Spielidee

Snake ist ein klassisches Spiel, das sich mit minimalen Grafiken darstellen lässt. Das Spiel passt daher zu den eingeschränkten 24x48 Display.

Der Spielverlauf ist simpel und einfach zu verstehen. Die Spielelemente sind als LEDs auf einer Fläche, die als Spielfeld dient, dargestellt.

Der Spieler steuert eine „Schlange“, die nach „Nahrung“ sucht. Wenn die Schlange die Nahrung berührt, wird die Schlange länger. Die Bewegung der Schlange wird mit den Pfeil-Tasten (up, down, left, right) gesteuert.

Das Spiel endet, wenn die Schlange die Wände oder ihren eigenen Körper berührt. Eine typische Spielrunde von Snake variiert nach dem gewählten Schwierigkeitsgrad.

Der Schwierigkeitsgrad, den man zu Beginn des Spiels festlegt, beeinflusst dabei die Geschwindigkeit der Bewegung von der Schlange auf dem Spielfeld. Um die Schlange auf hohen Geschwindigkeiten präzise steuern zu können, braucht der Spieler eine gute Hand-Augen-Koordination.

Aufgrund des hohen Bekanntheitsgrads, der simplen Darstellungsmöglichkeit auf dem gegebenen „LED-Spielfeld“ und dem intuitiven Gameplay haben wir entschieden, Snake zu implementieren.

Spielbeschreibung

Bei unserer Implementation von Snake gilt folgendes:

- Die Spielfeldgrenzen sind mit roten Punkten markiert.
- Der Hintergrund, auf welchem Essen erscheinen kann und die Schlange sich bewegen kann, ist mit grauen Punkten markiert.
- Die Schlange wird mit grünen Punkten dargestellt.
- Die Nahrung ist ein gelber Punkt.

Am Beginn des Spiels kann der/die Spieler/in seinen/ihren gewünschten Schwierigkeitsgrad auswählen:

- Easy (200 Millisekunden zwischen den Ticks)
- Medium (150 Millisekunden zwischen den Ticks)
- Hard (100 Millisekunden zwischen den Ticks)
- Hardcore (50 Millisekunden zwischen den Ticks)

Nach der Wahl des Schwierigkeitsgrads wird ein Countdown auf dem Bild gezeigt.

Nach dem Countdown beginnt sich die Schlange automatisch zu bewegen.

Die Richtung der Bewegung kann der Spieler mit den Pfeil-Tasten steuern.

Wenn die Schlange die Nahrung berührt, verschwindet die Nahrung und taucht wieder auf einen zufälligen validen Position auf.

Wenn das Spiel endet, erscheint eine Game Over Message-Box auf dem Bildschirm.

In einer weiteren Message-Box, sowie in der Konsole wird eine High-Score-Tabelle ausgegeben. Diese zeigt die High-Scores für den jeweils gewählten Schwierigkeitsgrad an. Diese wird spielübergreifend im Verzeichnis „C:\Users\user\AppData\Roaming\Snake“ bzw. bei anderen Betriebssystemen in „/home/user/Snake“ gespeichert.

In einer letzten Message-Box kann der Spieler entscheiden, das Spiel entweder neu zu starten oder zu beenden.

Kurze Beschreibung des Programmablaufs

SnakeMain.java ist die **main** Klasse des Programms.

InternalLedGameThread.run() wird erstens aufgerufen, gefolgt von der Erstellung eines neuen Objektes von der **GameBoard** Klasse.

Der Konstruktor von **GameBoard** ruft danach die Methoden ***initializeField()*** und ***initializeGame()*** sowie erstellt ein neues Objekt von **Timer**.

initializeGame() ist für folgendes zuständig:

- die Darstellung der Schwierigkeitsgradwahl
- die Initialisierung der Spielobjekte [durch Aufruf von ***initializeGameObjects()***]
- den Countdown

initializeField() erstellt das Spielfeld.

Snake.java beinhaltet den Programmcode für die automatische und spielergesteuerte Bewegung der Schlange. Die Schlage wird durch einen ArrayList implementiert. Ein Tile wird immer am vorne hinzugefügt und hinten wird eines entfernt.

FieldTile ist eine abstrakte Klasse mit den Unterklassen **BackgroundTile**, **BorderTile** und **EatTile**.

Jedem FieldTile im Array wird eine Farbe, bestehend aus einem short für Rot, einem für Grün und einem für blau, zwei Bytes für die jeweilige x und y Position auf dem Spielfeld, sowie ein Wert des enums CollisionType zugeordnet. Die enums sind:

- death (rote oder grüne LED)
- food (gelbe LED)
- nothing (graue LED)
- countdown (schwarze LED)

Die Methode ***calculateNextTile()*** in **GameBoard.java** ist für die Moment-zu-Moment-Gameplay zuständig. Die bestimmt, welche Aktionen durchzuführen als Reaktion auf die Handlung des Spielers.

