Kreuzzüge

Software-Projekt

Timo I. Denk, TGI12

Tobias A. Polatzek, TGI12

Informationstechnik Software

Fachlehrer: Joachim Krieg

Mittwoch, 8. Juli 2015

Inhaltsverzeichnis

Das Titelbild zeigt die Karte einer Kreuzzüge-Partie.

Eigenständigkeitserklärung

Hiermit versichern wir, Timo Denk und Tobias Polatzek, dass wir die vorliegende schriftliche Ausarbeitung selbstständig und nur mit den angegebenen Hilfsmitteln verfasst haben. Alle Passagen, die wir wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen haben, haben wir als Zitat mit Angabe der Quelle kenntlich gemacht.

1. Einführung

Das Thema dieser Arbeit ist die Dokumentation des Software-Projekts “Kreuzzüge”. Es entstand im Rahmen einer Projektarbeit im Fach ITS (Informationstechnik Software) in der 12. und 13. Klasse.

Kreuzzüge ist ein rundenbasiertes 2D Strategiespiel. Ziel ist die Neutralisierung des feindlichen Hauptquartiers. Dafür muss der Spieler Einheiten ausbilden, Städte, Dörfer und Ausbildungsstätten (Kaserne, Stallungen, Fabrik, Hafen, Tempel) einnehmen. Die beiden Konfliktparteien sind Kreuzritter und Sarazenen. Das Spiel spielt zur Zeit der Kreuzzüge.

Taktisches Spielen ist bei Kreuzzüge erfordert, da es auf der Spielkarte verschiedenen Untergründe, wie z. B. Wälder, Berge, Wege, etc. gibt. Diese bieten einer Einheit Vor- und Nachteile: In Bergen können sich Einheiten beispielsweise besser verteidigen, auf Wegen können sie weiter laufen. Da die meisten Einheiten einen begrenzten Proviant und manche auch begrenzte Munition haben, ist es erforderlich eine Einheit immer wieder in einer Stadt zu stationieren, da sich dort die Leben, Munition und Proviant der Einheit pro Runde um 1 erhöhen.

Die Idee von Kreuzzüge ist nicht neu: Vor über 10 Jahren veröffentlichte die media Verlagsgesellschaft mbH ein Spiel gleichen Namens, produziert wurde es von TASK four. Das Originalspiel hat keinen funktionsfähigen Multiplayer und ermöglicht keine lokalen Spiele zwischen zwei Personen. Ein weiterer Nachteil ist, dass immer 10x10 Spielfelder angezeigt werden und man bei größeren Karten (z. B. 30x10) scrollen muss. Die Idee bei diesem Projekt war es, das Spiel so originalgetreu wie möglich nachzuprogrammieren und die Schwachstellen zu verbessern.

Kapitel 2 stellt detailliert die Softwarearchitektur von Kreuzzüge dar. Dabei wird zwischen client- und serverseitigen Systeme unterschieden. Kapitel 3 beinhaltet den chronologischen Entwicklungsablauf von Kreuzzüge, der unter anderem durch ein Gantt-Diagramm dargestellt wird. Die Anleitung für das Strategiespiel befindet sich in Kapitel 4 und im letzten Kapitel finden sich noch einige Screenshots.

2. Softwarearchitektur

Die Kreuzzüge Systemarchitektur lässt sich in zwei große Bereiche gliedern:

- Clientseitige Anwendung
- Serverseitige Systeme

Auf der Client Seite befindet sich eine Windows Anwendung, das eigentliche Spiel.

Zu den serverseitigen Systemen zählt neben einem PHP-Skript, das eine MySQL-Datenbank anspricht, auch der FTP-Server. Alle drei laufen auf einem Raspberry Pi Model B und werden ausschließlich für die Multiplayer Funktionen von Kreuzzüge benötigt.

