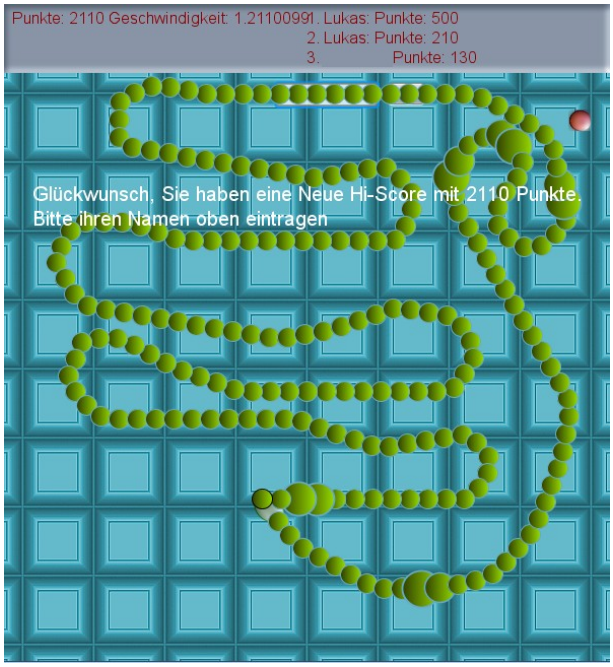
Entwicklung von Snake360 in JavaFX



1. Kurzbeschreibung

Es soll ein Snakeminigame entwickelt werden, bei welchem der Spieler eine Schlange über die Pfeiltasten der Tastatur oder über das Mausrad in alle Richtungen steuert. Wird ein Punkt gefressen, so wächst die Schlange. Daraufhin werden die Anzahl der Score und die Geschwindigkeit erhöht. Ziel des Spiels ist es, so viele Punkte wie möglich zu fressen, ohne das sich die Schlange selbst beißt. Zum Schluss wird die Score, falls diese hoch genug ist, in die Hi-Score Tabelle eingetragen .

2. Anforderungen an das System

Die Schlange soll über die Pfeiltasten oder das Mausrad in die jeweilige Richtung gesteuert werden. Wenn eine Kollision mit einem Punkt stattfindet, dann soll eine entsprechende Mund-Öffnen-Schließen-Animation erfolgen. Daraufhin soll für den Punkt eine neue zufällige Position berechnet werden. Des Weiteren sollen alle Glieder so skaliert werden, dass es so aussieht, alsob der Punkt von Kopf der Schlange, bis zum letzten Glieder durchgereicht wird. Erreicht der Punkt das letzte Glied, dann soll die Schlange um ein Glied erweitert werden, zudem sollen die Punkte und die Ablaufgeschwindigkeit erhöht werden.

Der Bewegungsablauf der Schlange soll folgendermaßen erfolgen. Erfolgt eine Richtungsänderung des Schlangenkopfes, so betrifft diese Änderung die Glieder erst, wenn diese an die selbe Position gelangt sind, wo der Schlangenkopf sich vorher befand. So entsteht eine Verzögerung der Glieder, je weiter sich ein Glied hinten befindet, desto später wird die Richtung dieses Gliedes aktualisiert.

Erfolgt eine Kollision mit dem Schlangenkopf und einem Glied der Schlange, dann ist das Spiel beendet. Daraufhin soll die erreichte Punktzahl gemessen werden und je nach höhe wird die Position für die Hi-Score Tabelle berechnet. Diese Tabelle soll nach Eintragung gesichert werden und beim nächsten Spielbeginn wieder geladen werden.

Die Spiel soll durch diverse Javafx Effekte begleitet werden.

Die Ablauf der Anwendung soll flüssig vonstatten gehen, um dies zu gewährleisten, muss getestet werden, welche und wieviele JavaFx Effekte zum Einsatz kommen. Zudem wird ein Augenmerk auf die Laufzeit der einzelnen Funktionen gelegt.

