CIOCO GAMING STUDIO

Dokumentation zu dem Probeprojekt:

Konzeption und Umsetzung des Spiels "Snake" in Java

unter Verwendung des JavaFX-Frameworks

in der Eclipse IDE

Probeprojekt IHK 2024

Ausbildungsberuf:
Fachinformatiker Anwendungsentwicklung

Ausbildender Betrieb:

[redacted]

Xxxxstr. 99

01234 Nürnberg

Projektverantwortlicher:

[redacted]

XXXXXX@XXX.XXX

Prüfungsteilnehmer:

Cîrjeu Daniel Marian

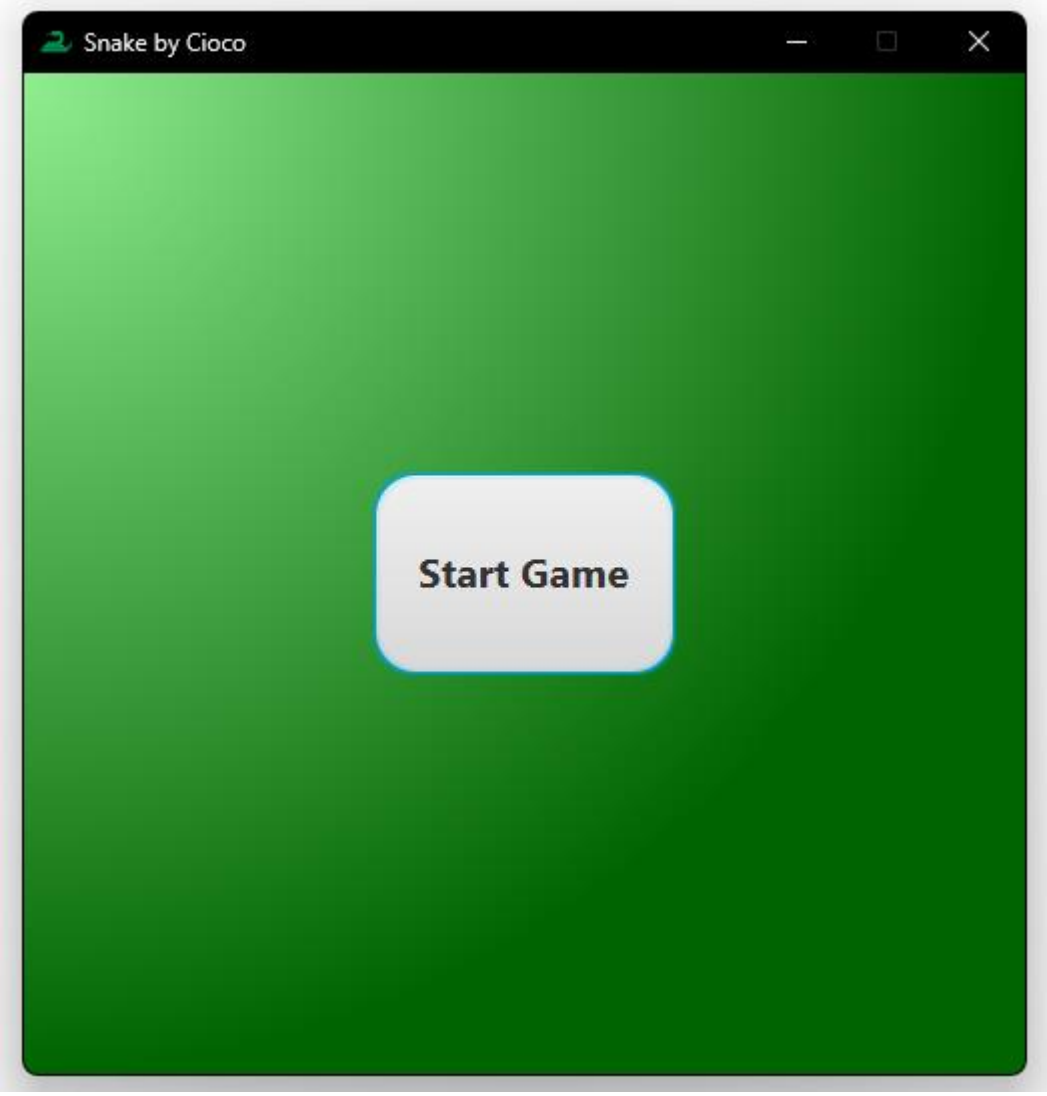
Zuhausestr. 01

43210 Nürnberg

Azubi-ID: XXXX

Geb. Datum: 17.08.1995

Inhalt



1. Aussgangssituation

1.1 Projektumfeld

1.1.1 Unternehmen: Lutz + Grub GmbH

Die Lutz + Grub GmbH ist ein renommiertes IT-Unternehmen, das sich auf die Entwicklung von Softwarelösungen spezialisiert hat. Mit Sitz in [Ort], [Land], hat das Unternehmen eine langjährige Erfahrung in der Bereitstellung innovativer IT-Dienstleistungen für verschiedene Branchen.

1.1.2 Unternehmensstruktur

Die Lutz + Grub GmbH besteht aus mehreren Abteilungen, darunter die Entwicklungsabteilung, die für die Konzeption und Umsetzung von Softwareprojekten verantwortlich ist. Die Abteilung wird von erfahrenen Fachleuten geleitet, die über umfassende Kenntnisse und Erfahrungen in den Bereichen Softwareentwicklung und -design verfügen.

1.1.3 Cioco Gaming Studio

Als Tochterunternehmen von Lutz + Grub GmbH ist das Cioco Gaming Studio spezialisiert auf die Entwicklung von Videospielen. Das Studio hat sich einen Namen gemacht durch die Kreation unterhaltsamer und innovativer Spiele für verschiedene Plattformen.

1.1.4 Projektteam

Das Projekt "Snake" wird von einem engagierten Entwickler geleitet und umgesetzt. Als Einzelentwickler trägt er die Verantwortung für den gesamten Entwicklungsprozess des Spiels.

1.1.5 Marktkontext

Das Projekt „Snake" findet in einem dynamischen Marktumfeld statt, das von ständigen technologischen Fortschritten und sich ändernden Spielererwartungen geprägt ist. Es ist wichtig, dass das Spiel den aktuellen Trends entspricht und gleichzeitig einzigartig ist, um sich in diesem wettbewerbsintensiven Markt zu behaupten.

