

1001011  
1101111  
1110010  
1100001  
1101101  
1110101

---

## Entwurfsspezifikationen

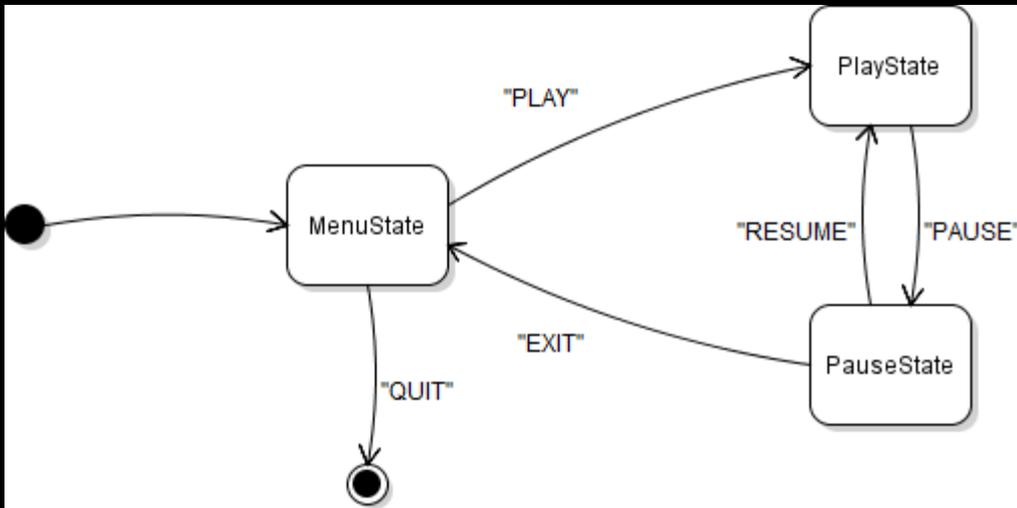
### Inhaltsverzeichnis:

- 1 Wichtige übergreifende Konzepte
- 2 Klassenübersicht
  - 2.1 Klassendiagramme
  - 2.2 Erläuterung der Funktionen einzelner Klassen
- 3 Zeitplan
  - 3.1 Bereichszuordnung der Teammitglieder
  - 3.2 Integrationsplan