3. Systemarchitektur

HiScoreManager

In dieser Klasse wird der Speicherpfad festgelegt. Des Weiteren wird berechnet, ob die aktuellen Punkte für einen Eintrag in die Hi-Score Tabelle genügen. Falls dies der Fall ist, kann der Spielername eingetragen werden. Die Position wird anhand der Höhe der erzielten Punkte berechnet. Zudem bietet diese Klasse die Funktionalität um die Tabelle zu speichern und wiederherzustellen.

GamePoint

Die Klasse GamePoint ist von der JavaFx Klasse CustomNode abgeleitet. Wodurch der Zugriff auf zusätzliche Funktionalitäten wie der Translation, Skalierung usw. gestattet ist. Der Hauptgrund ist jedoch, dass durch die Ableitung der GamePoint in die Szene zum Rendern eingefügt werden kann. Die Klasse selbst besteht aus einem Circle Shape. Weiterhin enthält sie eine Funktion um die Position auf dem Spielfeld zufällig zu berechnen. Der GamePoint wird begleitet durch einen linearen Farbverlauf sowie einem Bloom-Effekt, welcher den Punkt durch eine weitere Routine aufblinken lässt.

Snake

Dies ist die Basisklasse für die Klassen SnakeHead und SnakeTile. Die Klasse Snake ist aus den im vorherigen Abschnitt genannten Gründen von der JavaFx Klasse CustomNode abgeleitet. Die Klasse enthält Funktionen, für die Umrandungsfarbe eines Shapes zu setzen. Darüber hinaus werden die Position und die Skalierung gesetzt und ausgelesen. Außerdem wird in Snake die Skalierungsanimation eines Gliedes Berechnet.

SnakeHead

Diese Klasse ist von Snake abgeleitet. Der Kopf besteht aus mehreren verbundenen interpolierten Linien. Einige Referenzpunkte werden genutzt um eine Maul-Öffnen-Schließen Animation zu erstellen. Außerdem wird die Drehung des Kopfes in dieser Klasse berechnet. Der Schlagenkopf wird durch einen JavaFX ein Lichteffekt verfeinert.

SnakeTile

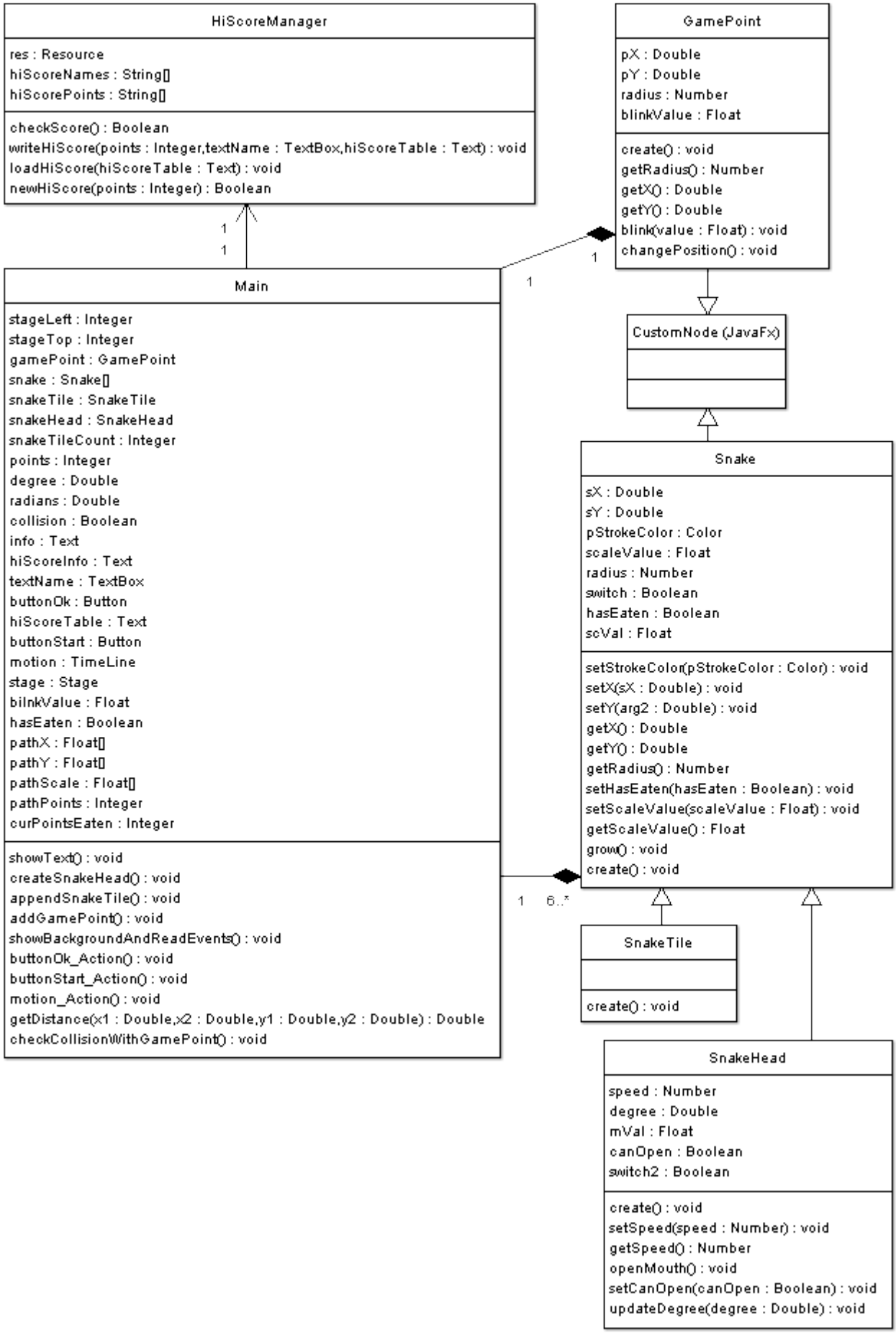
SnakeTile ist ebenfalls von der Klasse Snake abgeleitet. Sie repräsentiert das Glied der Schlange in Kreisform mit linearem Gradienten als Füllfarbe.

