Belegarbeit für das Projekt "Game of Life"

Hendrik Armbrecht und Toni Ngo

September 29, 2023

HTW Hochschule für Technik und Wirtschaft
Studiengang: Computer Engineering (1.Semester)
Kurs: Grundlagen der Programmierung
Dozent: Herr Prof. Dr. Sebastian Bauer

Contents

1 Einführung

Dies ist eine Belegarbeit für unser Projekt über die Erstellung des Spiels "Game of Life".
Conway's Game of Life ist ein mathematisches Modell auf einem Gitter, bei dem Zellen lebendig oder tot sein können. Die Entwicklung erfolgt anhand einfacher Regeln: Lebende Zellen überleben mit 2-3 Nachbarn, während einsame oder überbevölkerte Zellen sterben. Tote Zellen mit genau 3 Leben werden lebendig. Es entstehen sich entwickelte Muster und Strukturen, die in der Computerwissenschaft und Kunst verwendet werden.
In dieser schriftlichen Ausarbeitung erfahren Sie, wie unser Spiel funktioniert, die einzelnen Arbeitsschritte, sowie Herausforderungen und Probleme, die wir hatten.

2 Aufgabenstellung

Die Hauptaufgabe des Projekts war es, ein Spiel mithilfe von C und SDL2 zu implementieren. Zur Auswahl gab es 3 verschiedene Projekte: "Game of Life", "Abelian sandpile model" und "Mandelbrot". Das Projekt soll in einem Repository mit GitLab-Runner entwickelt werden. Auch gab es einige Anforderungen zum Spiel, welche im späteren Verlauf erläutert werden.

3 Bedienungsanleitung

Der kurz zusammengefasste Teil der Spielanleitung befindet sich in der ReadMe-Datei im Repository. Hier wird eine etwas detailliertere Beschreibung des Spiels stattfinden.

3.1 Spielphasen

Das Spiel unterteilt sich grob gefasst in drei Spielphasen: die Menüphase(1), die Bearbeitungsphase(2) und die aktive Spielphase(3). Man kann immer zwischen der zweiten und dritten Phase wechseln, während die Menüphase nur einmal im Start vorkommt.

Die **Menüphase** beschreibt den Startbildschirm des Spiels, bei der man die Größe der Spielfläche und der Kästchen einstellen kann. Es sollten jedoch nicht zu große Zahlen wie 1000x1000 gewählt werden, da unser Spiel nicht für riesige Projekte geeignet ist. Es ist lediglich eine Lite Version des Originals. Die ideale Einstellung ist unsere Standardeinstellung, wodurch es eigentlich nicht nötig ist, diese zu ändern. Um die Standardeinstellung zu benutzen, muss nur drei Mal Enter gedrückt werden, ohne Werte einzugeben. Um das Spiel zu starten, drückt man final noch einmal die Enter-Taste.

Die **Bearbeitungsphase** ist die Phase, wo man Zellen erzeugen und löschen kann. In der Bearbeitungszeit sind die lebenden Zellen grau statt schwarz, um zu verdeutlichen, in welcher Phase man sich befindet.

Die **aktive Phase** startet, nachdem der grüne Startknopf gedrückt wurde. In dieser Phase findet die Entwicklung der Zellen statt. Dabei kann die aktive Phase in zwei verschiedenen Varianten ablaufen: in der manuellen und der automatischen Variante. In der automatischen Variante läuft die Entwicklung alle 700 Millisekunden von alleine. In der manuellen Variante steuert man die Entwicklung mit der linken Maustaste pro Klick. Die manuelle Variante ist die Standardeinstellung und man wechselt zwischen den Varianten mit dem pinken Knopf.

3.2 Beschreibung der einzelnen Knöpfe

Es gibt insgesamt sechs Knoepfe, bei der alle unterschiedliche Symbole und Farben haben.

Angefangen mit dem **grünen Knopf**, der für den "Start" steht. Anders gesagt, er wechselt von der bearbeitenden Phase in die aktive Phase.

