Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

Fachbereich Design Informatik Medien

Anforderungsspezifikation Campusadventure

im Fach Softwaretechnik im SS2012

Vorgelegt von

Dominik Schuhmann

Tino Landmann

Simon Hardt

David Gens

bei	Prof. Dr. Wolfgang Weitz
am	17. Juni 2012
in der Version	248.

Rev	Autor	Datum	Beschreibung	Dateien
248	dgens001	17-06-2012	campusformat in der spec ergaenzt	/spec/meta/svnlog.tex /spec/kapitel/ui/room_door_fancy.ppm /spec/spec.pdf /spec/kapitel/ui.tex /spec/kapitel/ui/room_door_fancy.png
247	dgens001	17-06-2012	Map parsen weitergemacht	/trunk/CamptureTheFlag/src/CamptureTheFlag/log /trunk/CamptureTheFlag/src/CamptureTheFlag/da /trunk/CamptureTheFlag/src/CamptureTheFlag/da /trunk/CamptureTheFlag/src/CamptureTheFlag/tes /trunk/CamptureTheFlag/src/CamptureTheFlag/da
246	shard001	17-06-2012	minor fixes	/trunk/CamptureTheFlag/src/CamptureTheFlag/da /trunk/CamptureTheFlag/src/CamptureTheFlag/log
245	shard001	17-06-2012	ACHTUNG: neuer Ablauf zum Starten: Klasse Start.java übernimmt main() GameC erstellt Fenster im Konstruktor Fenster erstellt Hauptmenu im Konstruktor Alle anderen Panels werden erst mit neuesSpiel() oder spielLaden() erstellt (im GameC)	/trunk/CamptureTheFlag/src/CamptureTheFlag/da /trunk/CamptureTheFlag/src/CamptureTheFlag/tes /trunk/CamptureTheFlag/src/CamptureTheFlag/log /trunk/CamptureTheFlag/src/CamptureTheFlag/log

244	shard001	17-06-2012	Jpanels testweise bunt angelegt (direkt im Konstruktor) Umschalten per ESC geht TODO: Elemente in Funktionen auslagern / Konstruktor ohne grafische Elemente / ESC geht nur nach INGAME wenn ein Spiel läuft usw.	/trunk/CamptureTheFlag/src/CamptureTh /trunk/CamptureTheFlag/src/CamptureTh /trunk/CamptureTheFlag/src/CamptureTh /trunk/CamptureTheFlag/src/CamptureTh /trunk/CamptureTheFlag/src/CamptureTh /trunk/CamptureTheFlag/src/CamptureTh

Inhaltsverzeichnis

1 Einleitung **7**

2 Projektgrundlagen **8**
2.1 Zielsetzung . . . 8
2.2 Inhaltlicher Überblick . . . 8
2.3 Technische Anforderungen . . . 9
2.4 Architektur . . . 9
2.5 Erweiterbarkeit . . . 10

3 Abläufe und Funktionen **11**
3.1 Menümodus . . . 11
3.2 Ingamemodus . . . 13
3.2.1 Charakterbewegung . . . 13
3.2.2 Interaktion mit Items . . . 14
3.2.3 Interaktion mit NPCs . . . 16
3.2.4 Pausieren . . . 16
3.2.5 Speichern eines Spiels . . . 17
3.2.6 Laden eines Spiels . . . 18

4 Daten- und Domänenmodell **20**
4.1 Überblick . . . 20
4.2 Objekte und Gegenstände . . . 20
4.3 Inventar . . . 22
4.4 Felder, Räume und Campi . . . 22
4.5 NPCs und Dialoge . . . 22
4.6 Spieler und Scheine . . . 23

5 Benutzungsschnittstellen **24**
5.1 Ingame GUI . . . 24
5.2 Menü GUI . . . 27
5.3 Campusdatei *(v0.1)* . . . 28

Glossar **31**

1 Einleitung

In folgendem Dokument werden die Grundlagen und Anforderungen des Softwaretechnikprojektes definiert und spezifiziert.

Abschnitt 2 bietet eine Übersicht über die grundlegenden Anforderungen an das Projekt, sowie die technischen Anforderungen und unsere Umsetzungsideen. Der Abschnitt 3 befasst sich mit den allgemeinen Abläufen und Funktionen der Anwendung, die sich in Menü- und Ingamemodus aufteilen. Einen Überblick über das Domänenmodell zeigt Abschnitt 4 anhand einer genauen Beschreibung der einzelnen Elemente mit Hilfe eines UML-Diagrammes. Die Benutzerschnittstelle zum Spieler wird in Abschnitt 5 mittels einiger Zeichnungen beschrieben. Ebenso ist das Campusdateiformat hier definiert.

2 Projektgrundlagen

Dieses Kapitel bietet eine kurze Übersicht über definierte Ziele und Rahmenbedingungen. Es bietet einen ersten Architekturüberblick und stellt grobe Systembestandteile mit ihren Zuständigkeiten dar.

2.1 Zielsetzung

Ziel des Projekts ist eine selbst entwickelte Spieleapplikation, die ohne aufwändige Grafikdarstellung oder KI einen amüsanten Zeitvertreib bieten kann. Nicht ganz nebenbei soll die Applikation Ergebnis eines größeren Projekts sein, an dem sich softwaretechnische Methoden, kollaborative Arbeit und der nahezu vollständige Entwicklungszyklus üben und testen lassen. Fachliche, technische und andere qualitative Ziele sollen in diesem Dokument zusammengefasst werden.

2.2 Inhaltlicher Überblick

Die Autoren dieser Spezifikation haben sich darauf verständigt, ein Spiel nach der Machart einiger bereits bekannter Roleplaying-Adventure zu entwickeln. Das Spiel könnte also (je nach Storyline) endlos spielbar sein, oder ein festes Ende haben, wobei das hier spezifierte Spiel einen definiertes Spielende haben wird. Um einen gewissen Anreiz für den *Spieler* zu schaffen, sollte die Storyline diesen relativ frei durch den *Campus* führen. Dabei wird es in späteren Erweiterungen möglich sein, den eigenen *Charakter* zu leveln. Inhaltliches Ziel soll das Finden eines bestimmten *Schein*s sein. Der Spieler befindet sich immer auf genau einem *Feld*. Er kann nur mit *Items* oder *Non-Player-Characters (NPCs)*, die sich auf dem gleichen Feld befinden interagieren. Zudem hat er die Möglichkeit zu einem direkt benachbarten Feld zu wechseln. Mehrere Felder bilden einen *Raum*. Räume sind in sich geschlossene Areale. Räume haben einen oder mehrere *Eingänge* und müssen dem Spieler nicht zugänglich sein. Alle Räume zusammen bilden den Campus. Von Zeit zu Zeit sollte der Spieler einen oder mehrere Gegenstände (sog. Items) erhalten, die er in seinen *Inventar* aufnehmen kann. Manche Items sind wertvoll (hilfreich für den Verlauf des Spiels), andere nicht. Einige Items sollten sich kombinieren lassen, oder in anderer Form mit dem Spieler oder der *Spielwelt* interagieren können. Der Spieler sollte ein *Savegame* anlegen und laden können. Das Savegame beinhaltet die Campuskonfiguration und alle Elemente, die sich im Spiel befinden. Es hat ein Datum und eine Versionsnummer (für den Fall einer Erweiterung).

