technische universität
dortmund

Fachprojekt Report

Interactive Real-Time Gaming

Lukas Bünger
Thilaksan Kodeeswaran
Dilsan Mahadeva
30.09.2020

Supervisors:
Prof. Dr. Jian-Jia Chen
M.Sc. Junjie Shi

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl Informatik 12 (Eingebettete Systeme)
http://ls12-www.cs.tu-dortmund.de

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Hintergrund

Literatur [?] oder [?, ?] sowie Hinweise auf Quellen im Internet [?] und Verweis auf Kapitel ?? ab Seite ??.

Hinweise auf Diplom- [?], Bachelor- [?] und Masterarbeiten [?] sind auch möglich.

1.2 Aufbau und Umgebung der Arbeit

Er hörte „leise Schritte“ hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel[1]? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören.
Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst?
Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden? Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und

[1] Wer würde ihm schon folgen.

dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden.
Er hörte „leise Schritte“ hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören.
Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden. Beinahe wäre er dabei über den umgestürzten Mülleimer gefallen, der mitten im Weg lag. Er versuchte, sich in der Dunkelheit seinen Weg zu ertasten und erstarrte: Anscheinend gab es keinen anderen Ausweg aus diesem kleinen Hof als den Durchgang, durch den er gekommen war. Die Schritte wurden lauter und lauter, er sah eine dunkle Gestalt um die Ecke biegen. Fieberhaft irrten seine Augen durch die nächtliche Dunkelheit und suchten einen Ausweg. War jetzt wirklich alles vorbei, waren alle Mühe und alle Vorbereitungen umsonst?
Er presste sich ganz eng an die Wand hinter ihm und hoffte, der Verfolger würde ihn übersehen, als plötzlich neben ihm mit kaum wahrnehmbarem Quietschen eine Tür im nächtlichen Wind hin und her schwang. Könnte dieses der flehentlich herbeigesehnte Ausweg aus seinem Dilemma sein? Langsam bewegte er sich auf die offene Tür zu, immer dicht an die Mauer gepresst. Würde diese Tür seine Rettung werden? Er hörte leise Schritte hinter sich. Das bedeutete nichts Gutes. Wer würde ihm schon folgen, spät in der Nacht und dazu noch in dieser engen Gasse mitten im übel beleumundeten Hafenviertel? Gerade jetzt, wo er das Ding seines Lebens gedreht hatte und mit der Beute verschwinden wollte! Hatte einer seiner zahllosen Kollegen dieselbe Idee gehabt, ihn beobachtet und abgewartet, um ihn nun um die Früchte seiner Arbeit zu erleichtern? Oder gehörten die Schritte hinter ihm zu einem der unzähligen Gesetzeshüter dieser Stadt, und die stählerne Acht um seine Handgelenke würde gleich zuschnappen? Er konnte die Aufforderung stehen zu bleiben schon hören. Gehetzt sah er sich um. Plötzlich erblickte er den schmalen Durchgang. Blitzartig drehte er sich nach rechts und verschwand zwischen den beiden Gebäuden.

2 Spiel

In diesem Abschnitt wird das Snake Spiel beschrieben und die Funktionen von Elementen im Gameplay oder Buttons erklärt. Dabei wird gar nicht auf die Implementierung eingegangen, sondern nur erklärt wie das Spiel funktioniert.