UiFieldTranslator.java wandelt die FieldTiles in einem short Array um und verwendet die Methode ***InternalLedGameThread.showImage()***, um das finale Bild darzustellen.

Persönliche Kommentare zum Projekt und Zusammenfassung

Miguel Meindl:

Ich programmiere inzwischen schon seit über 6 Jahren. Angefangen habe ich mit Java. Ich war erstaunt, wie sehr ich an Erfahrung dazugewonnen habe, seit dem ich das letzte Mal diese Programmiersprache benutzt habe.

Auch musste ich leider feststellen, wie umständlich manche Dinge im Vergleich zu C#, meiner jetzigen Hauptsprache, umzusetzen waren. Man ist verwöhnt mit Features, wie Methoden Referenzen zu übergeben, um sich einen zweiten Rückgabewert zu erschleichen.

Teamarbeit und Organisation waren bisher für mich aufgrund meiner vorangegangenen Ausbildung zwar keine Fremdwörter, aber es ist immer wieder schön mit fremden Menschen zu arbeiten.

Muhammad Daniel:

Wie organisiere ich meinen Code? Wie verwende ich Klassen, sodass sie die Übersichtlichkeit dienen? Auf welche Weise kann das Spiel neu gestartet werden, ohne dass das ganze Programm beendet werden muss?

All diese Fragen (und andere) wurden mehr oder weniger während des Ablaufs dieses Projekts beantwortet.

Dass das Projekt eine Gruppenarbeit war, ist übrigens toll. Miguel hat mir sehr geholfen, wenn ich Probleme bei der Implementierung hatte. Seine gegebenen Feedbacks und Verbesserungsvorschläge habe ich sehr wertvoll gefunden.

Im Allgemeinen war dieses Projekt eine echt gute Lernerfahrung.

Zusammenfassung:

Im großen und Ganzen denken wir, dass das Projekt gelungen ist und wir viel daraus mitnehmen konnten. Es war angenehm im Umfang und wir hatten Spaß dabei, sowohl beim Spielen, als auch beim Erstellen.

Tabellarische Aufstellung der Arbeitsaufteilung

Mitglied	Zuständigkeiten	Geleistete Arbeitsstunden
Miguel Meindl	Implementation: - Snake.java - GameBoard.java - UiFieldTranslator.java - HighScoreIO.java Dokumentation: - Spielanleitung - Kurze Beschreibung des Programmablaufs	Implementation: 13 Stunden Dokumentation: 2 Stunden Insgesamt: 15 Stunden
Muhammad Daniel Bin Mohd Khir	Implementation: - FieldTiles.java und Unterklassen - Color.java - CollisionType.java - Countdown.java - Direction.java Dokumentation: - Spielidee & Spielbeschreibung - Klassendiagram	Implementation: 14 Stunden Dokumentation: 3 Stunden Insgesamt: 17 Stunden

GameBoard

-gameField: FieldTile[][]
-snake: Snake
-TIMER: Timer
-score: int
-DIFFICULTIES: String[]
-HIGHSCOREFRESHER: HighscoreIO
-TRANSLATOR: UiFieldTranslator
-runningFirstTime: boolean
-userDifficulty: 1

-initializeGame(): void
-initializeField(): void
-initializeGameObjects(): void
-mapMillisecondsToDifficulty(): int
+actionPerformed(ActionEvent): void
-proceddKeyBoardInput(): void
-calculateNextTile(byte, byte): void
-gameOver(): void
-restart(): void
-placeFood(): void

FieldTile

-collision: CollisionType
-color: Color
-POSX: byte
-POSY: byte

+change(CollisionType) void
+getColor(): Color
+setColor(Color): void
+getCollision(): CollisionType
+getPosX(): byte
+getPosY(): byte

BackgroundTile

BorderTile

EatTile

Snake

-SNAKETILES: List<FieldTile>
-facing: Direction

+move(FieldTile): void
+eat(FieldTile): void
-snakeifyTile(FieldTile): void
+getFacing(): Direction
+getHead(): FieldTile
+setFacing(Direction): void

Color

-RED: short
-GREEN: short
-BLUE: short

+getRed(): short
+getGreen(): short
+getBlue(): short

Countdown

-ABSPOSX: int
-ABSPOSY: int
-GAMEFIELD: FieldTile[][]
-TRANSLATOR: UiFieldTranslator
-stage: int
-TIMER: Timer
-GAMEBOARDTIMER: Timer
-COUNTDOWNCOLOR: Color

+doCountdown(): void
+actionPerformed(ActionEvent): void
-firstStage(Color): void
-secondStage(Color): void
-thirdStage(Color): void
-forthStage(Color): void
-fifthStage(Color): void

