# Zellomat3D

## Anforderungsspezifikation

| | | |
|---|---|---|
| **Projekt:** | 3D Cellular Automata Simulator – Diplomarbeit – SS/2005 | |
| **Auftraggeber:** | Hochschule Rapperswil HSR | |
| **Betreuer:** | Eduard Glatz – Prof. Dipl. Ing. ETH | eglatz@hsr.ch |
| **Mitarbeiter:** | Michael Florin | loop@loop.li |
| | Andreas Weinmann | a.weinmann@gmx.ch |
| **Ablage**: | AnforderungsSpezifikation - 06042005.doc | |

# Inhaltsverzeichnis

# Einführung

**Zweck**

Die Softwareanforderungsspezifikation gibt Auskunft über funktionale und nichtfunktionale Anforderungen des Projektes "Zellomat3D".

**Gültigkeitsbereich**

Dieses Dokument gilt für die Diplomarbeit "Zellomat3D", welche im SS/2005 an der Hochschule Rapperswil HSR durchgeführt wurde.

**Übersicht**

Beginnend mit einer allgemeinen Beschreibung der zu entwickelnden Applikation, beschreibt dieses Dokument die Benutzergruppen, die zu erwartenden Probleme und verschiedene Annahmen und Abhängigkeiten. Es folgen detaillierte Beschreibungen der einzelnen Anforderungen sowie Randbedingungen für den Entwurf.
Ausserdem beinhaltet dieses Dokument auch die Anforderungen, die an die Prototypen gestellt werden.

# Allgemeine Beschreibung

**Ziel und Zweck des Zellomat3D**

Der Zellomat3D beinhaltet verschiedenste Funktionen, um die Berechnung und die visuelle Darstellung von verschiedenen zellulären Automaten zu unterstützen. Er ermöglicht es, auf einfache Art und Weise selbst erstellte Automaten auf ihre Funktionsweise und Auswirkungen hin zu untersuchen. Durch eine hohe Verarbeitungsgeschwindigkeit und eine ansprechende 3D-Visualisierung der einzelnen Entwicklungszyklen des Automaten können schnell und einfach die Wechselwirkungen der zu Grunde liegenden Regeln beobachtet werden.

**Motivation**

Die Theorie der Cellular Automata befasst sich seit längerem mit selbstwachsenden Organismen auf einer computertechnischen bzw. mathematischen Basis. Praktische Anwendungen sind bei adaptiven und selbstoptimierenden Systemen zu finden, die im Autonomic Computing eine grosse Rolle spielen. Jedoch nur schon die Visualisierung selbstwachsender Systeme ist von Interesse.

**Produkte Umfeld**

Es sind viele verschiedene Implementationen von zweidimensionalen zellulären Automaten im Internet zu finden. Implementationen von dreidimensionalen Varianten existieren jedoch nur sehr wenige. Diese 3D-Automaten sind darüber hinaus ausnahmslos nur auf einfachste Regeln beschränkt. Das heisst, sie haben nur zwei Zustände. Eine Zelle des ZA existiert oder existiert nicht. Die Regeln, wann eine Zelle „geboren“ wird und wann sie „stirbt“, sind bei diesen Programmen bis zu einem gewissen Mass frei einstellbar. Die Anzahl der zu berechnenden Elemente ist auch stark beschränkt. Die Berechnungszeit aller gefundenen Implementationen für einen Zyklus überschreitet mit zunehmenden Lebensraumgrössen schnell den Geduldsfaden des Benutzers.

Der Zellomat3D wird sowohl schneller sein als auch komplexere Regeln anwenden können.

**Zielgruppe**

Die Zielgruppe für die Applikation Zellomat3D besteht aus Personen mit einer höheren Ausbildung, besonders einem allgemeinen Interesse an der zellulären Automatentheorie.

