Projekt: JBomberman
DomainAnalyse

Pascal Kistler
Silvan Adrian
Fabian Binna

1 Änderungshistorie

Datum	Version	Änderung	Autor
09.03.15	1.00	Erstellung des Dokuments	Gruppe
20.03.15	1.01	Vollendung des Dokuments	Fabian Binna

Inhaltsverzeichnis

1 Änderungshistorie **2**

2 Einführung **4**
2.1 Zweck 4
2.2 Gültigkeit 4
2.3 Übersicht 4

3 DomainModell **5**
3.1 Strukturdiagramm 5
3.2 Konzeptbeschreibung 6
3.2.1 Game 6
3.2.2 Party 6
3.2.3 Player 6
3.2.4 Sprite 7
3.2.5 AnimatedSprite 7
3.2.6 Bomberman 7
3.2.7 PowerUp 7
3.2.8 Bomb 8
3.2.9 Explosion 8
3.2.10 DestroyableBlock 8

4 Systemsequenzdiagramme **9**
4.1 UC01: Bomberman spielen 9

5 Systemoperationen **10**
5.1 Client 10
5.2 Server 10
5.3 Contracts 10

2 Einführung

2.1 Zweck

Dieses Dokument beschreibt die Domainanalyse für das Projekt JBomberman.

2.2 Gültigkeit

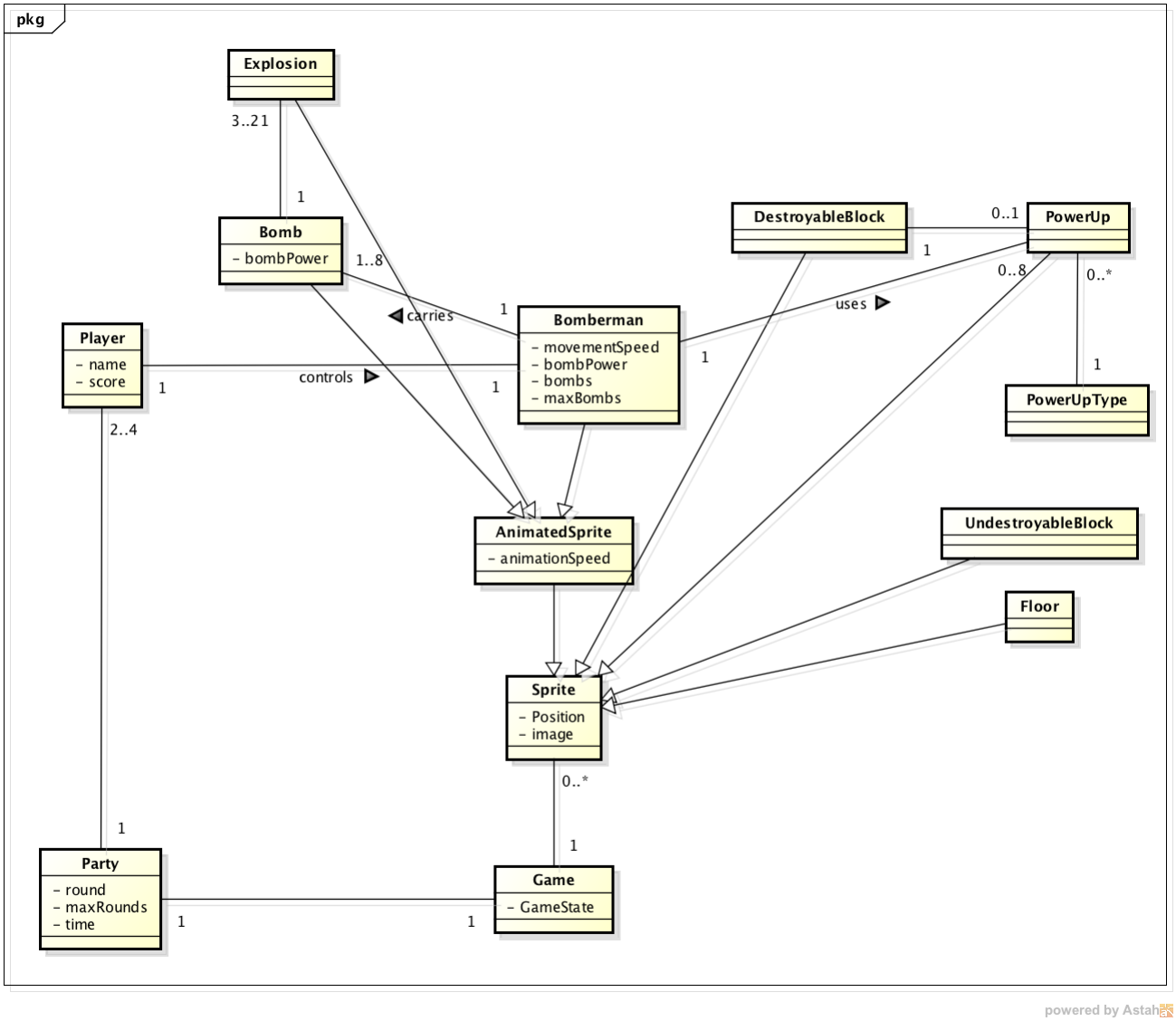
Dieses Dokument ist während des ganzen Projekts gültig und wird laufend aktualisiert.