2.3 Technische Anforderungen

Die Applikation soll unter Java Swing entwickelt werden. Sie sollte auf aktuellen Desktop-PCs ohne Leistungsprobleme spielbar sein (10MB freier Festplattenspeicher, 1GB RAM, 2GHz DualCore, 128MB Grafikspeicher, Bildschirmauflösung 1280x1024). Die Applikation sollte im Fenstermodus mit einer Fenstergröße von 1024x768 laufen und nicht vergrößer- oder verkleinerbar sein (entspricht Seitenverhältnis 4:3). Der Fokus soll aber nicht auf möglichst realistischer Echtzeitgrafik, sondern auf dem Spielerlebnis liegen. Darüber hinaus sollte der Spielplan anhand einer Spielplandatei konfigurierbar sein. Dort sollte man zunächst Wände und Spielstart, später vielleicht auch Non-Player-Character und Items positionieren können.

2.4 Architektur

Die Anwendungslogik soll derart beschaffen sein, dass der Spieler anhand einfacher Zugriffe (3-4 Klicks) zu den gesuchten Funktionen findet. Ziel ist nicht, dem Spieler über möglichst viele Einstellungsmöglichkeiten verfügen zu lassen. Es gibt ein Itemsystem, ein Raumsystem und ein Dialogsystem. In eventuellen Erweiterungen können zusätzliche Systeme implementiert werden. Die Architektur sollte später in Darstellung, Logik und Anwendungsdaten systematisch unterteilt werden. Generell lässt sich das Spiel inhaltlich in zwei Modi unterteilen, Ingame- und *Menümodus*. Der Menümodus sollte der Standardmodus sein, in dem die Applikation startet und von dem aus man in den Ingamemodus gelangt. Der Einfachheit halber sind die Abbruchsübergänge in Abbildung 2.1 nicht berücksichtigt.

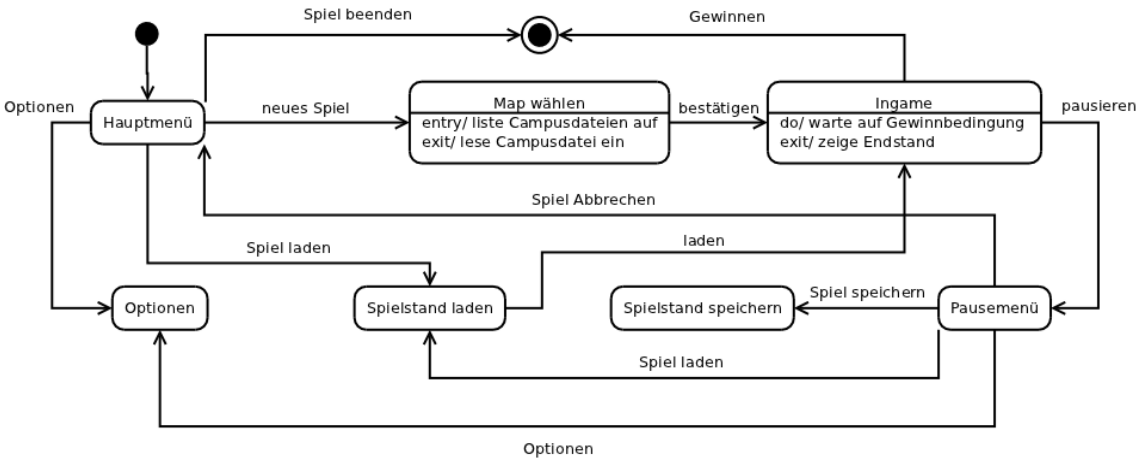

Abbildung 2.1: grobe Struktur der Systemzustände

2.5 Erweiterbarkeit

Der Charakter kann in späteren Erweiterungen z.B. verschiedene Fähigkeiten oder Eigenschaften haben. Diese könnten dann im Laufe des Spiels weiterentwickelt werden (z.B. anhand sog. CreditPoints). CreditPoints würde man durch das erfolgreiche Sammeln der Scheine erhalten, mehrere Scheine könnten dann ein Semester ergeben. Es könnte dann auch einen Abschluss in Form eines höchsten Semesters (z.B. Master of Sciene) geben. Wahlweise könnte der Spieler dann trotz Erreichen dieses Semesters weiterhin durch die Spielwelt navigieren und zufällig auftauchende Scheine (z.B. durch Lösen von Minispielen oder Rätseln) finden. Es könnte dann auch die Möglichkeit geben, unabhängig von den Savegames Charakterprofile zu speichern, zu editieren und zu laden. Daher soll es möglich sein, die Gegenstände, Objekte und Charaktere im Spiel von Version zu Version zu ergänzen und zu erweitern. So könnten in einer zukünftigen Erweiterung der NPCs feste Routen vorgesehen werden, auf denen diese sich wärend dem Spiel bewegen.

3 Abläufe und Funktionen

Dieses Kapitel bietet einen Einblick in das Zusammenspiel der Prozesse und Abläufe des Spiels. Anwendungsfälle der Interaktionen mit dem System, dabei wichtige Anwendungsfunktionen und die jeweiligen Akteure sollen einen klaren Umriss des später implementierten Spiels geben.

3.1 Menümodus

Nach dem Start der Anwendung findet sich der Spieler im Menümodus. Von dort aus kann er die Funktionen des Hauptmenüs erreichen und in andere Menüsichten wechseln. Unter Optionen könnte eine Reihe von Einstellungen getroffen werden, zumindest soll hier die Ingamesteuerung konfigurierbar sein. Später könnten auch zusätzliche Sound- oder Videoeinstellungen hier vorgenommen werden. Die Funktion *Spiel beenden* beendet die Applikation.

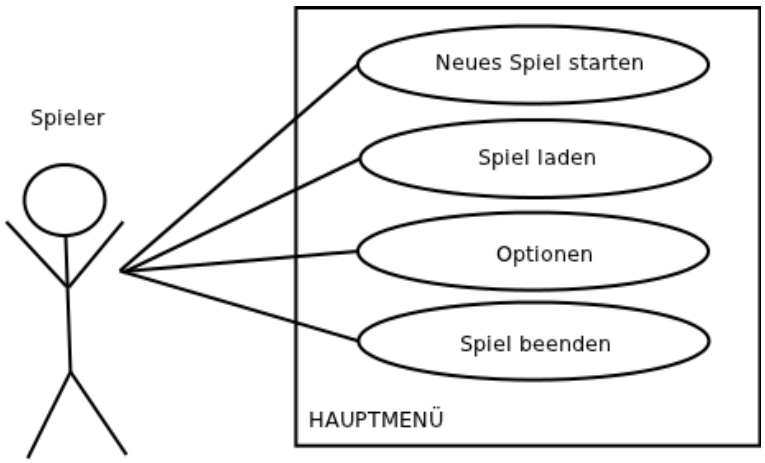

Abbildung 3.1: Übersicht der Hauptmenü-Funktionen

Neben dem Menümodus mit seinen verschiedenen Sichten gibt es den Ingamemodus, in dem das eigentliche Spiel abläuft. Die Funktionen *Spiel laden* und *Neues Spiel starten* wechseln vom Menü- in den Ingamemodus. Startet man ein neues Spiel, muss zunächst ein Campus geladen werden. Hierzu wählt man aus einer zuvor erstellten Liste eine Campusdatei aus.