## 1 Wichtige übergreifende Konzepte

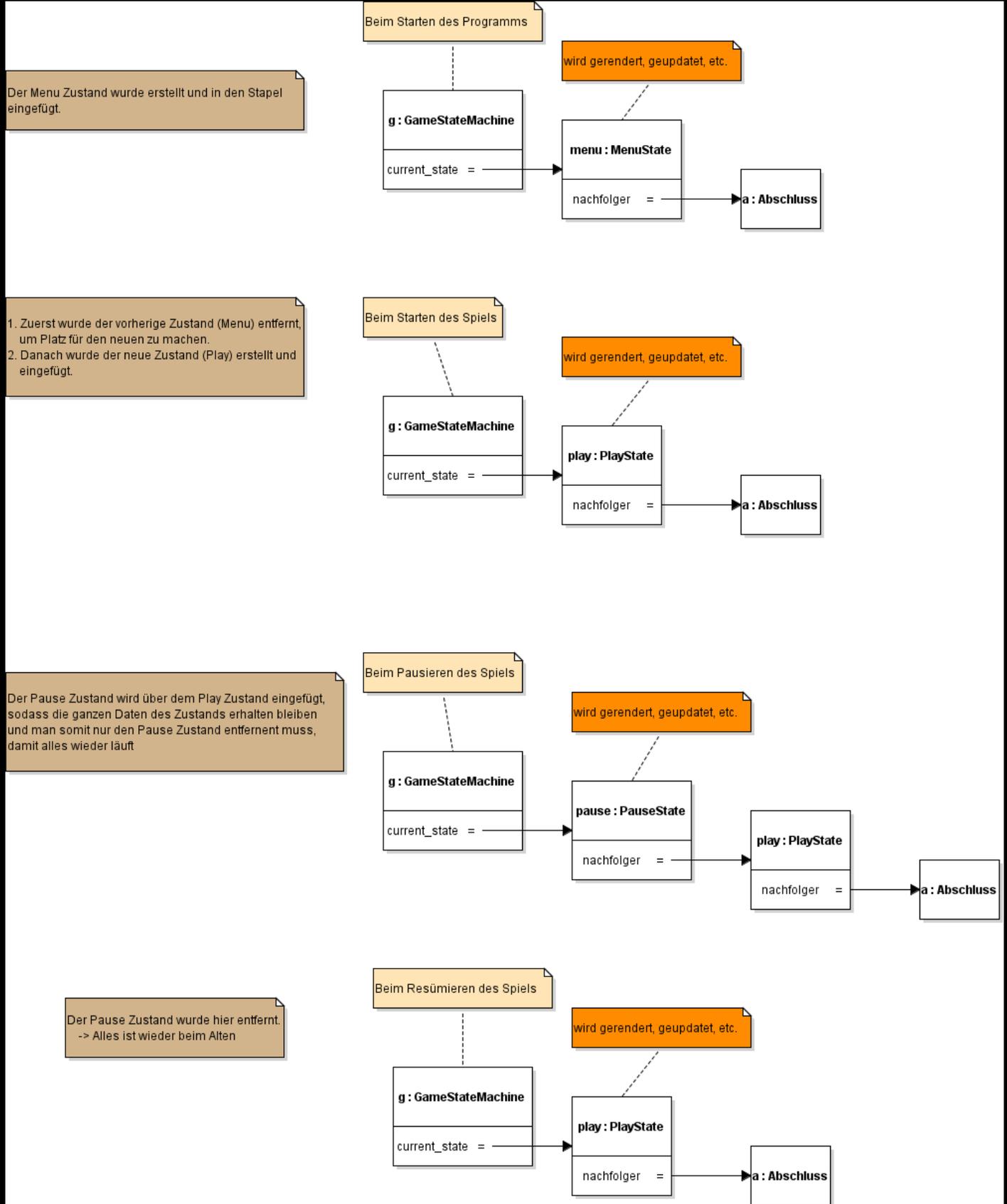
Wir implementieren unser Spiel auf Basis einer FSM (Finite State Machine). Das heißt, dass unser Spiel zwischen verschiedenen Zuständen (z.B. Pause, Menü, Play) unterscheiden kann und in der Folge auch die zustandsspezifischen Operationen durchführen wird.

Veranschaulichendes Zustandsdiagramm:



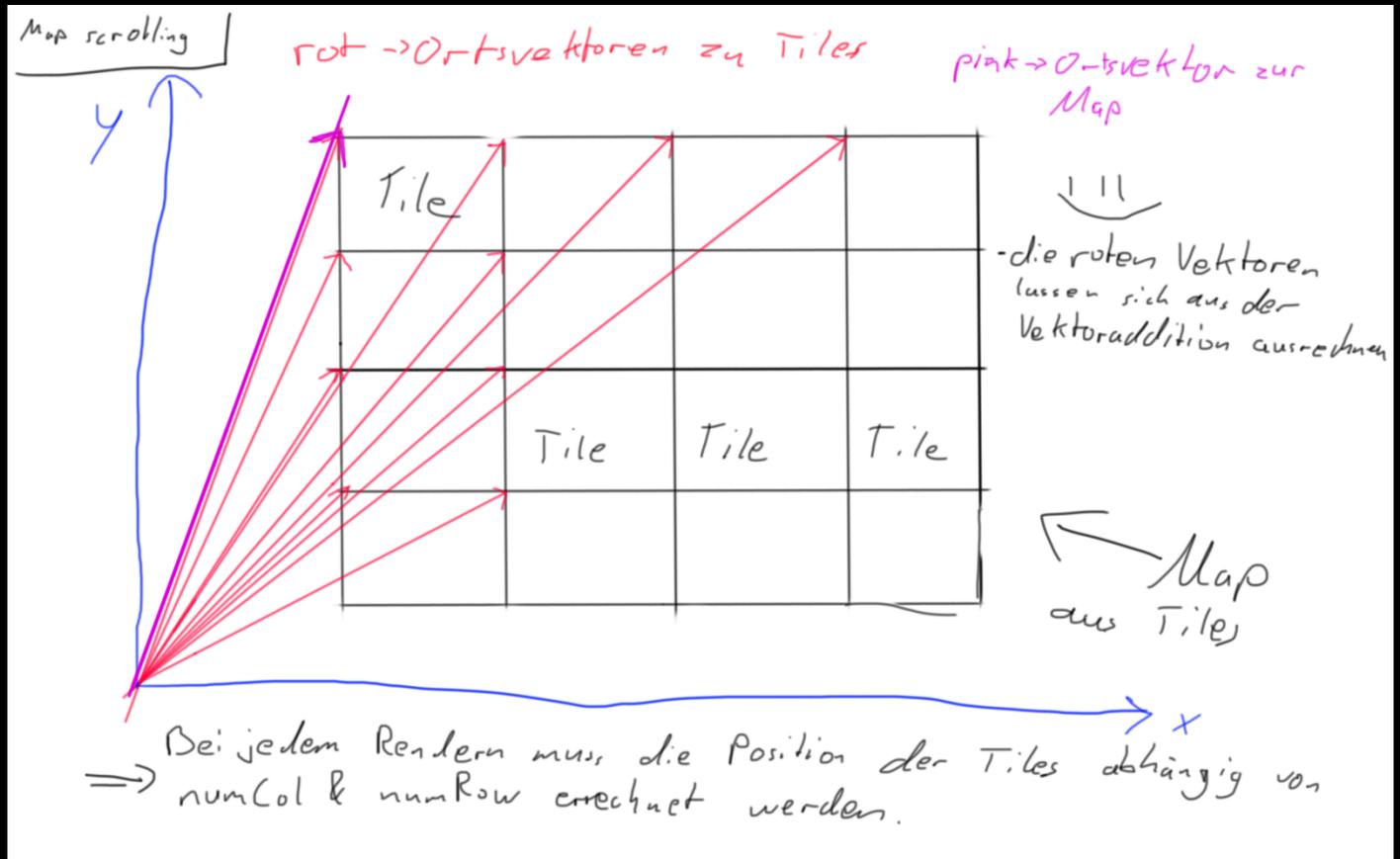
Die State Machine wird als Stapel implementiert. Der Einfachheit halber wurde auf die Trennung von Struktur und Inhalt verzichtet.

# Veranschaulichung der, als Stapel implementierten, State Machine



Unser Produkt ist ein 2D-Spiel. Zur Umsetzung der Map verwenden wir das Konzept einer Tile Map (in 64x64 große Bilder unterteilte Map). Auf das Verwenden einer Game Engine wurde gezielt verzichtet.

Das Rendern der Spielumgebung geschieht nach folgendem Schema:



## 2 Klassenübersicht

### 2.1 Klassendiagramme

Siehe „Klassendiagramm.png“.

### 2.2 Erläuterung der Funktionen einzelner Klassen

#### **Game:**

Eine zentrale Klasse, die z.B. dafür zuständig ist, dass das Spiel initialisiert wird. Sobald die Instanz dieser Klasse, die aufgrund von softwaretechnischen Vorkehrungen nur einmal existieren kann (Singleton), gelöscht wird, ist das Spiel beendet.

Außerdem stellt sie die Schnittstelle zwischen grundlegenden Klassen dar (z.B. GameStateMachine).

#### **TextureManager:**

Diese Klasse soll uns helfen mit Texturen umzugehen. Jede Textur, die im Laufe des Spiels verwendet werden soll wird in dem „TextureManager“ gespeichert und bei Bedarf mit einer entsprechenden Methode gerendert.

#### **GameObject:**

Diese Klasse beschreibt jedes Objekt des Spiels, mit dem der Nutzer interagieren kann. Aber auch Animationen gehören dazu.

Jedes der besagten Objekte erbt von dieser Klasse, um die Vorzüge der Polymorphie nutzen zu können.

#### **InputHandler:**

Diese Klasse existiert um sämtliche Eingaben vom User (Tastatur, Maus, etc.) entgegenzunehmen, damit die jeweiligen anderen Klassen etwas damit anfangen können.

