Entwickler-dokumentation

„3D Snake“

erstellt von:

Gruppe11:
Patrick Steinmetz, Matr.-Nr.: 190617
Elmar Braun, Matr.-Nr.: 190402

Inhaltsverzeichnis

1. Aufgabe

1.1 Laut Pflichtenheft

„Ein Klon des Spiels „Snake“ in 3D. Die Aufgabe des Spiels ist es zufällig erscheinende „Früchte „mit einer Schlange zu fressen. Das Ziel des Spiels ist es die Schlange durch das „Fressen“ zu verlängern, es gibt also nur einen Level bzw. eine Schwierigkeitsstufe. Die Schlange selbst ist immer in Bewegung und lässt sich durch die 4 Richtungstasten in die jeweilige Richtung lenken.
Die Schlange wird in einer vollständigen 3D-Umgebung gesteuert. Die steuerbaren Richtungen sind immer aus der Sicht des Schlangenkopfes zu sehen (links, rechts, oben, unten). Ein Abbremsen oder Stehenbleiben der Schlange ist nicht möglich. Durch das Fressen einer Frucht wird sie länger. Falls die Schlange eine der Kanten der Spielfeldbegrenzung berührt oder sich „selbst in den Schwanz beißt“ gilt das aktuelle Spiel als beendet.“

1.2 Erreichte Lösung

Die o.g. Kriterien wurden eingehalten.
Es wurden weiterhin einige Wunschkriterien erfüllt. Zum Beispiel erscheint ein Menü, vor Beginn des Spiels, sowie eine Pausefunktion und eine GameOver-Meldung. Die Spielumgebung wurde durch Texturen auf Schlange und im Hintergrund insgesamt attraktiver und lebendiger gestaltet. Auch eine Punktzählung wurde implementiert.

2. Erfüllte Kriterien

Kriterien	Erfüllt
Musskriterien	
• 3D Umgebung	✓
• 3D Würfel als Spielfeld	✓
• Steuerung der Schlange mit den Pfeiltasten	✓
• zufällig erscheinende Äpfel	✓
• Verlängerung der Schlange bei Kontakt(„Fressen“) von Apfel	✓
• Game-Over bei Kollision der Schlange mit Spielfeld-Begrenzung oder mit sich selbst	✓
Wunschkriterien:	
• Sound	-
• Menü	✓
• Punktezahl	✓
• Highscore-Tabelle(Name und Punktzahl wird gespeichert)	-
• Texturen	✓
• variable Spielfeldgröße	-
• mehrere Spielstufen = Erhöhung der Bewegungsgeschwindigkeit der Schlange	-
• mehrere „Leben“ der Schlange	-
• Auffüllen der Leben durch Zufallsgegenstände	-
• zufällige Kollisionsgegenstände	-
• ruckelfreie Animation der Schlange bei möglichst geringen Hardware-voraussetzungen *	✓

* das Bewegen der Schlange ist mit einem Ruckeln verbunden, da sie von Feld zu Feld springt.

3. Zeitlicher Aufwand (Soll/Ist)

Pakete	Patrick Steinmetz		Elmar Braun	
Stunden	geplant	tatsächlich	geplant	tatsächlich
Entwurf		7		7
Struktur C++		5		5
Grafik				
Viewing		4		4
Würfel	10	8	-	-
Schlange		-	10	2
Frucht	5	4		
Kollision				
Spielfeld-begrenzung	15	2	10	-
Frucht-berührung	10	2	5	-
Schlange mit sich selbst	5	-	10	3
Funktionen				
Schlangen- Bewegung	-	-	15	2
Schlange-Steuerung		-		5
Kopf-Rotation		12,5		23,5
Tail-Bewegung		7,5		1,5
Zufalls-Generator für Früchte	5	2	-	-
Würfel-Drehung	-	-	-	-
Punkte-Zählung	10	-	10	3
Bugfixing	15	8	15	8
SUMME	75	62	75	64