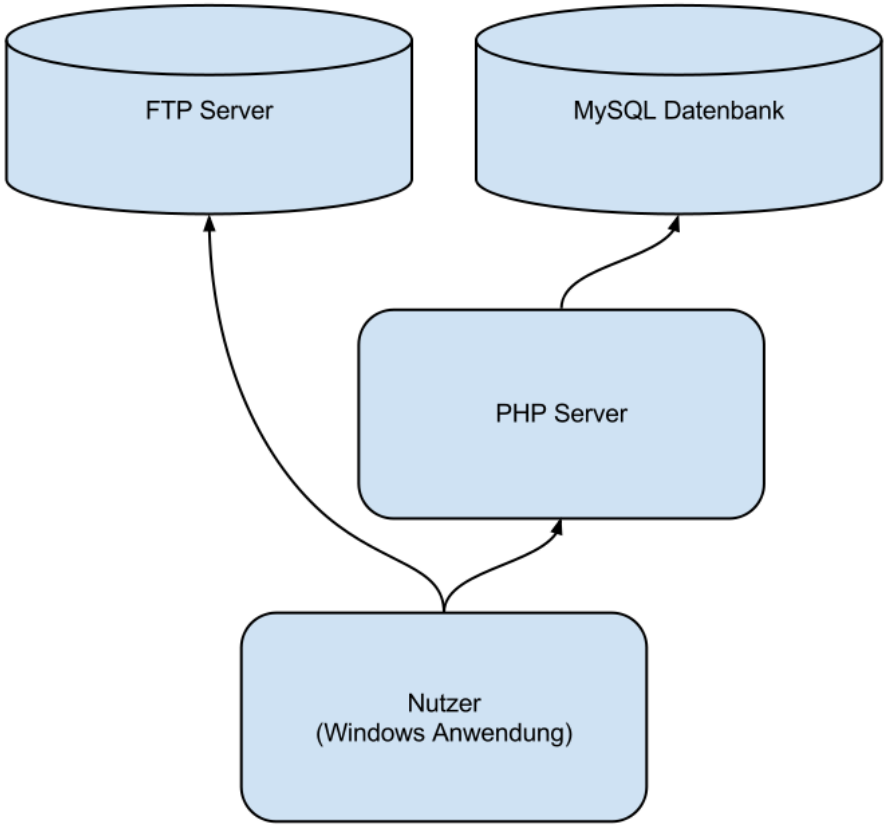


Abb.: Systemarchitektur und Beziehungen der einzelnen Systeme

2.1 Clientseitige Anwendung

Die C#-Anwendung Kreuzzüge ist in Form der Drei-Schichten-Architektur aufgebaut. Das bedeutet, dass grafische Benutzeroberfläche (GUI), Steuerung und Daten voneinander getrennt, also in verschiedenen Klassen vorliegen. Vorteil dieses Konzepts ist eine höhere Kohäsion, die Güte der Abbildung einer Teilaufgabe, der einzelnen Module bei größerer Unabhängigkeit voneinander.

Aufgrund der Komplexität des Projektes wurde für jeden Anwendungsstatus sowohl eine eigene GUI-, als auch eine eigene Steuerungs-Klasse programmiert. Diese untergeordneten GUI- und Steuerungs-Klassen werden von einer Hauptsteuerung kontrolliert. Kreuzzüge kennt folgende Anwendungszustände:

- *MainWindow*: Im Hauptfenster werden innerhalb eines *ContentControl*[1] alle nachfolgend gelisteten *Screens* angezeigt.
- *MenuScreen*: Das Hauptmenü bietet die Möglichkeit, sich für die Multiplayer-Dienste einzuloggen oder ein lokales Spiel zu beginnen.
- *MultiplayerScreen*: Im Multiplayer-Bildschirm kann der Anwender in einem “Server-Browser” nach offenen Partien suchen oder selbst ein Spiel hosten.
- *GameScreen*: Der Spiel-Bildschirm wird während einem Spiel angezeigt.
- *GameOverScreen*: Nach einem Spiel erscheint der Spiel-Beendet-Bildschirm.

Die Datenschicht wird bei Kreuzzüge von mehreren Klassen namens *Data*, *Server*, *Sound* und *R* repräsentiert.

Die Daten Klassen sind alle als *static* definiert. Das ist gemäß der Drei-Schichten-Architektur ungewöhnlich, bietet sich im Fall von Kreuzzüge aber an, da die Daten nicht sinnvoll in Form von Attributen gespeichert werden können. Die Daten liegen auch zur Laufzeit auf der Festplatte oder auf dem Server vor und es wäre deshalb ineffizient, mehrere Objekte zu erzeugen, die dieselben Daten an die Steuerung durchreichen.

Als Programmiersprache für die WPF[2]-Anwendung wurde *C#* gewählt. Programmiert wurde das Spiel in der Entwicklungsumgebung Visual Studio[3] Professional 2013 (Version 12.0) von Microsoft. Die EXE-Anwendung wurde unter Windows 7 getestet und ist auch auf dieses Betriebssystem optimiert.

Die Anwendung ist durchgehend mehrsprachig entwickelt, sodass es ohne größeren Aufwand möglich ist, weitere Sprachen zu den bereits implementierten (Deutsch und Englisch) hinzuzufügen. Alle Texte, die der Nutzer sieht, sind in sogenannte *ResourceDictionary*[4] (in Form von XAML-Dateien) ausgelagert. Jedem Text ist eine eindeutige ID zugewiesen, ein sogenannter *Key*. Eine Zeile, die den Schriftzug “New Game” definiert, sieht im englischen *ResourceDictionary* beispielsweise so aus:

```
<system:String x:Key="NewGame">New Game</system:String>
```

In der deutschen XAML-Datei steht die Übersetzung, mit gleichem Key:

```
<system:String x:Key="NewGame">Neues Spiel</system:String>
```

Die Wahl des Wörterbuches erfolgt in Abhängigkeit von der eingestellten Sprache des Betriebssystems. Falls kein Wörterbuch für die eingestellte Sprache existiert, wird auf Englisch zurückgegriffen.

---

[1] https://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol(v=vs.110).aspx

