Bericht Programmieren II Gruppenphase - Blöckchenspiel Gruppe 11

Jonas Plankert, Halil Yesilöz, Tim Schwetje, Erik Brinker, Vladislav Gornet
Datum:16.07.2023
Tutor:Jan Dukart

Im Rahmen der Gruppenphase des Moduls Programmieren 2 war es unsere Aufgabe ein Blöckchenspiel innerhalb von 6 Wochen in Java zu programmieren. Dabei sollte neben der Funktionsweise die Implementierung der graphischen Benutzeroberfläche (GUI) mittels JavaFX erfolgen. Konkret sollen folgende Basisregeln umgesetzt werden: Volle Reihen werden entfernt. Tetrominos können rotieren und haben Kollision. Es gibt ein oberes Limit, dessen Erreichen ein Game Over darstellt. Es müssen mindestens 5 verschiedene Tetrominos implementiert werden. Das GUI muss ein Hauptmenü enthalten, in welches man nach einem Game Over zurückkehrt und es muss auf beliebige Fenstergröße skaliert werden können, ohne dass sich das Seitenverhältnis ändert. Zusätzlich dazu muss ein lokaler 2- Spieler Modus implementiert werden, wobei entfernte Reihen dem anderen Spieler hinzugefügt werden.

Die wöchentlichen Aufgaben wurden zunächst in zwei getrennten Gruppen bearbeitet. Während die Spiellogik von Erik Brinker, Tim Schwetje und Halil Yesilöz implementiert wurde, widmeten sich Vladislav Gornet und Jonas Plankert der GUI. Die Motivation dahinter war die anfallende Aufgabenlast zu verteilen, sodass sich alle Gruppenmitglieder einem Thema hingeben konnten. Angesichts der kurzen Dauer von sechs Wochen, entpuppte sich dieses Verfahren als besonders zeiteffizient. Für die Implementierung der Spiellogik richteten wir uns jede Woche drei Grundproblemen, die wir zu Beginn der Gruppenphase definiert haben: Wie erstellen wir das Spielfeld? Wie erstellen wir die Tetromino-Steine? Wie können wir eine flüssige Bewegung der Steine garantieren? Dem Spielfeld liegt ein zwei-dimensionales GameBoard Array zugrunde, wobei GameBoard eine Enum-Klasse ist, die die Wertemöglichkeiten “AIR”, “PLAYER” und “SET” implementiert. Jeder Eintrag des Arrays wird demnach mit den Blöcken des aktuellen Tetromino-Steins gefüllt (“PLAYER), mit bereits platzierten Steinen (“SET”) oder er ist nicht befüllt (“AIR”). Es gibt insgesamt 7 unterschiedliche Tetromino-Steine, die jedoch alle dieselben Methoden benutzen. Aus diesem Grund hat es sich angeboten auf ein Tetromino-Interface zurückzugreifen. In diesem Interface werden die einzelnen Methoden wie turn, goRight, drop... bereitgestellt, die dann in den einzelnen Tetromino Klassen implementiert werden können. Die flüssige Bewegung erfolgt dadurch, dass bei jedem “Move” des Tetrominos in jegliche Richtung zunächst die Felder, die mit PLAYER belegt sind in AIR umgewandelt werden, der aktuelle Stein an der Stelle entfernt wird. Anschließend wird die gewünschte Bewegung durchgeführt und die neuen Felder mit PLAYER gesetzt. Bewegungen können nur passieren, falls sich keine gesetzten Steine an der neuen Position befinden oder die neue Position außerhalb des Spielfelds bzw. Arrays befinden.

Für den Single- und Multiplayer haben wir zwei eigene und voneinander unabhängige Controller entwickelt. Der Vorteil besteht darin die beiden Spielmodi voneinander getrennt zu bearbeiten und auf diese Weise strukturierter arbeiten zu können. Andererseits wiederholen sich dadurch bestimmte Codeabschnitte an manchen Stellen. Nichtsdestotrotz überwiegen die Vorteile. Aus denselben Gründen haben wir einen separaten Controller für das Hauptmenü geschrieben.

Das Game Over des Multiplayers haben wir so implementiert, sodass es keinen klaren Sieger gibt. Man kann selber entscheiden wer gewonnen hat, entweder die Person die länger überlebt oder die Person, welche den höheren Score erzielt hat. Dies wollten wir den Spielern jedoch nicht vorschreiben, weshalb der Game Over Screen sehr simpel gehalten wurde und nur die nötigen Informationen bereitstellt.

Bezüglich der Implementierung der UpdateGrid Methoden haben wir zwei voneinander unabhängige Methoden implementiert, da die Spiele ebenfalls voneinander getrennt sind. Außerdem war es für uns einfacher und effektiver zwei verschieden Methoden zu verwenden, welche jeweils genau auf ein Grid zugeschnitten wurden, anstatt eine Methode, welche beide Grids zur selben Zeit aktualisiert. Andernfalls könnten leichter Fehler entstehen, da diese immer nur zur gleichen Zeit aktualisiert werden würden.
Zudem haben wir in einigen Methoden an bestimmten Stellen ein Platform.runLater(() -> {}) implementiert, damit manche Befehle auf dem JavaFX Application Thread anstatt den Main-Thread ausgeführt werden, da ansonsten Fehler im Programm auftreten könnten.

