Zürcher Hochschule
für Angewandte Wissenschaften

Projektspezifikation

Cube Circle Tower Defense (CCTD)

Version 1.0
27. Oktober 2010

Rolf Koch, Lukas Spirig, Benjamin Felder, Fabian Eriksson, Nathanael Koch

Zusammenfassung

- Projektmanagement
- Anwendungsfälle
- Zusätzliche Spezifikationen
- Diagramme
- Systemoperationen
- Architektur
- Glossar

ZHAW, Technikumstrasse 9, CH-8401 Winterthur
Tel: +41 58 934 71 71, Fax: +41 58 935 71 71
http://www.zhaw.ch

Inhaltsverzeichnis

1 Projekt Management

1.1 Aktueller Status

In dieser Sektion befindet sich ein Vergleich des geplanten Aufwands und des tatsächlich erreichten Resultats sowie dessen jeweiliger Aufwand der für die einzelnen Tasks geleistet wurde.

Aufwände	**Aufw. Geplant**	**Aufw.**	**Aufw. p.P**	**Anz. P**
Spielprinzip definiert	? 0	15	3	5
Entwicklungs Umgebung definiert	?	6	3	2
Projekt Skizze: • Projekt Idee • Hauptanwendungsfall • Weitere Anforderungen • Ressourcen • Risiken • Grobplanung • Kundennutzen • Wirtschaftlichkeit	?	40	8	5
Präsentation Projekt Skizze	?	5	5	1
Applikations Grundgerüst erstellt	5	nicht erfüllt	?	?
Anwendungsfälle detailiert ausformuliert	8	12	6	2
Domänenmodel Entwurf	10	10	2	5
Projekt Management auf neusten Stand gebracht	7	7	7	1
Architektur	5	5	5	1
Supplementary Specification erstellt	6	6	6	1
System Sequenzdiagram	8	8	8	1
Glossar erstellt	3	6	2	3
Vision fertig gestellt	4	3	3	1
Domänenmodell erstellt	5	5	5	1
Java Tests bezüglich Client / Server verhalten	12	nicht erfüllt	?	?
Welche Daten werden übertragen	3	nicht erfüllt	?	?
Analyse Dokumentation fertigstellen	5	5	5	1
Analyse Präsentation erstellt	3	3	3	1
UML Klassendiagram Entwurf	3	3	1.5	2

Tabelle 1: Projekt Management: Aktueller Status

1.2 Massnahmen

Für die Design Phase wird nun vermehrt auch erster Javacode generiert. In der Iteration 3 (Analyse + Design) wird nun auch das Applikations Grundgerüst erstellt und erste Java Tests bezüglich Client / Server verhalten durchgeführt. Die Tasks wurden verschoben.

1.3 Weitere Planung

Analyse + Design (Iteration 3)

Wir befinden uns momentan in dieser Iteration 3 und somit in der Elaborations Phase. Zu diesem Zeitpunkt geht die Analyse Phase in die Design Phase über.

Phase	Woche	Iteration	Task	Geplanter Aufwand
Elaboration	42	3	Use Cases fertig	4
			Supplementary Specification fertig falls benötigt	3
			System-Sequenzdiagram erstellt	3
			Glossar erstellt	3
			Vision fertig	4
			Domänenmodell erstellt	3
			Erste Java Tests bezüglich Client / Server verhalten	12
			welche Daten werden übertragen	6
			Applikations Grundgerüst erstellt	5
	43		Abgabe Analyse	5
			Präsentation Analyse	1
			erste Version UML Klassendiagram	15
			erste Version Architekturdokumente	12

Tabelle 2: Projekt Management: Iteration 3

Woche 42: 40 Stunden. Pro Person: 8 Stunden

Woche 43: 33 Stunden. Pro Person: 6,6 Stunden

Total: 73 Stunden. Pro Person: 14,6 Stunden

Design (Iteration 4)

Ab Woche 44 startet die Design Phase mit der Iteration 4 bei der es hauptsächlich um das Code Design geht.

Phase	Woche	Iteration	Task	Geplanter Aufwand
Construction	44	4	UML Klassendiagram	10
			UML Interaktionsdiagram für ausgewählte Systemoperationen	10
			Architekturdokumente fertig 5	
			Implementation des UML Klassendiagrams in Java	15
			Test der Zusammenhänge: ev. Klassendiagram überarbeiten	5
	45		UML Klassendiagram	15
			UML Interaktionsdiagram für ausgewählte Systemoperationen	10
			Architekturdokumente fertig	5
			Design Präsentation fertig	5
			Erste Implementation der Klassen inklusive Methoden und Eigenschaften	15

Tabelle 3: Projekt Management: Iteration 4

Woche 44: 45 Stunden. Pro Person: 9 Stunden

Woche 45: 50 Stunden. Pro Person: 10 Stunden

Total: 95 Stunden. Pro Person: 19 Stunden

1.4 Anpassung der Projekt Aufwand Schätzung

Aufgrund der nun vorliegenden Zahlen muss die eher etwas übertriebene Aufwandschätzung korrigiert werden. Geschätzt war ursprünglich ein Aufwand von 600 Mannstunden verteilt auf 5 Personen. Aktuelle Zahlen zeigen, dass dieser Aufwand voraussichtlich nicht benötigt wird.

