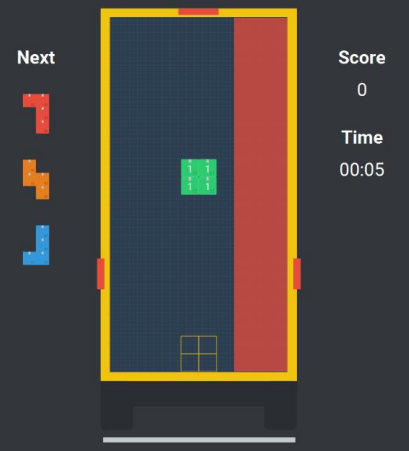
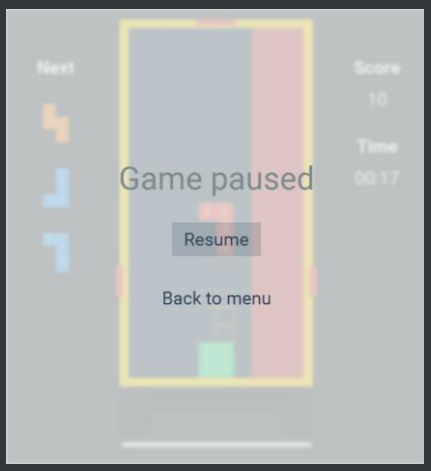
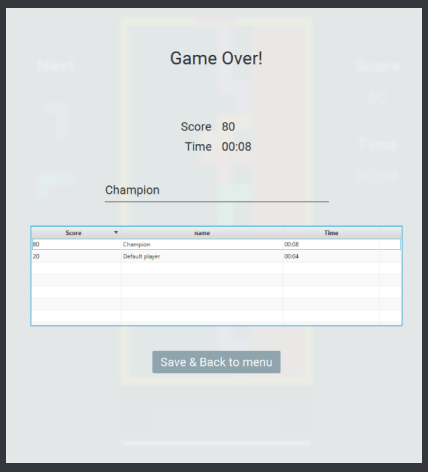
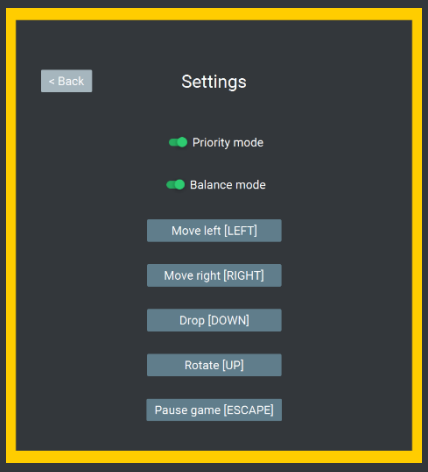
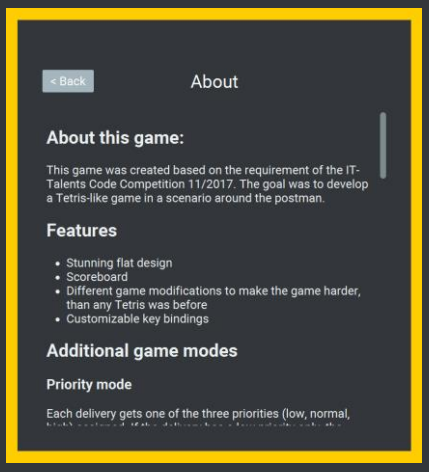
Postman Tetris











Setup

Um das Spiel starten zu können, müssen folgende Bedingungen erfüllt sein:

- OS: plattformunabhängig
- Java 8 muss installiert sein (Das Spiel ist noch nicht mit Java 9 getestet worden.)
- Das Spiel wurde für FullHD-Bildschirme optimiert. Es kann sein, dass es bei anderen Bildschirmauflösungen zu Darstellungsproblemen kommen kommt.
- Das Spiel verwendet die Schriftart ‚Roboto'. Wenn diese Schriftart nicht auf dem PC installiert ist, wird eine Standard-System-Font geladen, wodurch das Aussehen von den Screenshots abweichen kann.

Programmstart:
Zum Starten des Spiels muss lediglich die JAR-Datei per Doppelklick ausgeführt werden.