2.3 Übersicht

Dieses Dokument soll eine erste Analyse der Software zeigen. Das Strukturdiagramm stellt die wichtigsten Klassen dar, die miteinander interagieren müssen. Das Systemsequenzdiagramm beschreibt die wichtigsten Abläufe der Use Cases.

3 DomainModell

3.1 Strukturdiagramm



3.2 Konzeptbeschreibung

3.2.1 Game

Die Klasse Game ist die oberste Klasse und kontrolliert den Ablauf des Spiels.

Gamestate
Der GameState weist auf den aktuelle Spielestatus hin. Je nachdem in welchem Status sich das Spiel befindet werden andere Routinen durchlaufen.

3.2.2 Party

Die Party beinhaltet alle Spieler die aktuell am Spiel teilnehmen, und kontrolliert Zeit sowie die Spielrunden.

round
Die aktuelle Spielrunde.

maxRounds
Maximale Anzahl Runden die gespielt werden müssen.

time
Time zählt die Zeit von 3 Minuten herunter. Bei Ende der Zeit kriegen alle Bombermans die vollen PowerUps.

3.2.3 Player

Der Player dient zur identifizierung der Teilnehmer. Jedem Spieler ist ein Bomberman zugewiesen. Die Messages werden über die jeweilige Playerinstanz dem korrekten Bomberman zugewiesen.

name
Speichert den Namen des Spielers. Wird auch zur identifizierung verwendet.

score
Speichert die aktuelle Punktezahl des Spielers.

3.2.4 Sprite

Die Spriteklasse implementiert die Grundvoraussetzungen für jedes Spielelement. Jedes visuelle Element muss von der Spriteklasse erben.

Position
Position ist eine Klasse, die x und y Koordinaten speichert.

image
Image ist ein String, der den Namen des Bildes speichert, welcher intern zur identifizierung dient und bei einem ResourceManager die Referenz zum BufferedImage abholen kann.

3.2.5 AnimatedSprite

Das AnimatedSprite erbt von der Spriteklasse und implementiert weitere Funktionen, die ein animiertes Objekt ermöglicht.

animationSpeed
Die Geschwindigkeit mit der durch die einzelnen Bildteile geschaltet wird.

3.2.6 Bomberman

Die Bombermanklasse beschreibt den Zustand und die Fähigkeiten der Spielfigur.

movementSpeed
Wie schnell sich ein Bomberman fortbewegen kann.

bombPower
Wie weit reicht der Sprengradius.

bombs
Anzahl Bomben die der Bomberman zeitgleich auf dem Spielfeld verteilen kann.

maxBombs
Maximale Anzahl Bombem die der Bomberman tragen kann.

3.2.7 PowerUp

Das PowerUp beschreibt die Art der Fähigkeit die den Bomberman verbessert.

3.2.8 Bomb

Die Bombklasse repräsentiert die Bombe auf dem Spielfeld.

bombPower
Die Reichweite der Explosionsarme.