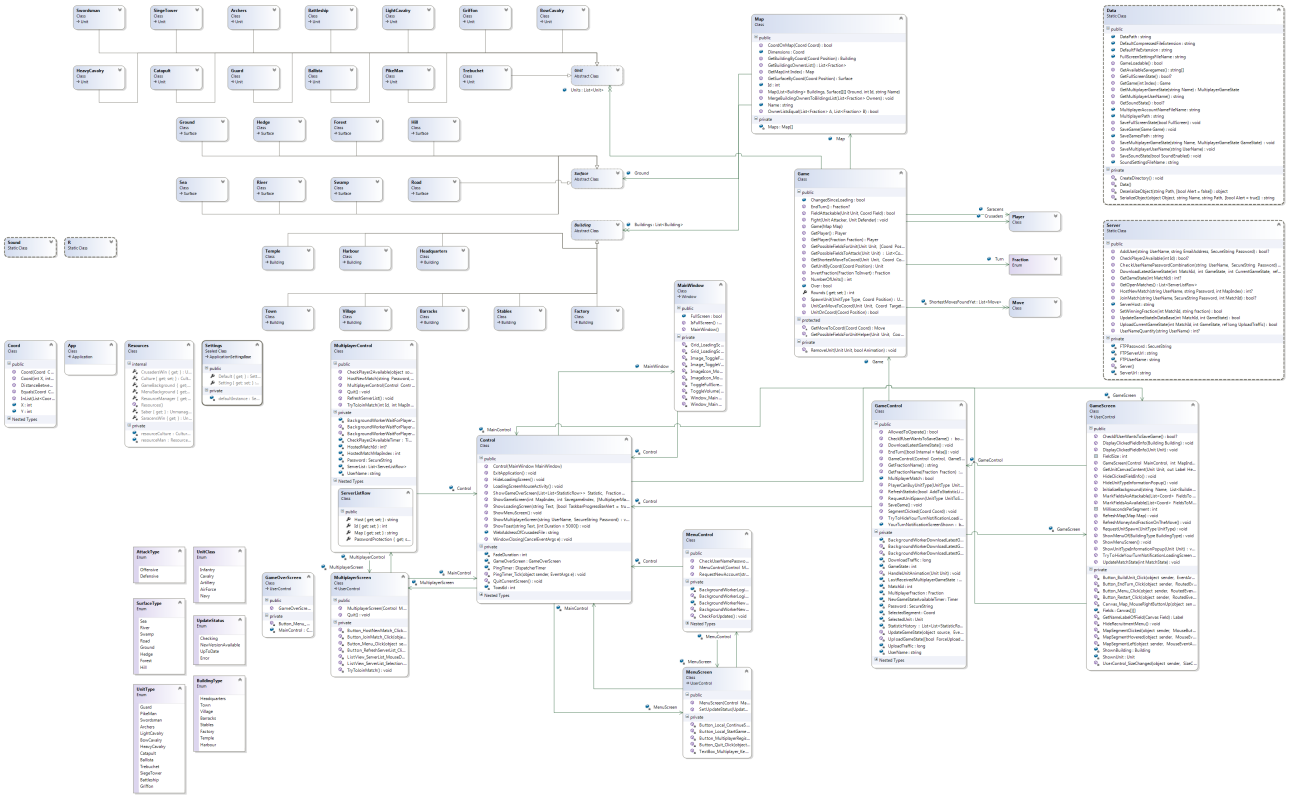
[2] https://msdn.microsoft.com/en-us/library/ms754130(v=vs.100).aspx

[3] https://www.visualstudio.com/

[4] https://msdn.microsoft.com/en-us/library/system.windows.resourcedictionary(v=vs.110).aspx

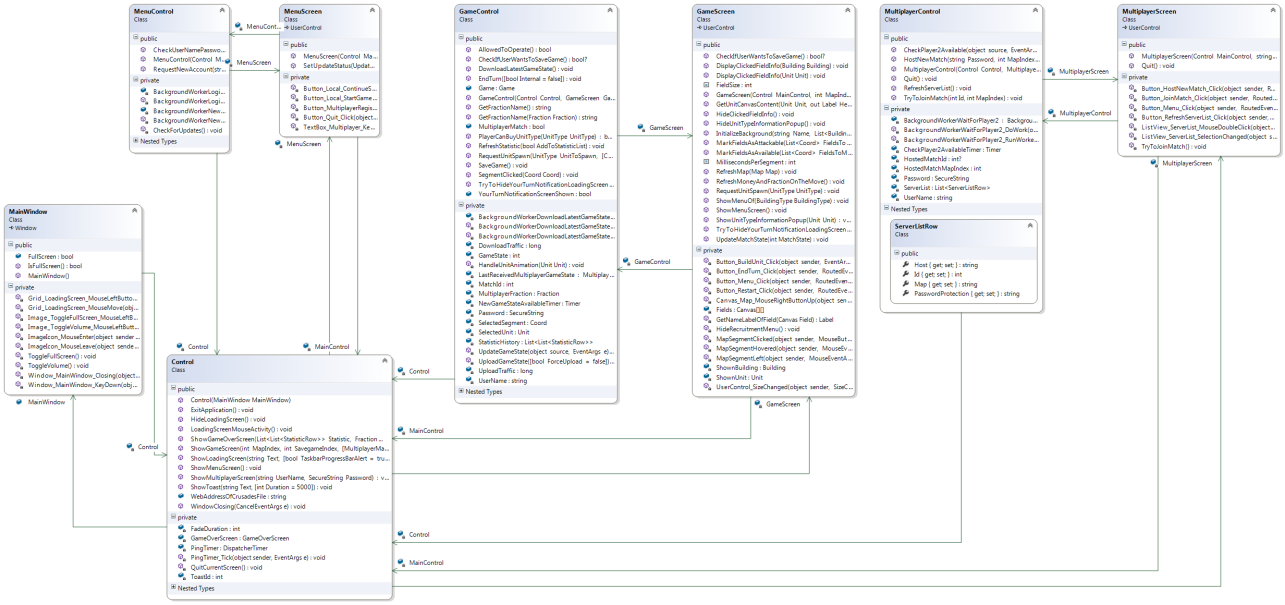
2.1.1 Klassendiagramme

Die über 50 Klassen von Kreuzzüge werden nachfolgend semantisch getrennt dargestellt und kurz erläutert.



