero wird z.B angegeben, dass der Feuerball in richtung der Maus fliegen soll.

XP-System:

In der update Methode werden wird eine Operation auf alle XP Components ausgeführt. Es wird eine neue referezn zur Component erstellt und ein long, der die xp speichert die noch übrig bis zum Level Up ist. Anschließend wird in einer Schleife immer wieder abgefragt, ob die die XP bis zum nächsten level noch nicht erreicht ist und es wird eine Private Methode performLevelUp() ausgeführt, welche das Level erhöht und die XP Left auf einen passenden Wert zurücksetzt.

Ansatz zur Modellierung

Es werden zwei Klassen SpeedSkill und InvisibilitySkill erstellt, welche von ISkillFunktion erben. Beide Klassen speichern die Hero Entität, um bei Aktivierung die Manapunkte des Heros heruntersetzen zu können. Beide Klassen überschreiben die Methode des Interfaces execute(), welche bei dem jeweiligen Tastendruck aufgerufen wird und die Vorgänge zur Aktivierung der Fähigkeit beinhaltet.

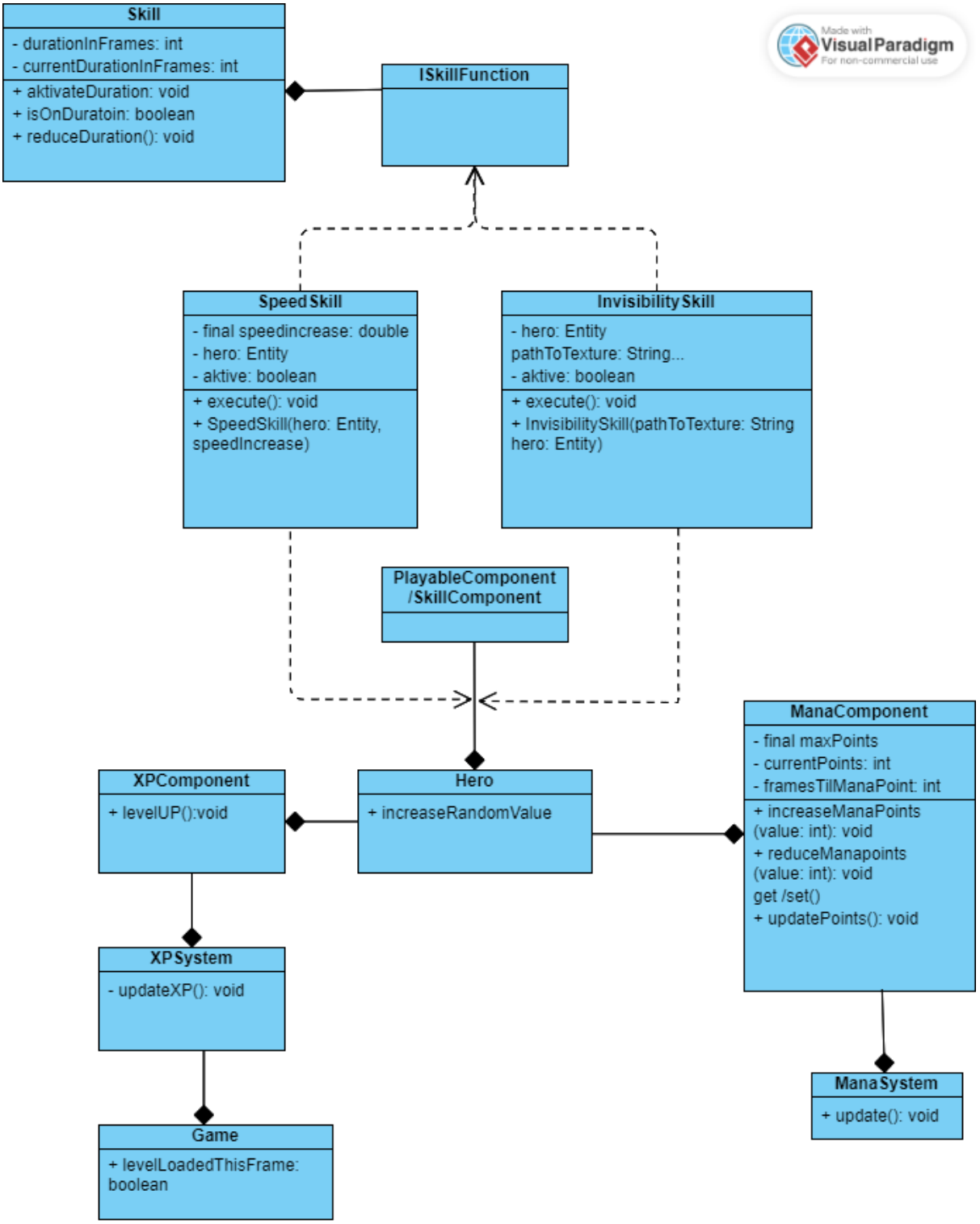
Bei der SpeedSkill Klasse werden lediglich zwei Attribute (xSpeed und ySpeed) in der Klasse SkillComponent des gegebenen Entities um einen bestimmten Wert erhöht. Ebenso werden auch Manapunkte bei der ManaComponent des Helden mit der Zeit runtergesetzt.