Danach kommt der **rote Knopf**, welcher für "Stopp" steht. Dieser dient zum Wechsel von der aktiven Phase in die bearbeitende Phase.

Der **blaue Knopf** ist der Reset-Knopf, welcher zur Zurücksetzung des Spielfeldes dient. Es werden im Grunde alle lebenden Zellen getötet. Dieser Knopf kann immer genutzt werden.

Die nächsten **zwei grauen Knöpfe** stehen für jeweils "Save" und "Load". Der erste speichert den Spielstand und der zweite ruft den gespeicherten Stand wieder auf. Da diese zwei ohne einander keinen Sinn ergeben, besitzen sie die

gleiche Farbe. Zwar sind beide Knöpfe jederzeit benutzbar, jedoch wird empfohlen, diese in der Bearbeitungsphase zu benutzen. Wenn man den Spielzustand in der aktiven Phase speichert, kann es sein, dass man nicht den gewünschten Stand speichert, sondern den danach.
Der letzte **pinke Knopf** wechselt zwischen der manuellen und der automatischen Variante. Dieser sollte in der aktiven Phase gedrückt werden.

3.3 Kommando-Flags

Es gibt vier Kommandos, welche man vor dem Spielstart hinzufügen kann. Also beispielsweise "build/release/gameoflife –cube-size 10". Ansonsten werden nun die Flags beschrieben:

- –**no-music**: stellt die komplette Hintergrundmusik des Spiels aus, da es Manuell über das Spiel nicht geht
- –**delay x** : ist die Einstellung der Entwicklungsgeschwindigkeit für die automatische Variante in der Aktiven Phase
- –**window-size x y** beschreibt die Spielfeldgröße, wobei x die Breite und y die Höhe darstellt
- –**cube-size** beschreibt die Größe der Kästchen

Da man im Menü auch die Einstellungen machen kann und es nicht zu Überschneidungen kommen soll, haben wir uns dafür entschieden, dass wenn "–cube-size" oder "–window-size" benutzt wird, die Menüphase übersprungen wird.

3.4 Extras

Hier geht es nur um kleine Erwähnungen über unser erstelltes Spiel. Was wir alles wirklich extra hinzugefügt haben, steht in einem anderen Kapitel. Es sollte nur kurz erwähnt werden, dass das Spielfeld größer ist, als es angezeigt wird. Es dient dazu, dass die Entwicklung nicht durch den Rand beeinflusst werden soll und es deshalb nach dem Rand etwas weitergeht.