3.2.9 Explosion

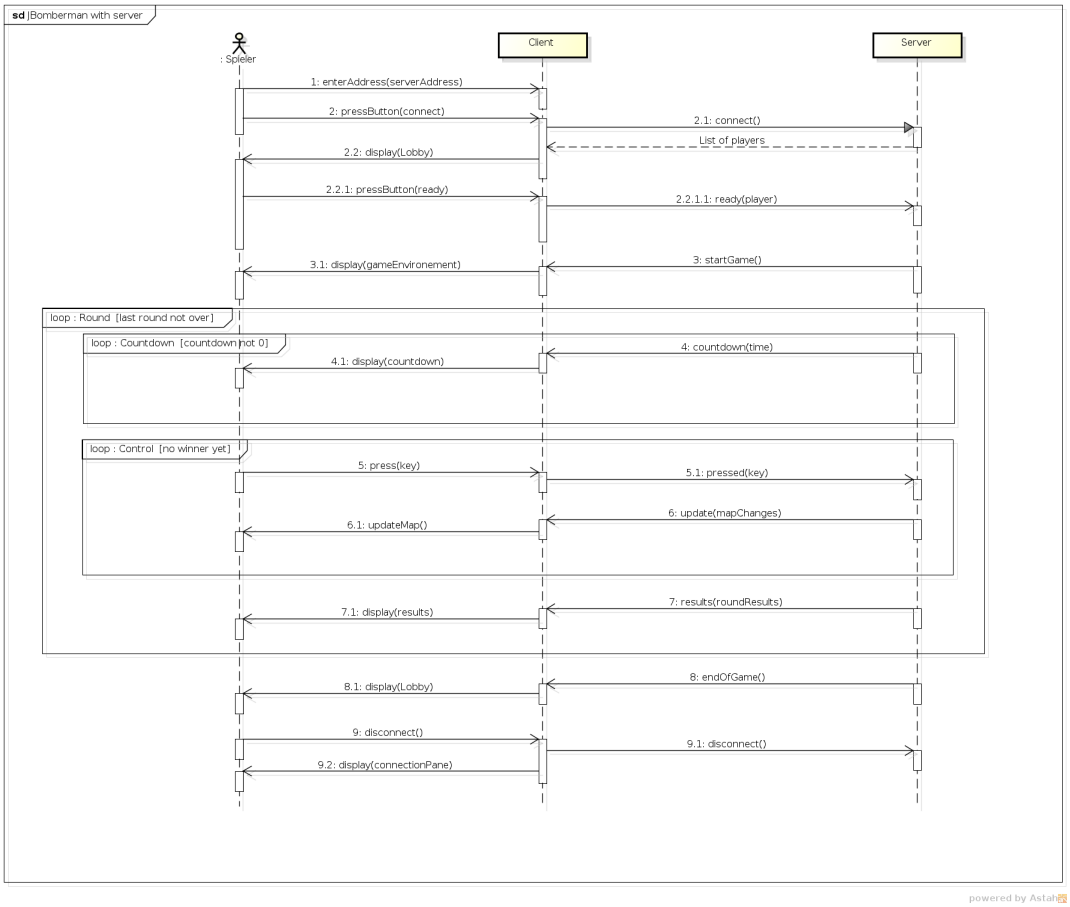
Beschreibt einen Teil eines Explosionsarms. Die bombPower der Bombe definiert wie viele Explosionen in jede Richtung erstellt werden müssen.

3.2.10 DestroyableBlock

Der DestroyableBlock kann von einer Bombe zerstört werden und hinterlässt manchmal ein PowerUp.

4 Systemsequenzdiagramme

4.1 UC01: Bomberman spielen



5 Systemoperationen

5.1 Client

- connect(server: Address)
- ready(player: Player)
- pressed(key: Message)
- disconnect()

5.2 Server

- startGame()
- countdown(time: integer)
- update(mapChanges: Message)
- results(results: Array)
- endOfGame()

5.3 Contracts

Operation	connect(server : Address)
Cross References	UC01
Preconditions	Eine Serveradressse wurde eingegeben
Postconditions	• Der Client ist mit dem Server verbunden • Der Server kennt den Client
Operation	ready(player : Player)
Cross References	UC01
Preconditions	Der Client befindet sich in der Lobby
Postconditions	• Der Status des Clients wurde auf dem Server auf **ready** gesetzt

Operation	startGame()
Cross References	UC01
Preconditions	Alle Client sind ready
Postconditions	• Bei den Clients wurde die Spielumgebung gestartet • Der Server befindet sich im Gamemode
Operation	countdown(time : integer)
Cross References	UC01
Preconditions	Das Spiel wurde gestartet
Postconditions	• Ein Countdown wird den Spielern angezeigt
Operation	pressed(key : Message)
Cross References	UC01
Preconditions	Der Client befindet sich im Spiel und der Countdown ist abgelaufen
Postconditions	• Der Event ist beim Server angekommen • Der Event befindet sich in der Eventqueue
Operation	update(mapChanges : Message)
Cross References	UC01
Preconditions	Der Client befindet sich im Spiel
Postconditions	• Die Informationen über Änderungen in der Map wurden an den Client übertragen • Der Client hat seine Map angepasst
Operation	results(results : Array)
Cross References	UC01
Preconditions	Die Runde ist vorbei
Postconditions	• Die Clients haben die Rangliste erhalten • Die Clients zeigen die Rangliste an

Operation	endOfGame()
Cross References	UC01
Preconditions	Alle Runden sind vorbei
Postconditions	• Alle Clients wurden über das Ende des Spiels informiert • Die Clients zeigen die Rangliste an

Operation	disconnect()
Cross References	UC01
Preconditions	Der Client befindet sich in der Lobby
Postconditions	• Der Client ist nicht mehr mit dem Server verbunden • Der Server hat den Client aus der Clientliste entfernt