1.2 Spielerwünsche

1.2.1 Nostalgie und Tradition

Die Spieler von "Snake" sind von der Nostalgie des Originalspiels aus den frühen Tagen der Videospiele geprägt. Sie schätzen die Einfachheit und den Suchtfaktor des klassischen Spiels und wünschen sich eine moderne Interpretation, die den Charme und die Tradition des Originals bewahrt.

1.2.2 Zugänglichkeit und Benutzerfreundlichkeit

Spieler legen Wert auf eine benutzerfreundliche Benutzeroberfläche und einfache Bedienung. Sie erwarten eine intuitive Steuerung, um das Spiel sofort genießen zu können, ohne lange Lernkurven oder komplexe Anleitungen durchlaufen zu müssen.

1.3 Projektziele und Vision

Das Ziel des Projekts „Snake" ist es, eine überarbeitete Version des klassischen Spiels zu entwickeln, die die Spieler mit modernem Design und verbesserten Graphics anspricht. Die Vision ist es, die Nostalgie des Originalspiels mit zeitgemäßen Elementen zu verbinden und eine neue Generation von Spielern zu begeistern.

1.4 Ist-Zustand

1.4.1 Aktuelle Version von Snake

Die aktuelle Version des Snake-Spiels, das von "Cioco Gaming Studio" entwickelt wurde, basiert auf älterer Technologie und Designkonzepten. Obwohl das Spiel immer noch einen gewissen Reiz hat, insbesondere für Nostalgiker, entspricht es nicht mehr den modernen Standards und den Erwartungen der heutigen Spieler.

1.4.2 Technologische Einschränkungen

Die vorhandene Version des Spiels ist technisch begrenzt und nutzt veraltete Technologien, was sich negativ auf die Leistung, Grafikqualität und Benutzererfahrung auswirkt. Die begrenzte Flexibilität und Skalierbarkeit der vorhandenen Plattform erschweren es, das Spiel auf verschiedene Geräte und Plattformen zu portieren und damit eine breitere Zielgruppe anzusprechen.