4 Teillösungen

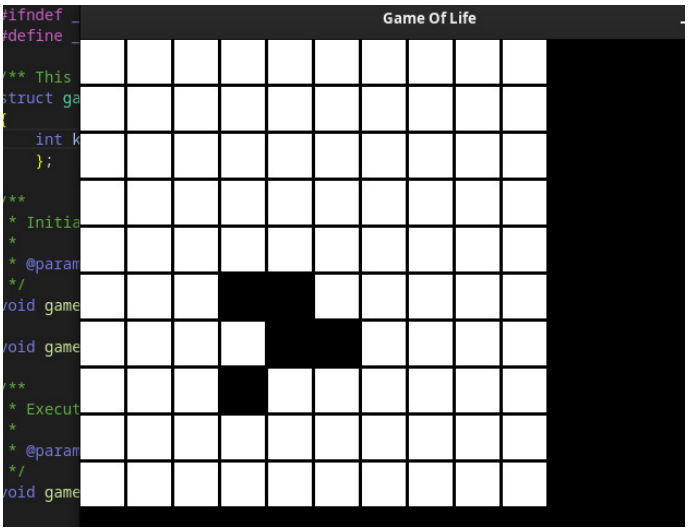

Figure 1: Spielfeld Prototyp **bild1**

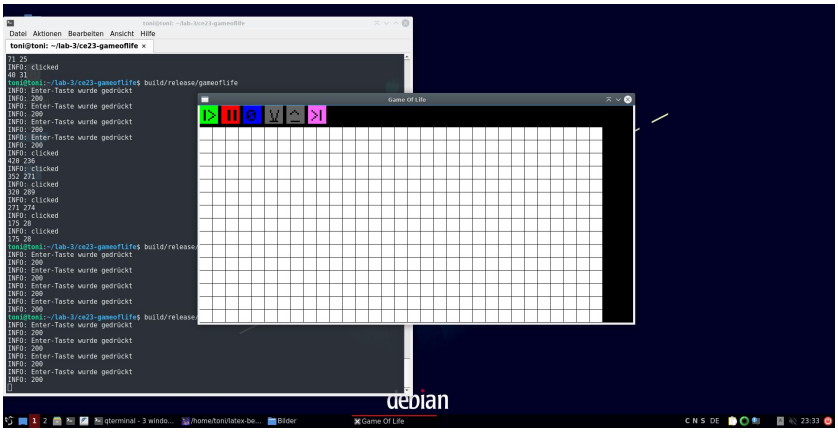

Figure 2: Endprodukt des Spielfelds **bild2**

5 Beschreibung der Quelltexte

In diesem Abschnitt geht es um die grobe Beschreibung unserer wichtigsten Quelltexte. Weitere Informationen stehen im Quelltext als Kommentar. Es werden deshalb auch nur kurz die wichtigsten Funktionen erwähnt.

5.1 gameoflife.c

In dieser Datei befindet sich die main-Funktion, sowie auch der Großteil unseres Projektes. Abgesehen von der main-Funktion befinden sich dort drei weitere Funktionen **"game"**, **"game_draw"** und **"game_draw_menue"**.

Die "void_game"-Funktion ist groeßer als sie eigentlich sein sollte, denn sie beinhaltet die Musikwiedergabe und die Nutzerinteraktion zwischen den drei Phasen, sowie die Interaktion mit den Knöpfen. Anfangs werden in der Menüphase die Spieleinstellungen konfiguriert. Dann geht es in die nächste Phase, wo es zur Schleife geht. Es gibt nämlich zwei Sprungziele **"endmenu:;"** und **ENDPLANT:;**, wodurch man immer zwischen der Bearbeitungsphase und der aktiven Phase mit der Start- und Stopp-Taste wechseln kann.

Die "game_draw"-Funktion rändert bzw. gestaltet das Spielfeld und dort wurden auch die Knöpfe mit deren Symbolen erstellt. Um die Symbole der Knöpfe darzustellen, mussten wir etwas kreativ werden, da es beispielsweise keine Pfeile nach unten oder oben gab, sowie keine nutzbaren Symbole. Da wir nicht die Schriftart wechseln bzw. eine neue suchen wollten, haben wir Buchstaben/Zeichen übereinandergelagert, um daraus Symbole zu kreieren. Es sind stolze 50 Zeilen Quellcode geworden, die man bestimmt irgendwie hätte einfacher darstellen können.

In "game_draw_menu" fand die Gestaltung des Menübildschirms statt. Hier hatten wir recht wenig Probleme gehabt, außer bei der Beschriftung, da wir anfangs keine Zeilenumbrüche hinbekommen hatten. Ansonsten wird der Text Zeilenweise grün, nachdem man die Enter-Taste drückt, um einen kleinen Überblick zu haben.

5.2 game_state.c

In dieser Datei befinden sich fünf Funktionen, die ausgeführt werden, sobald man im Spiel einen bestimmten Knopf betätigt. Die Funktion **"game_state_init"** setzt alle Koordinaten im Spielfeld auf 0 bzw. macht die Kästchen weiß. Dies geschieht einmal beim Spielstart und wenn man den blauen Knopf drückt. **"game_state_plant"** setzt ein Kästchen auf 1, wenn es 0 war und andersherum. Diese Funktion wird ausgeführt, wenn ein Kästchen in der Bearbeitungsphase angeklickt wird. So werden die Zellen lebendig/geboren oder gelöscht/sterben. Die Funktion **"game_state_update"** ist für die Entwicklung bzw. Evolution der Zellen zuständig. Alle Kästchen werden pro Sekunde abgetastet und es wird danach