Bei der Klasse InvisibilitySkill wird bei Initialisierung außerdem der Pfad zu einer halbtransparenten Animation für die AnimationComponent des Hero angegeben. Eine Andere Möglichkeit wäre den Alpha Wert der vorhandenen Textur zu verändern.Wenn execute ausgeführt wird, werden die neuen Methoden setIdleRight() und setIdleLeft in der Klassse AnimationComponent und VeolocityComponent aufgerufen. Als Parameter dieser Methoden werden die neuen Pfade angegeben. Die Methoden initialisieren eine neue AnimationComponent und eine VelocityComponent mit den neuen Animations Pfaden.

In der Skill Klasse müssen dann auch zwei Attribute final duration and current Duration ergänzt werden mit passenden set und get Methoden. Bei Skills, die keine duration haben, wie z.B. FireballSkill wird die duration einfach auf 0 gesetzt Die duration wird dann auch wie beim Cooldown in dem gegebenen Entity gesetzt.
Die Fähigkeiten sollen mit execute wieder deaktiviert werden können, dafür gibt es dann ein weiteres attribut, welches angibt ob die fähigkeit gerade aktiviert ist oder nicht.

Um ein LevelXP System zu benutzen, muss man definieren, wann eine Entität XP bekommt und was bei einem LevelUP passieren soll.
Die update Methode der XPSystem Klasse fragt ab ob im gleichen frame onLevelLoad() ausgeführt wurde und gibt dem Helden 40 XP. Für ein Level up wird eine Methode im Helden oder in den SkillComponenten definiert, welche random eines der Werte verbessert.

UML:


Skill
- durationInFrames: int
- currentDurationInFrames: int
+ aktivateDuration: void
+ isOnDuratoin: boolean
+ reduceDuration(): void
ISkillFunction
Speed Skill
- final speedincrease: double
- hero: Entity
- aktive: boolean
+ execute(): void
+ SpeedSkill(hero: Entity, speedIncrease)
Invisibility Skill
- hero: Entity
pathToTexture: String...
- aktive: boolean
+ execute(): void
+ InvisibilitySkill(pathToTexture: String hero: Entity)
PlayableComponent /SkillComponent
ManaComponent
- final maxPoints
- currentPoints: int
- framesTilManaPoint: int
+ increaseManaPoints (value: int): void
+ reduceManapoints (value: int): void
get /set()
+ updatePoints(): void
XPComponent
+ levelUP():void
Hero
+ increaseRandomValue
XPSystem
- updateXP(): void
ManaSystem
+ update(): void
Game
+ levelLoadedThisFrame: boolean

2 Konzeptskizze für "Freundlicher NPC"

Beschreibung der Aufgabe:

Es soll um einen NPC-Geist erstellt werden, welcher freundlich ist, d.h. er soll den Hero nicht angreifen.

Der Geist unterscheidet zwischen drei Zuständen: verfolgen des Heros mit Abstand, verschwinden oder zufällig durch das Level wandern.

Erreicht der Geist den Grabstein, wird eine Belohnung ausgelöst.

Beschreibung der Lösung:

Der Geist soll einem Spieler folgen, welcher diesen zum Grabstein bringen kann.

Diese werden zufällig generiert. Schafft man es, den Geist zum

Grabstein zu bringen, werden als Belohnung alle Monster einer zufällig ausgewählten Art im Level

Entfernt.

Methoden & Techniken:

Zum Einsatz kommt JavaDocs, zur besseren Übersicht im Code.

Ansatz und Modellierung:

Eine abstrakte Klasse NPC und eine Klasse Geist. Die Klasse Geist erbt die

Eigenschaften der Klasse NPC und die Klasse NPC erbt selbst von Entity, um verwaltet zu

werden. Zusätzlich erstellen wir eine neue Klasse Grabstein, welche ebenfalls von Entität erbt.