#### **GameStateMachine:**

Diese Klasse verwaltet die verschiedenen Spielzustände (Play, Pause, Menu, etc.) mithilfe eines Stapels. Wir wollen nämlich die Informationen eines alten Spielzustandes nicht verlieren, nur, weil wir uns kurzzeitig in einem anderen befinden.

#### **Environment (Namensraum):**

Die einzelnen Spielzustände verwalten „Map“s (Stapel aus „Map“s).

Eine „Map“ wiederum ist aus mehreren „Layer“n (Ebenen) aufgebaut.

Die „Layer“ werden in einer spezifischen Reihenfolge gerendert, damit alle Texturen richtig übereinander gelagert werden (z.B. Die Spielfigur über den Hintergrund).

Mit den verschiedenen „TileLayer“n (Backgroundlayer, Collisionlayer,

Bridgelayer, Interactionlayer, Foregroundlayer) wird unsere Tile Map umgesetzt.

Die einzelnen „TileLayer“ verwalten „Tile“s (64x64px große Texturen), aus welchen sich unsere Spielumgebung zusammensetzt.

## 3 Zeitplan

### 3.1 Bereichszuordnung der Teammitglieder

Wir arbeiten nach „Kanban“, einem Vorgehensmodell zur agilen Softwareentwicklung, bei dem die Projektteilnehmer keinem festen Aufgabenbereich zugeordnet sind, sondern Aufgaben (Tickets) aus jedem Bereich erledigen. Wir managen das in einem Online-Ticketing-System. Jeder Teilnehmer weist sich Tickets aus der „TODO“-Spalte zu. Sie werden dann parallel bearbeitet („Work in progress“).

### 3.2 Integrationsplan

Wir richten uns bei der Entwicklung nach folgenden Meilensteinen:

7 Open ✓ 1 Closed		Sort ▾
<h4>Klassen für allgemeine Spielkonzepte</h4> <p>📅 Due by April 9, 2017 ⌚ Last updated 1 minute ago</p>	<div><div style="width: 86%;"></div></div> <p>86% complete 2 open 13 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	
<h4>Map fertiggestellt</h4> <p>📅 Due by May 9, 2017 ⌚ Last updated 2 minutes ago</p>	<div><div style="width: 0%;"></div></div> <p>0% complete 4 open 0 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	
<h4>Story</h4> <p>📅 Due by June 30, 2017 ⌚ Last updated 4 months ago</p> <p>Die Möglichkeit eine Story zu haben sollte zu diesem Zeitpunkt vorh...(more)</p>	<div><div style="width: 0%;"></div></div> <p>0% complete 0 open 0 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	
<h4>Das Spiel ist spielbar</h4> <p>📅 Due by September 30, 2017 ⌚ Last updated 4 months ago</p> <p>Nun sollten wir so weit sein, dass die Story komplett implementiert wurde und das Spiel in dem Zustand spielbar ist.</p>	<div><div style="width: 0%;"></div></div> <p>0% complete 0 open 0 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	
<h4>Feinheiten</h4> <p>📅 Due by December 31, 2017 ⌚ Last updated about 1 month ago</p> <p>Spielmusik einbauen. Nun sollen (viele) aufkommende Fehler (Bugs) b...(more)</p>	<div><div style="width: 100%;"></div></div> <p>100% complete 0 open 1 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	
<h4>Fertigstellung</h4> <p>📅 Due by January 8, 2018 ⌚ Last updated 4 months ago</p> <p>Abgabe der Arbeit</p>	<div><div style="width: 0%;"></div></div> <p>0% complete 0 open 0 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	
<h4>Vorstellung</h4> <p>📅 Due by February 26, 2019 ⌚ Last updated 4 months ago</p> <p>Das Spiel muss dem P-Seminar vorgestellt werden.</p>	<div><div style="width: 0%;"></div></div> <p>0% complete 0 open 0 closed</p> <p><a href="#">Edit</a> <a href="#">Close</a> <a href="#">Delete</a></p>	

Anmerkung: Der Meilenstein „Feinheiten“ ist noch nicht fertiggestellt.