Main

Main repräsentiert, den Kern der Anwendung. In Main wird die grafische Szene sowie teil der Logik des Programms repräsentiert. In Main wird die Schlange mit zusätzlich 5 Glieder erstellt. Zusätzlich werden die Klassen GamePoint, HiScoreManager in Main erstellt und initialisiert. Main kommuniziert mit den Klassen und verwaltet diese für den Programmablauf. Es werden alle Komponenten mittels JavaFX aktualisiert und gezeichnet. Zu den JavaFX Komponenten gehören:

- Das Spielfeld mit Hintergrundbild.
- Die bewegte und animierte Schlange, sowie der Spielpunkt.
- Die Statusleiste mit Lichteffekt, in welcher die Punkte, die Bewegungs-geschwindigkeit der Schlange und die Hi-Score Tabelle angezeigt wird.
- Ein Knopf um das Spiel zu starten.
- Ein Textfeld um den Spielernamen für die Hi-Score Tabelle eintragen zu können
- Ein Knopf um dies zu bestätigen.

Die Architektur der Applikation wird im folgenden als Klassendiagramm dargestellt.



4. Realisierung

In diesem Kapitel werden der Ablauf der Anwendung und alle Aktionen erläutert.

4.1 Entwicklung des Spielfeldes

Im ersten Schritt wurde mit dem Malprogramm Gimp eine Hintergrundtextur gezeichnet. Anschließend wurde die Statusanzeige zum Anzeigen der Punkte, der Bewegungsgeschwindigkeit der Schlange und der Hi-Score Tabelle erstellt.

4.2 Erstellung der Schlange

Der Schlangenkopf besteht aus einer Anzahl an interpolierten Linien, welche geeignet miteinander verbunden sind. Davon sind zwei Ankerpunkte an eine Variable gebunden, welche in der Routine "openMouth()" bis zu einem bestimmten Wert hochgezählt wird und dann wieder herunter, dadurch entsteht kurz bevor die Schlange einen Punkt frisst eine Maul-Öffnen-Schließen Animation. Die Punkte bilden ein umschlossenes Gebiet, auf welches ein Lichteffekt berechnet wird. Der Befehl "rotate" vom Szeneknoten wird an den Winkel, welcher über die Benutzerinteraktion mit dem Mausrad oder der Tastatur gesteuert wird, gebunden. Dadurch wird der Schlagnekopf in die entsprechende Richtung gedreht.