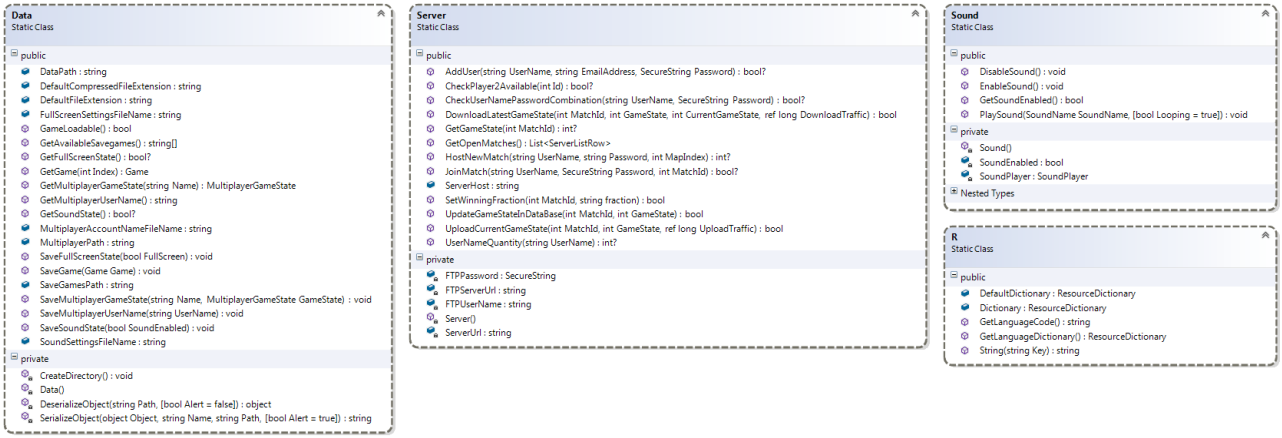
2.1.1.1 Steuerungs- und GUI-Klassen

- *Control*: Haupt-Steuerung
- *MainWindow*: Haupt-Fenster
- *MenuControl*: Logik des Hauptmenüs
- *MenuScreen*: Darstellung des Hauptmenüs
- *GameControl*: Steuerung des Spiels
- *GameScreen*: Darstellung des Spiels
- *MultiplayerControl*: Logik des Multiplayer-Menüs (Server-Liste, Match-Hosting, Match-Joining, etc.)
- *MultiplayerScreen*: Darstellung des Multiplayer-Menüs



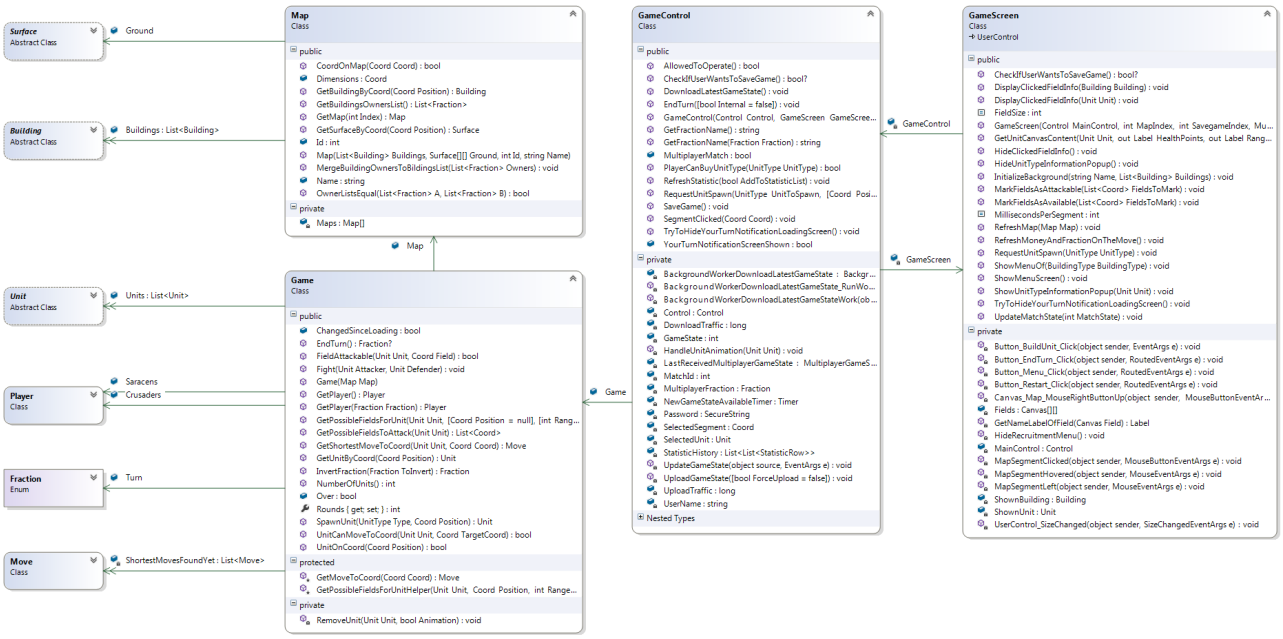
2.1.1.2 Daten-Klassen

- *Data*: Erlaubt das Einlesen und Speichern von Spielständen und Einstellungen des Benutzers
- *Server*: Sendet HTTP Anfragen an den PHP-Server und lädt Dateien auf den FTP-Server hoch und herunter
- *Sound*: Verwaltet die Wiedergabe von Audio-Dateien
- *R* (Ressourcen-Klasse): verwaltet Wörterbücher



2.1.1.3 Spiel-Klassen

- *GameControl*: Steuerung des Spiels
- *GameScreen*: Darstellung des Spiels
- *Game*: Logik und Daten (Einheiten, Karte, etc.) des Spiels
- *Map*: Kartenbezogene Daten des Spiels
- *Surface*, *Building*, *Unit*: Abstrakte Klassen für Untergrund-, Gebäude- und Einheitentypen
- *Player*: Informationen über einen Spieler
- *Move*: Informationen über den Zug einer Einheit