3.1 Zusätzlich eingefügte Funktionen

Pakete	Patrick Steinmetz		Elmar Braun	
Stunden	**geplant**	**tatsächlich**	**geplant**	**tatsächlich**
Allgemeine Textur-Implementation	**x**	**2**		
Snake- Textur	**x**	**2**		
Hintergrund- Textur	**x**	**2**		
3 Früchte			**x**	**2**
StartMenü	**x**	**2**		
GameOver			**x**	**1**
Pause			**x**	**1**
Schlangenschatten			**x**	**2**
Koordinationshilfe Früchte			**x**	**2**
Pflichtenheft		**13**		**13**
Anwenderdokumentation		**4**		**4**
Entwicklerdokumenation/UML		**7**		**7**
Meetings		**2**		**2**
Quellcode-Überarbeitung		**2**		**2**
Endbesprechung/Resümee		**1,5**		**1,5**
SUMME		**37,5**		**37,5**

3.2 Zusammenfassung

	Gesamt	**Patrick Steinmetz**	**Elmar Braun**
Planung*	69	34,5	34,5
Implementation*	112	55	57
Tests/Bugfixing*	20	10	10
Gesamt	205	101,5	103,5

*Planung
Meetings, Endbesprechung, Entwicklerdokumentation, Anwenderdokumentation, Pflichtenheft, Entwurf, Problembesprechungen

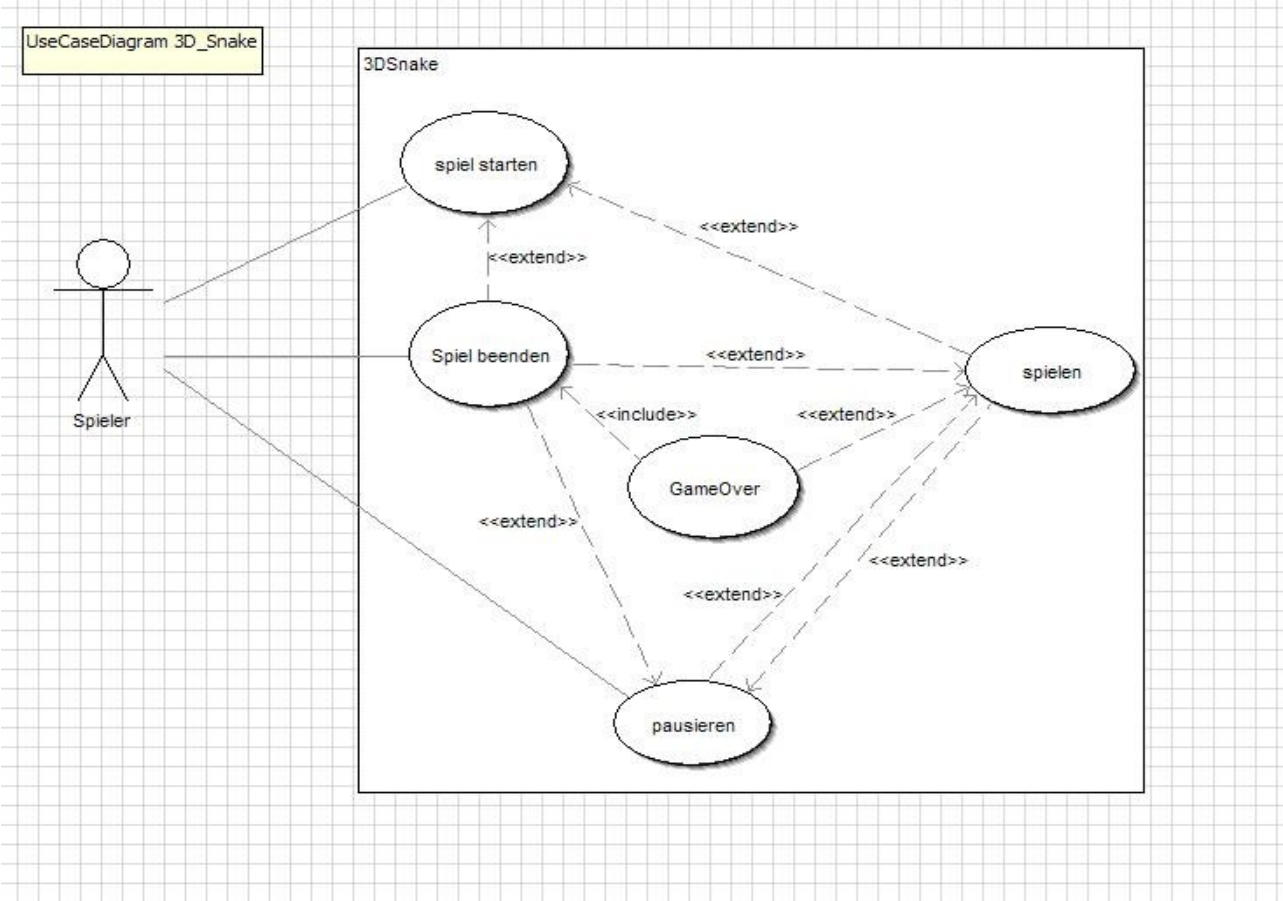
*Implementation
alles andere ausser „Planung“ und „Test“