Wir setzen daher den geschätzten Aufwand neu auf ca 450 Stunden fest. Dies entspricht etwa einem Aufwand von 90 Stunden pro Iteration Total -¿ 45 Stunden pro Woche Total -¿ 9 Stunden pro Person pro Woche.

1.5 Risiken

1.6 Nachfrage

Nach einer kurzen Suche mit Google ergeben sich etliche Treffer für “Tower Defense“-Spiele in etlichen Formen. Zum Beispiel towerdefence.net listet 196 unterschiedliche “Tower Defense“-Implementationen (Stand: 01.10.2010). Ausserdem unterstützen mehrere Multiplayer Strategy Spiele einen Tower Defense Modus (Warcraft 3, Starcraft 2, ...).
Ein weiteres Problem stellt die Gewinngenerierung dar. Die meisten “Tower Defense“-Spiele die sind entweder eine Erweiterung zu einem bereits existierenden Spiel oder können Gratis heruntergeladen werden.

1.7 Projektumfang

Der Projektumfang ist für die kurze Zeit von einem Semester gross gehalten. Deswegen kann es schwierig werden das ganze Produkt mit allen vorgesehenen Features in der kurzen Zeit zu implementieren.
Stellt sich im Verlaufe des Projektes heraus, dass das Projekt nicht im gegebenen Zeitrahmen zum vollständigen Abschluss gebracht werden kann, werden gewisse Features weggelassen.

1.8 Mitarbeiter Motivation

Da das Projekt ‘nur’ auf Basis eines Wahlmoduls entwickelt wird und keine reale Bezahlung stattfindet, kann es passieren, dass sich einzelne Mitarbeiter des Teams nicht motivieren können für das Projekt zu arbeiten. Dies kann eventuell dazu führen, dass nicht alle geplanten Tasks durchgeführt werden und Tasks immer weiter nach hinten geschoben werden, was schlussendlich zum Risiko ‘Projektumfang’ führt und eine Reduktion der Features getätigt werden muss um das Projekt doch noch vollständig abschliessen zu können.