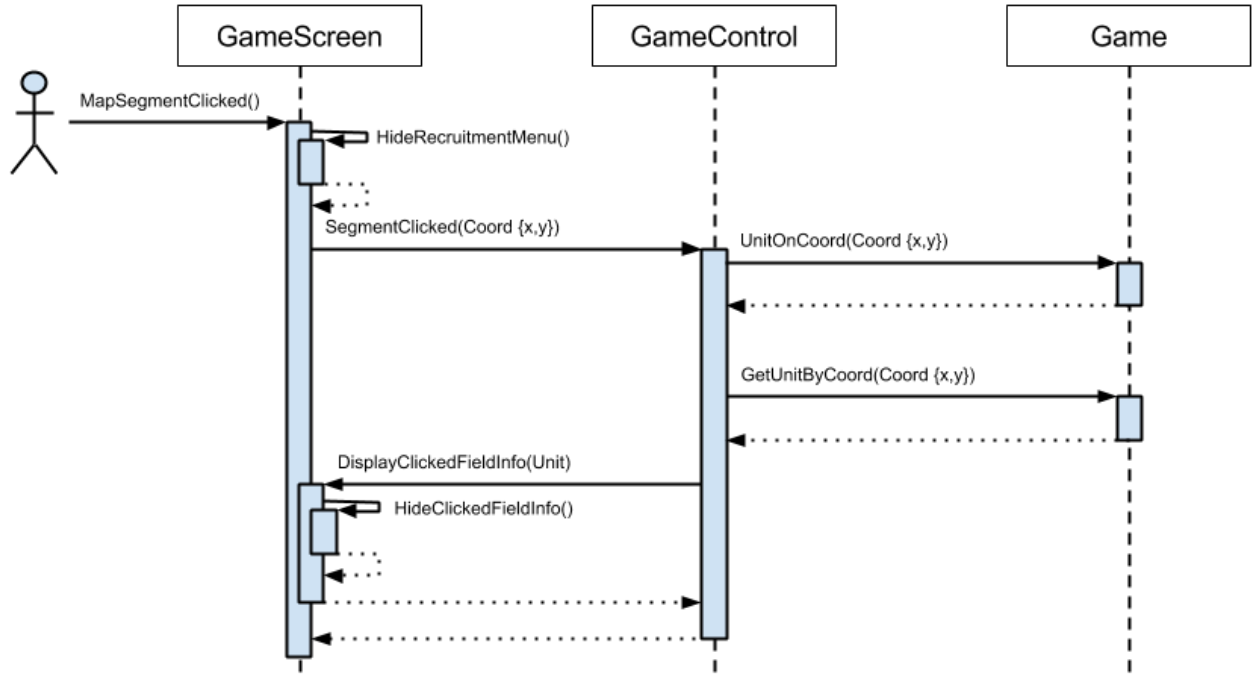
2.1.1.4 Einheiten-, Gebäude- und Bodentyp-Klassen

Die verschiedenen Einheiten-, Gebäude- und Bodentypen erben von den drei Abstrakten Klassen *Unit*, *Building* und *Surface* und sind dadurch polymorph.

2.1.2 Sequenzdiagramme

Beispielhaft sollen hier die Aufrufe der Routine *MapSegmentClicked* der *GameScreen*-Klasse in Form eines Sequenzdiagramms visualisiert werden. Die Funktion *MapSegmentClicked* wird aufgerufen, wenn der Nutzer auf ein Feld der Karte klickt.

Im Beispiel wird dabei auf ein Feld geklickt, auf dem eine Einheit steht, was durch einen Aufruf der Methode *UnitOnCoord* des *Game*-Objekts bestimmt wird. Das Objekt der Einheit wird daraufhin durch den Aufruf der Methode *GetUnitByCoord* vom *Game*-Objekt an das *GameControl*-Objekt übergeben. Dieses gibt es an das *GameScreen*-Objekt weiter, welches nach dem zurücksetzen der Informationsanzeige die Informationen über die angeklickte Einheit anzeigt.



2.2 Serverseitige Systeme

Die serverseitigen Systeme laufen alle auf einem Raspberry Pi Model B[5], der für dieses Projekt mit dem Linux-Betriebsystem Raspbian[6] aufgesetzt wurde. Um die Erreichbarkeit des Raspberry Pi über eine statische URL[7] zu gewährleisten wird der Service TwoDNS[8] verwendet. Auf dem Raspberry Pi läuft ein PHP-Server, eine MySQL-Datenbank und ein FTP-Server.

2.2.1 PHP-Server

Auf dem PHP-Server läuft ein Skript, das die dahinter betriebene MySQL-Datenbank verwaltet. Das stellt einen sicherheitstechnischen Vorteil dar, da die Windows Anwendung nicht direkt auf die Datenbank zugreift. Es ist aufgrund des zwischengeschalteten PHP-Skripts jederzeit möglich, den Datenverkehr zu kontrollieren und wenn nötig zu regulieren. Beispielsweise liese sich eine maximale Anfragen-Anzahl pro IP-Adresse pro Stunde festlegen. Außerdem vereinfacht der PHP-Server die Entwicklung von Kreuzzüge für andere Plattformen, die untereinander kompatibel wären.

---

[5] https://www.raspberrypi.org/products/model-b/

[6] https://www.raspbian.org/

[7] http://simsso.dd-dns.de/crusades/server.php

[8] https://www.twodns.de/

Die Kommunikation zwischen Anwendung und PHP-Server findet in Form von GET-HTTP-Anfragen statt. In der URL werden die für den Server relevanten Informationen (GET-Parameter) übergeben. Der PHP-Server stellt die folgenden Funktionen bereit.