Bis zur Fertigstellung des Programms wurden folgende Aufgaben in jeder Woche bearbeitet:

(Woche 1)	Einrichtung und Einarbeitung in Git/GitLab, Javafx Einigung auf gemeinsame IDE (IntelliJ IDEA) Bearbeitung von Assignment_6 und Erstellung erster Codeelemente für Blöckchenspiel
Woche 2	Anfang GUI und Spiellogik: Spiellogik: Erstellung des Spielfelds als 2d-Array Erstellung der Tetromino Blöcke Bewegung der Blöcke nach rechts, links und unten implementiert Drehoption auch eingefügt GUI: Erstellung des Hauptmenüs mit Buttons zur Auswahl von Exit, Single- und Multiplayer. Durch Drücken der Escape Taste gelangt man ins Hauptmenü, leichter Wechsel zwischen Single- und Multiplayer ist auch möglich Hintergrundbild, Icon und Titeltext eingefügt Skalierung auf beliebige Fenstergröße ist möglich, ohne dass sich das Seitenverhältnis des Spielfelds ändert
Woche 3	 Spiellogik: Tetrominos drehen sich nicht mehr außerhalb des Arrays, drehen nur möglich, wenn Tetromino darf Wenn eine Reihe voll, ist fallen alle Blöcke darüber eine Zeile runter GUI: GUIs zusammengefügt und Fenstergröße wurde angepasst Exit Funktion implementiert und Design überarbeitet Zwei weitere Szenen für Single- und Multiplayer Paths überarbeitet
Woche 4	Spiellogik:

	Blöcke können sich zusätzlich zu dem Rand nicht mehr in andere Blöcke drehen Timer implementiert, womit die Blöcke von alleine fallen und ein neuer Stein spawnt, sobald der aktuelle gelandet ist Random stein wird gespawnt, sobald einer gelandet ist Game over implementiert, womit das Programm stoppt, sobald kein weiterer Stein spawnen kann GUI: GUI mit der Spiellogik verknüpft 2 weitere Scenes für das Spielfeld hinzugefügt Tasteneingabe und Game over wird implementiert
Woche 5	Allgemein: Javadoc Kommentare hinzugefügt Spiellogik: Wallkick Funktion implementiert, wodurch das Drehen an den Enden des Gameboards möglich wird Zahlreiche Bug fixes bezüglich des Drehens Score implementiert GUI: Multiplayer implementiert Bugfixing, wo falsche Threads genutzt wurden Multiplayer Logik und GUI verknüpft Game over und Score screen implementiert GUI des Singleplayers verbessert

Im gesamten Prozess begegneten wir unterschiedlichen Schwierigkeiten. Zu Beginn war der Scene Builder uns unbekannt und musste von Grund auf erlernt werden, um damit vernünftig umgehen zu können. Zudem erhöht die Einteilung in zwei Gruppen die Effizienz, erwies sich jedoch bei der Verknüpfung von Spiellogik und GUI als Problem. Beide Gruppen mussten sich in den jeweils anderen Codeabschnitt einlesen, was viel Zeit in Anspruch nahm und die Verknüpfung erschwerte. Anders als in der Assignmentphase, war der Codeumfang bei Weitem größer und stellte beim Bugfixing eine Herausforderung dar. Das gemeinsame Arbeiten an einem Projekt mit Git war für alle Teammitglieder ungewohnt, stellte jedoch mit mehr Übung kein Problem mehr dar.

Insgesamt war das Projekt eine wertvolle Erfahrung, sowohl in Bezug auf die technische Umsetzung als auch auf das Teamwork. Durch die enge Zusammenarbeit konnten wir Hindernisse überwinden, kreative Lösungen finden und letztendlich ein erfolgreiches Ergebnis erzielen.

User Manual

Wenn das Programm gestartet wird, gelangt man in das Hauptmenü, wo der Spieler den Single- oder Multiplayer Button klicken kann, um eine Runde Tetris zu starten. Beim Singleplayer erscheint nun das rechteckige Gameboard samt Score, einem Play Button, um das Spiel zu starten und einem Back Button, um ins Hauptmenü zurückzukehren. Mit den Tasten A, S und D lässt sich das Tetromino nach links, rechts oder nach unten bewegen, währen die W Taste das Tetromino dreht. Volle Reihen werden entfernt und mit 100 Punkten belohnt, während jeder gesetzte Stein den Score um 10 Punkte erhöht. Der Spieler versucht nun durch geschicktes Drehen und Legen der Steine möglichst viele Reihen zu füllen und die Decke des Gameboards nicht zu erreichen, da dies ein GameOver auslöst. Analog dazu werden im Multiplayer Modus zwei Gameboard angezeigt, wobei das rechte Gameboard mit den Tasten I, J, K, L bedient wird. Die Besonderheit im Multiplayer liegt darin, dass entfernte Reihen dem Gegenspieler hinzugefügt werden.