Entwicklung:
Für die Entwicklung wurde die Entwicklungsumgebung IntelliJ verwendet. Zusätzlich wurde das Lombok-Framework[1] und das dafür verfügbare Plugin verwendet. Um das Programm neu kompilieren zu können, sind ist dieses Plugin und die in der pom.xml spezifizierten Maven-Dependencies erforderlich.
Zusätzlich wurde SonarLint als Code-Lint-Checker verwendet.

[1] Lombok ist ein Java-Framwork, welches automatisch Getter- und Setter beim Preprocessing generiert und somit das für Java typische „Aufblähen" des Codes minimiert.

Features

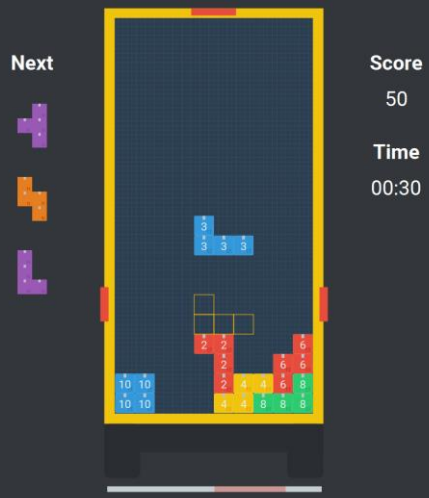
- Tetris im Postboten-Look
- Flat-Design
- Neue Spielmodi, die das bekannte Tetris auf ein neues Level bringen
- Miss dich mit deinen Freunden im Highscore!
- Passe die Steuerung so an, wie du es möchtest!

Spielmodi

Balance-Mode

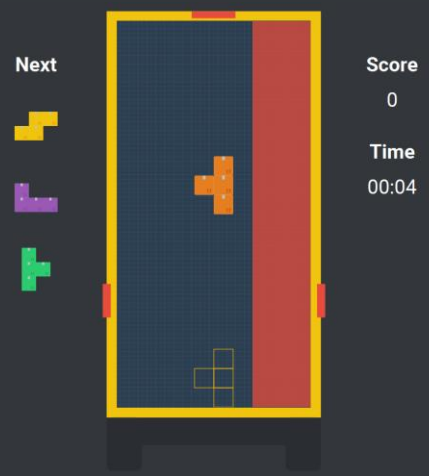
Im Balance-Mode wird jedem Parcel ein Gewicht zugewiesen. Die Aufgabe des Spielers besteht darin, die Päckchen möglichst gleichmäßig im Truck zu verteilen. Die momentane Verteilung wird dabei, durch die BalanceBar unter dem Truck angezeigt. Ab einer bestimmten Anzahl von Parcels im Truck wird geprüft, ob sich die momentane Verteilung im spezifizierten Rahmen hält. Wenn der Truck zu unbalanciert ist, kann es sein, dass die Päckchen in den Kurven herunterfallen – das Spiel wird beendet.

Die Balance bezieht sich hierbei auf die Verteilung der Päckchen im Truck, also ob mehr links oder rechts liegen.



Priority-Mode

In diesem Modus erhält jedes Päckchen eine der drei Prioritäten (gering, mittel oder hoch). Päckchen mit hoher Priorität müssen unbedingt auf die rechte Seite des Trucks, damit der Weg zum Bürgersteig noch weiter verkürzt wird und der ungeduldige Kunde sein Päckchen möglichst schnell erhält. Dahingegen dürfen Parcels mit geringer Priorität nicht in der rechten Hälfte des Trucks gelagert werden. Päckchen mit normaler Priorität dürfen überall im Truck abgelegt werden. Die verbotene Seite wird mit einem roten Rechteck angezeigt. Es ist dem Spieler in diesem Fall nicht möglich, das Parcel in diesen Bereich zu bewegen.