2 Anwendungsfälle

2.1 Use Case Diagram

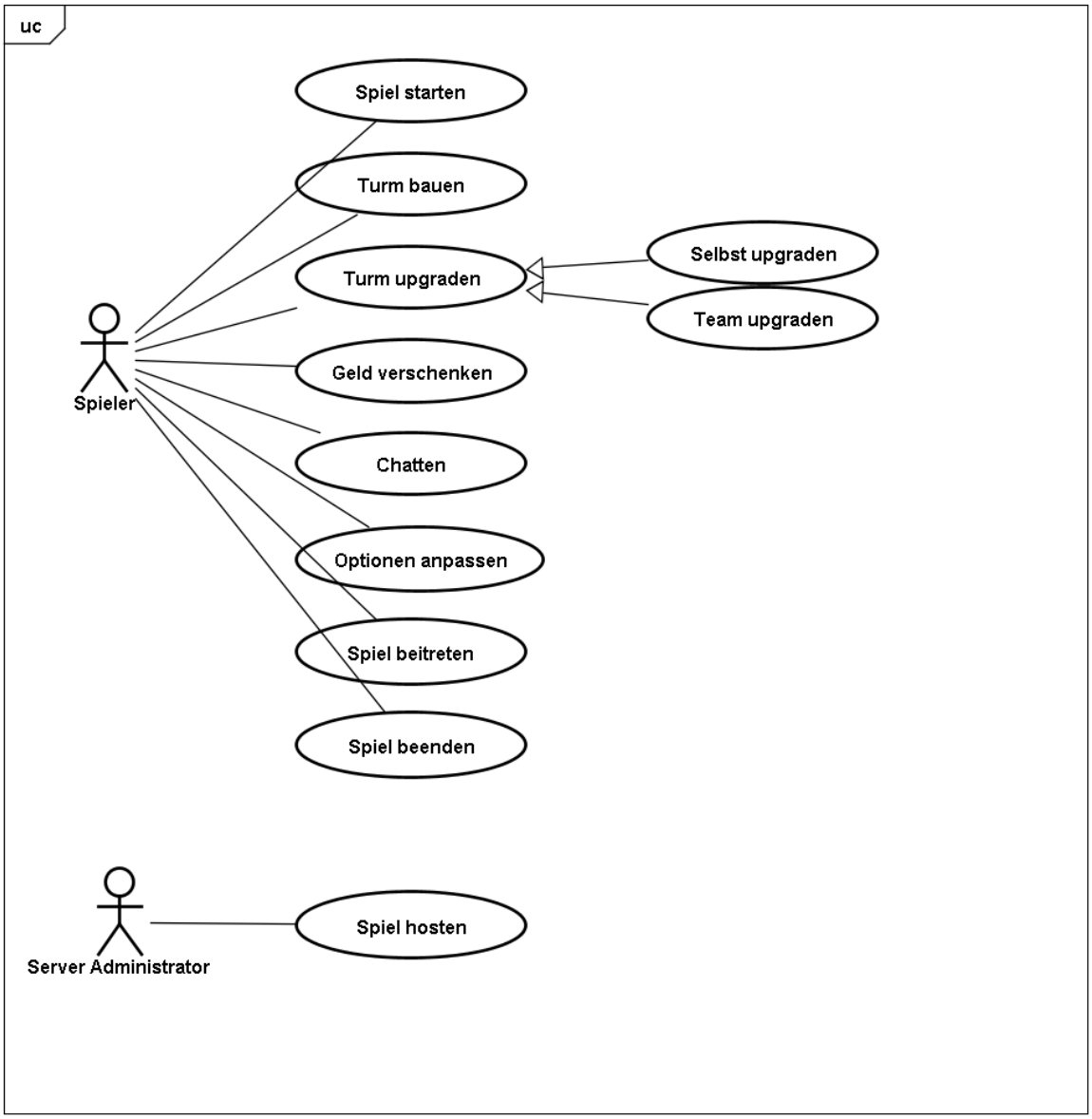


Abbildung 1: Use Case-/Anwendungsfall-Diagram

Es existieren zwei Hauptaktoren. Ein Spieler und ein Server Administrator. Total können maximal vier Spieler an einem Spiel teilnehmen. Pro Spiel benötigt es mindestens einen Server Administrator. Der Server Administrator zählt gleichzeitig auch als Spieler. Für ein Einzelspieler Spiel muss braucht es also einen Server Administrator der gleichzeitig auch als Spieler fungiert. Für Multiplayer Spiele können dann jedoch noch bis zu drei zusätzliche Spielern beitreten nebst dem Server Administrator.

Wir haben als Haupt Use Case ‘Spiel starten’ ausgewählt. Als Casual haben wir ‘Spiel hosten’, ‘Turm bauen’ und ‘Turm upgraden’ ausgewählt. Als Brief stehen somit noch ‘Spiel beitreten’, ‘Geld verschenken’, ‘Chatten’, ‘Optionen anpassen’, ‘Spiel beenden’ und die beiden Upgrade Funktionalitäten ‘Selbst upgraden’ und ‘Team upgraden’ zur Verfügung. Diese beiden Upgrade Funktionalitäten stellen zwei Ausprägungen des ‘Turm up-

graden' dar, da sie unterschiedlich funktionieren, je nach dem ob man einen eigenen Turm oder den Turm eines Kollegen upgraden möchte.

2.2 Use Case UC1: Spiel starten

Primary Actor: Spieler

Stakeholder and Interests:

- Spieler: Möchte so schnell wie möglich ein Spiel starten und keine grosse Verzögerung durch die Authentifizierung
- Server: Möchte entsprechende Benutzernamen im korrekten Format und die Verbindung verwalten

Precondition:

- Der Spieler möchte ein Spiel starten
- Es muss ein Server aktiv sein

Postcondition:

- Programm ist offen und eine Verbindung steht

Main Success Scenario:

1. Spieler startet Programm
2. Programmfenster erscheint
3. Spieler gibt, falls nicht gespeichert, Benutzername ein
4. Spieler wählt Einzelspiel oder Multiplayer
   (a) Spieler wählt Einzelspiel
      i. Programm startet ein Einzelspiel
      ii. Programm verwaltet Spiel
   (b) Spieler wählt Multiplayer
      i. Spieler kann wählen, ob er ein Spiel hosten möchte (UC2) oder einem Spiel beitreten möchte (UC3)
      ii. Spieler wartet dann in der Lobby, bis genügend andere Spieler anwesend sind

Extensions:

- Wenn während eines Multiplayer Matches das Programm abstürzt oder die Verbindung verloren geht, soll der Server das erkennen und entsprechend reagieren.
- Wenn der Spielerhost die Verbindung verliert, bricht das Spiel ab.

Special Requirements:

Die Fenstergrösse des Programms soll variabel sein

Technology and Data Variations List:

-

Frequency of Occurrence:

Zu Beginn

Open Issues:

-

2.3 Use Case UC2: Spiel hosten

Primary Actor: Server Administrator

Ein Spieler startet einen Spiel-Server auf den sich die Clients einloggen können.

Der Server überprüft die Benutzernamen und verwaltet alle aktiven Verbindungen. Wenn alle Spieler bereit sind, wird das Spiel gestartet.

Der Server-Client übernimmt dann das Wave-Handling und das Creep-Handling.

Wenn eine Verbindung zu einem Spieler verloren geht, informiert er die anderen Spieler.

Wenn ein Spieler seine Verbindung trennt, entfernt er die Türme des entsprechenden Spielers und auch den zugehörigen Wave-Spawn-Punkt.

2.4 Use Case U3: Spiel beitreten

Brief

Primary Actor: Spieler

Ein Spieler kann einem Match beitreten. Dies kann er über die Adresse tun.

2.5 Use Case UC4: Turm bauen

Primary Actor: Spieler

Das Spielfeld muss zusammengestellt werden. Das Spielfeld besteht aus einer (oder mehreren) Hintergrundgrafik, begrenzte Bauflächen, Spawn-Punkte für die Creeps, eine Route für die Creeps, Spielinfozeile und einem Baumenü.