*Tests/Bugfixing
Quellcode-Überarbeitung, Tests, Bugfixing

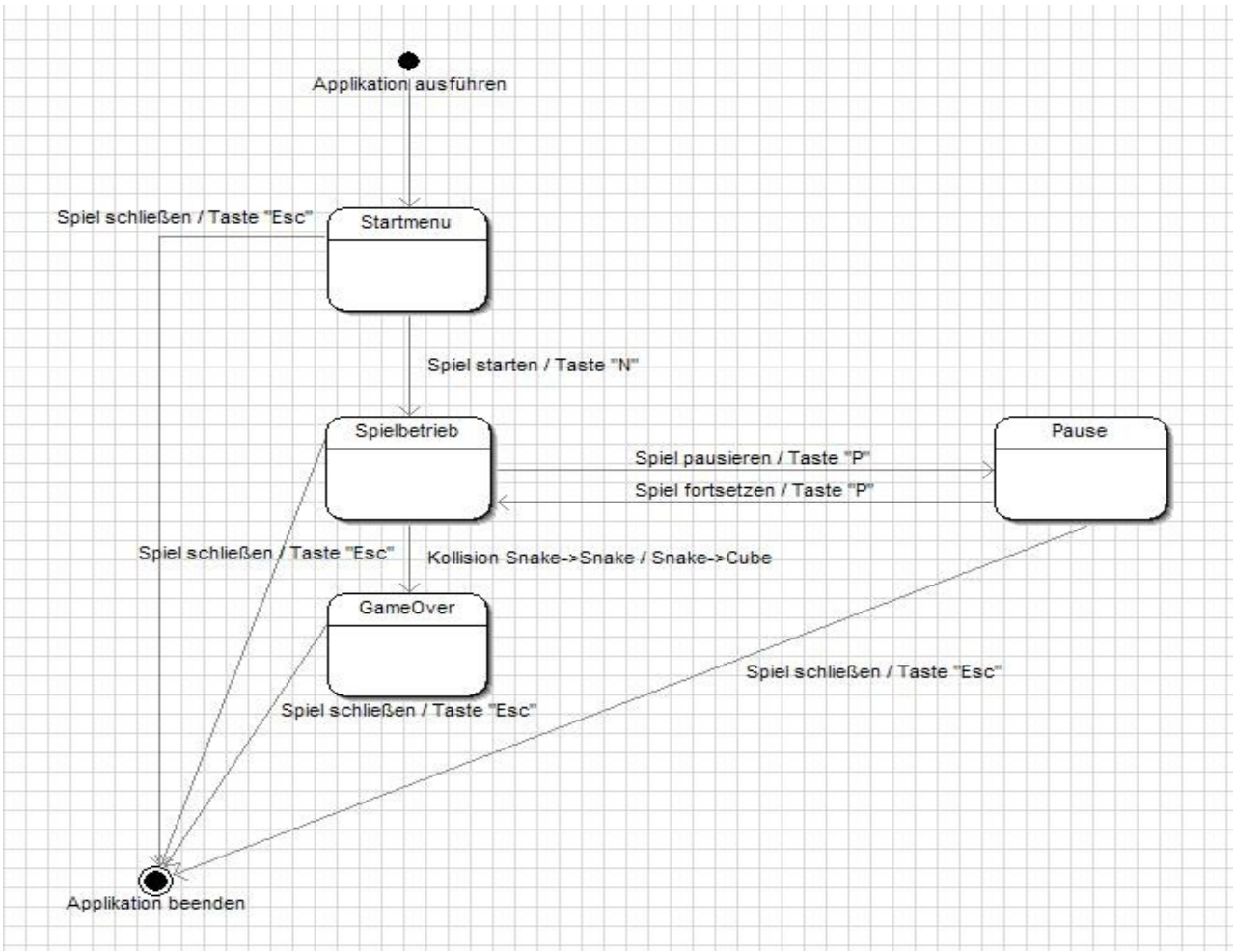
4. UML- Diagramme