Ein Schlangenglied ist durch einen Kreis mit einem linearen Farbverlauf gekennzeichnet. Ein Lichteffekt wurde nachträglich entfernt, da beim Testen der Anwendung, die Performance mit zunehmender Schlangengröße anfing darunter zu leiden.

Die Schlange enthält die Funktion "grow()" in welcher, der Skalierfaktor bis zu einem gewissen Wert wächst und dann wieder fällt. Wenn ein Spielpunkt gefressen wurde, erfolgt die Skalierung aller Glieder nacheinander.

Der Schlangen kopf und die Schlangenglieder werden in eine dynamische Liste und gleichzeitig in die grafische Szene eingefügt. Das Einfügen direkt in die Szene funktioniert dadurch, dass die Klasse Snake von der JavaFX Klasse CustomNode abgeleitet ist.

4.3 Spielablauf

Wenn der Benutzer auf den Startknopf drückt, veranlasst die Aktion, dass die Schlange mit fünf Glieder erstellt wird, zusätzlich werden der Spielpunkt zufällig platziert und die Gameloop gestartet. Falls eine Hi-Score Tabelle existiert, wird diese ebenfalls geladen. Die Bewegung der Schlange wird über das Mausrad oder den Pfeiltasten der Tastatur gesteuert. Wenn beispielsweise das Mausrad gedreht wird, dann werden abhängig von

der Drehrichtung 15 Grad erhöht oder erniedrigt. Anschließend wird der Winkel in Bogenmaß umgerechnet und an die Schlangenkopf-Klasse weitergegeben um den Kopf zu drehen. In der Spielschleife wird die Richtung der Schlange über eine Kreisberechnung ermittelt.

```
snakeHead.setX(snakeHead.getX() + snakeHead.getSpeed() * Math.cos(radians));
snakeHead.setY(snakeHead.getY() + snakeHead.getSpeed() * Math.sin(radians));
```

Es werden die X-, Y-Position und der Skalierfaktor in jeweils eine dynamische Liste eingetragen, um den Pfad der Schlange festzuhalten, oder besser gesagt um die "Spur" der Schlange aufzuzeichnen. Zugleich werden die Pfadpunkte erhöht.

```
insert snakeHead.getX() into pathX;
insert snakeHead.getY() into pathY;
insert snakeHead.getScaleValue() into pathScale;
pathPoints += 1;
```

Wenn die Schlange außerhalb vom Spielfeld kommt, tritt sie gespiegelt wieder ins Spielfeld ein.

Damit die einzelnen Schlangenteile, dem Pfad des Schlangenkopfes verzögert folgen, werden in einer Schleife, bereits vergangene Pfadpunkte ausgelesen und die Positionen der Glieder, sowie der Sklalierfaktor mit diesen aktualisiert.

```
snake[j].setX((pathX[pathPoints-(j*15)]));
snake[j].setY((pathY[pathPoints-(j*15)]));
snake[j].setScaleValue(pathScale[pathPoints-(j*15)]);
```

Wenn im weiteren Spielverlauf genügend Pfadpunkte vorhanden sind, sodass komplette Schlange und alle Glieder im Pfad abgedeckt sind, dann werden nach dem FIFO-Prinzip alte Pfadpunkte entfernt und neue hinzugefügt. Dies erweist sich als unabdingbar, da sonst die drei Listen immens wachsen würden. Wenn jedoch ein neues Glied an die Schlange hinzukommt, dann werden wieder 15 neue Pfadpunkte benötigt um auch dieses Glied im Pfad unterzubringen.