$_GET[“action”]	GET-Parameter	Aktion	Rückgabe
client-ip			• IP-Adresse des Clients
username-quantity	• user-name		• 0: Benutzername verfügbar • 1: Benutzername in Verwendung
add-user	• user-name • email-address • password	Fügt einen neuen Nutzer hinzu	• 0: Fehler • 1: Erfolg
check-username-password-combination	• user-name • password		• 0: Falsche Kombination • 1: Richtige Kombination
get-number-of-users			• Anzahl der registrierten Benutzer
get-open-matches-list			• Liste der offenen Multiplayer-Spiele
username2id	• user-name		• Zum Benutzernamen gehörige ID
host-new-match	• user-name	Fügt ein neues Spiel zur Spiele-Tabelle hinzu	• -1: Fehler • Zahl >= 0: ID des hinzugefügten Spiels
check-player-2-available	• match-id		0: Spieler 2 nicht beigetreten 1: Spieler 2 beigetreten
join-match	• user-name • password • match-id	Fügt den beitretenden Spieler in der Spiel-Tabelle hinzu	• 0: Fehler • 1: Erfolg
get-game-state	• match-id		• -1: Fehler • Zahl >= 0: ID des Spielstandes
update-game-state	• match-id • game-state	Ändert die ID des Spielstandes auf den übergebenen Wert	• 1

2.2.2 MySQL-Datenbank

Die Datenbank von Kreuzzüge heißt *crusades* besteht aus zwei Tabellen, deren Aufbau nachfolgend dargestellt wird.

2.2.2.1 Nutzer-Tabelle: *user*

Die Tabelle *user* speichert Nutzerdaten. Dazu zählt der Benutzername (*user-name*), die Email-Adresse (*email-address*), das Passwort (*password*) und den Zeitpunkt der Registrierung (*time-of-registration*).

#	Spaltenname	Typ	Attribute	Null	Standardwert	Extra
1	**id**	int(10)	unsigned	no	none	auto increment
2	**user-name**	text		no	none	
3	**email-address**	text		no	none	
4	**password**	text		no	none	
5	**time-of-registration**	timestamp		no	current timestamp	

2.2.2.2 Spiele-Tabelle: *games*

In der Tabelle *games* werden laufende und abgeschlossene Multiplayer-Partien gespeichert. Das beinhaltet neben der ID beider Spieler eine Referenz zum, auf den FTP-Server hochgeladenen, aktuellen Spielstand (*game-state*), die ID der Karte (*map*) und weitere Informationen.

Die Zeilen *host* und *player-2* sind Fremdschlüssel und stehen für die jeweiligen Einträge aus der *user*-Tabelle.

#	Spaltenname	Typ	Attribute	Null	Standardwert	Extra
1	**id**	int(10)	unsigned	no	none	auto increment
2	**host**	int(10)	unsigned	no	none	
3	**player-2**	int(10)		yes	null	
4	**map**	int(10)	unsigned	no	none	
5	**game-state**	int(10)	unsigned	yes	null	
6	**password**	text		no	none	
7	**host-won**	tinyint(1)		yes	null	
8	**last-host-activity**	bigint(20)	unsigned	no	none	
9	**last-player-2-activity**	bigint(20)	unsigned	no	none	
10	**round**	int(10)	unsigned	no	0	

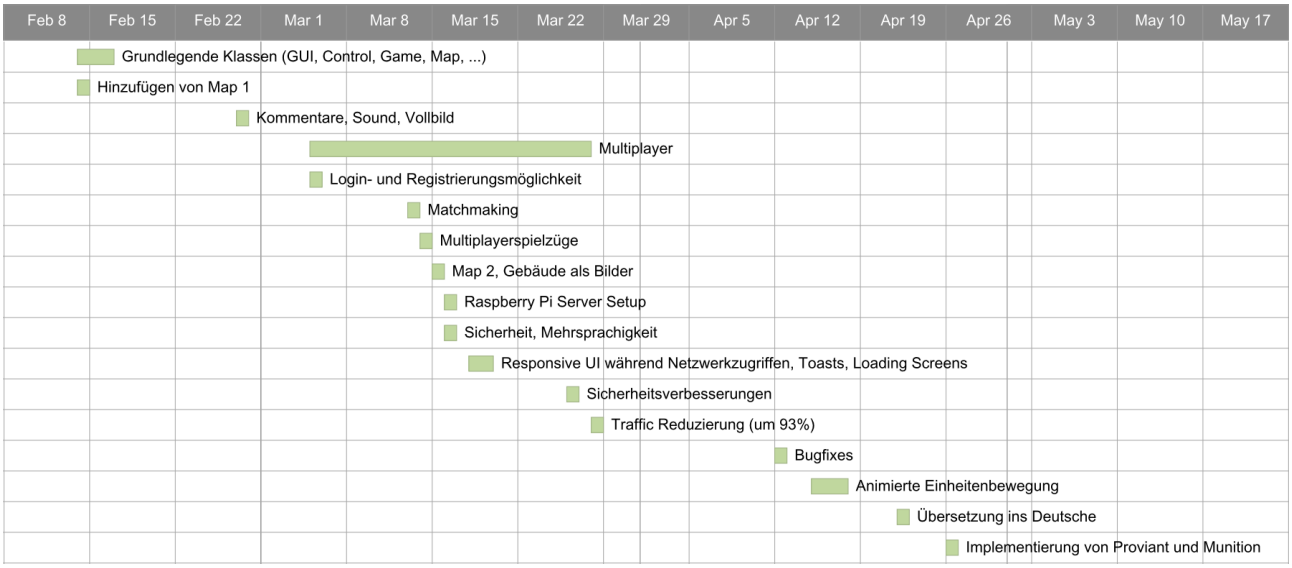
2.2.3 FTP-Server

Der FTP-Server dient zum Austauschen von Spielständen. Bei jedem Zug, den ein Spieler in einer Multiplayer-Partie durchführt, werden die relevanten Daten serialisiert und als Datei auf der Festplatte abgespeichert. Der gespeicherte, neue Spielstand wird anschließend komprimiert und auf den FTP-Server hochgeladen. Sobald die Anwendung des anderen Spielers durch die, in der MySQL-Datenbank eingetragene Veränderung darüber informiert ist, dass ein neuer Spielstand verfügbar ist, wird der Spielstand heruntergeladen, dekomprimiert, ins Spiel geladen (Deserialisierung) und anschließend von Server gelöscht, um den Speicherplatz wieder freizugeben.