2.1 Spiel - Hauptmenü

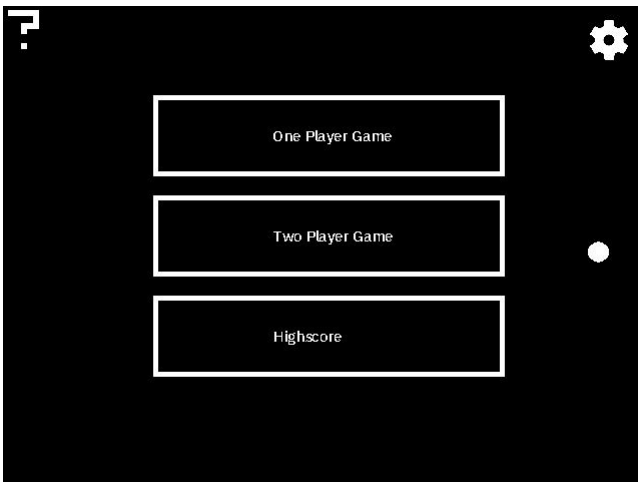

Abbildung 2.1: Hauptmenü

Das Hauptmenü 2.1 erscheint beim starten des Spiels und hat fünf Buttons. Liegt der Mauszeiger über dem Fragezeichen-Button wird eine Anleitung des Spiels angezeigt. Dort wird die Steuerung für die jeweiligen Spieler angezeigt und spezielle Elemente aus dem Mehrspielermodus erklärt.

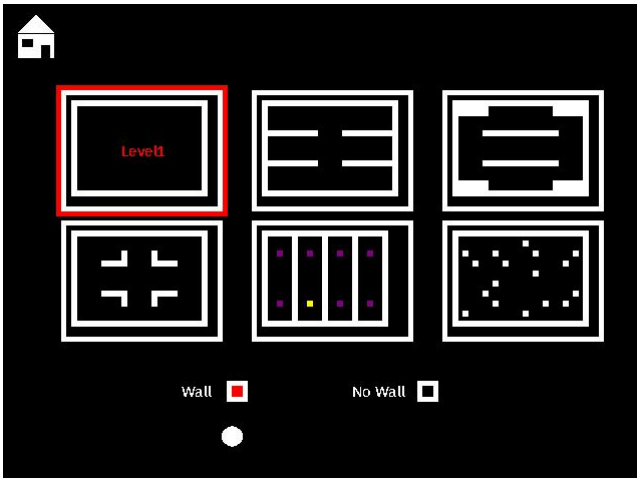

Abbildung 2.2: Einstellungen

Wird auf das Zahnrad rechts oben geklickt, öffnet sich das Einstellungsmenü 2.2, indem verschiedene Levels ausgewählt werden können und die Außenwand aktiviert und deaktiviert werden kann. Ist die Außenwand deaktiviert, kann die Schlange beim kollidieren mit der Außenwand nicht sterben. Durch klicken auf das Haus links oben kehrt der Spieler zum Hauptmenü zurück.

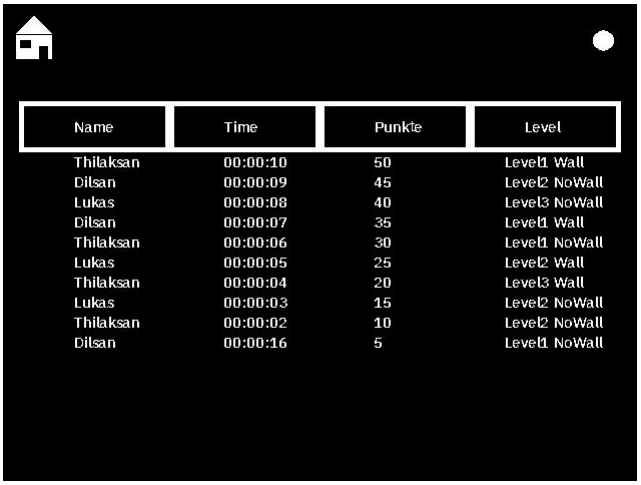

Abbildung 2.3: Highscore

Wird im Hauptmenü auf den Highscore-Button geklickt, wird der Highscore 2.3 aus den bisher gespielten Spielen angezeigt. Dabei sind die Highscores primär nach Punkten und sekundär nach Zeit sortiert. Außerdem steht in jeweils der letzten Zeile der Highscores das Level, indem gespielt wurde. Mit Klicken auf das Haus links oben kehrt der Spieler zurück zum Hauptmenü.
Klickt der Spieler auf den Öne Player GameButton öffnet sich ein Menü, indem der Spie-

ler seinen Namen eintragen kann. Dies kann durch Klicken auf die jeweiligen Buttons auf dem Bildschirm oder durch Eingabe über die Tastatur geschehen. Das Spiel wird dann gestartet, wenn der StartGame-Button betätigt wurde. Wählt der Spieler den Mehrspielermodus, durch Betätigen des "Two Player GameButton im Hauptmenü, öffnet sich wieder ein Menü zum Eintragen der Spielernamen und das Spiel kann durch das Betätigen des StartGame-Buttons gestartet werden.