**Allgemeine Einschränkungen**

Die grösste, aber einzige Einschränkung ist die Leistung der Hardware. Denn auch mit der bestmöglichen Optimierung sind die Berechnung der Zyklen sowie deren Darstellung, extrem rechenintensiv. Ausserdem benötigt beides sehr viel Arbeitspeicher.

# Anforderungen an evolutionäre Prototypen

**Rohdaten Berechnung**

Die Rohdatenberechnung muss in jedem Zyklus für alle Zellen den neuen Zustand aus den aktuellen Zuständen der Nachbarn bestimmen. Die Regeln für diese Bestimmung werden durch den verwendeten ZA gegeben. Die Rohdatenberechnung enthält auch die Datenhaltung aller Zellen.

**Berechnungszeit**

Die Berechnungszeit für einen Zyklus darf eine Zeitspanne von zehn Sekunden nicht überschreiten. Es muss ein effizienter Algorithmus für dieses Problem gefunden werden.

**Anzahl Elemente**

Die Lebensraumgrösse des ZA kann eingestellt werden. Es stehen folgende Seitenlängen des Berechnungsraumes zur Auswahl:

**8, 16, 32, 64, 128**

**Speicherbedarf**

Der Speicherbedarf der Applikation darf nicht zu gross sein. Mehr als 512MB Arbeitsspeicher dürfen nicht überschritten werden.

**Aufbau geschwindigkeit**

Die Framerate der Darstellung muss mehr als 24fps[1] betragen. Dies ist nötig, um dem Betrachter den Eindruck einer flüssigen Bewegung zu geben, wobei sich die flüssige Bewegung nur auf die Kamerasteuerung auswirken muss und nicht auch auf die Updaterate der Zyklen eines ZA's.
Konkret bedeutet das, dass man um die visuelle Darstellung eines Zyklus eines ZA's herumfliegen kann, auch wenn das Update des nachfolgenden Zyklus erst ein paar Sekunden später erfolgt.

[1] fps: Frames per Second, Bilder pro Sekunde, mehr als 24 um dem Auge einen flüssigen Ablauf vorzugaukeln

# Nichtfunktionale Anforderungen der Applikation

**Grundlegende Prinzipien**

Das System darf keine lizenzpflichtigen Librarys oder Codeteile enthalten. Verwendete Opensource-Elemente müssen klar deklariert und in der Codestatistik angegeben werden.

**Programmier-sprache**

Wegen der einfacheren Ansteuerung der Grafikkarte und der viel schnelleren Entwicklungszeiten wird die Applikation in C# geschrieben. Ausserdem unterstützt C# das neue Managed DirectX 9.0 umfassend und somit auch die neusten Grafikkartenfunktionen.

**Datenhaltung**

Die ganze Datenhaltung wird vollumfänglich im Arbeitsspeicher des Computers gehalten. Es werden ausschliesslich die Konfigurationsdateien auf der Festplatte gespeichert.

**Speicherbedarf**

Der Speicherbedarf der Applikation darf nicht zu gross werden. Mehr als 512MB Arbeitsspeicher dürfen nicht überschritten werden.

**Framebuffer**

Die Speichergrösse der Grafikkarte spielt keine Rolle, ausser für die Anzahl gebufferter Frames. Der Buffer ist für eine Speichergrösse der Grafikkarte von 64MB ausgelegt.

**Portierung**

Der Zellomat3D wird für das Windows Betriebssystem 2000 und XP kompiliert.

**Sicherheit**

Da der Zellomat3D keine kritischen oder persönlichen Daten enthält, werden keine Sicherheitsanforderungen gestellt.

**Minimale Hardware Anforderungen**

Für den Betrieb des Zellomat3D werden folgende minimale Hardwareanforderungen benötigt:

PC-System mit:

- einem Pentium 4 2.2GHz oder einem AMD Athlon XP 2000+ Prozessor
- einer NVIDIA GeForce4 oder ATI Radeon 9500 Grafikkarte mit 64MB RAM (128MB empfohlen)
- 512MB Arbeitsspeicher

**Installation**

Wichtigstes Kriterium für die Inbetriebnahme bzw. die Installation ist die einfache und intuitive Handhabung. Es wird ein Wizard gesteuertes Setup-Programm für die Installation der Applikation zur Verfügung gestellt.