Die Creeps müssen dynamisch über das Spielfeld bewegt werden und bei Tod entfernt werden.

Das Programm stellt anhand der Fenstergrösse einen Ausschnitt des Spielfelds dar.

Der Spieler kann mit vordefinierten Tasten durch das effektive Spielfeld scrollen oder mit der Maus an einen entsprechenden Rand fahren, damit sich das sichtbare Spielfeld verschiebt. Wichtig dabei ist, dass die Geschwindigkeit angepasst ist, damit eine Feinkontrolle möglich ist.

Der Spieler soll dann aus einer Liste von Türmen auswählen können und einen Turm per Drag-and-Drop auf dem Spielfeld platzieren können.

Ein Turm kann nur in der eigenen Baufläche platziert werden und darf nicht mit anderen Türmen überlappen.

Wenn man einen Turm ”dragt”, soll die, für den Turm benötigte, Fläche unter dem Cursor angezeigt werden.

Daraus soll auch ersichtlich sein, ob der vorgesehene Bauplatz für den selektierten Turm auch gültig ist.

Wenn der Turm erfolgreich platziert wurde, soll er nach einer kurzen Konstruktionszeit aktiv werden.

2.6 Use Case UC5: Turm upgraden

Primary Actor: Spieler

Der Prozess des Upgraden läuft wie folgt ab:

1. Der Spieler klickt auf den Tower, den er gerne upgraden möchte
2. In der Übersichtsliste erscheint eine Liste von möglichen Upgrades und die Option den Turm wieder zu verkaufen

3. Nachdem ein Upgrade ausgewählt wurde, wird der Tower upgegradet, sofern genug Ressourcen zur Verfügung stehen
4. Der Tower ist deaktiviert, bis das Upgrade abgeschlossen ist.
5. Sobald das Upgrad fertig gestellt ist, fängt der Tower wieder an zu schiessen und ist entsprechend der Art des Towers und des Upgrades verbessert

2.7 Use Case UC6: Turm upgraden: Selbst upgraden

Brief

Primary Actor: Spieler

Ein Spieler kann seine Türme upgraden. Dieses Upgrad ist eindimensional und erhöht die Effizienz des primären Merkmals des Towers.

2.8 Use Case UC7: Turm upgraden: Team upgraden

Brief

Primary Actor: Spieler

Einem Spieler wird bei Spielbeginn zufällig ein Attribut zugewiesen. (Splash, Speed, Range, Slow, Poison) Er kann einen Tower eines andere Spieler upgraden und diesem Tower zusätzlich zu dessen Merkmal das Attribut hinzufügen, das der Spieler besitzt.

2.9 Use Case UC8: Geld verschenken

Brief

Primary Actor: Spieler

Ein Spieler kann einem anderen Spieler Ressourcen schicken. Er kann dabei frei den Spieler und die Ressourcenanzahl wählen.

2.10 Use Case UC9: Chatten

Brief

Primary Actor: Spieler

Die Spieler sollen untereinander chatten können. Sowohl in der Lobby als auch im Spiel. Durch drücken einer bestimmten Taste soll eine Eingabezeile erscheinen, in welche der Spieler eine Nachricht schreiben kann. Wenn die Nachricht mit Enter bestätigt wird, wird diese in einer Scroll-Liste angezeigt, wobei die neuste Nachricht immer unten ist und der Scroll-Balken sich bei einer neuen Nachricht automatisch nach unten verschiebt. Bei jeder Nachricht wird auf der linken Seite der Spielername des Authors angegeben, damit auch klar ist, von wem die Nachricht stammt.

2.11 Use Case UC10: Optionen anpassen

Brief

Primary Actor: Spieler

Der Spieler kann diverse Einstellungen anpassen.

2.12 Use Case UC11: Spiel beenden

Brief

Primary Actor: Spieler

Der Spieler kann jederzeit das Spiel beenden. Dabei werden alle allfälligen Verbindungen getrennt.

3 Zusätzliche Spezifikationen

3.1 Einführung

Diese Sektion beschreibt CCTD Spiel Anforderungen, die nicht in den Use Cases erfasst sind.

3.2 Funktionalität

Protokollierung und Fehlerhandhabung

Alle Fehler in einem persistenten Speicher protokollieren.

Sicherheit

Um Spieler eindeutig zu identifizieren muss sich jeder Spieler einen Benutzer Namen anlegen. Über diesen Benutzernamen wird der Spieler dann als eindeutiger Spieler behandelt.

3.3 Usability

Real Time Spielgeschehen

Das Spiel soll sich als Echtzeit Spiel anfühlen. Das bedeutet jeder Command wird sofort und ohne grosse Zeitverzögerung ausgeführt und Spiel Elemente können auch ihrerseits jederzeit Aktionen ausführen ohne, dass der Spieler etwas dafür tut. z.B: Türme Schiessen, Creeps bewegen sich.