1.4.3 Feedback von Spielern

Das Feedback von Spielern und Community-Mitgliedern hat gezeigt, dass es Bedarf an Verbesserungen und Aktualisierungen gibt. Spieler haben ihre Wünsche und Erwartungen in Bezug auf das Design und die Funktionalität des Spiels geäußert und hoffen auf eine überarbeitete Version, die ihren Anforderungen besser entspricht.

1.4.4 Wettbewerbsumfeld

Im Hinblick auf das Wettbewerbsumfeld ist es wichtig anzumerken, dass es eine Vielzahl von Snake-ähnlichen Spielen gibt, die auf verschiedenen Plattformen verfügbar sind. Einige dieser Spiele bieten innovative Funktionen und ansprechende Grafiken, was den Druck auf "Cioco Gaming Studio" erhöht, mit ihrer neuen Version des Spiels zu überzeugen und sich von der Konkurrenz abzuheben.

1.4.5 Marktpotenzial

Trotz der bestehenden Herausforderungen und des Wettbewerbsumfelds besteht ein großes Marktpotenzial für eine überarbeitete Version von Snake. Das Spiel hat eine treue Fangemeinde und ist bei Spielern aller Altersgruppen beliebt. Durch die Modernisierung des Spiels besteht die Möglichkeit, das Interesse an Snake zu revitalisieren und eine neue Generation von Spielern anzusprechen.

1.5 Soll-Zustand

1.5.1 Neues Design und Grafik

Die überarbeitete Version von Snake wird ein modernes Design und verbesserte Grafiken aufweisen, die die Spielerfahrung bereichern und das Spiel visuell ansprechender machen. Durch die Verwendung zeitgemäßer Designkonzepte und hochwertiger Grafiken streben wir danach, das Spiel auf ein neues Niveau zu heben und die Spieler zu begeistern.

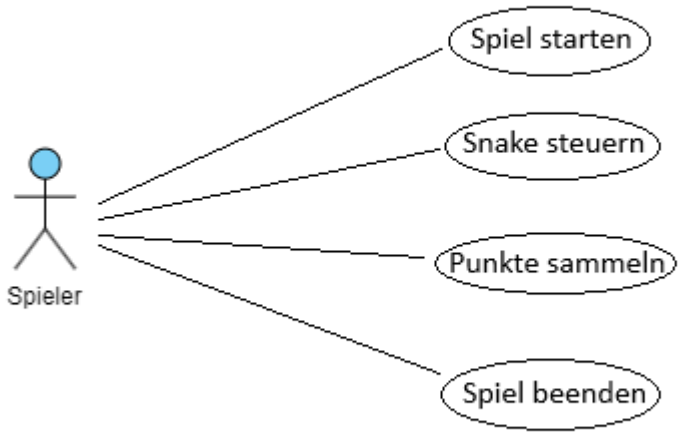
1.5.2 Plattformübergreifende Komplatibilität

Um eine breitere Zielgruppe anzusprechen, wird die überarbeitete Version von Snake auf verschiedenen Plattformen verfügbar sein, einschließlich mobiler Geräte, Tablets und PCs. Durch die Optimierung des Spiels für verschiedene Bildschirmgrößen und Betriebssysteme streben wir danach, eine nahtlose Spielerfahrung auf allen Geräten zu gewährleisten.

1.5.3 Fortlaufende Aktualisierungen und Support

Nach der Veröffentlichung der überarbeiteten Version von Snake werden wir kontinuierlich Updates und Verbesserungen bereitstellen, um das Spiel zu optimieren und auf das Feedback der Spieler einzugehen. Durch einen engagierten Support und regelmäßige Inhaltsaktualisierungen streben wir danach, das Spiel langfristig erfolgreich zu halten und die Spielerzufriedenheit zu gewährleisten.

Use Case Diagramm für Snake



2. Teilaufgaben

2.1 Spielmechanik definieren

In diesem Abschnitt werden die grundlegenden Regeln festgelegt, die das Spiel steuern. Das beinhaltet die Bewegung der Schlange und wie sie wächst, wenn sie Nahrung frisst. Die Spieler steuern die Schlange mit den Pfeiltasten, und das Spiel endet, wenn die Schlange die Spielfeldgrenzen erreicht oder mit sich selbst kollidiert.