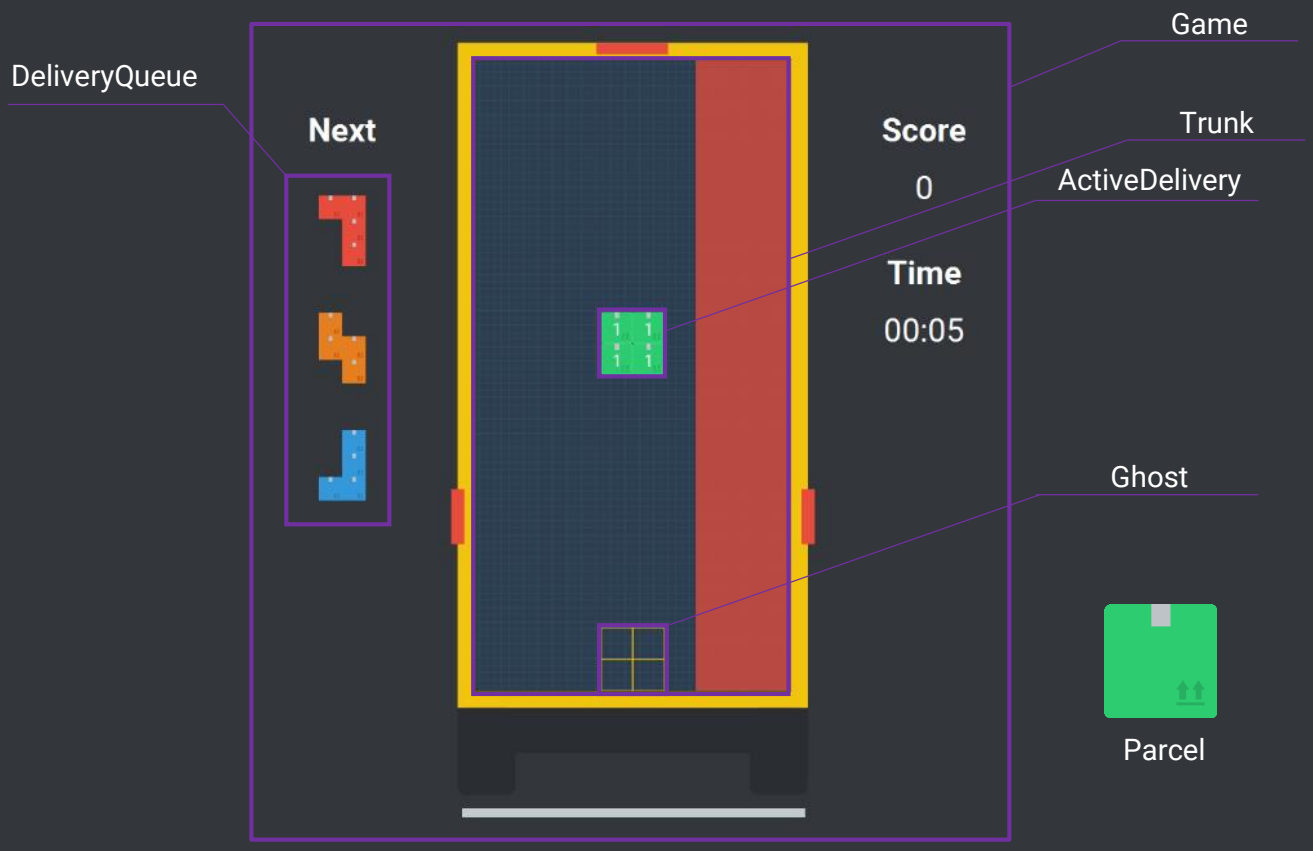
Grundlegende Architektur

Komponenten

Die Komponenten des Spiels lassen sich grob in zwei Kategorien unterteilen:

- [App]
  Die Komponenten, welche für die App an sich benötigt werden. Diese Kategorie ist verantwortlich für die Menüführung, den Lifecycle der Applikation sowie dem Bereitstellen der Parameter für das Spiel.
- [Game]
  Alle Komponenten, die etwas mit dem tatsächlichen Spielfluss zu tun haben.

Beide Komponenten sind nach dem MVC-Prinzip entwickelt, was einem durch die Verwendung von JavaFX sowieso nahegelegt wird. Im nachfolgenden werde ich mich hauptsächlich auf die Game-Komponenten beschränken, da die JavaFX-Komponenten zwar auch wichtig sind, dabei jedoch keine grundlegenden Architekturentscheidungen getroffen werden mussten. Nachfolgend möchte ich einen groben Überblick über die Architektur geben, um den Einstieg in den Code zu vereinfachen.
Anschließend würde ich erwarten, dass der Code so geschrieben ist, dass man ihn auch ohne weitere Dokumentation gut lesen kann. An den komplexeren Stellen habe ich zudem Klassenkommentare hinzugefügt.