Spieler Focus

Jeder Spieler ist für seine Ecke des Spielfeldes verantwortlich. Ein Spieler kann das Spielfeld jedoch erkunden und so zu seinen Kollegen schauen. Dies muss flüssig und vorallem schnell möglich sein, da er nicht viel Zeit hat um sich umzusehen ohne seine Türme zu stark zu vernachlässigen. Es sollte sehr rasch möglich sein für einen Spieler sich auf dem Spielfeld zu orientieren. Hierzu sollten Tastenkürzel bereitstehen um direkt von Basis zu Basis der Kollegen zu wechseln und auch ein Tastenkürzel um wieder zurück zur eigenen Basis zu wechseln.

Generell sollte es dem Spieler jedoch überlassen werden, wo er hinschauen möchte. Er sollte nicht gezwungen sein ständig auf einen Punkt zu starren.

Informationen zu Spielelementen

Ein Spieler sollte alle Spielelemente anklicken können. Hat er ein Element ausgewählt, werden ihm zusätzliche Informationen zu dem Element präsentiert. So kann er beispielsweise Türme seiner Kollegen anschauen und untersuchen oder Creeps anschauen, um festzustellen wie stark sie sind.

3.4 Wartbarkeit

Anpassbarkeit

Das Spiel sollte schnell und einfach erweiterbar sein. Es sollen neue Creeps eingefügt werden können, neue Tower Modelle sollen erstellt werden können und neue Tower Updates sollten einfach hinzugefügt werden können.

Konfigurierbarkeit

Alle Einstellungen die man im Spiel vornehmen kann sollten in einem Config File gespeichert werden und sollten auch dort änderbar sein.

3.5 Implementierungsbedingungen

Das CCTD Spiel wird in Java entwickelt werden, da diese Programmiersprache von allen beteiligten beherrscht wird.

3.6 kostenlose Open-Source-Komponenten

Im Allgemeinen setzen wir auf möglichst viele kostenlose Java-basierte Open-Source-Komponenten.

Folgende Komponenten werden mit hoher Wahrscheinlichkeit für das Spiel oder zu dessen Entwicklung verwendet:

- Junit (Testing Framework)
- Easymock (Testing Component)
- Cobertura (Test Coverage Utility)
- log4j (Protokollierungs Framework)

3.7 Interfaces

Grundsätzliche Design Entscheidung:

Das CCTD Spiel wird grundsätzlich für Computer Bildschirme (12" und grösser mit einer Mindestauflösung von 1024x768) entwickelt. Das Spiel wird auf eine Maus und Tastatur Bedienung ausgelegt.

Erweiterbarkeit für spätere Releases:

Das Spiel kann erweitert werden auf Touchscreen Bedienung und kleinere Bildschirme wie IPads oder Smartphones.

Dies geschieht jedoch nicht im Release 1.

3.8 Softwareschnittstellen

Geplant sind keine speziellen Softwareschnittstellen.

3.9 Rechtlich

Lizenzen

Da unsere Software auch wieder als Open-Source publiziert wird ist es möglich andere Open-Source Software komponenten die unter den unten genannten Lizenzen veröffentlicht sind (aber nicht nur) zu verwenden.

- Apache License, 2.0
- BSD licenses (New and Simplified)
- GNU General Public License (GPL)
- GNU Library or Lesser"General Public License (LGPL)
- MIT license
- Mozilla Public License 1.1 (MPL)
- Common Development and Distribution License
- Eclipse Public License
- Creative Commons Attribution 2.5 License
- Public Domain

Für mehr details besuchen sie bitte diese Seite: http://en.wikipedia.org/wiki/Comparison_of_free_software_licenses

4 Diagramme

4.1 Domänenmodell

Dieses Diagramm beschreibt das Problem (Die Umsetzung des Spiel), das gelöst werden muss.