2.1.1 Schlange und ihre Bewegung

Die Schlange besteht aus einer Liste von Eckpunkten, die ihre Position auf dem Spielfeld darstellen. Sie bewegt sich durch Verschieben dieser Punkte und wächst, wenn sie Nahrung frisst. Die Spieler steuern die Richtung der Schlange mit den Pfeiltasten. Das Spiel endet, wenn die Schlange die Spielfeldgrenzen erreicht oder mit sich selbst kollidiert. Wenn eines dieser Ereignisse eintritt, wird das Spiel beendet.

```
141     // tick
142     public static void tick(GraphicsContext gc) {
143         if (gameOver) {
144             gc.setFill(Color.DARKRED); // Game Over color
145             gc.setStroke(Color.BLACK);
146             gc.setLineWidth(2);
147             gc.setFont(Font.font("Impact", 40)); // Game Over font and size
148             gc.fillText("YOU DIED :'(\nGAME OVER", 150, 250);// Game Over position
149             gc.strokeText("YOU DIED :'(\nGAME OVER", 150, 250);
150             return;
151         }
152 
153         for (int i = snake.size() - 1; i >= 1; i--) {
154             snake.get(i).x = snake.get(i - 1).x;
155             snake.get(i).y = snake.get(i - 1).y;
156         }
157 
158         switch (direction) { // Game Over if the snake touches a border
159         case up:
160             snake.get(0).y--;
161             if (snake.get(0).y < 0) {
162                 gameOver = true;
163             }
164             break;
165         case down:
166             snake.get(0).y++;
167             if (snake.get(0).y > height) {
168                 gameOver = true;
169             }
170             break;
171         case left:
172             snake.get(0).x--;
173             if (snake.get(0).x < 0) {
174                 gameOver = true;
175             }
176             break;
177         case right:
178             snake.get(0).x++;
179             if(snake.get(0).x > width) {
180                 gameOver = true;
181             }
182             break;
183         }
184 
```

Der tick-Methode behandelt die Bewegung der Schlange auf dem Spielfeld sowie die Überprüfung, ob das Spiel beendet werden muss.

- *Überprüfen des Spielendes (Game Over):*
  Die gameOver-Variable wird überprüft, um festzustellen, ob das Spiel vorbei ist. Wenn das Spiel vorbei ist, wird eine "Game Over"-Nachricht auf dem Bildschirm angezeigt.

```
123         gameScene.addEventFilter(KeyEvent.KEY_PRESSED, key -> {
124             if (key.getCode() == KeyCode.UP) {
125                 direction = Dir.up;
126             }
127             if (key.getCode() == KeyCode.DOWN) {
128                 direction = Dir.down;
129             }
130             if (key.getCode() == KeyCode.LEFT) {
131                 direction = Dir.left;
132             }
133             if (key.getCode() == KeyCode.RIGHT) {
134                 direction = Dir.right;
135             }
136         });
```

- *Bewegung der Schlange:*
  Eine Schleife wird verwendet, um durch die Segmente der Schlange zu iterieren und ihre Positionen zu aktualisieren, indem sie jeweils um eins verschoben werden. Die Bewegungsrichtung der Schlange wird basierend auf Pfeiltasten festgelegt. Für jede Richtung (oben, unten, links, rechts) wird die Position des Kopfs der Schlange entsprechend angepasst. Es wird überprüft, ob die neue Position des Kopfs der Schlange die Spielfeldgrenzen erreicht hat. Wenn ja, wird das Spiel beendet.

```
191         // suicide
192         for (int i = 1; i < snake.size(); i++) { // Game Over if the snakes hits itself
193             if (snake.get(0).x == snake.get(i).x && snake.get(0).y == snake.get(i).y) {
194                 gameOver = true;
195             }
196         }
```

- *Kollisionserkennung zwischen der Schlange und sich selbst*
  In diesem Abschnitt durchläuft eine Schleife alle Teile der Schlange außer dem Kopf (Index 0). Für jedes Segment der Schlange wird überprüft, ob seine Position mit der Position des Kopfes übereinstimmt. Wenn ja, bedeutet dies, dass die Schlange mit sich selbst kollidiert ist, und das Spiel endet, indem die Variable gameOver auf true gesetzt wird.