geschaut, ob sie lebendig, sterben oder bleiben können. **"game_save"** und **"game_load"** werden durch den beiden grauen Knöpfen "save" und "load" ausgeführt. Der Spielstand wird in der Datei **"game_safe"** gespeichert. Wir haben es so gemacht, dass man nur einen Spielstand speichern kann, da es für ein kleines Projekt ausreicht. Ansonsten hätten wir Dateien per cmdargs-flag wie in lab-4 erstellt. Über einen weiteren flag hätten wir die gewünschte Datei bzw. Spielstand suchen und einlesen können (wie in lab-5).

5.3 restliche Quelltexte

Von den restlichen Quelltexten wird nicht viel mehr erwähnt, da diese in den Header-Dateien oder über Kommentare beschrieben wurden. Weitere Dateien wären **"cmdargs.c"** und ihre Testdatei **"cmdargs_test.c"** oder auch die Core-Datei, welche eine Funktion besitzt, die die lebenden Nachbarn einer Koordinate zählt.

6 Aufgabeneinteilung

Aufgrund von zeitlichen Gründen haben wir es als effizientesten Weg gesehen es so einzuteilen, dass Hendrik das Spiel entwickelt und Toni das drumherum macht. Da wir an unterschiedlichen Zeiten in den Urlaub fuhren, hätten wir kommunikative Schwierigkeiten gehabt, da wir in einem unterschiedlichen Tempo arbeiten würden.

6.1 Zuständigkeitsbereich

Hendrik war wie schon erwähnt für die Entwicklung des Spiels zuständig. Dazu zählen die einzelne Erstellungen der Funktionen und die Visualisierung des Spiels. Auch ist er für die Beschreibung der Funktionen zuständig. Das heißt, er hat die meisten Spielanforderungen fertiggestellt, sowie weitere Extras, wie die Musik. Toni war zuständig für die Unit-Tests in der Core-Datei, für die Nutzung der Kommandozeilenflags durch cmdargs. Ansonsten war er für die Bearbeitung des Belegs, die Spielanleitung, sowie die Korrektur des Codes zuständig.
Auch wenn jeder seinen eigenen Arbeitsbereich hatte, war der regelmäßige Austausch wichtig, damit man stets auf dem gleichen Stand bleibt. Dadurch konnte man sich auch immer gegenseitig bei Problemen unterstützen.

6.2 Arbeitsschritte

Angefangen haben wir mit einem allgemeinen Brainstorm, um den Arbeitsaufwand einschätzen zu können. Danach erfolgte die Aufgabeneinteilung. Da wir kaum Zeit für Treffen haben, hatten wir die Termine genutzt, um einander auf den neusten Stand zu bringen. Falls jemand nicht weiterkam, konnte der andere meistens weiterhelfen, wodurch wir nie

lange an Problemen fest hingen. Im Allgemeinen fing es mit Informationsbeschaffung an, danach mit der Umsetzung der Ideen und endete mit gegenseitigen Korrigieren. Es war zwar chaotisch und wir würden es nicht wiederholen wollen, aber am Ende ist alles gut gegangen.

7 Probleme und Herausforderungen

Hier kommen die Top 5 Highlights unserer Probleme:

1. **Prokrastination** : Da wir den Start sehr weit nach hinten verzögert haben, hatten wir am Ende Zeitprobleme.
2. **Ideen**: Dadurch, dass wir immer mehr Ideen ins Spiel implementieren wollten, hatten wir auch mehr Arbeit vor uns gehabt, als es nötig war.
3. **SDL2**: Auch wenn wir schon etwas mit SDL2 zu tun hatten, war es trotzdem schwierig hineinzufinden.
4. **Core-Datei/Test**: Wir hatten zu wenig Funktionen gehabt, die man testen könnte, da sie keinen Wert zurückgegeben haben. Man hätte sie bestimmt in eine testbare Funktion umändern können, aber aufgrund von Zeitproblemen waren wir mit einer Funktion in der Core-Datei zufrieden.
5. **Schlaflosigkeit**: Uns ist kein weiteres Problem eingefallen.