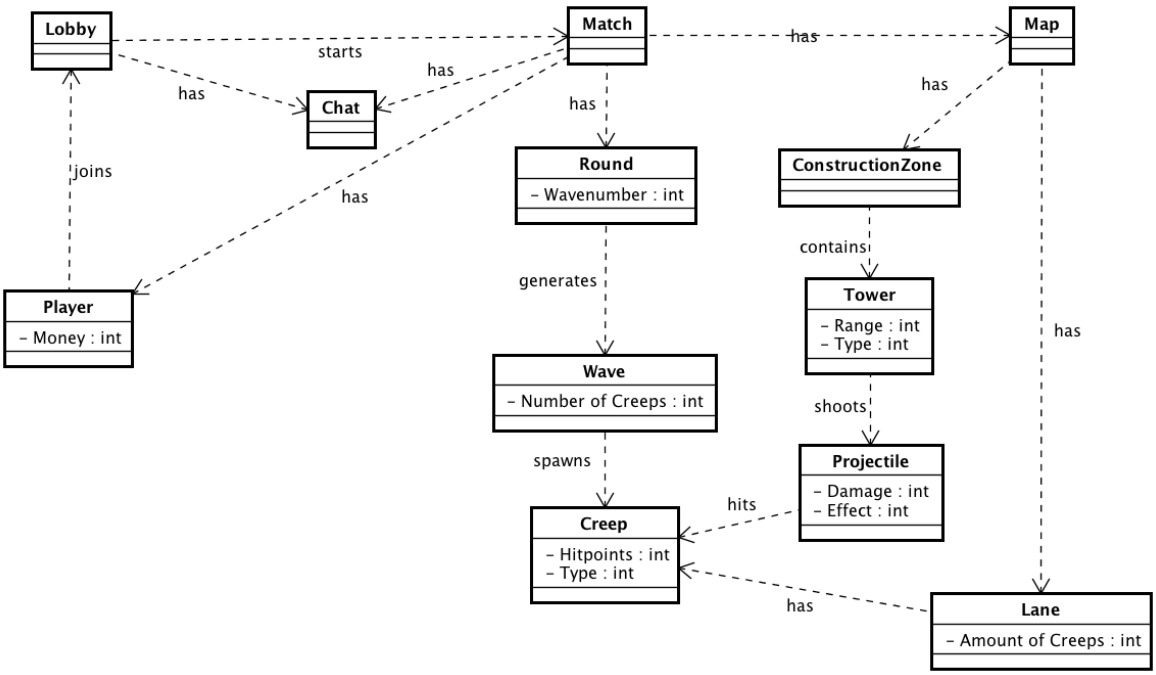


Abbildung 2: Domänenmodell

Erklährung des Diagrammes

Spieler

- Diese Domäne beschreibt den Spieler, der am Spiel teilnimmt.

Lobby

- Im Multiplayerfall betreten die Spieler die Lobby, wo sie warten bis die gewünschte Anzahl Spieler erreicht ist.

Chat

- Der Chat ermöglicht es den Spielern in der Lobby, wie auch im Spiel miteinander zu kommunizieren.

Match

- Organisiert das Spiel.
- Beinhaltet die Map und kümmert sich um die einzelnen Runden.

Round

- Generiert die einzelnen Waves anhand der Anzahl Spieler.

- Beinhaltet die Map und die Lane. Weiss in welchen Runde sich das Spiel befindet.

Wave

- Erzeugt die einzelnen Creeps, die zur Wave gehören.
- Hat Informationen über Anzahl und Typ von Creeps die gespawnt werden.

Creep

- Gegner, der sich auf dem Spielfeld befindet.
- Hat Attribute wie Lebenspunkte und Typ.

Map

- Ist das eigentliche Spielfeld.
- Beinhaltet die Constructionzone, sowie auch die Lane.

Lane

- Der Weg auf welchem sich die Creeps bewegen.
- Kennt die Anzahl der Creeps, die sich momentan auf der Lane befinden.

Constructionzone

- Fläche auf der die Spieler Türme bauen können.

Tower

- Turm welcher auf der Constructionzone platziert werden kann.
- Schiesst auf die Creeps

Projectile

- Projektil, welches von den Türmen geschossen wird.
- Haben verschiedene Eigenschaften wie Schaden oder Effekte auf Creeps.

4.2 System-Sequenzdiagramm

Dieses Sequenzdiagramm beschreibt die Systemoperationen des Use Cases UC1 ”Start Game”. Dabei wird ersichtlich, welche interaktionen der Benutzer in der Startphase des Programms tätigen kann und was dies bewirkt.