3. Entwicklungsablauf

Dieses Kapitel dokumentiert die Entwicklung des Kreuzzüge Projekts chronologisch. Während die Programmierung der Grundfunktionen nur wenige Tage einnahm, war die Entwicklung des Multiplayers sehr zeitintensiv. Über einen Zeitraum von knapp einem Monat wurden die Funktionen Registrierung, Login, Matchmaking, Multiplayer-Spielfunktion, Sicherheitsverbesserungen und eine Traffic Reduzierung implementiert. Letztere senkt die Größe der übertragenen Daten um 93% durch Anpassung von Datentypen, GZip[9]-Komprimierung und Nicht-Versenden von unveränderten Daten. Bugfixes, animierte Einheitenbewegung, die Übersetzung ins Deutsche und die Implementierung von Proviant und Munition folgten im nächsten Monat.

Das Gantt-Diagramm stellt die Entwicklungsschritte chronologisch dar:



Termin für die Zwischenabgabe des Projekts war der 8. Juli 2015.

[9] https://www.gnu.org/software/gzip/

4. Anleitung

Zu Beginn des Spiels besitzt jede der Parteien, Kreuzritter und Sarazenen, ein Hauptquartier, eine Kaserne, mindestens eine Stadt und etwas Gold. Das Ziel des Spiels ist es, das gegnerische Hauptquartier einzunehmen. Das Hauptquartier ist deshalb das wichtigste Gebäude, das es zu verteidigen gilt. Welches Gebäude welcher Partei gehört, kann an der Beflaggung erkannt werden. Die Kreuzflagge kennzeichnet Gebäude der Kreuzritter, die Halbmondflagge die der Sarazenen.

In der Kaserne, den Stallungen und der Fabrik können Truppen mit unterschiedlichen Eigenschaften für Gold rekrutiert werden. Städte und Dörfer zahlen jede Runde Steuern in die Kriegskasse. Neutrale oder feindliche Gebäude können erobert werden, indem eine Infanterie-Einheit mehrere Runden in dem Gebäude stationiert wird.

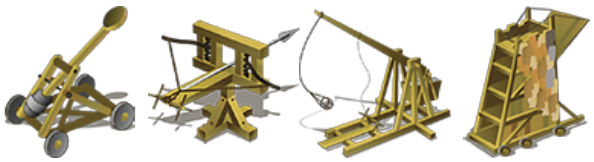
Die Felder der Karte haben verschiedene Eigenschaften, wie schnellere Fortbewegung oder höhere Defensivkraft. Nicht jede Einheit kann über jedes Feld laufen: Ein Katapult beispielsweise kann nicht durch den Wald gezogen werden.

4.1 Einheiten

4.1.1 Infanterie

4.1.2 Kavallerie

4.1.3 Artillerie



4.1.4 Marine

4.1.5 Lufteinheit

4.2 Gebäude

4.2.1 Hauptquartier

Das Hauptquartier zahlt pro Runde zwei Gold Abgaben. Eine Partie hat verloren, wenn sie kein Hauptgebäude mehr besitzt.

4.2.2 Stadt und Dorf

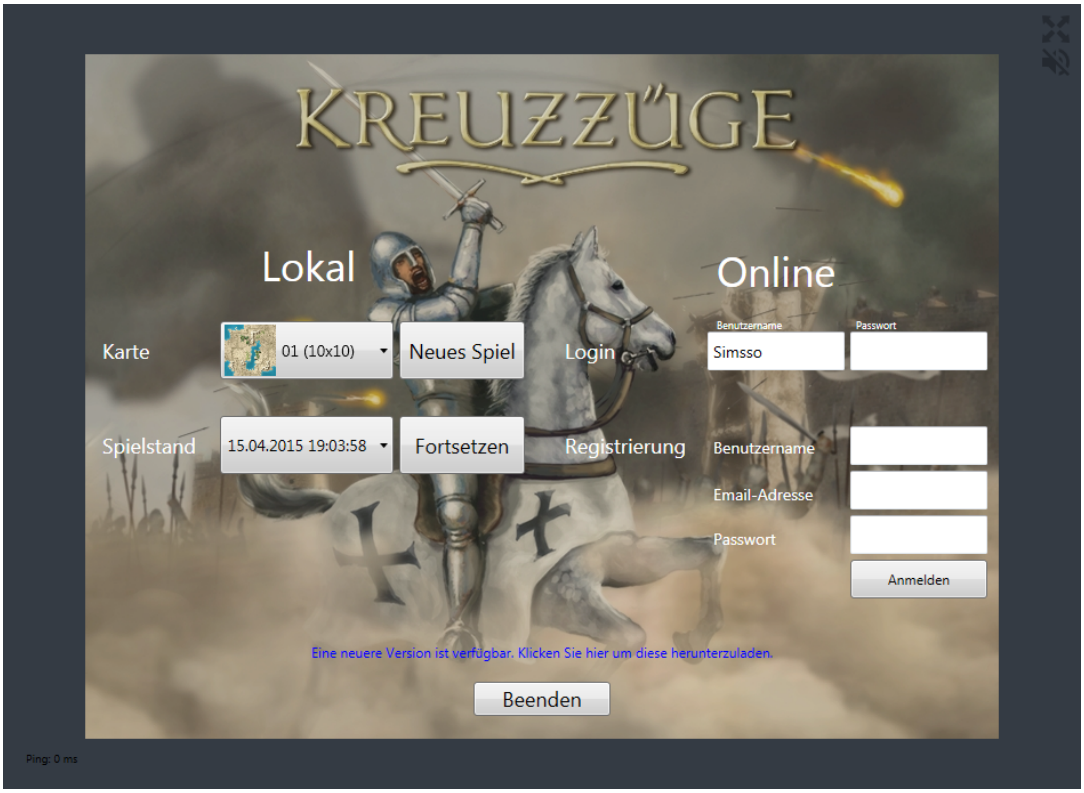
Städte zahlen pro Runde vier Gold Abgaben, Dörfer nur ein Gold.