2.1.2 Generierung von Nahrung

Für die Generierung der Nahrung wird ein zufälliger Ort auf dem Spielfeld gewählt. Nach dem Verzehr durch die Schlange erscheint die Nahrung an einer neuen zufälligen Position. Die Spieler erhalten Punkte für jede gefressene Nahrung. Die Position der Nahrung wird sorgfältig ausgewählt, um eine herausfordernde Spielerfahrung zu gewährleisten.

```
242     // food
243     public static void newFood() {
244         start: while (true){ // new food on random location foodX * foodY on the canvas (if there is no snake)
245             foodX = rand.nextInt(width);
246             foodY = rand.nextInt(height);
247 
248             for (Corner c : snake) {
249                 if (c.x == foodX && c.y == foodY) {
250                     continue start;
251                 }
252             }
253             foodColor = rand.nextInt(5); // new color
254             score ++;
255             break;
256         }
257     }
```

Die Methode newFood() generiert eine neue Nahrung für die Schlange auf dem Spielfeld. Zuerst wird eine zufällige Position für die Nahrung gewählt. Dann wird überprüft, ob diese Position bereits von der Schlange besetzt ist. Falls die Position von der Schlange besetzt ist, wird eine neue Position gewählt, bis eine freie Position gefunden wird. Sobald eine freie Position gefunden wurde, wird die Nahrung dort platziert. Der Punktestand wird erhöht, da die Schlange die Nahrung gefressen hat.

2.1.3 Punktezählung und Anzeige

Die Punktezählung und Anzeige verfolgen die Leistung des Spielers während des Spiels. Jedes Mal, wenn die Schlange Nahrung frisst, wird der Punktestand erhöht. Die Punkte werden in Echtzeit auf dem Bildschirm angezeigt, in der rechten oberen Ecke des Spielfeldes. Diese Positionierung ermöglicht es dem Spieler, seinen Fortschritt im Spiel leicht zu überwachen. Die Anzeige der Punktzahl trägt zur Motivation des Spielers bei und ermöglicht es ihm, seine Leistung im Spiel zu messen.

```
207         // score
208         gc.setFill(Color.WHITE); // color
209         gc.setFont(Font.font("Verdana", 15)); // font and size
210         gc.fillText("SCORE: " + score, 390, 30); // position
```

Die Methode setFill() wird verwendet, um die Farbe des Textes auf Weiß festzulegen. Mit setFont() wird die Schriftart "Verdana" und die Größe 15 festgelegt. Dann wird fillText() aufgerufen, um den Text "SCORE: " gefolgt von der aktuellen Punktzahl (score) an die rechte obere Ecke des Fenster zu zeichnen.

2.2 Benutzeroberfläche entwickeln

```
63          // Start Game window
64          StackPane root = new StackPane();
65          root.setStyle("-fx-background-color: radial-gradient(radius 100%, lightgreen, darkgreen);");
66
67          // button
68          Button startButton = new Button("Start Game");
69          startButton.setStyle("-fx-border-radius: 20px; -fx-background-radius: 20px; -fx-font-weight: bold; -fx-font-size: 20px;");
70          startButton.setPrefWidth(buttonWidth);
71          startButton.setPrefHeight(buttonHeight);
72          startButton.setOnAction(event -> startGame(primaryStage));
73
74          // button add
75          root.getChildren().add(startButton);
76
77          // scene Start Game window
78          Scene scene = new Scene(root, buttonWindowWidth, buttonWindowHeight);
```

Die Benutzeroberfläche des Spiels wird mithilfe von JavaFX erstellt. Ein Startfenster wird mit einem Startbutton angezeigt. Die Benutzeroberfläche wird ansprechend gestaltet, um dem Spieler ein angenehmes visuelles Erlebnis zu bieten und die Interaktion mit dem Spiel zu erleichtern. Beim Klicken auf den Startbutton wird das Spiel gestartet.