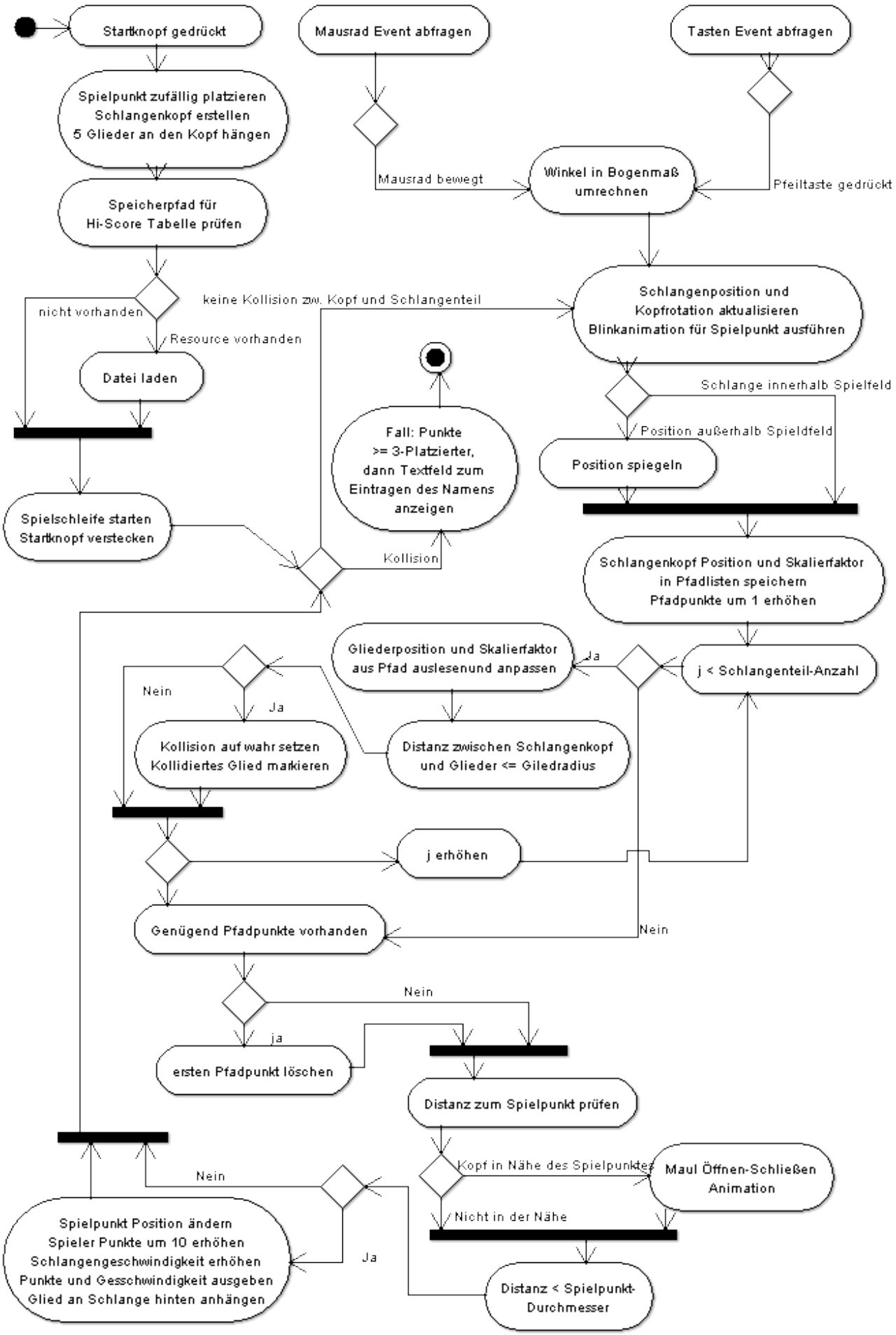
In einer weiteren Funktion "checkCollisionWithGamePoint()" wird die Distanz zwischen dem Schlangenkopf und dem Spielpunkt (GamePoint) gemessen. Befindet sich der Schlangenkopf unmittelbar vor dem GamePoint, dann wird die Funktion "openMouth()" des Schlangenkopfes aufgerufen, welche den Ablauf der Maul-Öffnen-Schließen Animation des Schlangenkopfes veranlasst. Des Weiteren wird die Variable "hasEaten"