8 Erweiterungen und Extras

Dies sind unsere umgesetzten Ideen kurz zusammengefasst:

- **Musik**: Durch das Hinzufügen von Hintergrundmusik, wird ein weiteres Paket gebraucht, welche man sich herunterladen muss. Falls es nicht vorhanden ist, wird es von der Makefile erkannt und gibt eine Benachrichtigung aus. Außerdem mussten wir etwas am GitLab-Runner ändern, damit dieser auch das Paket hat. Die Musik kann nach einer Weile stören, deshalb empfehlen wir "–no-music", welche man beim Starten des Programms hinzufügen kann.
- **Spielanleitung**: Es war uns wichtig, dass man schnell einen Überblick vom Spiel haben sollte. Deshalb haben wir eine kleine, aber feine Anleitung im Repository fertiggestellt. Sie beinhaltet nicht die Spielregeln, sondern eher mehr die Funktionsweise der Knöpfe des Spiels.

9 Zusammenfassung

Auch wenn man in der Projektaufgabe nur zwischen drei Spielen wählen durfte, war der Spielraum, was die Gestaltung des Spiels angeht, sehr groß. Wir haben uns für "Game of Life" entschieden, da Hendrik eine Arte Dokumentation dazu gesehen hat. Wir haben diese Entscheidung bis zum Ende nicht bereut.

Man kann in den Teillösungen einen Vergleich zwischen unserer Arbeit am Anfang(1) und am Ende(2) sehen. Wir sind recht zufrieden mit dem Endprodukt, auch wenn wir viele Punkte besser hätten machen können. Zumal war es die geringe Anzahl an Funktionen für die Core-Datei, was auch dazu führte, dass es kaum Tests gab, abgesehen vom cmdargs. Auch ist die Datei "gameoflife.c" ziemlich groß geworden, verglichen mit den anderen Dateien. Man hätte einige Funktionen extrahieren sollen für einen besseren Überblick. Auch war die Aufgabeneinteilung nicht sehr sinnvoll, auch wenn es besser geklappt hat als erwartet. Zwar war es einfacher für alle Parteien sich auf eine Sache zu konzentrieren, jedoch versteht der eine vielleicht dann weniger vom Quelltext oder der andere weiß nicht, wie man einen ordentlichen Beleg schreibt, weil man es nie geübt hatte. Jedenfalls funktionierte es recht gut, da man nur zu zweit war und sich regelmäßig miteinander austauschen konnte.

Uns war es wichtig, dass man gut ins Spiel einsteigen kann. Das heißt, dass alles leicht und verständlich vermittelt werden musste. Beispielsweise haben wir extra versucht, die Knöpfe nach der richtigen Farbe, Symbol und Reihenfolge zu gestalten. Es wäre verwirrender gewesen, wenn die Knöpfe unnatürliche Farben hätten und man erst einmal herausfinden müsste, was der jeweilige überhaupt tut. Auch haben wir uns für Symbole entschieden, da diese universell und besser verständlich sind als Buchstaben. Natürlich wussten wir, dass auch dies nicht ausreichen würde, weshalb wir eine kleine Anleitung im Repository erstellt haben. Ideen, die wir hatten, waren rein- und herauszoomen, einen Timer, einen Knopf, wo die Evolution rückwärts verläuft (was gar nicht so toll aussah wie erwartet) oder auch andere Kästchenfarben, die man einstellen kann.
Man ist sehr erleichtert, nach so einem Projekt. Es war sehr lehrreich und herausfordernd. Es wäre sicherlich schwieriger gewesen, wenn das Spiel kein Spaß gemacht hätte.