2.3 Grafiken und Animationen integrieren

```
90      private void startGame(Stage primaryStage) {
91          snake.clear();
92          speed = 10;
93          gameOver = false;
94          direction = Dir.left;
95          snake.add(new Corner(width / 2, height / 2));
96          snake.add(new Corner(width / 2, height / 2));
97          snake.add(new Corner(width / 2, height / 2));
98
99          newFood();
100
101         StackPane gameRoot = new StackPane();
102         Canvas c = new Canvas(width * cornerSize, height * cornerSize);
103         GraphicsContext gc = c.getGraphicsContext2D();
104         gameRoot.getChildren().add(c);
105
106         new AnimationTimer() {
107             long lastTick = 0;
108
109             public void handle(long now) {
110                 if (lastTick == 0) {
111                     lastTick = now;
112                     tick(gc);
113                     return;
114                 }
115                 if (now - lastTick > 1000000000 / speed) {
116                     lastTick = now;
117                     tick(gc);
118                 }
119             }
120         }.start();
```

Die Integration von Grafiken und Animationen wurde direkt im Spielcode vorgenommen. Die Animationen für die Bewegung der Schlange und andere Spielaktionen wurden mit dem AnimationTimer implementiert, um eine flüssige Darstellung zu ermöglichen. Diese Integration erfolgte innerhalb der startGame()-Methode, wo die Spielumgebung initialisiert wird.

2.4 Spiellogik programmieren

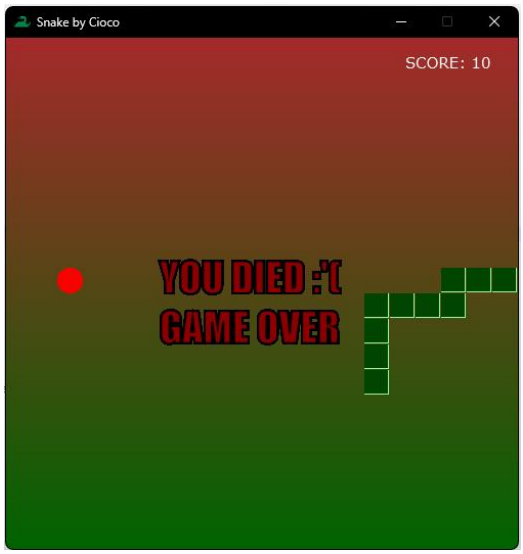
Die Spiellogik wurde direkt im Spielcode implementiert, um die Funktionalität des Spiels zu steuern. Die Spiellogik umfasst die tatsächliche Implementierung der zuvor definierten Spielmechaniken. Während in der Spielmechanik die grundlegenden Regeln festgelegt wurden, die das Spiel steuern, wird in der Spiellogik der Code geschrieben, um diese Regeln umzusetzen.

```
185         // eat food
186         if (foodX == snake.get(0).x && foodY == snake.get(0).y) { // snake grows
187             snake.add(new Corner(-1, -1));
188             newFood();
189         }
```

In diesem Spiellogik wird überprüft, ob die Position der Nahrung mit der Position des Kopfes der Schlange übereinstimmt. Wenn ja, wird die Schlange um ein neues Segment erweitert und neue Nahrung wird generiert.

2.5 Testen und Fehlerbehebung

Der Code wird gründlich getestet, um sicherzustellen, dass alle Spielmechaniken reibungslos funktionieren. Fehler werden durch Debugging-Techniken wie Konsolenausgaben und Überprüfung von Variablenwerten identifiziert und behoben. Der Testprozess wird wiederholt, bis das Spiel fehlerfrei läuft.



3. Durchführung

3.1 Analysephase

3.1.1 Modulfindung und Zielsetzung

In dieser Phase sind die erforderlichen Module identifiziert und die Ziele für das Projekt festgelegt. Dies umfasst die Analyse der bestehenden Struktur des Snake-Spiels sowie die Definition der Funktionalitäten und Features, die in der überarbeiteten Version implementiert werden sollen.

3.1.2 Ist-Analyse

Die Ist-Analyse beinhaltet die detaillierte Untersuchung der aktuellen Version des Snake-Spiels, einschließlich ihrer Funktionen, Benutzeroberfläche und Leistung. Es werden Stärken, Schwächen und Verbesserungspotenziale identifiziert, um als Grundlage für die Überarbeitung des Spiels zu dienen.

3.1.3 Soll-Analyse

In der Soll-Analyse sind die Ziele und Anforderungen des Snake-Spielprojekts definiert. Festlegung der Spielmechanik, einschließlich der Bewegung der Schlange, der Generierung von Nahrung und des Spielendes, Integration von Grafiken und Animationen, um das Spielerlebnis zu verbessern und Testen des Spiels auf Fehler und Behebung von gefundenen Fehlern.