(erstellt mit Omondo und StarUML)

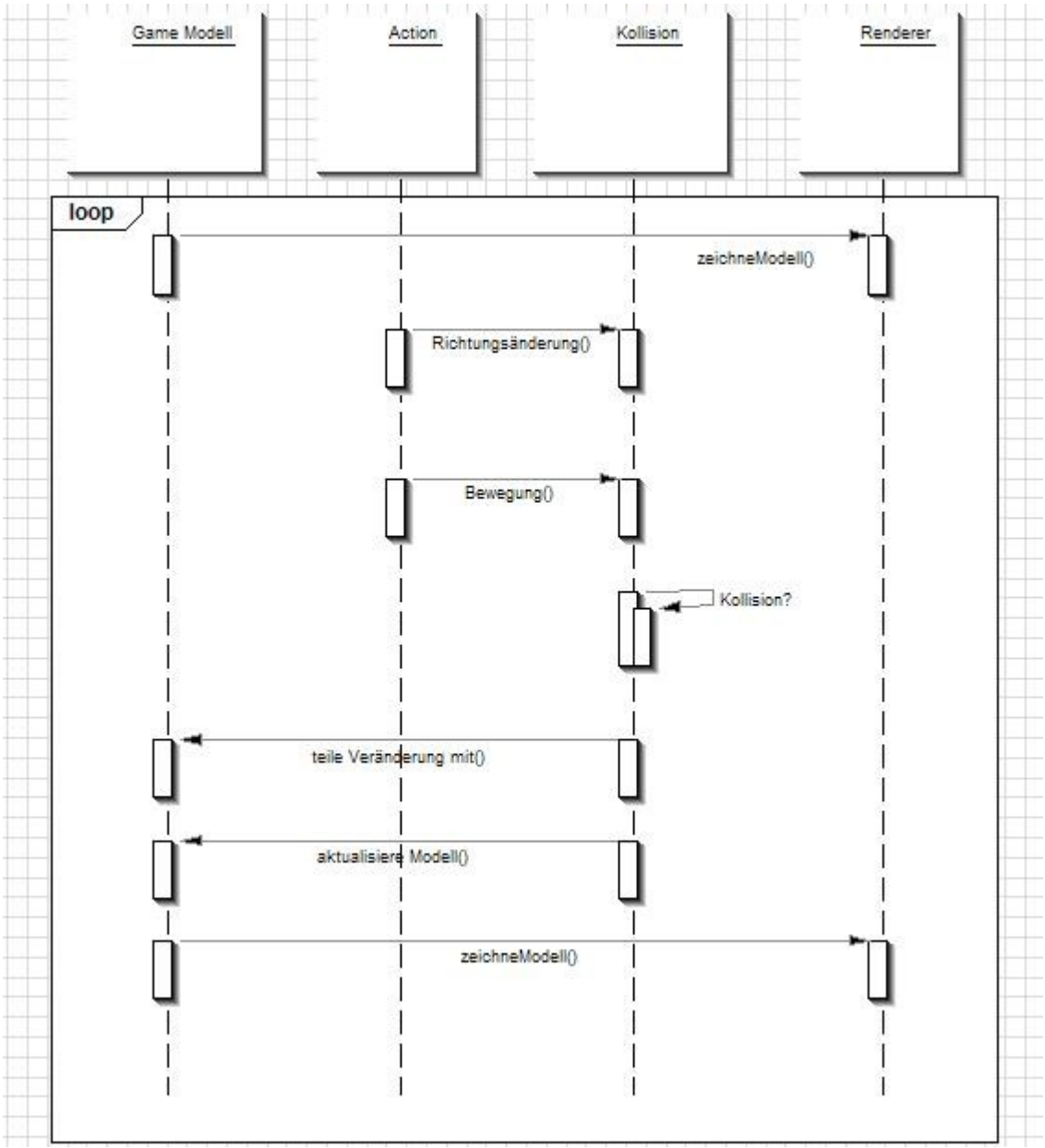
4.1 Use-Case-Diagramm:



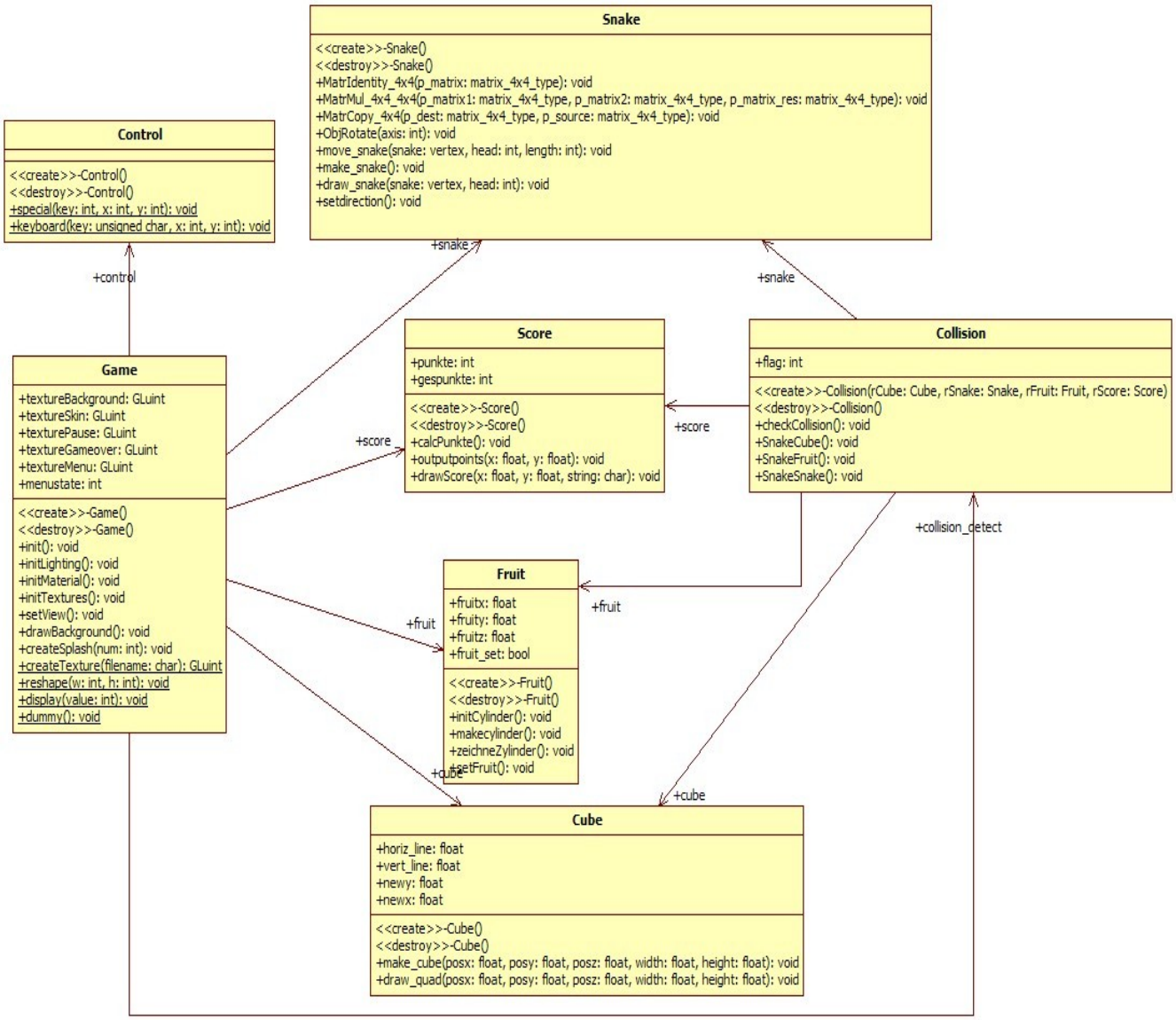
4.2 State-Diagramm:



4.3 Sequenz-Diagramm



4.4 Klassendiagramm:



5. Testplan

Alle Tests wurden auf mindestens zwei PC´s mit unterschiedlicher Hardware mit Windows XP Professional und Suse Linux 10.1 sowie einer Knoppix-Distribution(Live-CD) getestet.

Durchgeführte Tests	Steinmetz	Braun
Korrekte Darstellung	✓	✓
Korrekte Punktezählung	✓	✓
Dauertest	✓	✓
Korrekte Kollisionserkennung	✓	✓
Steuerungstest	✓	✓
Statustest (Menü,Gameover,Pause)	✓	✓
Vollbild-Test	✓	✓

5.1 Testbeschreibung

Korrekte Darstellung
Korrekte Darstellung von

- **Farben**: Farben auf texturierten Objekten, Farben auf untexturierten Objekten
- **Objekten**: Schlange(Kopf und weitere Glieder) Würfel, Früchte, Punktestand, Menüeinblendungen
- **Texturen**: auf Schlange, Schwanzglieder, Hintergrund, Menüeinblendungen
- **Fenster**: vergrößern, verkleinern s. Vollbild-Test
- **Spielabläufe**: Schlangenkopf-Rotation, Schlangen-Bewegung, Frucht-Darstellung vor/nach Kollision, Schatten der Schlange, Orientierungshilfen der Früchte vor/nach Kollision

Korrekte Punktezählung

- Punkteerhöhung nach Kollision mit Frucht
- Punkteerhöhung nach über 10 Gliedern (größer 55 Punkte)

Dauertest

- Spiel wurde von jedem Tester länger als 10 Minuten gespielt, dabei wurden keine Abbrüche oder sonstige ungewollte Fehlverhalten festgestellt.

Kollisionserkennung

- Die Kollision der Schlange mit Früchten, Würfel sowie dem eigenen Schwanz wurde von jeder möglichen Bewegungsrichtung getestet.
- An der Spielfeldbegrenzung wurde die Kollision zusätzlich an verschiedenen Koordinaten getestet.
- Alle Folgeereignisse der o.g. Kollisionen sind erfolgreich eingetreten.
- Die Orientierungshilfen der Früchte sind für die Kollisionserkennung nicht relevant.

Steuerungstest
- Alle Tasten lösen ihre vorgegebene Aktion aus. Abbrüche des Spiels sind durch Spezial-Tasten weiterhin möglich.

Statustest
- Menü, Gameover, Pause werden korrekt angezeigt.
- Pause pausiert das Spiel wie vorgegeben und nur während des Spielbetriebs, d.h. Während das Spiel sich im Status Menü oder GameOver befindet, ist keine Pause möglich.
- Aus dem Status GameOver kommt man nicht mehr zurück ins Spiel.

Vollbild-Test
- Das Spiel funktioniert im Vollbildmodus.
- Die Skalierung durch die Reshape-Funktion funktioniert nur eingeschränkt, da die Seitenverhältnisse eingehalten und nicht verändert werden können.