Die Spielwelt wird anhand dieser Datei automatisch erstellt. Bestätigt der Spieler seine Campuswahl, wechselt das System automatisch in den Ingamemodus und startet auf dem gewählten Campus.

UseCase: Neues Spiel starten

Titel:	Neues Spiel starten
Akteur:	Spieler
Fachlicher Auslöser:	Spieler hat *Neues Spiel starten* gewählt
Vorbedingung:	Das System hat Zugriff auf mindestens eine Campusdatei
Standardablauf:	1. Spieler: Wählt einen Campus aus
	2. Spieler: Bestätigt Campuswahl
	3. System: Liest die Campusdatei ein
Abbruch:	1a. Spieler: Wählt *Zurück*
	2a. System: Wechselt ins Hauptmenü
Nachbedingung:	System geht in Zustand *Ingame* oder Hauptmenü
NF-Anf.:	Ladezeit $<$ 30 Sekunden
Parameter:	Campusdatei
Nutzungshäufigkeit:	bei jedem neuen Spiel

3.2 Ingamemodus

Der Ingamemodus ist der wichtigste Zustand der Applikation. Hier bieten sich dem Spieler alle Möglichkeiten, er kann den Campus erkunden, Gegenstände aufnehmen oder ablegen, mit NPCs reden, neue Räume betreten usw. Im Wesentlichen lassen sich die Funktionen wie in Abb. 3.2 dargestellt zusammenfassen.

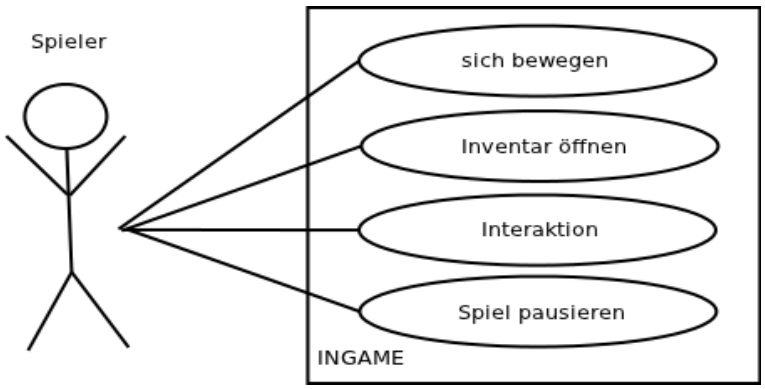

Abbildung 3.2: Übersicht der Ingame-Funktionen

3.2.1 Charakterbewegung

Der Spieler kann sich grundsätzlich über entsprechende Tasten von Feld zu Feld bewegen oder seine Sicht innerhalb eines Feldes steuern. Dabei behält der Spieler bei seitlichen Bewegungen über Felder stets seine Blickrichtung bei (sogenanntes Strafing). Die Blickrichtung kann er ändern, indem er sich auf einem Feld nach Links oder Rechts dreht. Grundsätzlich gibt es nur vier Blickrichtungen auf jedem Feld, die mit Nord, Süd, Ost und West bezeichnet werden. Bei Bewegung nach vorn oder hinten behält der Spieler ebenfalls seine Blickrichtung bei (die Ansicht wechselt also nicht um 180° wenn der Spieler rückwärts gehen möchte).

UseCase: Charakterbewegung

Titel:	Charakterbewegung
Akteur:	Spieler
Fachlicher Auslöser:	Spieler wählt Bewegungsrichtung
Vorbedingungen:	Spieler ist Ingame
Laufen:	1. Spieler: Möchte nach vorn oder hinten laufen
	2. System: Versetzt den Spieler auf das Feld seiner Wahl
	3. System: Wechselt die Ansicht zum neuen Feld
Strafen:	1. Spieler: Möchte nach rechts oder links strafen
	2. System: Versetzt den Spieler auf das Feld seiner Wahl
	3. System: Wechselt die Ansicht zum neuen Feld
Drehen:	1. Spieler: Möchte Blickrichtung nach links oder rechts wechseln
	2. System: Dreht die Ansicht auf dem aktuellen Feld
	um $90\,^\circ$ in die gewählte Richtung
Fehlerfall Bewegen:	2.a System: Feld ist dem Spieler nicht zugänglich (z.B. Wand)
	3.a System: Mitteilung an den Spieler, Ansicht wechselt nicht
Nachbedingung:	Ansicht gewechselt (im Fehlerfall nicht)
NF-Anf.:	Reaktionszeit System $<$ 2 Sekunden
Parameter.:	Feld, Blickrichtung, Ansicht
Nutzungshäufigkeit:	Ingame sehr häufig (ca. 5 von 10 Interaktionen)

3.2.2 Interaktion mit Items

Items, die sich im Besitz des Spielers befinden, können verwendet oder miteinander kombiniert werden. Dazu muss der Spieler ein Item aus dem Inventar in die Hand nehmen und über die Benutzeninteraktion mit einem Gegenstand aus dem Inventar kombinieren oder auf Objekte oder NPCs auf dem Spielfeld anwenden. Häufig wird, wenn ein Item benutzt wird, dieses verbraucht werden und aus dem Inventar verschwinden. Es gibt jedoch auch viele Gegenstände, welche mehrmals verwendet werden können, die nicht nach ihrer Benutzung verschwinden. Um den begrenzten Inventarplatz zu verwalten, können Items auch auf dem aktuellen Feld abgelegt werden. Dazu wird es in die Hand genommen und dann über die Drop-Funktion abgelegt. Das Item ist dann auf dem Feld wieder sichtbar und kann aufgenommen werden. Dazu wählt der Spieler in der Infosicht den gewünschten Gegenstand aus, diesen hat er dann wieder in der Hand und kann entscheiden ob er ihn direkt benutzen, im Inventar ablegen oder wegwerfen möchte.

Szenario: Item finden Der Spieler bewegt sich frei über den Campus. Dabei liegen auf dem Feld vor ihm drei Items. Der Spieler möchte nun das Item aufnehmen. Es öffnet sich eine Liste, in der alle Items aufgelistet werden. Der Spieler browst nun durch die Liste und wählt einen Schlüssel aus. Die anderen Items braucht er nicht. Jedoch bekommt er, bevor er

den Gegenstand aufgenommen hat, eine Meldung das sein Inventar bereits voll ist und der Schlüssel daher nicht aufgenommen werden kann. Aus diesem Grund wirft er das Sandwich aus seinem Inventar. Das Sandwich erscheint nun auf dem Feld und ein Slot im Inventar ist frei geworden. Nun kann er den Schlüssel aufnehmen. Der Schlüssel ist dabei vom Boden verschwunden und stattdessen nun in seinem Inventar.