3.2 Planungsphase

3.2.1 Projektplanung

In der Projektplanung sind Zeitrahmen festgelegt, Aufgaben aufgeteilt, Risiken identifiziert, Qualitätsstandards festgelegt und Ressourcen bereitgestellt, um den reibungslosen Ablauf des Snake Projekts sicherzustellen.

3.2.2 Kostenkalkulation

Die Kostenkalkulation für das Snake-Projekt umfasste eine detaillierte Aufschlüsselung der Ausgaben für verschiedene Projektressourcen wie Softwarelizenzen für Eclipse IDE und JavaFX, Entwicklungswerkzeuge, Hardwareanforderungen wie Computerressourcen und etwaige externe Dienstleistungen für Schulungen oder Beratung. Basierend auf diesen Informationen wurde ein Budgetplan erstellt, der die finanziellen Anforderungen des Projekts berücksichtigt und eine angemessene Ressourcenallokation sicherstellt.

3.3 Umsetzungs- und Testphase

3.3.1 Implementierung des Snake-Spiels

Während der Implementierung des Snake-Spiels wurde der bestehende Code analysiert und überarbeitet, um die Anforderungen des Projekts zu erfüllen. Neue Funktionen wie die Steuerung der Schlange, die Generierung von Nahrung, die Überprüfung auf Kollisionen und das Anzeigen von Punkten wurden hinzugefügt. Die Implementierung erfolgte in Java unter Verwendung des JavaFX-Frameworks für die Benutzeroberfläche und der Eclipse IDE für die Entwicklungsumgebung.

3.3.2 Testen und Fehlerbehebung

Während der Test- und Fehlerbehebungsphase wurde das Snake-Spiel auf Herz und Nieren geprüft, um sicherzustellen, dass es reibungslos funktioniert und alle Funktionen wie erwartet ausgeführt werden. Dabei wurden verschiedene Testfälle durchgeführt, um die Funktionalität der Schlange, die Generierung von Nahrung, die Kollisionserkennung und die Anzeige von Punkten zu überprüfen. Fehler und Probleme wurden identifiziert, analysiert und behoben, um eine optimale Spielerfahrung zu gewährleisten. Der Prozess des Testens und Debuggens wurde iterativ durchgeführt, bis das Spiel den Qualitätsstandards entsprach und bereit für die Auslieferung war.

3.4 Auswertungsphase

3.4.1 Qualitätskontrolle

Während der Qualitätskontrolle wurden verschiedene Aspekte des Snake-Spiels überprüft, um sicherzustellen, dass es den definierten Qualitätsstandards entspricht. Dazu gehörten Tests der Spielmechanik, der Benutzeroberfläche, der Grafiken und Animationen sowie der Gesamtperformance des Spiels.

3.4.2 Soll-Ist-Vergleich

Der Soll-Ist-Vergleich beinhaltet die Überprüfung, ob das entwickelte Snake-Spiel den ursprünglich definierten Anforderungen und Zielen entspricht. Dabei sind die tatsächlich erreichten Ergebnisse mit den geplanten Zielen verglichen, um Abweichungen zu identifizieren. Anhand dieses Vergleichs konnten mögliche Schwachstellen oder Verbesserungspotenziale ermittelt werden. Der Fokus lag darauf, festzustellen, ob das Spiel alle funktionalen und ästhetischen Anforderungen erfüllt und eine optimale Spielerfahrung bietet.