**Stabilität**

Unter keinen Umständen darf die Applikation das Betriebssystem zum Absturz bringen.
Bei Unterschreitung der minimalen Hardwareanforderungen kann es zu Fehlfunktionen kommen.
Bei Einhaltung der Hardwareanforderungen darf die Visualisierung in weniger als 2% der Anwendungsfälle Fehlverhalten aufweisen.
Die Berechnung der ZA darf keine Fehlkalkulationen beinhalten.

**Bedienung**

Die Software soll durch eine einfache, intuitive Benutzeroberfläche bedient werden können. Technisches Verständnis sollte für die Bedienung der Applikation auf keinen Fall eine Voraussetzung sein, ausser um eigene Automaten zu entwickeln. In diesem Falle sollte der Benutzer über Kenntnisse der zellulären Automatentheorie.

# Funktionale Anforderungen an Applikation

« « Realisierung zwingend
« Realisierung wünschenswert

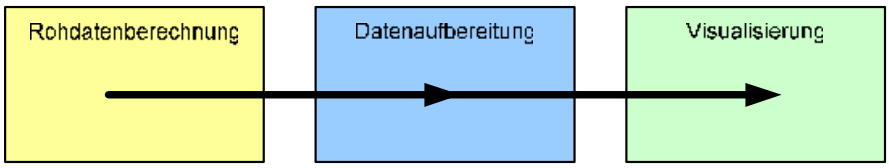
**Prioritäten**

**Programmmodule**

Der Zellomat3D besteht aus drei Hauptfunktionen. Die erste Funktion ist die Rohdatenberechnung. Diese ist für die Berechnung der einzelnen Zyklen eines zellulären Automaten zuständig.
Die zweite Funktion ist die Datenaufbereitung in ein grafisches Format, das für die nachfolgende Visualisierung gebraucht wird.
Die dritte Funktion ist die dreidimensionale Visualisierung der berechneten Zyklen.

1. **Modul: Rohdatenberechnung** « «
2. **Modul: Datenaufbereitung** « «
3. **Modul: Visualisierung** « «

**Ablauf**




Die Applikation erzeugt die Rohdaten, bereitet die Daten auf ihre Visualisierung vor und zeigt diese schliesslich auf dem Monitor an. Ausser beim Initialisieren finden während des gesamten Programmablaufes keine Festplattenzugriffe statt. Die berechneten Frames gehen nach ihrer Visualisierung verloren und müssen bei einem Neustart des ZA wieder von Neuem berechnet und aufbereitet werden.

**Hohe Framerate**

Die Framerate der Darstellung muss mehr als 24fps[2] betragen. « «
Dies ist nötig, um dem Betrachter den Eindruck einer flüssigen Bewegung zu geben, wobei sich die flüssige Bewegung nur auf die Kamerasteuerung auswirkten muss und nicht auch auf die Updaterate der Zyklen eines ZA's.
Konkret bedeutet das,dass man um die visuelle Darstellung eines Zykluses eines ZA's herumfliegen kann, auch wenn das Update des nachfolgenden Zyklus erst ein paar Sekunden später erfolgt.

**Berechnungszeit**

Die Berechnungszeit für einen Zyklus darf eine Zeitspanne von « über zehn Sekunden nicht überschreiten.

[2] fps: Frames per Second, Bilder pro Sekunde, mehr als 24, um dem Auge einen flüssigern Ablauf vorzugaukeln.

**Kamerasteuerung**

Zusätzlich sollte man sich im dreidimensionalen Raum frei bewegen können. So kann der Blickwinkel selbst bestimmt werden und man erhält den Eindruck, man sei der Beobachter eines Minikosmos der zellulären Automaten. Die Steuerung besteht aus einer Kombination von Tastatureingaben und Mausbewegungen.