Game Diese Klasse ist für die grundlegende Spiellogik (also den Spielfluss) verantwortlich.
Delivery Besteht aus mehreren Parcels und stellt eine von sechs Formen dar.
Parcel Bestandteil einer Delivery.
Trunk Dient als Schnittstelle zwischen Spiel und Darstellung. Enthält mehrere Methoden, zum Prüfen ob ein Parcel/ eine Delivery im Trunk oder innerhalb eines bestimmten Bereiches ist. Der Trunk umfasst eine Fläche von 10 x 20 Parcels.
ActiveDelivery Das ist die Delivery, die mometan vom Spieler gesteuert werden kann und fällt.
Ghost Ein Schatten für activeDelivery der anzeigt, wo die Delivery landen würde.
DeliveryQueue Generiert neue Deliveries und hält sie in einer Queue.

Game-Klasse

Die Gameklasse implementiert hauptsächlich die Game-Loop und eine Pipe. In jeder Iteration der Game-Loop wird die activeDelivery durch die Pipe geschickt. In der Pipe sind mehrere Pipe-Components enthalten, wie z.B. das Spawnen von neuen Deliveries, nachdem die vorherige gelandet ist, das Updaten des Ghosts oder das Reagieren auf Nutzereingaben. Diese Struktur ermöglicht es nicht nur, die eigentliche Logik dezentral zu halten, um die Lesbarkeit zu steigern, sondern auch eine gute Skalierbarkeit für weitere Features.
Die Kopplung zwischen Game und PipeComponent ist dadurch sehr stark.

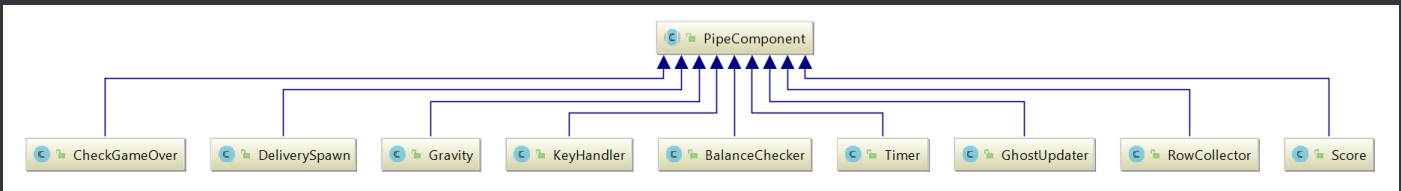


Abbildung 1 Die Pipe-Komponenten

Die Hauptbestandteile des Tetris-Spiels, also das Game, die Delivery und das Parcel wurden in jeweils zwei Klassen aufgeteilt. In eine {Komponente}Data-Klasse (GameData) und in die tatsächliche Komponentenklasse (Game). Das Ziel war es dabei, die Repräsentation der Daten eines Objektes und die anwendbaren Methoden zu trennen, um die Übersichtlichkeit zu erhöhen, da die Bestandteile sonst recht komplex und umfangreich wären.

DeliveryShape

Die Form der Deliveries wird durch eine von AbstractDeliveryShape erbende Instanz bestimmt. Es gibt sechs verschiedene Formen (LShape, InvertedLShape, ZShape, InvertedZShape, SquareShape und Spike Shape. In der AbstractDeliveryShape werden die Methoden für die Rotation einer Form implementiert.