3.4.3 Übergabe

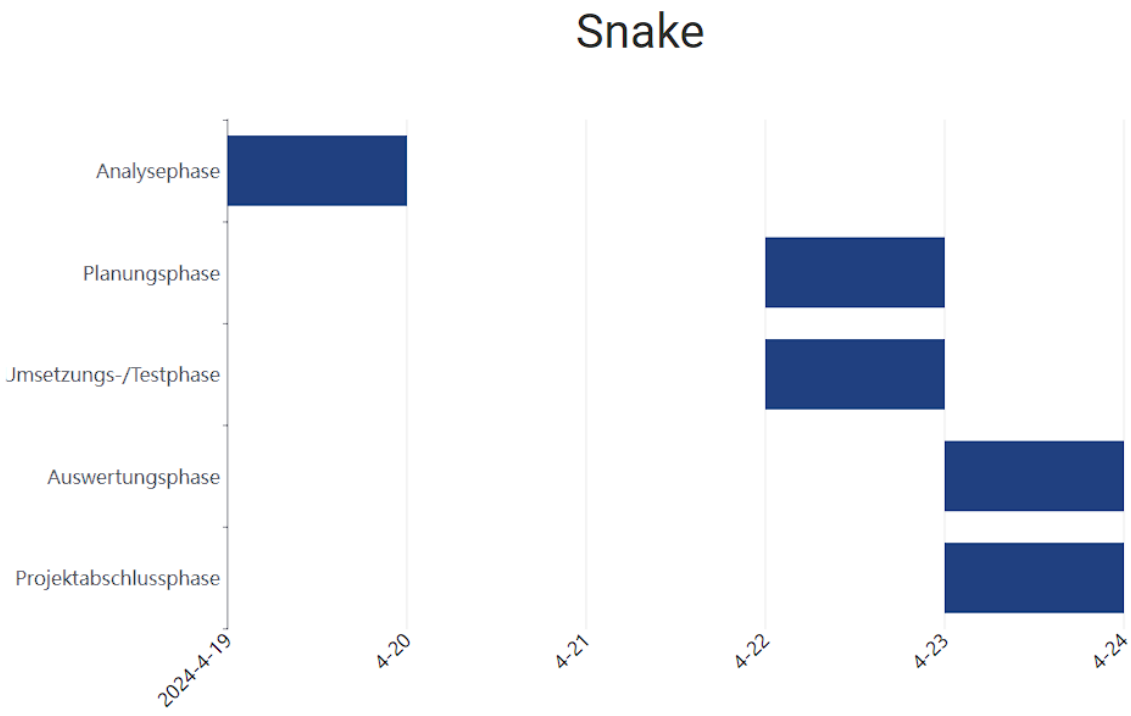
Die Übergabe markiert den Abschluss des Projekts und den Übergang des entwickelten Snake-Spiels an Lutz + Grub GmbH. In diesem Schritt sind alle erforderlichen Dokumentationen, Dateien und Ressourcen bereitgestellt, um sicherzustellen, dass das Spiel reibungslos übergeben werden kann. Dabei sind auch mögliche Wartungs- und Supportvereinbarungen festgelegt, um sicherzustellen, dass das Spiel auch nach der Übergabe erfolgreich betrieben und gewartet werden kann. Lutz + Grub GmbH war für die Koordination und Abwicklung des Projekts verantwortlich, während die Tochterfirma Cioco Gaming Studio die Spielentwicklung durchführte.

3.5 Projektabschlussphase

3.5.1 Projektdokumentation

In dieser Phase sind die gesamte Dokumentation des Projekts erstellt, Anleitungen, Codes und anderen relevanten Unterlagen. Diese Dokumentation dient als umfassendes Nachschlagewerk für zukünftige Referenzen und ermöglicht eine detaillierte Analyse des Projektsverlaufs und der Ergebnisse.

3.5.2 Gantt-Diagramm



4. Sonstiges

4.1 Fazit

Im Fazit meiner Snake-Projektdokumentation kann ich festhalten, dass die Entwicklung des Spiels erfolgreich verlaufen ist. Alle definierten Ziele wurden erreicht, und das Spiel wurde mit modernen Grafiken und Funktionen optimiert. Während des Projekts habe ich wertvolle Erfahrungen gesammelt und meine Fähigkeiten im Bereich der Softwareentwicklung weiterentwickelt. Für die Zukunft sehe ich noch Potenzial zur Verbesserung und Erweiterung des Spiels, um die Spielerfahrung noch weiter zu optimieren, beispielsweise durch neue Features wie das Neustarten nach Game Over und das Einführen von Highscores. Insgesamt war die Entwicklung des Snake-Spiels eine herausfordernde, aber äußerst lohnende Erfahrung.

4.2 Quellenverzeichnis

- *Programmiersprache:* Java (https://dev.java/)
- *IDE:* Eclipse IDE (https://eclipseide.org/)
- *Framework:* JavaFX (https://openjfx.io/)
- *Snake Logo:* freepik.com (https://www.freepik.com/icon/snake_427533)
- *Gantt-Diagramm:* Visual Paradigm Online
  (https://www.visual-paradigm.com/)