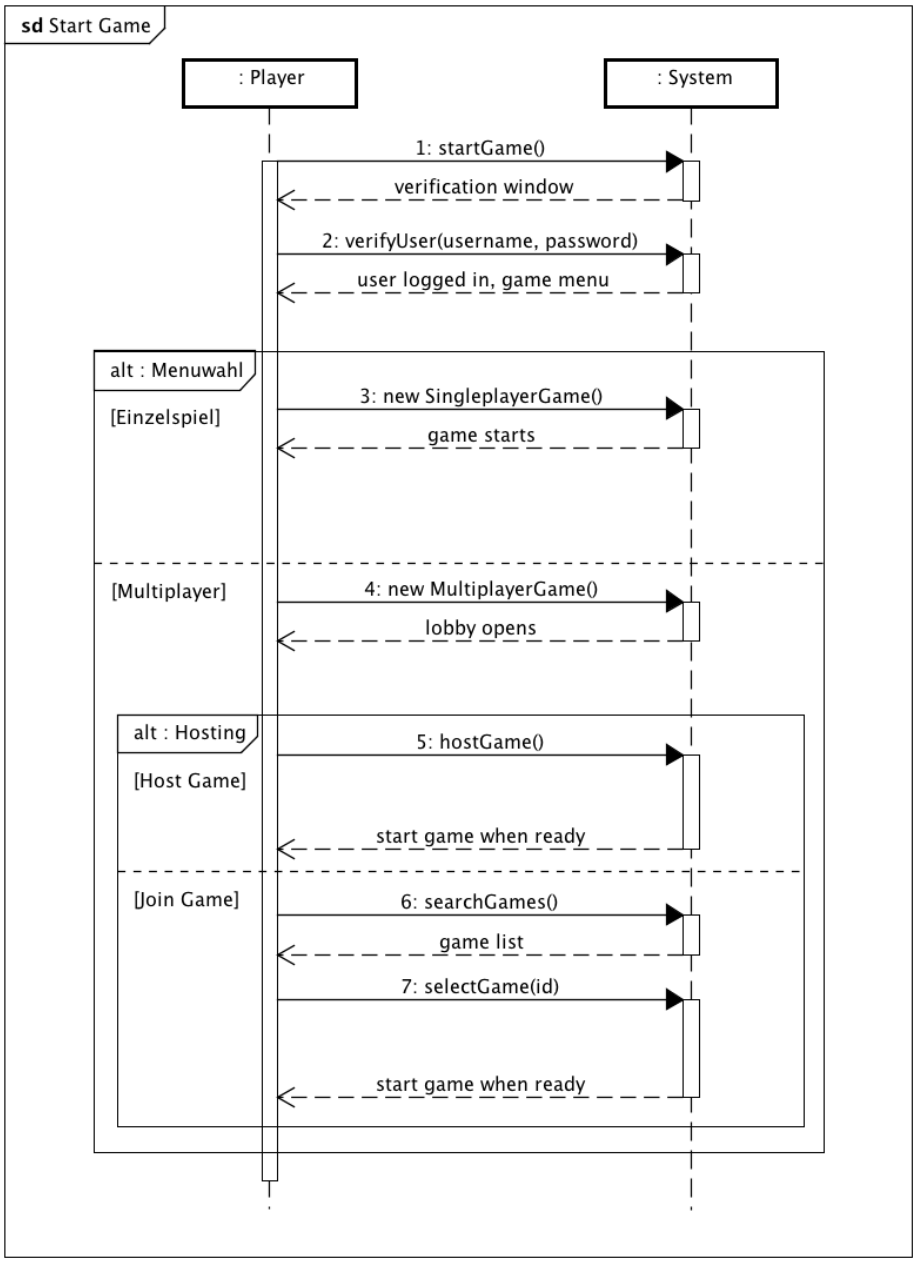


Abbildung 3: Use Case UC1 “Start Game”: Sequenzdiagramm

5 Systemoperationen

Die folgenden Contracts beschreiben alle Systemoperationen des Use Cases UC1 “Start Game”, welche im Sequenzdiagramm aufgeführt sind. Hierbei wird aufgezeigt, welche Operationen der User im Startprozess auslösen kann und was die Antwort des Systems darauf ist.

5.1 Contract CO1: startGame

Operation:	`void startGame()`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	-
Nachbedingungen:	❑ Das Spiel wurde gestartet
	❑ Das Anmeldefenster ist erschienen

5.2 Contract CO2: verifyUser

Operation:	`boolean verifyUser(Sting username, String password)`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	- Das Spiel wurde bereits gestartet
	- Es besteht eine Verbindung zum Internet
Nachbedingungen:	❑ Der Spieler ist mit dem Server verbunden
	❑ Das Spielmenü erscheint

5.3 Contract CO3: new SingleplayerGame

Operation:	`new SingleplayerGame()`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	- Das Spiel wurde bereits gestartet
	- Der Benutzer ist mit dem Server verbunden
Nachbedingungen:	❑ Ein Einzelspieler-Spiel wurde gestartet

5.4 Contract CO4: new MultiplayerGame

Operation:	`new MultiplayerGame()`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	- Das Spiel wurde bereits gestartet
	- Der Benutzer ist mit dem Server verbunden
Nachbedingungen:	❑ Die Multiplayer Lobby wird angezeigt

5.5 Contract CO5: hostGame

Operation:	`void hostGame()`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	- Das Spiel wurde bereits gestartet
	- Der Spieler hat den Multiplayer Modus gewählt
Nachbedingungen:	❑ Der Spieler hostet ein Spiel und wartet auf weitere Spieler
	❑ Sind alle Spieler bereit, wird das Multiplayer Spiel gestartet

5.6 Contract CO6: searchGame

Operation:	`Game[] searchGame()`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	- Das Spiel wurde bereits gestartet - Der Benutzer ist mit dem Server verbunden - Der Spieler hat den Multiplayer Modus gewählt
Nachbedingungen:	❏ Es erscheint eine Liste mit den zur Zeit gehosteten Spieler, welche noch nicht gestartet sind.