UseCase: Item finden

Titel:	Item Finden
Akteur:	Spieler
Fachlicher Auslöser:	Spieler schaut sich um
Vorbedingungen:	Spieler ist Ingame auf einem Feld mit Items
Standardablauf:	1. Spieler: Schaut sich um und bekommt alle Items auf dem Feld, in Form einer Liste (in der Infosicht) angezeigt 2. Spieler: Wählt ein Item aus dieser Liste und nimmt dieses in sein Inventar auf 3. System: Übergibt Item von Feld an Spieler
Abbruch:	2.a Spieler: Möchte kein Item aufnehmen und bricht ab 3.a System: Wechselt die Anzeige von Infosicht zu Inventarsicht
Nachbedingung:	Spieler erhält Item
NF-Anf.:	Reaktionszeit System $<$ 2 Sekunden
Parameter:	Inventar / Item
Nutzungshäufigkeit:	Ingame häufig (ca. 3 von 10 Interaktionen)

UseCase: Inventarinteraktion

Titel:	Inventarinteraktion
Akteur:	Spieler
Fachlicher Auslöser:	Spieler möchte einen seiner Gegenstände verwenden
Vorbedingungen:	Im Inventar befinden sich verwendbare Gegenstände
Ablauf:	1. Spieler: Wählt sein Inventar an 2. System: Inventarsicht wird aufgerufen 3. Spieler: Wählt einen Gegenstand 4. System: Wechselt zur Infosicht
Nachbedingung:	Bei leerem Handslot wird dieser mit Item gefüllt und Gegenstand wird aus dem Inventar entfernt, bei besetztem Handslot wird das Item im Handslot ersetzt und wandert in das Inventar
NF-Anf.:	Reaktionszeit System $<$ 2 Sekunden
Parameter:	ausgewählter Gegenstand
Nutzungshäufigkeit:	beliebig oft, sofern sich Gegenstände im Inventar befinden

3.2.3 Interaktion mit NPCs

Befindet sich ein *Non-Player-Character (NPC)* auf dem Feld des Spielers, kann der Spieler über die Interaktionssicht mit diesem in einen Dialog treten. Ein Dialog wird einen kurzen Einleitungstext haben. Im Verlauf des Dialogs kann der NPC dem Spieler Handlungsanweisungen geben, ihm Fragen stellen oder Items übergeben. Dem Spieler werden dabei mehrere Frage- oder Antwortmöglichkeiten zur Verfügung gestellt. Gewisse Fragen oder Antworten treiben den Dialog voran, während andere ihn zum Ende bringen. Bestimmte Dialoge können Items beinhalten. Wird ein solcher Dialog erreicht, vergleicht das System das (Ist)-Item, welches der Spieler dem NPC anbietet mit dem (Soll)-Item, dass der NPC verlangt um eine bestimmte Aufgabe als erfüllt anzuerkennen. Es kann jedoch auch nach dem Erfüllen einer Aufgabe bzw. damit eine Aufgabe erfüllt wird, der Spieler ein bestimmtes Item vom NPC erhalten.

UseCase: Interaktion mit NPC

Titel:	Interaktion mit NPC
Akteur:	Spieler und NPC
Vorbedingungen:	Spieler steht auf gleichem Feld wie der NPC
Ablauf:	1. Spieler: Spieler spricht NPC an 2. System: *Dialog* des NPC wird aufgerufen 3. Spieler: Spieler wählt eine *Erwiderung* aus 4. System: Wird fortgeführt bis Dialog abgeschlossen
Ablauf (mit Item):	1. Spieler: Spieler wendet Gegenstand auf NPC an 2. System: NPC reagiert ggf., je nach Art des Gegenstandes 3. Spieler: Spieler wählt eine Erwiderung aus, sofern möglich 4. System: Wird fortgeführt bis Dialog abgeschlossen oder beendet
Nachbedingung:	Dialog mit NPC wird an aktuellem Stand angehalten
NF-Anf.:	Systemseits 1-2 Sekunden Reaktionszeit
Parameter:	NPC, evtl ausgewählter Gegenstand
Nutzungshäufigkeit:	Ingame häufig (3 von 10 Interaktionen)

3.2.4 Pausieren

Pausiert der Spieler das laufende Spiel, so wechselt er kurzzeitig in den Menümodus mit der Pausesicht, wo er z.B. seinen Spielstand speichern oder einen alten laden kann. Das Spiel selbst bleibt, während er im Pausemenü ist, erhalten und kann jeder Zeit fortgesetzt werden. Über den Pausemodus kann zudem das Spiel abgebrochen werden. Dabei gelangt der Spieler zurück ins Hauptmenü. Das Pausemenü ist gleichzeitig die einzige Möglichkeit für den Spieler, die Applikation aus dem Ingamemodus heraus regulär zu beenden.

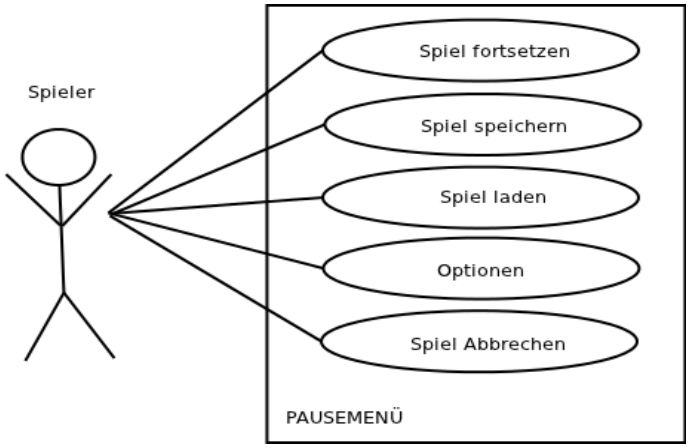

Abbildung 3.3: Übersicht der Pausemenü-Funktionen

UseCase: Spiel pausieren

Titel:	Spiel pausieren
Akteur:	Spieler
Fachlicher Auslöser:	Spieler möchte Spiel unterbrechen
Vorbedingungen:	Spieler ist *Ingame*
Standardablauf:	1. Spieler: wählt Pausenmenü
	2. System: wechselt in den *Menümodus*
	3. Spieler: wählt eine Pausefunktion aus
Abbruch:	1.a Spieler: Spieler wählt *Spiel fortsetzen*
	2.a System: wechselt in den *Ingamemodus*
Nachbedingung:	Spiel pausiert, System im Menümodus
NF-Anf.:	Reaktionszeit System $<$ 1 Sekunde
Nutzungshäufigkeit.:	min. einmal pro Spiel

3.2.5 Speichern eines Spiels

Das Spiel kann ausschließlich gespeichert werden, wenn der Spieler sich im Ingamemodus befindet. Dazu wählt er die entsprechende Taste zur Spielunterbrechung. Darüber gelangt er dann in das Pausemenü. An dieser Stelle gibt es verschieden Auswahloptionen u.a. die das Spiel zu speichern. Geht der Spieler in das Spiel speichern Menü, kann er an dieser Stelle einen vorhandenen Spielstand aus einer Liste überschreiben oder einen neuen anlegen und so sein derzeitiges Spiel speichern. Ihm steht es jedoch frei auch einen Spielstand zu löschen oder ohne zu speichern das Menü zu verlassen.

3.2.6 Laden eines Spiels

Spiele können im CampusAdventure auf zwei Arten geladen werden. Zum einen direkt nach dem Starten der Spieldatei. Der Spieler erreicht dabei zunächst das Hauptmenü, in dem er neben anderen Optionen, die des Spiel ladens hat. Das Spiel laden Menü selbst, ist vom Aufbau her gleich dem Speichermenü. Es kann ein Spielstand aus der Liste gewählt werden und dieser wird entweder gelöscht, geladen oder das Menü wird ohne jegliche Aktion wieder verlassen, was einen Abbruch darstellt. Die zweite Variante ist, dass während einer Spielunterbrechung ein anderer Spielstand geladen werden kann. Hierzu befindet sich der Spieler,auf Grund der Spielunterbrechung, im Pausemenü. In diesem ist dann das Spiel laden Menü integriert und nutzbar.