2.2 Spiel - Einzelspielermodus

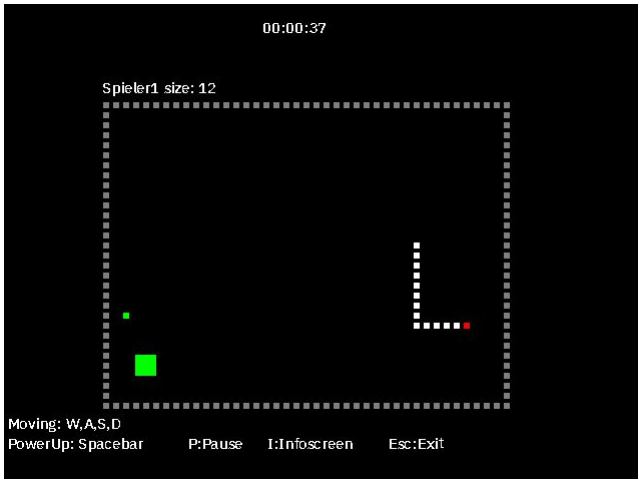

Abbildung 2.4: Einzelspieler Spiel

Der Einzelspielermodus ist im Grunde das traditionelle Snake Game. Die Schlange lässt sich steuern durch Betätigen der Tasten A,S,D,F und bewegt sich alle fünf Frames. Die grünen Elemente sind das Essen der Schlange, welches die Schlange wachsen lässt. Dabei wächst die Schlange um eine Größe beim Verzehren des kleinen Food-Elements und um zwei beim Verzehren des großen Superfood-Elements. Das Spiel kann durch die entsprechenden Levels erschwert werden.Level fünf hat ein lilanes Teleport-Element, welches die Schlange zu einem anderen Teleport-Element teleportiert. Dabei teleportieren die oberen Elemente zum nächsten rechten Element und die unteren Elemente zum nächsten linkem Element. Level 6 verändert sein inneres Wandmuster nach dem Verzehren des Food-Elements jedoch nicht nach Verzehren des Superfood-Elements. Das Spiel endet, wenn die Schlange mit der Wand oder mit sich selber kollidiert.