HighscoreIO

+refreshHighscore(String, int): String
-writeFile(List<Integer>, String): void
-readFileIfExists(List<Integer>, String): void
-createDirectoryIfNotExists(String): void

<enumeration> Direction

left
right
up
down

UiFieldTranslator

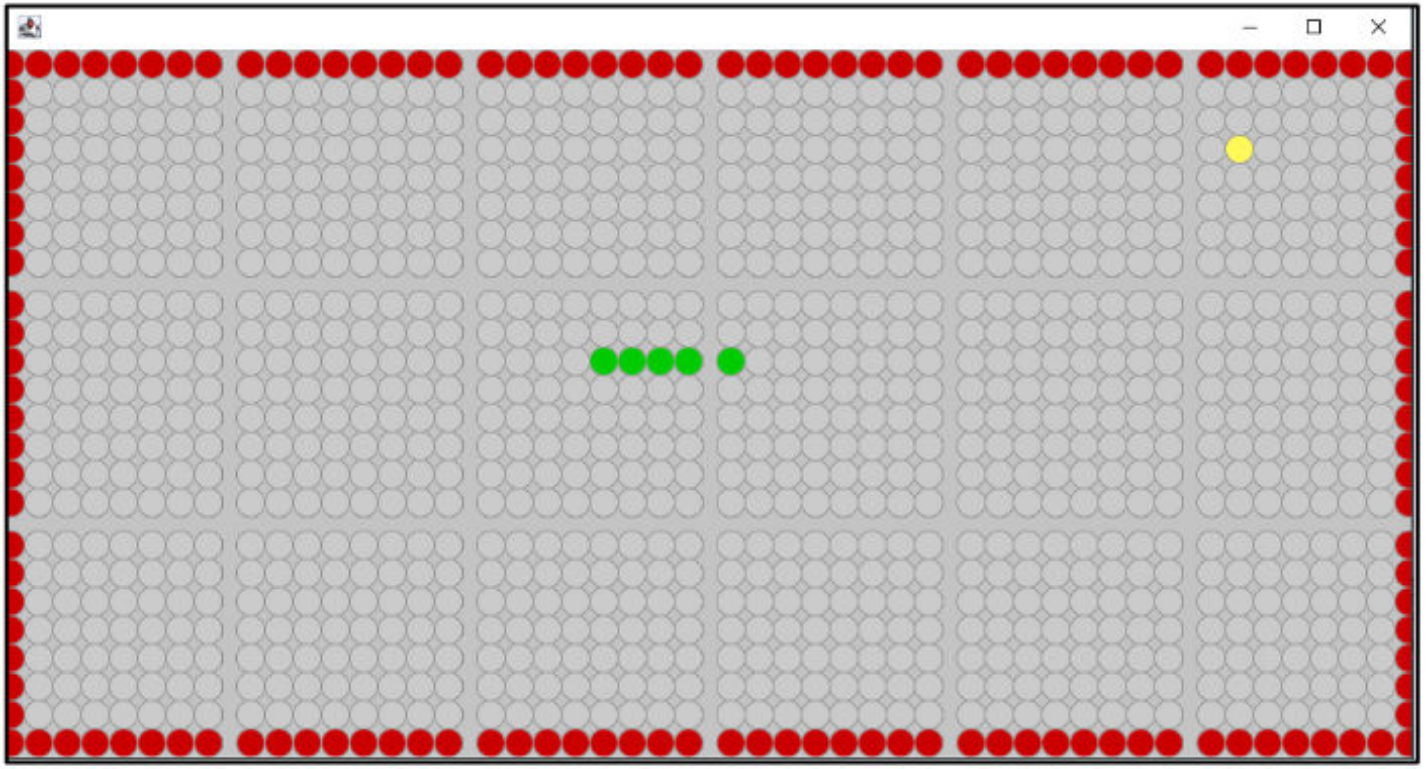
+translateToShortArray(FieldTile[][]): void

<enumeration> CollisionType

death
food
nothing
countdown

+mapColorFromCollision(): Color

Anmerkungen:
Konstruktoren werden weggelassen.
Die Pfeile für die drei Tiles sollen einen nicht ausgefüllten Kopf haben.



Spielanleitung zu Snake:

1. Ziel:

Als eine Schlange müssen Sie nach Nahrung suchen.
Wenn Sie Nahrung essen, werden Sie immer länger!
Vermeiden Sie mit Ihren Kopf an die Wände oder Ihren eigenen Körper zu stoßen!
Streben Sie nach einem High-Score!

2. Steuerungen: Pfeil-Tasten

up = nach oben bewegen
down = nach unten bewegen
left = nach links bewegen
right = nach rechts bewegen

3. Bedeutung der Punkte

grün = die Schlange
gelb = die Nahrung
rot = die Wände