4.2.3 Ausbildungsstätten (Kaserne, Stallungen, Fabrik, Hafen, Tempel)

In der Kaserne kann Infanterie, in den Stallungen Kavallerie, in der Fabrik Artillerie, im Hafen das Schlachtschiff und im Tempel der Greif rekrutiert werden.

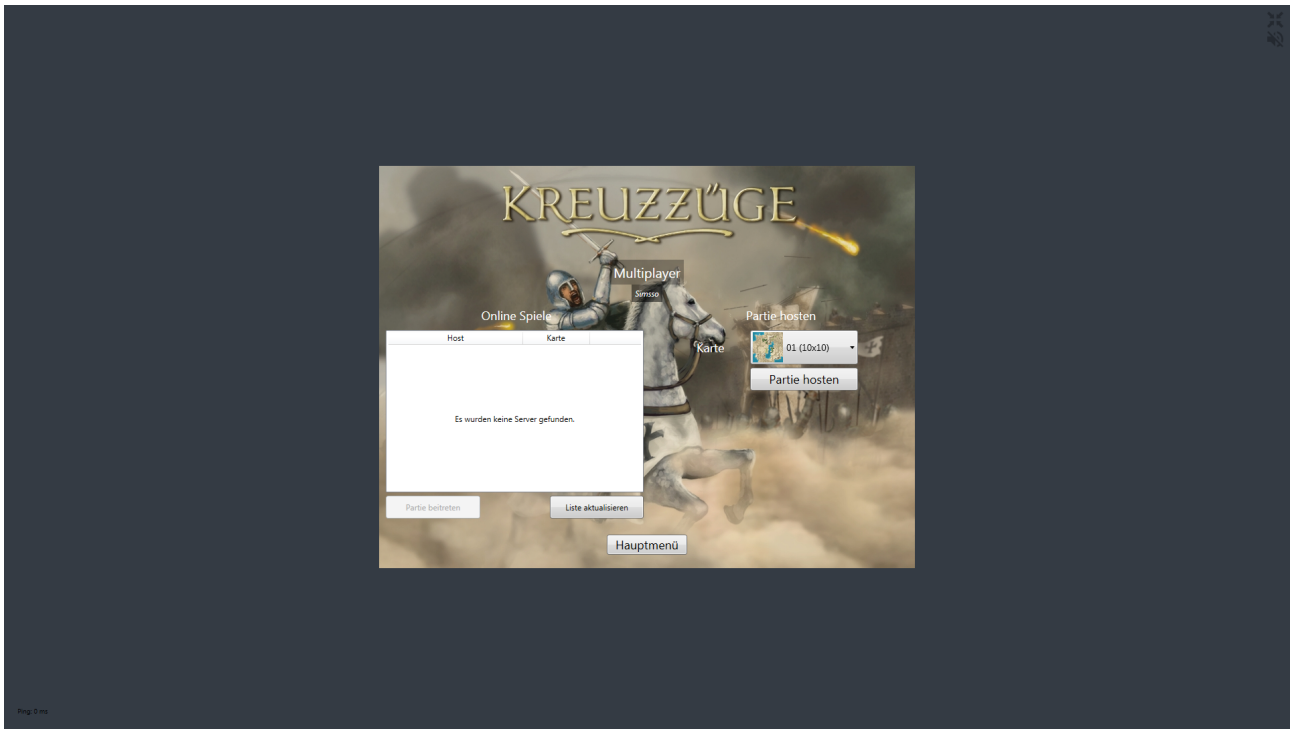
5. Screenshots

5.1 Hauptmenü



5.2 Spielbildschirm

5.3 Multiplayer-Menü



Abbildungsnachweis

Die Klassendiagramme wurden in der Entwicklungsumgebung Visual Studio[10] Professional 2013 (Version 12.0) von Microsoft erstellt und daraus exportiert.

Die Abbildung der Systemarchitektur sowie das Sequenzdiagramm wurden mit der “Drawings” Funktion von Google Drive[11] erstellt.

Das Copyright für die Bilder von Einheiten und Gebäuden liegt bei TASK four[12], dem Entwickler der ursprünglichen Kreuzzüge Version.

Die Screenshots (Kapitel 5) wurden selbst aufgenommen.

Das Gantt-Diagramm aus Kapitel 3 wurde mit dem Online-Tool “Gant-Chart-Software” von Smartsheet[13] kreiert.

Onlineverfügbarkeit

Eine PDF Version dieser Ausarbeitung steht auf http://www.simsso.de/?weiterleitung=Kreuzzeuge zum Download bereit.

[10] https://www.visualstudio.com/

[11] https://www.google.com/drive/

[12] http://taskfour.net/

[13] https://www.smartsheet.com/gantt-chart-software