4 Daten- und Domänenmodell

In diesem Kapitel bemühen wir uns um eine formale Darstellung der Gegenstandswelt der Anwendung. Es werden die Objekthierarchie und die wichtigsten Konzepte, die in der Spielwelt auftauchen, beschrieben.

4.1 Überblick

Im *Spiel* gibt es *GameObjects* und einen Campus (später eventuell mehrere). Der Campus besteht aus Räumen, die wiederum aus Feldern bestehen. Ein Feld kann eine feste Anzahl an GameObjects aufnehmen. Im Spiel tauchen verschiedene GameObjects auf, z.B. Charaktere, *Objekte* oder *Gegenstände*. Sowohl der Spieler als auch NPCs sind Charaktere. Das Spiel ist gewonnen, sobald die Gewinnbedingung erfüllt wurde. Die Gewinnbedingung besteht darin einen bestimmten Schein zu erlangen. Ein Spiel kann neu gestartet, abgespeichert oder geladen werden. Das Ziel des Spiels könnte später auch durch einen Abschluss repräsentiert werden. Der Abschluss würde sich beispielsweise immer aus sechs Semestern zusammensetzen und diese würden wiederum aus mehreren Scheinen bestehen.

4.2 Objekte und Gegenstände

Grundsätzlich gibt es Objekte und Gegenstände. Objekte sind stationäre Gebilde auf einem Feld, die der Spieler nicht mitnehmen kann. Gegenstände dagegen sind portabel, der Spieler kann sie in sein Inventar packen. Auch NPCs können Gegenstände mit sich tragen. Häufig auftauchende Gegenstände sind zum Beispiel Items. Der Spieler soll mit jeder Form von Gegenstand und den meisten Objekte interagieren können. Die Reaktion der Gegenstände auf bestimmte Interaktionen kann intern durch einfaches Nachschauen in einer Tabelle geschehen. Jeder Gegenstand kann dann eine oder mehrere besondere Formen und eine standardmäßige Form von Interaktion haben um alle Kombinationen möglichst einfach abdecken zu können. In folgendem UML-Diagramm wird die Domainstruktur bildlich dargestellt. Die grünen Klassen zeigen mögliche spätere Erweiterungen an. Sie werden hier nur zur Orientierung abgebildet und sind nicht Teil der Spezifikation.

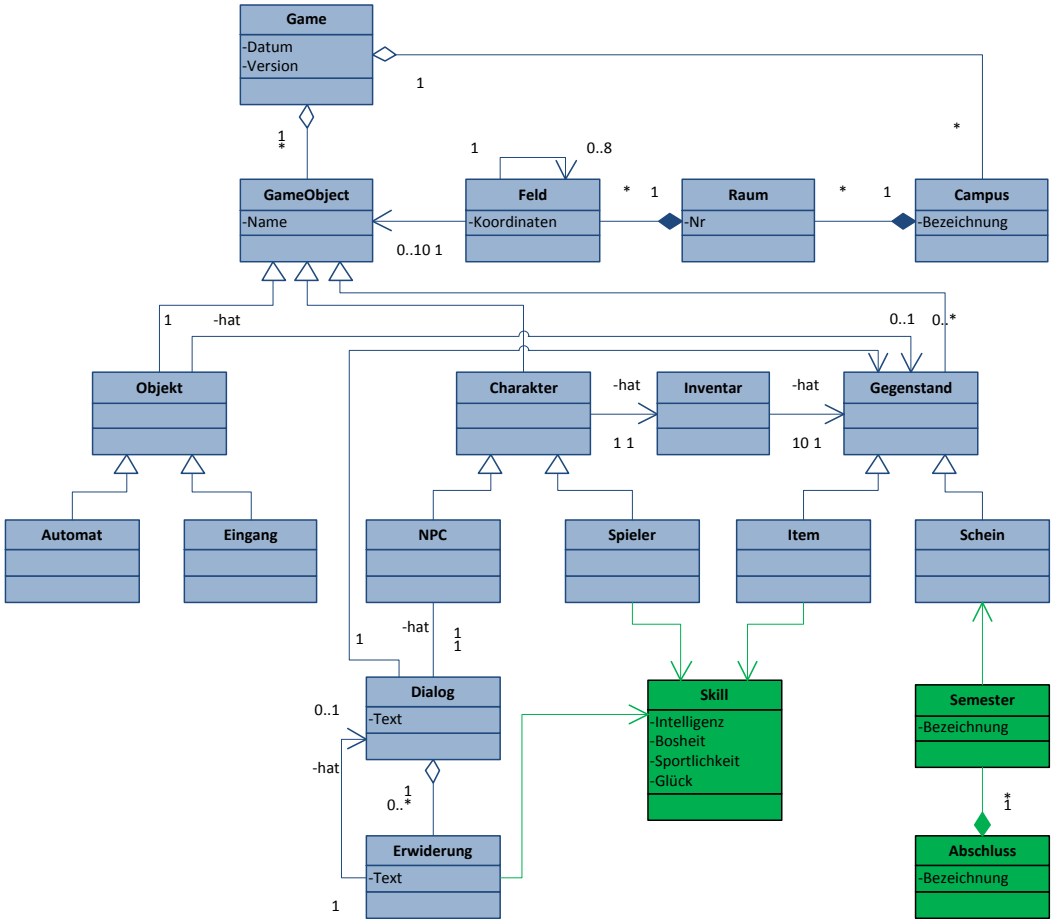

Abbildung 4.1: UML-Diagramm der Domain

4.3 Inventar

Das Inventar bietet dem Spieler die Möglichkeit Items in seinen Besitz zu bringen. Es beinhaltet eine feste Anzahl an Slots. Jeder Slot kann mit nur einem Item belegt werden. Die Anzahl der Items im Besitz des Spielers ist also durch die Kapazität des Inventars (plus eins im Handslot) direkt beschränkt. Der Spieler kann Items aus dem Inventar ablegen (droppen) oder in die Hand nehmen (benutzen). Auch NPCs können ein Inventar haben und Gegenstände mit sich herumtragen. Der Spieler kann Items also nicht nur durch das Aufnehmen von Feldern, sondern auch durch die Interaktion (z.B. den Dialog) mit NPCs erlangen.

4.4 Felder, Räume und Campi

Felder sind die kleinste räumliche (ebene) Einheit im Spiel. Felder haben eine feste Anzahl Slots (eine Quadratzahl, in unserem Spiel neun) und können maximal Slots + 1 viele Elemente aufnehmen (der Spieler muss immer auf ein Feld navigieren können, auch wenn alle Slots belegt sind). Jedes Feld hat kein bis maximal acht angrenzende Felder. Felder haben Koordinaten. Der Übergang zwischen zwei Feldern muss nicht immer möglich sein, so können diese durch eine Wand getrennt sein, was dem Spieler ersichtlich sein sollte. Der Spieler steht immer genau auf einem Feld und kann alle unmittelbar angrenzenden Felder in Blickrichtung sehen (aber nicht notwendigerweise erreichen oder betreten). Auf diesem Feld hat er eine von genau vier Blickrichtungen. Diese werden mit N, S, O, W bezeichnet. Diagonal angrenzende Felder kann er gegebenenfalls sehen, jedoch nicht direkt betreten oder darauf zugreifen. Räume bestehen aus zusammenhängenden Feldern, die einander zugänglich und vom Rest des Campus durch Wände abgetrennt sind. Räume sind mit einander durch Eingänge verbunden. Eingänge können verschlossen oder offen sein. Eingänge können auch mit Items (z.B. Schlüsseln) kombiniert werden. Alle Räume zusammen bilden einen Campus. In der vorgelegten Version spezifizieren wir genau einen Campus pro Spiel.