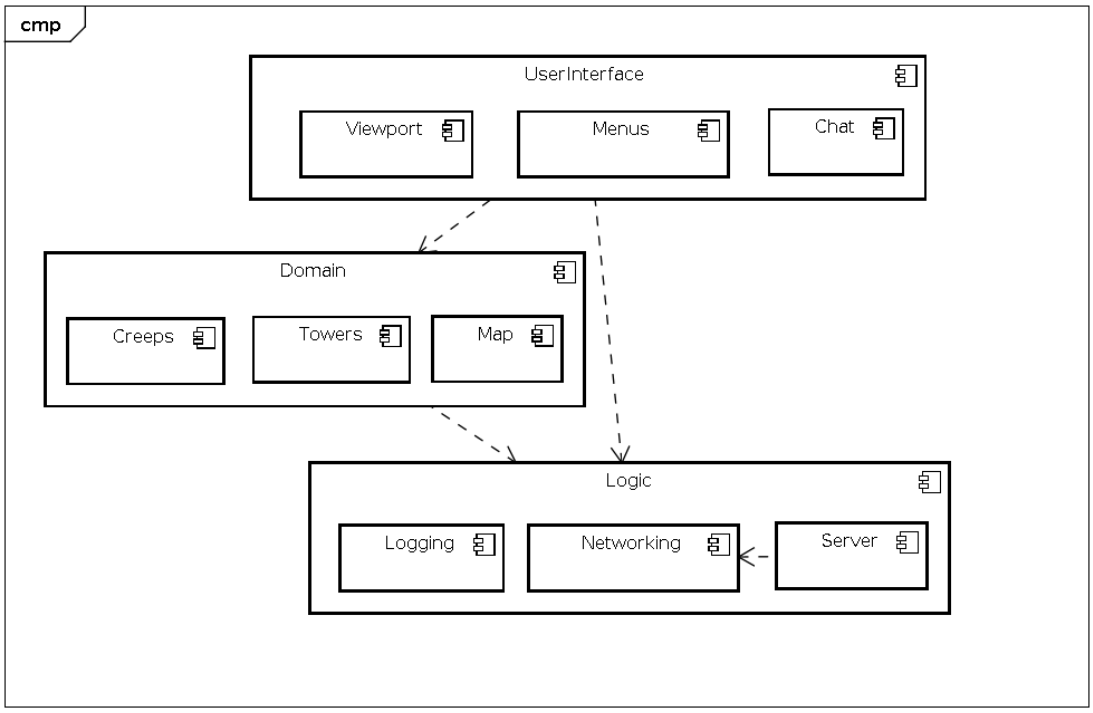
5.7 Contract CO7: selectGame

Operation:	`Game selectGame(int id)`
Querverweise:	Use Cases: Start Game
Vorbedingungen:	- Das Spiel wurde bereits gestartet - Der Benutzer ist mit dem Server verbunden - Der Spieler hat den Multiplayer Modus gewählt - Der Spieler hat gewählt, dass er einem Spiel beitreten will
Nachbedingungen:	❏ Der Spieler hat ein Spiel gewählt und wartet auf weitere Spieler ❏ Sind alle Spieler bereit, wird das Multiplayer Spiel gestartet

6 Architektur

Das CCTD-Projekt ist Netzwerk-Applikation, welche aus zwei Hauptteilen besteht: Das effektive Spiel und eine Server-Komponente, welche die Lobby zur Verfügung stellt und alles andere, was für die Initialisierung eines Spieles nötig ist. Die Server-Komponente muss unabhängig von der Spiel-Komponente sein, damit Erweiterung am Server einen minimalen Einfluss auf das Spiel selber hat und damit in einem späteren Schritt dediziert gestart werden kann.
Es ist im Allgemeinen angebracht, dass ein solches System auf dem Schichten-Model aufbauen sollte. Dies ist eine starke Unterstützung, um die Komplexität des System zu reduzieren und so minimal komplexen Programmcode zu erhalten. Moderne System werden in der Regel in 3 Schichten aufgeteilt: UserInterface, Domain, Logic.

6.1 Logische Architektur



UI	
Viewport	Zeichnet und definiert den sichtbaren Teil der Karte.
Menus	Bietet Aktionen (Build,Upgrade,...) während dem Spiel an.
Chat	UI-Elemente für den Spieler-Chat.
Domain	
Creeps	Alle Arten von Creeps.
Towers	Alle Arten von towers inklusive ihren Upgrade Möglichkeiten.
Map	Die Karte und alle Elemente, die sich direkt auf die Karte beziehen.
Logic	
Logging	Bietet eine einheitliche Logging-Schnittstelle für die gesamte Applikation.
Networking	Stellt eine Schnittstelle für die Kommunikation mit physikalisch entfernten Teil der Applikation zur Verfügung.
Server	Beobachtet und Kontrolliert den Spielablauf, bietet eine Lobby

Glossar

CCTD	Cube Circle Tower Defense	1
Creep	gegnerisches Monster, welches abgeschossen werden soll	16
Lane	Weg, auf welchem sich die Creeps bewegen	11
Open-Source	bedeutet, dass eine Software nach einem Prinzip der Offenheit entwickelt wurde. Sie wird dann meist mit einer Lizenz versehen, die die weiterverwendbarkeit der Software relativ detailiert regelt.	9
Spawn-Point	Sind die Punkte auf der Karte, wo die Creeps auf das Spielfeld tretten	6
to spawn	erzeugen, generieren von creeps	11
Tower	Siehe Turm	4
Turm	Eine Einheit, die vom Benutzer gebaut wird und autonom auf Creeps schiesst.	4
Viewport	Virtuelles Fenster für den Benutzer wodurch sie/er einen Bereich des Spiels sieht	16
Wave	(Welle), besteht aus einer variirenden Anzahl an Creeps, die zusammen auf das Spielfeld tretten	6
Wave	Spielrunde. Bestimmte Anzahl creeps, die bei jedem Spieler spawnen	11

Abbildungsverzeichnis

Tabellenverzeichnis