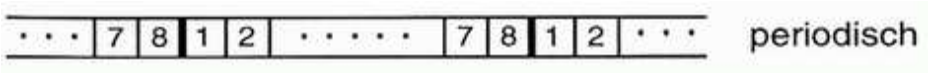
« «

**Automaten**

Es müssen verschiedene Automaten entwickelt werden, bei denen die Auswirkungen der dazugehörigen Regeln klar ersichtlich sind. Diese sind innerhalb der Applikation als „Presets“ auswählbar.

« «

**Randbedingungen**




« «

Der Lebensraum des Automaten hat keine Begrenzungen. Das bedeutet, dass er über zyklische (periodisch) Randbedingungen verfügt. Vorstellen kann man dies das als 3-dimensionales Pacman Spielfeld. („Oben raus und unten wieder rein.“)

**Orientierung**

Alle Präsentationen der Zyklen der ZA werden in eine passende Umgebung eingebettet. Damit wird dem Benutzer die Möglichkeit geboten, sich in den drei Achsen des Koordinatensystems zu orientieren.

«

**Parametrisierung**

Es muss sichergestellt werden, dass die minimalen Hardware-bedingten Grenzen nicht aufgrund falscher Parametrisierung der zellulären Automaten überschritten werden können. Daher muss eine falsche Parametrisierung des ZA, die zu einem Fehlverhalten der Applikation führt, verunmöglicht werden. Dies geschieht durch Eingabeüberprüfungen der einzelnen Parameter des ZA.

« «

**Erweiterbarkeit der ZA**

Es wird die Möglichkeit geboten, ohne Programmierkenntnisse eigene Automaten zu entwickeln und dem Programm hinzuzufügen. Mittels einer XML-Konfigurationsdatei können selbsterstellte Automaten in das Programm eingebunden werden.

«

**Erweiterbarkeit des Programmes**

Es muss darauf geachtet werden, für spätere Weiterentwicklungen die Grundsteine so zu legen, dass darauf auch weiter aufgebaut werden kann. Die ganze Applikation wird aus einzelnen Modulen mit klar definierten Schnittstellen entwickelt. Dies ermöglicht einen Austausch einzelner Module (ohne Auswirkungen auf die anderen).

« «

# Use Cases

**UC1: Öffnen eines ZA**

Beschreibung: Der User lädt einen zellulären Automaten in die Applikation
Akteur: User
Vorbedingung: Applikation gestartet

| User Intentions | System Responsibility |
|---|---|
| **any time {cancel}** | **any time {report error}** |
| | |
| 1. startet Auswahl ZA | |
| | 2. öffnet Dateibrowser |
| 3. wählt ZA aus | |
| 4. bestätigt Auswahl | |
| | 5. lädt Parameter des ZA |
| | 6. überprüft Parameter des ZA |
| | 7. initialisiert interne Strukturen |

**UC2: Reset ZA**

Beschreibung: Der User oder die Applikation kann einen zellulären Automaten wieder auf seinen Initialzustand setzen.
Akteur: User / Applikation
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. löst Reset aus | 1. löst Reset aus |
| | 2. UC5: Pause |
| | 3. initialisiert ZA neu |

**UC3: Play ZA**

Beschreibung: Der User kann die einzelnen Zyklen automatisch nacheinander abspielen lassen.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. löst Play aus | |
| | 2. zeigt ZA-Zyklen nacheinander in eingestelltem Intervall an |

**UC4: Step ZA**

Beschreibung: Der User kann die einzelnen Zyklen von Hand nacheinander anzeigen lassen.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. löst Step aus | |
| | 2. UC5: Pause |
| | 3. zeigt nächsten Zyklus an |

**UC5: Pause ZA**

Beschreibung: Der User oder die Applikation die Anzeige der Zyklen unterbrechen.
Akteur: User / Applikation
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. löst Pause aus | 1. Löst Pause aus |
| | 2. stoppt Anzeige der anstehenden Zyklen |