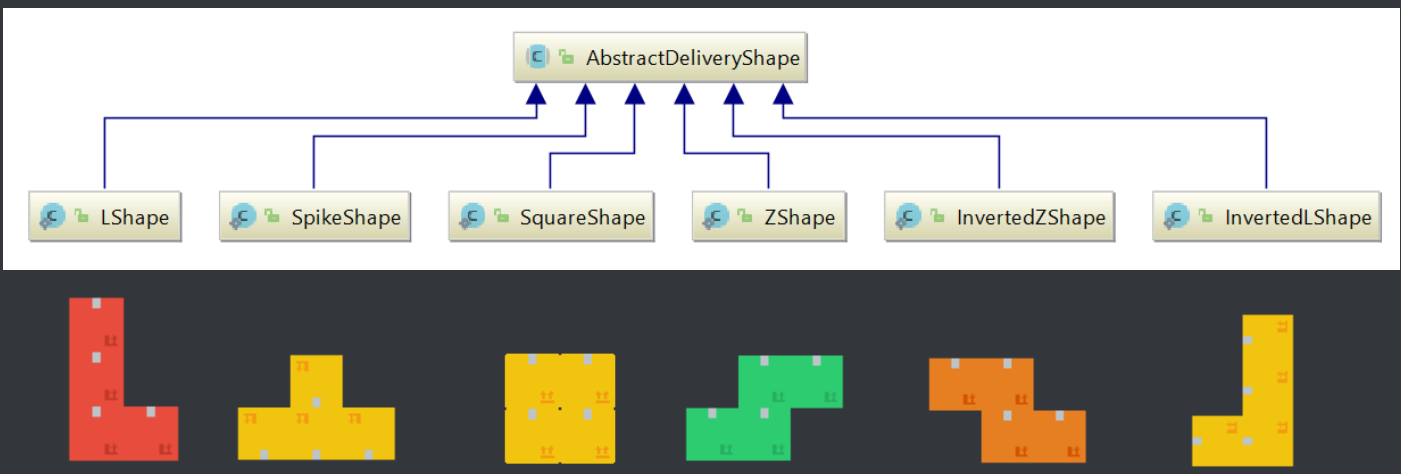


Abbildung 2 Die verschiedenen DeliveryShapes

Resourcen

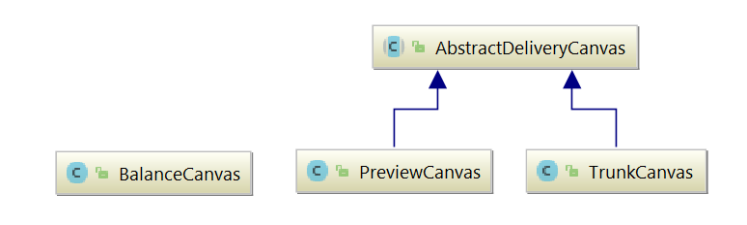
Das Spiel benötigt verschiedene Resourcen, welche im Classpath enthalten sind. Neben den Bildern für die Parcels und den Truck, ist die HTML-Seite für den About-Menüpunkt enthalten. Interessant ist zu dem auch die GameParameter.JSON. In den Game-Parameter kann die Schwierigkeit des Spiels vom Entwickler angepasst werden. Im Prinzip werden Konstanten aus dem Code in eine JSON-Datei ausgelagert. In dieser kann zum Beispiel die Griddimensionen, die Fallgeschwindigkeit und das Punktesystem angepasst werden.

Ähnlich verhält es sich mit der defaultKeymap.json in der, wie es der Name bereits verrät, ein initiales Mapping von UserActions auf eine Tastenbelegung enthalten ist.

In der Lokalisation-Datei befinden sich alle im Spiel verwendeten Wörter. Diese saubere Trennung würde es einem später leichter ermöglichen, das Spiel auch auf andere Sprachen zu übersetzen.

Die Darstellung

Für die Darstellung des Spielverlaufes, wurden drei Komponenten entwickelt, welche alle vom Canvas erben.



Das AbstractDeliveryCanvas lädt beim Initiieren bereits alle Bilder für die Parcels ein. Das TrunkCanvas ist für das Zeichnen des Spielfeldes verantwortlich. Das PreviewCanvas zeigt den momentanen Stand der DeliveryQueue an.

Das BalanceCanvas hingegen ist eine ProgressBar-ähnliche Canvas-Komponente, welche sich unterhalb des Trucks befindet. Sie zeigt im Balance-Modus an, wie die Parcels im Truck verteilt sind.

Ausblick

Mit mehr Zeit könnte man das Spiel selbstverständlich noch ausbauen. So steht als nächstes vor allem ein adaptives Design an, damit das Spiel auf allen Bildschirmauflösungen gleich gut aussieht.
Des Weiteren könnte ich mir gut vorstellen, dass man das Päckchen je nach Gewicht schneller fallen lässt, oder Powerups einbaut. Hierbei muss jedoch gut die Balance zwischen neuen Features und einer steigenden Komplexität abgewägt werden. Durch zu viele Änderungen am Spielfluss kann es leicht passieren, dass der Spieler überfordert ist und frustriert aufgibt.