Alle NPCs haben folgende Attribute:

xSpeed: Die Geschwindigkeit auf der X-Achse

ySpeed: Die Geschwindigkeit, auf der Y-Achse

Alle NPCs haben folgende Methoden beziehungsweise Components:

setupVelocityComponent (damit die NPCs sich bewegen können)

setupAnimationComponent: (damit die NPCs animiert werden)

setupHitboxComponent: (damit die NPCs eine Hitbox haben)

NPC (Konstruktor): Führt die restlichen Methoden aus und initialisiert zusätzlich folgende Components:

PositionComponent: Wird benötigt, damit das Monster eine Position im Level hat

AIComponent: Legt fest, wie sich der NPC verhält (keine Kämpfe und keine Transition)

Die Geister haben zusätzlich folgende Attribute:

pathToIdleLeft: Dateipfad, wo die Assests mit Blickrichtung links liegen

pathToIdleRight: Dateipfad, wo die Assests mit Blickrichtung rechts liegen

pathToRunLeft: Dateipfad, wo die Assets für das Rennen mit Blickrichtung links liegen

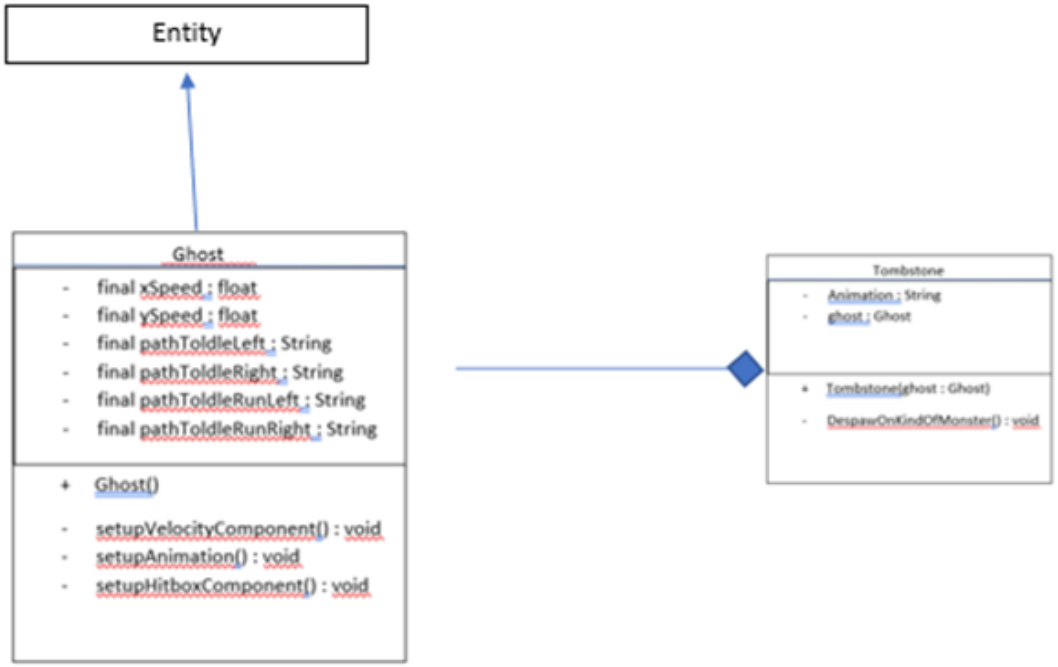
pathToRunRight: Dateipfad, wo die Assests für das Rennen mit Blickrichtung rechts liegen

Werte des Geistes:

xSpeed= 0.4 ySpeed= 0.2

UML


Game
- hasGhost : boolean
- tomb : Tombstone
- spawnGhost() : void


Entity
Ghost
- final xSpeed : float
- final ySpeed : float
- final pathToIdleLeft : String
- final pathToIdleRight : String
- final pathToIdleRunLeft : String
- final pathToIdleRunRight : String
+ Ghost()
- setupVelocityComponent() : void
- setupAnimation() : void
- setupHitboxComponent() : void
Tombstone
- Animation : String
- ghost : Ghost
+ Tombstone(ghost : Ghost)
- DespawnOnKindOfMonster() : void


IIdleAI
GhostWalk
- Path : GraphPath<Tile>
- randomPath : boolean
+ idle() : void