**UC6: Anzeigeintervall einstellen**

Beschreibung: Der User stellt die Zeit ein, die zwischen zwei Anzeigen von Zyklen des ZA vergehen soll.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. stellt Interval ein | |
| | 2. setzt Interval |

**UC7: Schnappschuss eines Zyklus**

Beschreibung: Der User schiesst einen Schnappschuss vom aktuell angezeigten Zyklus.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. wählt Schnappschuss | |
| | 3. speichert das aktuelle Bild |

**UC8: Informationen über das Programm anzeigen**

Beschreibung: Der User kann sich Informationen über das Programm anzeigen lassen.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| | |
| 1. wählt Informationen | |
| | 2. UC5: Pause |
| | 3. zeigt Informationen |
| 4. User schliesst Anzeige | |

**UC9: Hilfe anzeigen / ausblenden**

Beschreibung: Der User lässt sich die Hilfe zur Steuerung des Programmes anzeigen.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| | |
| 1. wählt Hilfe | |
| | 2. blendet Hilfe ein / aus |

**UC10: Umgebung anzeigen / ausblenden**

Beschreibung: Der User kann die Umgebung als Orientierungshilfe einschalten / ausschalten.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. wählt Umgebung | |
| | 2. Umgebung anzeigen / ausblenden |

**UC11: Koordinatenachsen anzeigen / ausblenden**

Beschreibung: Der User aktiviert / deaktiviert die Koordinatenachsen.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. wählt Koordinatenachsen | |
| | 2. Koordinatenachsen anzeigen / ausblenden |

**UC12: Lebensraumkubus anzeigen / ausblenden**

Beschreibung: Der User aktiviert / deaktiviert den Lebensraumkubus.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application}** | |
| | |
| 1. wählt Lebensraumkubus | |
| | 2. Lebensraumkubus anzeigen / ausblenden |

**UC13: Anzeige der Eigenschaften verändern**

Beschreibung: Der User passt die Anzeigeeigenschaften seinen Bedürfnissen an.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| **any time {close application, cancel}** | |
| | |
| 1. wählt Anzeige Eigenschaften | |
| | 2. UC5: Pause |
| | 3. zeigt Eigenschaften an |
| 4. wählt Einstellungen | |
| 5. bestätigt Einstellungen | |
| | 6. übernimmt Einstellungen |
| | 7. initialisiert die Anzeigeumgebung neu |
| | 8. UC2: Reset ZA |

**UC14: Wechsel in Fenster- / Vollbildmodus**

Beschreibung: Der User wechselt in Vollbild oder Fenstermodus.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| | **any time {report error}** |
| | |
| 1. löst Wechsel aus | |
| | 2. UC5: Pause |
| | 3. initialisiert die Anzeigeumgebung neu |
| | 4. UC2: Reset ZA |

**UC15: Änderung der Grösse des Fensters**

Beschreibung: Der User vergrössert oder verkleinert das Fenster.
Akteur: User
Vorbedingung: ZA geladen

| User Intentions | System Responsibility |
|---|---|
| | **any time {report error}** |
| | |
| 1. vergrössert oder verkleinert das Fenster | |
| | 2. UC5: Pause |
| | 3. initialisiert die Anzeigeumgebung neu |

**UC16: Kamerasteuerung**

Beschreibung: Der User steuert die Kamera.
Akteur: User

| User Intentions | System Responsibility |
|---|---|
| | |
| 1. bewegt die Kamera | |
| | 2. setzt Parameter |
| | 3. rotiert Ansicht |

**UC17: Beenden der Applikation**

Beschreibung: Der User beendet die Applikation.
Akteur: User

| User Intentions | System Responsibility |
|---|---|
| | |
| 1. beendet die Applikation | |
| | 2. UC5: Pause |
| | 3. räumt auf |
| | 4. beendet die Applikation |