4.5 NPCs und Dialoge

NPCs sind Nichtspielercharaktere, die auf einem Feld stehen. Der Spieler kann mit ihnen in Form von Dialogen oder durch Items interagieren. Dialoge haben keine oder mehrere Erwiderungen, die der Spieler dem NPC geben kann. Eine Erwiderung kann einen Folgedialog haben. Dialoge werden also in Form eines Baums repräsentiert, in der es einen Einstiegsdialog (Vaterknoten) mit Erwiderungen (den Pfaden) und Folgedialogen (Kindknoten) geben kann. Manche Dialoge können Gegenstände enthalten, der Spieler erhält also den Gegenstand, vorausgesetzt er wählt die richtige Abfolge von Erwiderungen. NPCs sollen stationär sein, in späteren Erweiterungen könnten sich NPCs aber auch auf festen Routen durch die Map von Feld zu Feld bewegen.

4.6 Spieler und Scheine

In der hier vorgelegten Version gibt es nur einen Schein, den es zu finden gilt. Sobald der Schein gefunden wurde, ist das Spiel automatisch beendet. In zukünftigen Erweiterungen könnten Scheine dem Spieler CPs einbringen. Das Finden der Scheine könnte dann an eine Reihe von inhaltlichen Verpflichtungen geknüpft sein (z.B. "finde Item A, damit dir NPC B Schein C überreicht"). Das Ganze könnte dann in Form von Semestern und Abschlüssen weiter ausgebaut werden, wie schon zuvor angedeutet.

5 Benutzungsschnittstellen

Hier findet man die Dialogspezifikation, GUI-Skizzen, die Dialogabläufe und die für das Spiel wichtige Formatspezifikation der Campusdatei.

5.1 Ingame GUI

Ingame gibt es vier sog. *Aktivitätsbereiche*. Im *Sichtbereich* (gelb) wird dem Spieler seine aktuelle Position und die angrenzenden Felder angezeigt. Zudem werden alle Items und NPCs auf dem aktuellen Feld dargestellt. Nicht begehbare Felder sind durch Wände oder verschlossene Türen abgetrennt. Offene Türen stellen betretbare Felder dar.

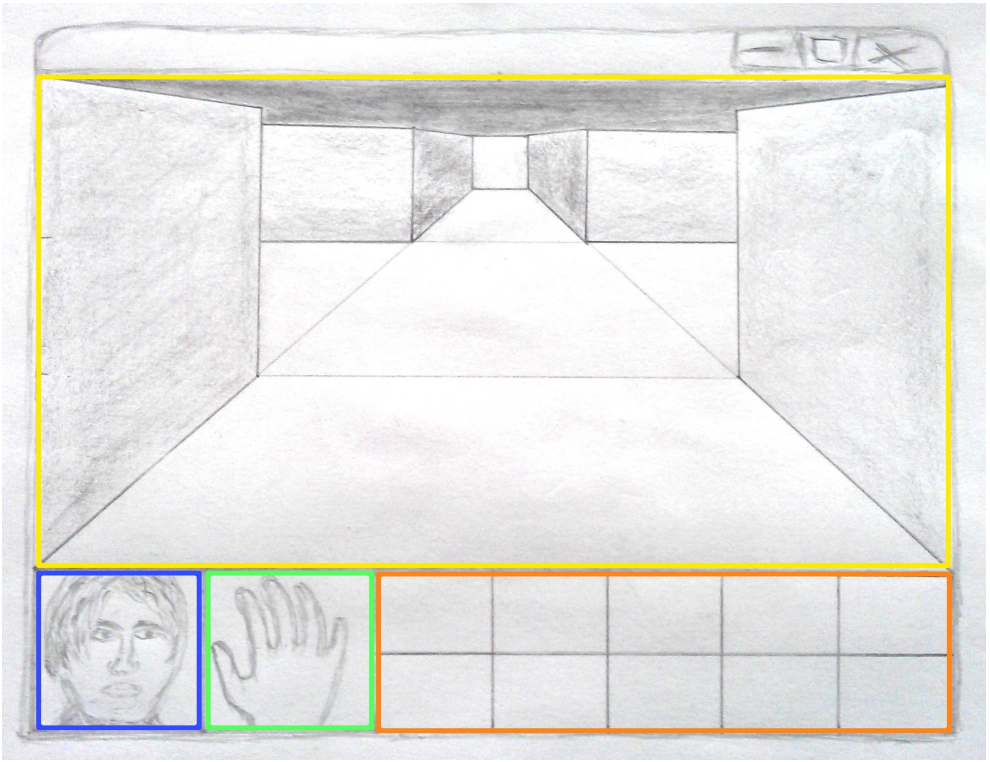

Abbildung 5.1: Die vier Aktivitätsbereiche

Im *Infobereich* (orange) hat der Spieler entweder eine Übersicht der einzelnen Inventarslots (Inventarsicht) oder in Interaktion textuelle Informationen (Infosicht). Der *Handbereich* (grün) zeigt dem Spieler an, ob er ein Objekt in der Hand hält und falls ja, welches. Der *Statusbereich* (blau) zeigt ein Porträt des Charakters.