auf wahr gesetzt, wenn tatsächlich eine Kollision mit dem Spielpunkt entsteht. Diese bewirkt den Aufruf der Funktion "grow()", in welcher der Skalierfaktor bis zu einem bestimmten Punkt hochgezählt und dann wieder heruntergezählt wird. Zugleich wird dieser Faktor in die "pathScale" Liste eingetragen, sodass später jedes Glied in einer Schleife, den Skalierfaktor aus dem Pfad auslesen kann und mit diesem Faktor skaliert wird. So entsteht der Eindruck, alsob der GamePoint vom Maul der Schlange bis zum letzten Glied wandert. Wenn das letzte Glied skaliert wurde, dann wird ein neues Schlangenteil hinten angehängt. Des Weiteren werden die Spielerpunkte, die Geschwindigkeit der Schlange erhöht und der Status aktualisiert.

Außerdem wird in der Schleife die Distanz zwischen dem Schlangenkopf und der Glieder gemessen, wenn diese kleiner als der Radius eines Gliedes wird, dann ist die Schlange mit sich selbst kollidiert und das betroffne Glied wird markiert. Das Spiel ist dann beendet, es wird die aktuelle Score mit den Punkten in der Hi-Score Tabelle gemessen. Sind diese hoch genug, dann erscheint ein Textfeld, in welches der Benutzer seinen Namen eintragen kann. Der aktuelle Punktestand wird mit dem in der Tabelle verglichen und daraus die Punkteplatzierung errechnet. Die Tabelle kann nur 3 Einträge enthalten, sodass der Drittplatzierte aus der Tabelle gelöscht wird, falls die aktuellen Punkte gleich diesem oder höher sind und der Spielername sich von dem in der Tabelle unterscheidet.

Es wurde mit dem Media Objekt von JavaFX experimentiert um Sounds abzuspielen. Wenn ein GamePoint gefressen wurde, sollte ein entsprechender Sound abgespielt werden, leider entstand dadurch eine Verzögerung. Außerdem wurden nur spezielle Wave-Formate angekommen. Daher wurde die Erweiterung durch Sounds wieder entfernt.

In der sich darunter befindlichen Abbildung wird der Ablauf des Spiels durch ein Aktivitätsdiagramm veranschaulicht.


Startknopf gedrückt
Mausrad Event abfragen
Tasten Event abfragen
Spielpunkt zufällig platzieren
Schlangenkopf erstellen
5 Glieder an den Kopf hängen
Mausrad bewegt
Winkel in Bogenmaß umrechnen
Pfeiltaste gedrückt
Speicherpfad für Hi-Score Tabelle prüfen
nicht vorhanden
keine Kollision zw. Kopf und Schlangenteil
Schlangenposition und Kopfrotation aktualisieren
Blinkanimation für Spielpunkt ausführen
Resource vorhanden
Datei laden
Schlange innerhalb Spielfeld
Position außerhalb Spieldfeld
Fall: Punkte >= 3-Platzierter, dann Textfeld zum Eintragen des Namens anzeigen
Position spiegeln
Spielschleife starten
Startknopf verstecken
Kollision
Schlangenkopf Position und Skalierfaktor in Pfadlisten speichern
Pfadpunkte um 1 erhöhen
Gliederposition und Skalierfaktor aus Pfad auslesenund anpassen
Ja
j < Schlangenteil-Anzahl
Nein
Ja
Kollision auf wahr setzen
Kollidiertes Glied markieren
Distanz zwischen Schlangenkopf und Glieder <= Giledradius
j erhöhen
Genügend Pfadpunkte vorhanden
Nein
Nein
ja
ersten Pfadpunkt löschen
Distanz zum Spielpunkt prüfen
Kopf in Nähe des Spielpunktes
Maul Öffnen-Schließen Animation
Nein
Nicht in der Nähe
Spielpunkt Position ändern
Spieler Punkte um 10 erhöhen
Schlangengeschwindigkeit erhöhen
Punkte und Gesschwindigkeit ausgeben
Glied an Schlange hinten anhängen
Ja
Distanz < Spielpunkt-Durchmesser