2.3 Spiel - Mehrspielermodus

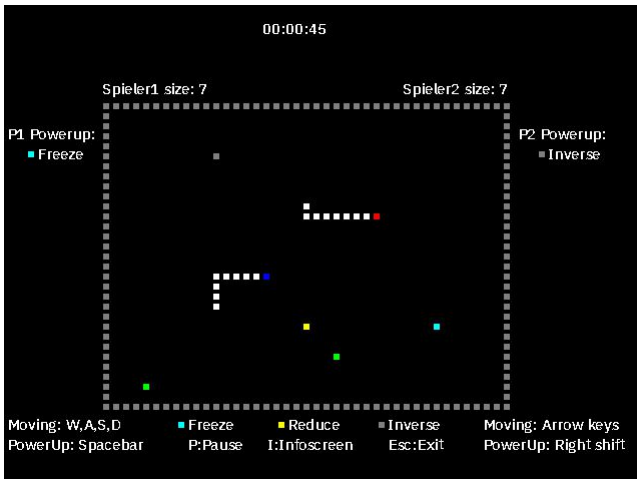

Abbildung 2.5: Mehrspieler Spiel

Beim Mehrspielermodus spielen zwei Spieler gegeneinander und versuchen das Spiel durch töten der Schlange des Gegenspielers zu gewinnen. Spieler eins bewegt die rote Schlange mit den Tasten A,S,D,F und Spieler zwei bewegt die blaue Schlange mit den Pfeiltasten. Im Mehrspielermodus gibt es außerdem Spezial-Elemente, wie das cyanfarbene Freeze-Element, das gelbe Reduce-Element und das graue Inverse Element. Diese Elemente können aufgesammelt werden und mit Leertaste für Spieler eins und mit Shift für Spieler zwei eingesetzt werden. Dabei kann nur ein Element gleichzeitig in der Tasche eines jeweiligen Spielers sein. Beim Einsatz des Freeze-Elementes kann die gegnerische Schlange sich für 25 Frames nicht bewegen. Beim Einsatz des Reduce Elementes veringert sich die Länge der gegnerischen Schlange um eine Größe. Das Inverse-Element invertiert die Steuerung des Gegenspielers für 250 Frames. Das Spiel ist beendet, wenn eines der Spieler verliert. Kollidiert die Schlange eines Spielers mit der Wand, sich selber oder dem Körper der gegnerischen Schlange, verliert derjenige Spieler. Kollidieren beide Schlangen Kopf an Kopf gewinnt derjenige Spieler mit der größeren Schlange.

2.4 Spiel - Spielende

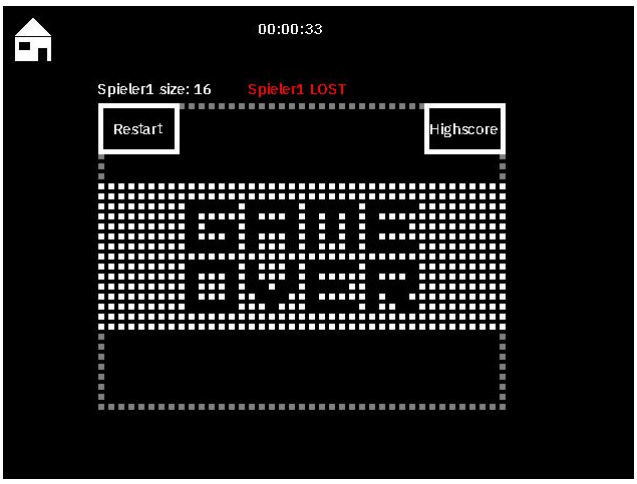

Abbildung 2.6: Spielende

Ist das Spiel zu Ende erscheint eine kleine Animation und mehrere Buttons. Der Restart-Button startet das Spiel neu und der Highscore-Button zeigt den Highscore an. Außerdem steht über dem Spielfeld welcher Spieler gewonnen hat.

3 Struktur

In diesem Abschnitt wird beschrieben, wie das Spiel in seiner Struktur aufgebaut ist, entspricht wie jedes einzelne Element sich zum Gesamtspiel aufbaut. Dabei wird von jedem Essenselement bis hin zum Bildschirm die Implementierung und Erzeugung erklärt.

3.1 Elemente

3.2 Relation der Elemente

3.3 Zustände

4 Spiel Logik

In diesem Kapitel gehen wir auf die Hauptfunktion des eigentlichen Spielablaufs, $vGameScreen()$, und der dafür benötigten einzelnen Funktionen ein. Wie in den meisten Computerspielen besteht die Hauptfunkion im groben aus einer einzigen großen Schleife, in der zum der aktuelle Spielzustand abhängig von den Eingaben der Spieler geändert wird, und zum anderen der aktuelle Spielzustand grafisch aufgearbeitet und auf den Bildschirm gebracht wird. Vor dem Start dieser Schleife findet noch eine Initialisierung statt, welche den Spielzustand auf den Anfang des Spiels setzt.