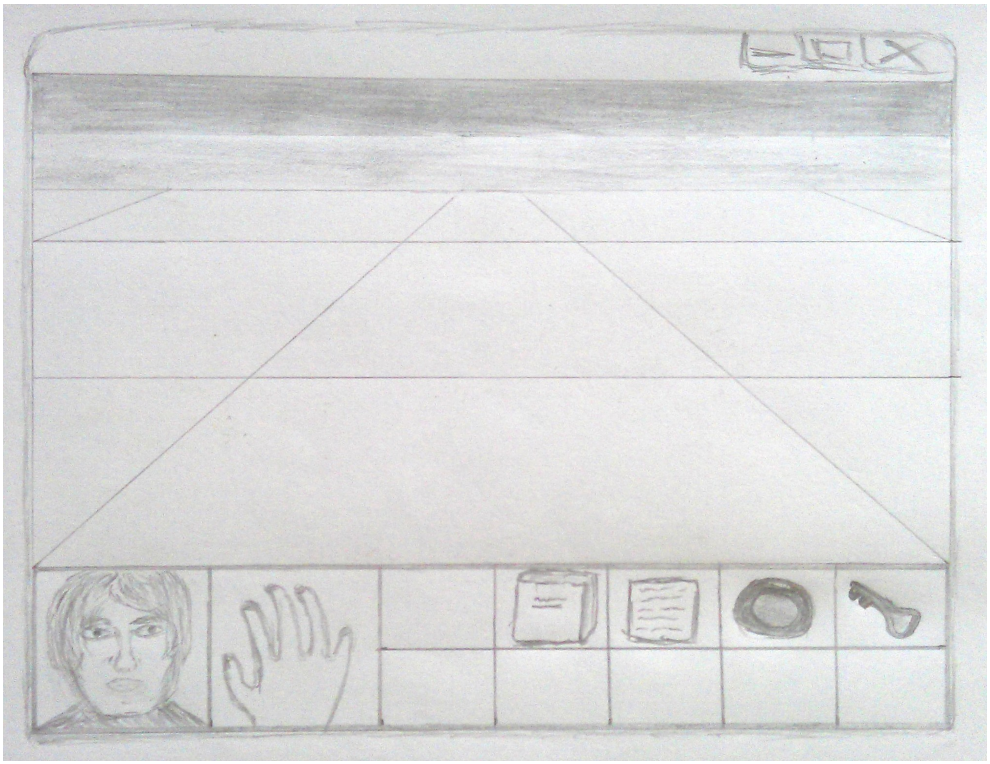

Abbildung 5.2: Ein leerer Sichtbereich

Das Inventar in Abbildung 5.2 enthält einige Items, die der Nutzer bereits aufgesammelt hat. Er hat aber kein Objekt in der Hand, daher die Darstellung der leeren Hand. Da er vor einem freien Bereich steht, ist die Sichtweite am größten. Selbst in diesem Fall kann der Spieler aber nur drei Felder weit sehen (das auf dem er sich momentan befindet eingeschlossen). Dahinter befinden sich eventuell weitere Felder, die aber durch eine Art Nebel verhangen sind. Der Horizont ist also nicht sichtbar. Darüber befindet sich der Himmel bzw. innerhalb von Gebäuden die Decke.

Abbildung 5.3: Interaktion mit einem NPC

Abbildung 5.3 zeigt die Interaktion mit einem NPC. Der Infobereich zeigt jetzt textuelle Informationen zur Interaktion an. Auch Dialoge werden in dieser Form dargestellt. Der Status- und Handbereich bleiben in dieser Sicht erhalten, was praktisch ist um zum Beispiel eine Tür mit einem Schlüssel zu öffnen: Man hat direkt im Blick, ob der Schlüssel schon in der Hand liegt, also verfügbar für diese Interaktion ist. Wäre dies nicht der Fall, müsste dieser erst aus dem Inventar in die Hand genommen werden.

5.2 Menü GUI

Die Menüs stellen die im Abschnitt 3 dargestellten Funktionen zur Verfügung. Wie dort angesprochen startet der Spieler im Hauptmenü. Die Menüs sollen alle im gleichen Stil angelegt werden, daher werden hier nur zwei Menüskizzen gezeigt. Die einzelnen Einträge können wie gesagt aus Abschnitt 3 übernommen werden.

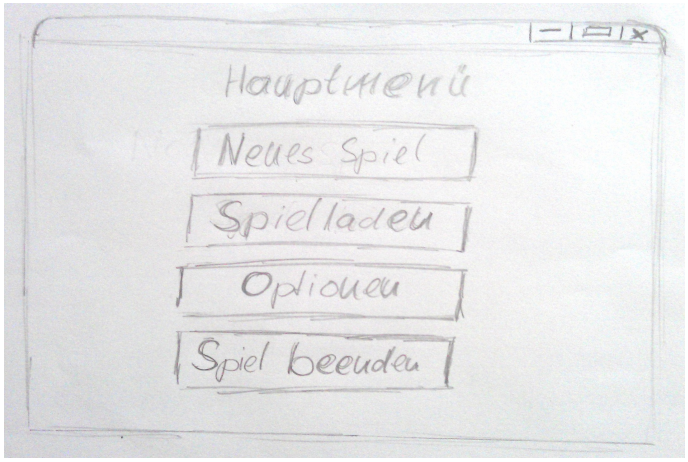

Abbildung 5.4: Das Hauptmenü

Laden- und Speicherscreen wird es eine tabellarische Sicht mit den Savegames geben. Man hat jeweils die Möglichkeit eines auszuwählen und zu laden bzw. ein neues anzulegen oder einzelne Savegames zu löschen.

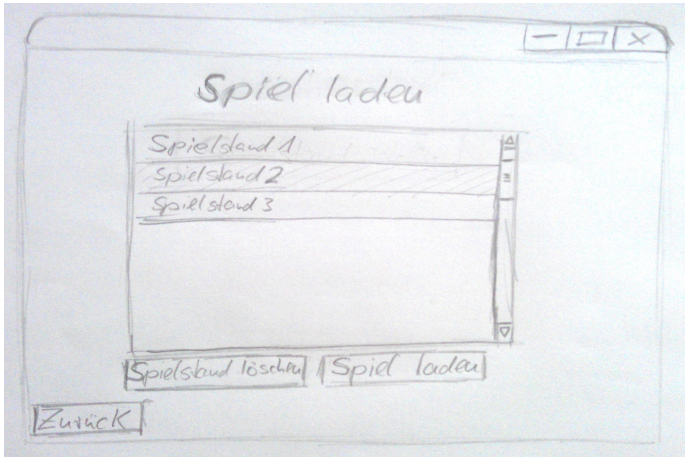

Abbildung 5.5: Das Lademenü

5.3 Campusdatei (v0.1)

Die Campusdatei ist ein grundlegendes Konzept im Spiel, um den Campus konfigurierbar zu machen. Dabei gibt es eine Standardvorgehensweise: Den Campus in Form von ASCII-Zeichen zu codieren und dann die so entstandene Textdatei zu parsen. Da das Erstellen eines neuen Campus von Hand so aber sehr umständlich werden kann (und Probleme mit der Positionierung auftreten könnten), sollte der Campus als Bitmapdatei codiert werden. Letzten Endes ist auch eine Bitmap eine Textdatei in der die Pixel Felder im Spiel darstellen und kann geparsed werden. Entscheidender Vorteil ist die bessere Les- und somit Benutzbarkeit.

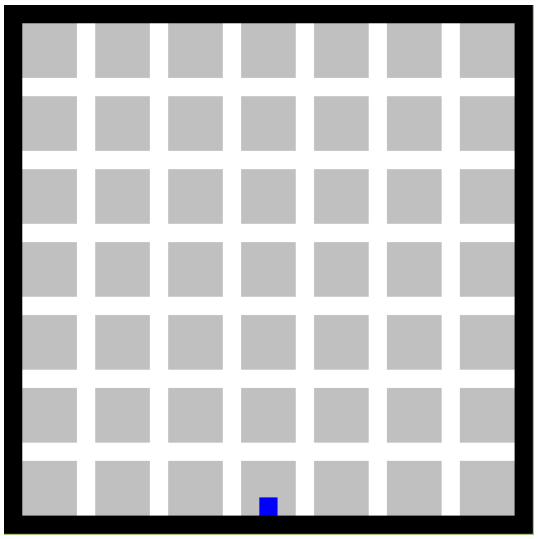

Abbildung 5.6: Ein leerer Campus

Das 29x29 Pixel große Minimalbeispiel einer solchen Datei wurde mit KolourPaint erstellt. Das Format der Datei ist PPM (Portable Pixel Map) in der Version P6 (Spezifikation: `http://netpbm.sourceforge.net/doc/ppm.html`). Die 3x3 großen Pixelfelder in Grau (*0xC0C0C0*) stellen Felder auf dem Campus dar. Die ein Pixel breiten Zwischenräume bieten Platz für Wände (*0x000000*) und Türen (grün *0x00FF00* oder rot *0xFF0000*). Die Starposition des Spielers wird durch einen blaues (*0x0000FF*) Pixel innerhalb eines Feldes verdeutlicht.