6. Verwendete Techniken – Beschreibung

Unser Spiel „3D Snake“ besteht hauptsächlich aus den Objekten: Cube, Snake, Fruit und Score. Alle diese werden in der Hauptklasse „Game“ erstellt und später der Klasse „Collision“, die ebenfalls in der Game erstellt wird, mitgeteilt.
„Snake“, „Fruit“ und „Cube“ befinden sich in einem spielinternem Koordinatensystem. In jedem Durchlauf der „display-Funktion gleicht die „Collision“ die Koordinaten dieser kollisionsabhängigen Objekte miteinander ab.

Es kann dadurch zu drei verschiedenen Kollisionsarten kommen:
- Kollisionen „Snake -> Snake“ und „Snake -> Cube“ führen zu einem „Game Over“ und damit dem Ende des Spiels. Dieses wird dem Benutzer durch eine „Game Over“-Meldung und der erreichten Punktzahl mitgeteilt.
- Kollision „Snake -> Fruit“ hingegen hat zur Folge, dass der Benutzer eine Punktzahlerhöhung erhält und die Zufallsposition mit der kollidierten Frucht wird neu berechnet und gesetzt. Außerdem wird die Länge des Schwanzes der Schlange um eins erhöht.

Die Schlange (Klasse: Snake) besteht aus einem Array von 3-Dimensionalen Vektoren.
Die Hauptfunktionen der Schlange sind funktional getrennt in „Bewegung“, „Kopf-Rotation“ und „Zeichnen“ implementiert.
Durch Matrix-Multiplikationen wird eine realistische Drehung des Kopfes in alle möglichen Richtungen erreicht.

7. Verwendete Techniken – Besonderheiten

- Spielbar unter Windows und Linux
- Weitesgehend Objektorientierte Struktur
- Durch die Verwendung einer glutTimerFunc() wird die Laufgeschwindigkeit auf unterschiedlich schnellen PC-Systemen angeglichen.
- Rotation des Kopfes basierend auf Matrix-Multiplikation

8. Abschließende Projektbesprechung über den Projektverlauf

8.1 Aufgetretene Probleme

„Kopf-Rotation-Problem“
Da die Rotation des Kopfes um die xyz-Achse nicht durch glRotate-Anweisungen zu lösen war, musste ein völlig neuer Ansatz gewählt werden.
Erst nach mehreren neuen Ideen und einem Gespräch mit Herrn Prof. Rodrian wurde eine zufriedenstellende Lösung gefunden. Letztendlich wurde für die Schlange eine eigene Rotationsmatrix, welche mit der Spielmatrix multipliziert wird, implementiert.

„Snake-Tail-Problem“
Einer der ersten Versuche den Schlangenschwanz zu implementieren führte dazu, dass sich der Schwanz der Schlange beim Richtungswechsel zu früh in die neue Richtung bewegte. Durch Implementation der Schlange als Array von Vektoren konnte dieses Problem jedoch behoben werden.

„TimerFunc-Problem“
Um das Programm unter Linux und Windows unabhängig von der System-Geschwindigkeit zu betreiben, mussten wir verschiedene Arten der Drosselung der Animationsgeschwindigkeit testen. Unsere Lösung ist eine glutTimerFunc mit davon abhänigigen Timer für die Bewegung.

8.2 Zeitplanung

Die effektive Planungs- und Implementierungszeit stimmt mit der geplanten Zeit weitesgehend überein.
Nicht berücksichtigt wurden sämtliche Dokumentations-Zeiten.
Trotz o.g. Probleme konnte der Zeitplan aufs Ganze gesehen gut eingehalten werden.
In Bezug auf Milestones und geplante Fertigstellungstermine einzelner Arbeitspakete stimmt die Planung nicht mit den tatsächlichen Terminen überein. Gründe hierfür sind unvorhergesehene Klausuren, Krankheitsausfall, Hardwareausfall sowie berufliche Aktivitäten.