4.1 Initialisierung

Das erste, was nach Starten des Spiel geschieht, ist das Erstellen eine Art *seed* für die $rand()$ Funktion der C - Programmiersprache. Da die $rand()$ Funktion die verstrichene Zeit seit Aufruf der Funktion in der sie aufgerufen wird als Basis für die Berechnung der Pseudozufallszahl nimmt, die ersten Aufrufe der $rand()$ Funktion immer die gleichen Werte zurückgeben, und somit die Schlangen der Spieler sowie das erste $foodElement$ immer an der gleichen Stelle starten. Um den entgegenzuwirken berechnet die Funktion als erstes die letzten beiden Ziffern des Produkts der x- und y-Koordinaten der Mausposition beim klicken des Start-Buttons und dekrementiert diese Zahl bis zur null. Durch diese minimale verstrichene Zeitspanne ändern sich die Ergebnisse der ersten $rand()$-Aufrufe und damit der Startzustand des Spielfelds, vorrausgesetzt man drückt beim Start nicht auf den exakt gleichen Pixel wie vorher. Anschließend werden, je nach gewähltem Level, die Wände sowie alle weiteren festen Elemente des Levels an ihre entsprechenden Stellen ins $fieldArray$ geschrieben. Daraufhin werden die Variablen für die Position der beiden Spieler, $p1X, p1Y, p2X$ und $p2Y$, sowie das erste $foodElement$, auf zufällige Positionen im Spielfeld gesetzt und die zugehörigen verketteten Listen für die Schlangen mit bereits 2 weiteren SSchlangenteilenïnitialisiert. Dies geschieht auch im Singleplayer-Modus für die Schlange von Spieler 2, da der Code ansonsten aufgrund einer möglicherweise nicht initialisierten Spieler-2-Schlange nicht kompiliert. Einzig der Eintrag im $fieldArray()$ wird im Singleplayer-Modus ausgelassen, da offensichtlich kein Spieler 2 existiert. Nun ist das Spielfeld soweit initilisiert dass das eigentliche Spiel beginnen kann.

4.2 Gameloop

5 Probleme und Lösungen

Im folgendem werden einige Probleme, die bei der Implementierung des Projektes aufgekommen sind, beschrieben und des weiteren werden verschiedenen Lösungsansäzen dazu vorgestellt.

5.1 Bewegung der Schlange

Bei der Implementation der Schlange wurde am Anfang erst ein Quadrat erstellt, danach musste die Bewegung hinzugefügt werden. Da am Anfang noch keine Bildschirmrate gab, konnte die Schlange in jede Richtung in Echtzeit laufen, je nach Eingabe der Richtung. Um die Schlange in direkter Diagonalen Richtungen zu vermeiden, wurde die Bewegung in Horizontaler- und Vertikaler-Achse beschränkt. Das Problem war, dass die Schlange in die entgegengesetzte Richtung laufen kann, d.h. die Schlange konnte durch sich selbst durchlaufen.
Der erste Lösungsansatz war mit Hilfe von einer Variable die aktuelle Richtung zu speichern um je nachdem in die andere Achse zu laufen. Damit wird verhindert direkt in die entgegengesetzte Richtung zu laufen. Jedoch wurde das Problem noch nicht behoben, da durch gleichzeitige benutzen von mehreren Richtungseingaben die Variable eine falsche Richtung erhält und somit die entgegengesetzte Richtung erlaubt. Dies kann so schnell passieren, sodass die Schlange die zweite Richtung nicht wahrnimmt und sofort in die entgegengesetzte Richtung läuft.
Deshalb gab es es eine endgültigen Lösungsansatz, welcher mit zwei Variablen und mit der Methode, die jeden Schritt der Schlange verarbeitet, arbeitet. Die Variablen sind dabei einmal $p1Direction$ (für Player1 und Player2 p2Direction) für die aktuelle Richtung und $p1NextDirection$ (für Player1 und Player2 p2NextDirection) für die nächste Richtung. Die Idee mit dem verhindern der Achse bleibt. Jedoch wird die aktuelle Richtung erst der neuen Richtung gesetzt, sobald die Schlange sich um ein Feld bewegt.