Es soll nicht möglich sein, Wände quer über ein Feld zu zeichnen. Allein die Zwischenräume bieten Platz für Wanddefinitionen. Sollen Felder untereinander frei zugänglich sein, so bleibt der Zwischenraum weiss (*0xFFFFFF*). Sollen Felder zwar durch eine Wand getrennt, aber mittels einer Tür verbunden sein, so wird dies durch ein grünes (*0x00FF00*) Pixel im Fall einer offenen, oder ein rotes (*0xFF0000*) Pixel im Fall einer verschlossenen Tür dargestellt. Türen und Wände sind die einzigen Gegenstände die im Zwischenraum positioniert werden dürfen. Türen belegen zusätzlich auf den Feldern die sie verbinden einen Slot (in der gleichen

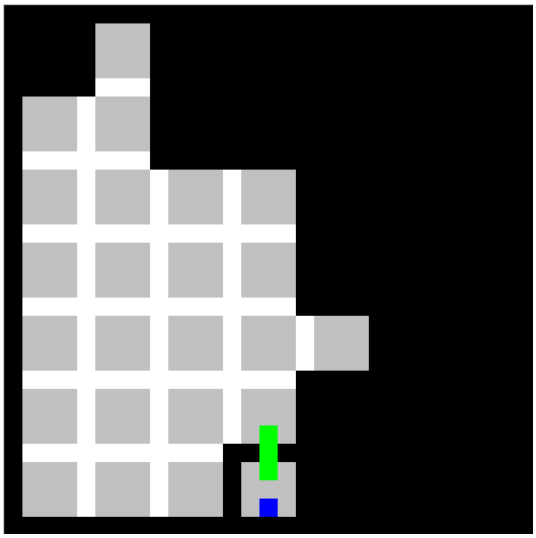

Abbildung 5.7: Zwei Räume mit Tür

Farbe die die jeweilige Tür markiert). Zu beachten ist auch, dass der auf diese Art und Weise gezeichnete Campus immer einen ein Pixel großen Rand hat (*0x000000* oder *0xFFFFFF*) und die Felder nur immer jeweils um drei Pixel gegeinander verschoben werden können, die Feldkanten sind also immer bündig zu einander angelegt. Das gleiche gilt für Wände. Ein Feld ist immer 3x3 Pixel groß, ein Wandstück immer 3x1 Pixel lang. Es können allerdings mehrere dieser Wandteile aneinander gefügt werden, wie in Abbildung 5.7 zu sehen, falls größere Bereiche zum Beispiel nicht begehbar sein sollen.

Die Mapdatei muss eine Versionsnummer haben. Da die Versionsnummer für unser Spiel im PPM Format nicht vorgesehen ist, muss jede Mapdatei als Dateiendung "ctfmap*xx*"haben. Die Nummer *xx* ist die zugehörige Versionsnummer des Spiels (ohne trennenden Punkt) und die Campusdatei ist genau für diese Version zulässig. Je nach Version können weitere Objekte auf den Feldern der Map positioniert werden, Grundlage ist aber immer die Version 0.1. Die angegebenen Pixelgrößen gehören zum definierten Campusformat dazu und müssen immer berücksichtigt werden. Dateien, die nicht genau das angegebene Format haben, können nicht eingelesen werden.

Glossar

Aktivitätsbereich Ingame wird das Fenster in vier dieser Bereiche unterteilt, von denen jeder einen anderen Aspekt des Spiels darstellt. 24

Campus Campus im Bezug auf das CampusAdventure. Synonym und Welt. 8, 11, 20, 22

Charakter Das Alter Ego (seltener: Avatar) des Spielers *Ingame*. 8, 10, 20

Dialog Meist eine Frage oder ein einfacher Satz mit (mehreren) möglichen Antworten, welche auf einen weiteren Dialog verweisen oder das Ende der Unterhaltung markieren. 16

Eingang Ein möglicher Zugang zu einem *Raum*. Eingänge können verschlossen sein. 8

Erwiderung Eine einfache Aussage oder Frage als Reaktion auf die eine NPC. 16

Feld Felder sind räumliche (ebene) Einheiten quadratischer Größe. Felder können sowohl Items als auch NPCs oder it Quests enthalten. Der Spieler bewegt sich von Feld zu Feld. 8, 14, 16, 20

GameObject Elemente im Spiel, z.B. Charaktere, Gegenstände oder Scheine . 20

Gegenstand Ein virtueller Gegenstand, der wenigstens eine Form der Interaktion mit dem Spieler zulässt. 20

Handbereich Einer der vier Aktivitätsbereiche des Spiels Ingame. Hier wird entweder eine leere Hand oder das momentan in der Hand gehaltene Objekt dargestellt. 25

Infobereich Einer der vier Aktivitätsbereiche des Spiels Ingame. Hier wird das Inventar mit seinen Slots oder textuelle Information bei Interaktionen dargestellt. 25

Ingame Bezeichnet den Zustand des Spiels, in dem gespielt werden kann. Übergeordnet auch Ingamemodus genannt. 12, 15

Inventar Eine Art virtueller Rucksack mit beschränkter Kapazität, in dem z.B. *Items* aufgenommen werden können. 8, 14, 15

Item Ein Gegenstand, der vom Spieler in sein Inventar aufgenommen werden kann. 8, 9, 14, 16, 20, 22–24

Menümodus Der Zustand der Applikation in dem nicht gespielt werden kann. Man kann aber in den Ingamemodus wechseln. 9

NPC Non-Player-Character. 9, 16

NPCs Non-Player-Characters. 8, 20, 24

Objekt Elemente im Spiel, mit denen der Spieler interagieren kann, die er jedoch nicht in sein Inventar aufnehmen kann. 20

Raum Ein abgeschlossenes Areal innerhalb der Map. Räume müssen dem Spieler nicht zugänglich sein. Räume bestehen aus *Feldern*. 8, 20

Savegame Bezeichnet eine Datei, in der ein laufendes Spiel gesichert wird, um es später fortzusetzen. 8, 10, 27

Schein Scheine müssen vom Spieler erlangt werden, um das *Spielziel* zu erreichen. Scheine sind atomar und bringen dem Spieler *CreditPoints (CPs)*. 8, 10, 20, 23

Sichtbereich Einer der vier Aktivitätsbereiche des Spiels Ingame. Hier wird die virtuelle Szene in einer Pseudo-3D-Perspektive dargestellt. 24

Spiel Die Interaktion mit der Anwendung nach dem Start eines neuen Spiels und dem Erreichen des Spielziels. 20

Spieler Der Spieler in Form der Spielfigur, welche vom Anwender kontrolliert wird. 8, 10–18, 20, 22–24

Spielwelt Die Gesamtheit der virtuellen Umgebung bestehend aus dem Campus, Räumen, Items, NPCs, Quests und Spieler. 8, 10, 12, 20

Statusbereich Einer der vier Aktivitätsbereiche des Spiels Ingame. Hier wird ein Portät des Charakters dargestellt. 25