5.2 Elemente Erzeugen

Relativ früh in der Entwicklung des Spiels machten sich Performanceprobleme beim Erstellen neuer Elemente, vor allem des $foodElement$, bemerkbar. Bei zunehmend langer Schlange erhöhte sich die Rechenzeit für das Finden einer freien Stelle auf dem Spielfeld

so stark, dass zum Beispiel nach dem finden des $foodElements$, eine bemerkbare Pause entstand in der das Spiel nicht weiterlief und somit stockte. Diese Pause war allerdings nicht gleichbleibend lang, sondern erschien zufällig und mit unterschiedlicher Dauer, jedoch mit ansteigender Länge der Schlange öfter und länger. Dies war zurückzuführen auf unseren Algorithmus zur Bestimmung einer freien Stelle sowie die Speicherung der Schlange in einer Liste. Der Algorithmus suchte pseudo-randomiesiert eine Stelle des Spielfeldes aus und überprüfte daraufhin, ob ein Element der Schlange bereits auf diesem Feld ist. Falls dies nicht der Fall war, war das Feld geeignet, ansonsten wurde einfach ein anderes Feld ausgesucht und dies solange wiederholt bis ein freies Feld gefunden wurde. Durch eine längere Schlange stieg somit die Wahrscheinlichkeit, dass ein neues Feld ausgesucht und der überprüfungsprozess deshalb mehrere Male durchgeführt werden musste. Das größte Problem hierbei war aber er überprüfungsprozess selber, sowie die Speicherung der Schlange in einer verketteten Liste. Um zu überprüfen, ob das ausgesuchte Feld nicht schon durch die Schlange belegt war, musste die gesamte Schlange Element für Element überprüft werden. In Kombination mit der Wahrscheinlichkeit auf mehrere überprüfungen entstand dadurch bei längeren Schlangen ein so großer Rechenaufwand, dass die Verzögerung durch den Spieler bemerkbar und somit der Spielfluss ins stocken geraten ist. Zur Lösung des Problems führten wir eine neue Variable ein, welche auch in der weiteren Entwicklung noch sehr hilfreich wurde: ein zweidimensionales Array mit Integer-Werten, das $fieldArray$. Das $fieldArray$ dient als kompakte Darstellung des aktuellen Zustands des gesamten Spielfelds. Dabei repräsentierte jeder der 39 x 29 Integer-Werte eine Stelle auf dem Spielfeld und deren aktuellen Zustand. Ist der Wert 0 ist das Feld frei, ist der Wert 1 ist die Stelle von der Schlange von Spieler 1 belegt. Alle Werte und zugehörigen Zustände sind in zu sehen. Durch das $fieldArray$ ist nun der Prozess des überprüfens einer Stelle nicht mehr abhängig von der Länge der Schlange, sondern kann immer in konstanter Zeit durch einmaliges checken des zugehörigen Wertes im $fieldArray$ erledigt werden. Das Probelm der mehrmaligen überprüfungen durch Auswählen eines bereits belegten Feldes ist damit auch trivialisert, da das überprüfen nun schnell genug von statten geht. Durch das $fieldArray$ wurde auch die weitere Implementierung des Spiel vereinfacht, da durch neue Wertezuweisungen im Array relativ einfach neue Objekte wie Wände und $PowerUps$ in die Spiellogik eingeführt werden konnten, und das überprüfen der Stellen an denen diese Objekte sich befinden durch das $fieldArray$ direkt abgedeckt ist.

5.3 Spielleistung

Beim Ausführen des Spiels kam es sehr oft zu Problemen mit der Spielleistung. Insbesondere wenn das Spiel mit anderen Programmen, wie BigBlueButton, gleichzeitig lief. Dies führte in den meisten Fällen zur Verlangsamung des Spielfluss. Desweiteren flackerte das Spiel sehr stark, sodass ältere Frames anstatt des neuen Frames gezeichnet wurden. Wei-

tere Probleme waren häufig auftretende Speicherzugriffsfehler vor allem, wenn das Spiel über eine virtuelle Maschine lief. Jener Fehler führte zur Terminierung des Spiels.

Appendix

Abbildungsverzeichnis

Algorithmenverzeichnis

List of Source Codes

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem Titel

_____

_____

_____

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift