Projekt: Kniffel4Fun

Judith Balß, Merle Friedrichsen, Susann Leonhardt

1 Einführung

Im Rahmen des Projektes „Würfeln4Fun“ soll das klassische Würfelspiel „Kniffel“ (engl. Yatzy) als interaktives Spiel auf dem Smartphone entwickelt werden. Da der Name „Kniffel“ markenrechtlich geschützt ist, wurde diese Version in "Würfeln4Fun" umbenannt. Das Spiel soll die originalen Regeln verwenden und von mehreren Spielern gemeinsam gespielt werden können. Der im Rahmen dieses Projektes entwickelte Prototyp stellt exemplarisch die Nutzung durch 3 Spielern da.

Die Entwicklung der Anwendung erfolgt mit der Software Android Studio. Um die gemeinsame Arbeit an dem Projekt möglichst effektiv und effizient zu gestalten, wurde das Projekt auf dem Repositorium GitHub hochgeladen und anschließend mit Android Studio verbunden. Auf diese Weise konnten mehrere Personen unabhängig voneinander und ohne Datenverlust am Projekt arbeiten.

2 Technische Hinweise

Die entwickelte Anwendung „Würfeln4Fun“ ist für Geräte ab einer Mindest-Bildschirmgröße von 3,3‘‘ mit einer Bildschirmauflösung von 240 x 400 und einer Android-Version ab API-Level 15 geeignet. Die Anwendung läuft ausschließlich im „Portrait“-Modus.

3 Konzept

Zu Beginn haben wir die Idee zum Spiel auf Papier visualisiert und dabei den Ablauf des Spiels skizziert. Aus diesen Überlegungen ist ein erster Entwurf zum Design entwickelt worden (siehe Anlage 1), der im Laufe der Umsetzung angepasst wurde. Anhand des Design wurde die Nutzerinteraktion mit der Application besprochen, und die daraus resultierende Spielelogik erfasst.

Nach diesen ersten Vorarbeiten wurden zunächst die für das Spiel benötigten Ressourcen, Grafiken und Sound, recherchiert und im Projekt hinterlegt. Dabei wurden nur Ressourcen verwendet, die unter einer entsprechenden Lizenz (CC-BY oder vergleichbar) frei verfügbar sind. Entsprechende Hinweise zu den Quellen finden sich in der README-Datei des Spiels sowie unter dem Info-Button auf dem Start-Bildschirm der Anwendung.

3.1 Programmaufbau und Ablauf

Der Programmaufbau umfasst verschiedene Java-Klassen. Das Model-View-Controller-Konzepts (MVC) trennt die Ansicht (View), und die Spielelogik (Model). Die Kommunikation läuft über den Controller, der auf Nutzereingaben reagiert, das Model entsprechend anpasst, und die nötigen Anpassungen in der Ansicht anstößt. Im Sinne des MVC können bei der Entwicklung von Android-Apps die Activities als Zusammenführung von View und Controller verstanden werden, da diese Klassen sowohl die einzelnen Bedienelemente (z.B. Buttons) erstellen, und auf Nutzerinteraktion reagieren. Entsprechend dem Model werden diese Activities aufgebaut bzw. angepasst.

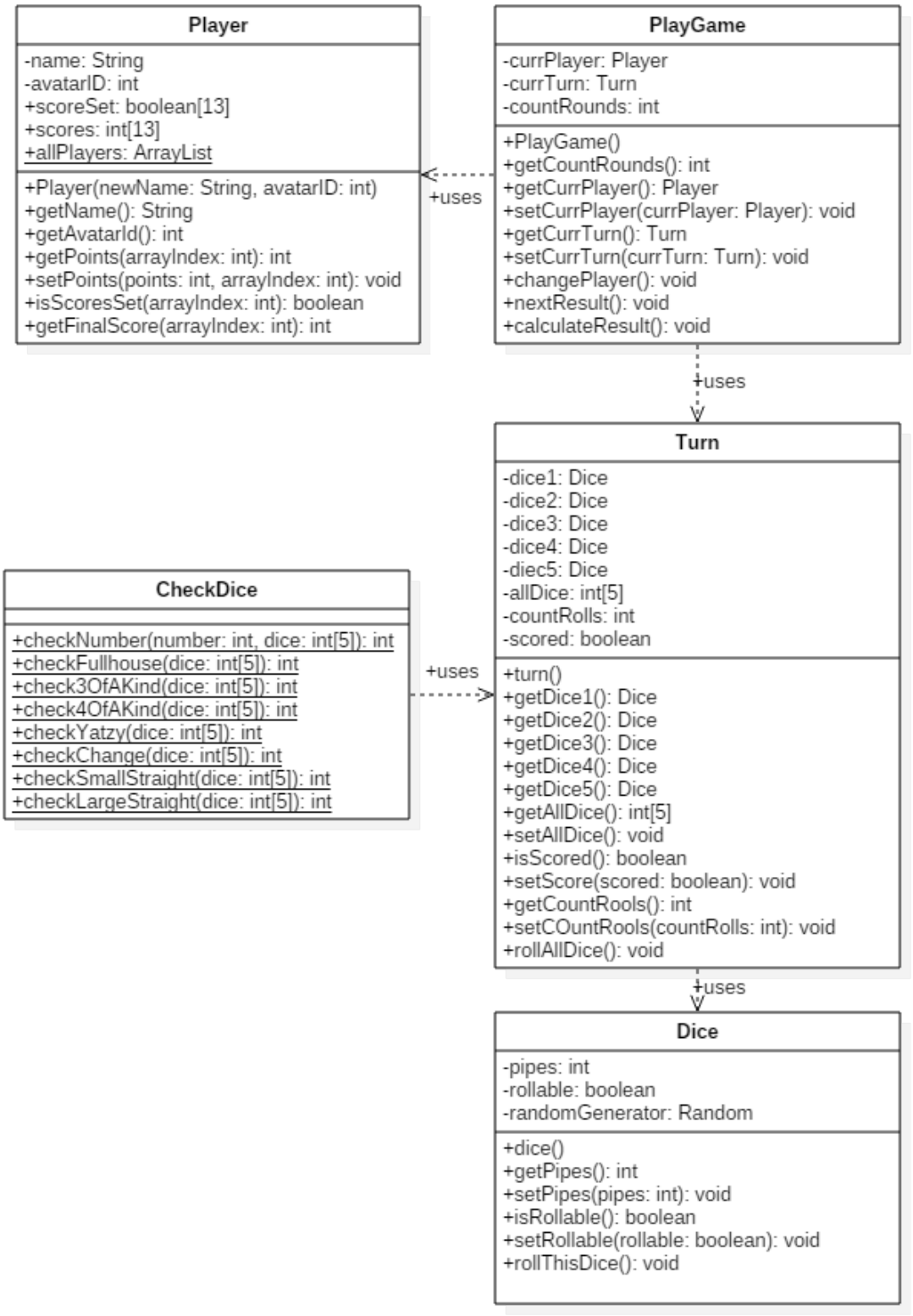
Die erstellten Klassen können also entweder dem Model oder View/Controller zugeordnet werden. Zu **View/Controller** gehören folgende Klassen:

- MainActivity (Start Activity, Spieler werden angelegt)
- AvatarAdapter (benötigt für MainActivity)
- AvatarItem (benötigt für MainActivity)
- KniffelGame (Activity zum Würfeln und Eintragen der Ergebnisse)
- ScoreGame (Spielstände am Ende des Spiels werden angezeigt)

Zu den Klassen des **Model** gehören:

- Dice (Objekt vom Typ Würfel)
- Player (Objekt vom Typ Spieler)
- Turn (Objekt vom Typ Spielzug)
- CheckDice (Überprüfung des Würfelergebnisses
- PlayGame (Objekt vom Typ Spiel)

Anhand des UML-Diagramms sieht man, wie die Klassen des Models aufgebaut sind, und wie sie ineinander greifen:



Der Programmablauf und die Interaktion zwischen View/Controller und Model wird exemplarisch anhand der Beschreibung eines Zuges erläutert:

Variable *Methode* **Klasse**

Voraussetzungen:

- in der ArrayList AllPlayers ist mindestens ein Objekt vom Typ Spieler,
- die Activity „**KniffelGame**“ ist aktiv

Mit eine m Klick auf den Button „scoreBtn“ (beschriftet mit „nächster Spieler“) wird die Methode *nextPlayer()* in der Klasse **KniffelGame** ausgeführt. Dort wird zunächst der Spieler gewechselnt mit **PlayGame**.*changePlayer*(). Diese Methode befüllt die Variable currPlayer in **PlayGame** mit dem nächsten Spieler in der ArrayList AllPlayers. Ist das Ende des ArrayList erreicht, befüllt sie die Variable mit dem ersten Spieler in der ArrayList. Zusätzlich wird ein neues Objekt vom Typ Zug erstellt. Das bedeutet, es werden 5 neue Würfel erzeugt. Als Vergleich zum analogen Kniffel dient die Übergabe der Würfel an den nächsten Spieler.

Nachdem der Spieler gewechselt wurde, wird der Wechsel in der Oberfläche angezeigt: der Name wird geändert durch **PlayGame**.*getCurrPlayer().getName()* und auch der Avatar wird angepasst durch **PlayGame**.*getCurrPlayer().getAvatarID().* Das Textfeld, welches dem Nutzer anzeigt, wie häufig er gewürfelt hat, wird auf 0 zurückgesetzt. Die Methoden *showScores()* und *showAllDices()* ändern die Anzeige: mit showScores werden die Punkte, die bereits für den Spieler eingetragen sind, angezeigt. Buttons, bei denen bereits ein Score eingetragen ist, werden ausgegraut. Mit *showDice()* werden bei den Würfeln zunächst ein Bild mit Fragezeichen angezeigt, da der Spieler noch nicht gewürfelt hat.

In diesem Moment ist es für den Spieler möglich, auf den „rollDiceBtn“ zu drücken. Es ist weder möglich, Würfel rauszulegen (also nicht mit ihnen zu würfeln), noch Ergebnisse einzutragen. Dies ist erst möglich, nachdem ein erstes Mal gewürfelt wurde. Mit einem Klick auf den „rollDiceBtn“ wird zunächst überprüft, ob bereits mehr als dreimal gewürfelt wurde und ob bereits ein Ergebnis eingetragen wurde. Laut den Kniffelregeln darf höchstens bis zu dreimal gewürfelt werden, dann muss ein Ergebnis eingetragen werden. Und wenn ein Ergebnis eingetragen ist, dann darf nicht mehr gewürfelt werden, denn der nächste Spieler ist an der Reihe.

Sind beide Bedingungen wahr, so wird gewürfelt: mit der Methode player.play(…) wird der Sound von geworfenen Würfeln abgespielt. **PlayGame**.*getCurrTurn().rollAllDice()* sorgt dafür, dass jedes Dice-Objekt, welches „rollable“ ist, eine neue Zufallszahl zwischen 1 und 6 erhält. Mit *showAllDices()* wird die Ansicht so angepasst, dass die entsprechenden Abbildungen bei den Würfeln zu sehen sind. Das Textfeld, welches anzeigt wie häufig gewürfelt wurde, wird mit **PlayGame**.*getCurrTurn().getCountRolls()* aktualisiert.

An dieser Stelle kann der Spieler sich entscheiden, ob er direkt nochmal würfeln möchte, ob er einzelne Würfel rauslegen möchte, oder bereits ein Ergebnis eintragen möchte. Das Würfeln erfolgt wie oben beschrieben. Für das rauslegen von Würfeln kann der Spieler auf den entsprechenden Würfelbutton klicken. Wenn der Würfel „rollable“ ist, dann wird der Würfel über **PlayGame**.*getCurrTurn().getDice3().setRollable(false)* vom Würfeln ausgeschlossen. Für die optische Rückmeldung an den Spieler wird mittels ColorFilter der Würfel ausgegraut. So kann der Spieler z.B. zwei gewürfelte Sechsen rauslegen, und mit den restlichen drei Würfeln versuchen, noch weitere Sechsen zu würfeln. Es ist aber auch möglich, den Würfel beim nächsten Würfeln wieder zu würfeln, in dem der Spieler auf den Würfelbutton klickt.

Entscheidet sich der Spieler, ein Ergebnis einzutragen, so klickt er auf den entsprechenden Knopf, beispielsweise „smallStreetBtn“. Bei einem Klick auf diesen Button wird zunächst überprüft, ob in diesem Zug bereits ein Ergebnis eingetragen wurde über **PlayGame**.*getCurrTurn().isScored()* und für die kleine Straße noch keine Punkte eingetragen wurden über **PlayGame**.*getCurrPlayer().isScoresSet(9)*. Trifft eines von beiden zu, so hat das Klicken des Knopfes keine Auswirkung. Trifft beides zu, dann wird mit der Methode **checkDice**.*checkSmall Straight*(**PlayGame**.*getCurrTurn().getAllDice()*) geprüft, ob die Voraussetzungen für eine kleine Straße gegeben sind.

Handelt es sich um eine kleine Straße, so wird das Ergebnis beim Spieler eingetragen und dieses im Array scoreSet markiert mithilfe von **PlayGame**.*getCurrPlayer().setPoints(30,9)* . Ist der Returnwert von checkSmallStraight(…) 0, so wird die Methode **KniffelGame**.*setScoresNull(...)* aufgerufen. Diese öffnet einen AlertDialog, in dem der Nutzer sich auswählen kann, ob er als Wert tatsächlich 0 setzten möchte. Er kann an dieser Stelle aber auch abbrechen, und ein anderen Ergebnis eintragen lassen. Bestätigt der Nutzer seine Auswahl, so wird über eine Switch-Case-Anweisung das Array scores und scoresSet an entsprechender Stelle ausgefüllt, sowie mit **PlayGame**.*getCurrTurn().setScored(**true**)* angezeigt, dass der Zug beendet ist. Durch das setzten von setScored wird beim klicken des Buttons „nächster Spieler“ ein neuer Zug des nächsten Spielers ausgelöst.

3.2 Layout

Gemäß dem o.g. Entwurf enthält die Anwendung vier verschiedene Ansichten (Activities): Start-Bildschirm (MainActivity), Spiele-Bildschirm (KniffelGameActivity) und Score-Bildschirm (ScoreGameActivity).

Während der Entwicklung der Anwendung wurden verschiedene Layouts auf Umsetzbarkeit für das geplante Design getestet. Um eine ansprechende Darstellung der (Image)Buttons und der dazugehörigen TextViews in der KniffelGameActivity und der ScoreGameActivity zu ermöglichen, wurde letztendlich das LinearLayout für alle Activities verwendet. Da die EndGameActivity keine weitern Funktion enthält wurde im Laufe der Entwicklung entschieden auf diese zu verzichten und auf der ScoreGameActivity eine Möglichkeit zur Beendigung des Spiels eingebaut.

Die MainActivity enthält zum einen das GUI-Element …. um dem Spieler die Möglichkeit zu geben, einen Spielernamen einzugeben und zum anderen das GUI-Element Spinner mit dessen Hilfe eine Spiele-Avatar aus einer vorgegebenen Dropdown-Liste ausgewählt werden kann. Der Start-Button für das Spiel „Lets Go…“ ist animiert.

Die KniffelGameActivity enthält neben den TextViews die GUI-Elemente Button und ImageButton, die je nach Funktion mit verschiedenen Klassen und Methoden verbunden sind bzw. durch diese den Spielverlauf steuern.

Die ScoreGameActivity enthält TextViews und ImageViews mit deren Hilfe die erreichten Score-Werte der verschiedenen Spieler angezeigt werden, sowie einen Button um zum Startbildschirm zurück zu kehren und ein neues Spiel zu beginnen.

3.3 Fehler

Zum jetzigen Zeitpunkt sind uns mehrere Fehler bekannt. Zum einen kann über den „zurück-Button“ jederzeit ein neuer Spieler hinzugefügt werden. Dies funktioniert nicht nur in der ersten Runde, sondern auch schon im fortgeschrittenen Spiel. Das Spiel wird beendet, sobald in einer Runde ein Spieler alle Ergebnisse eingetragen hat, so dass der später hinzugefügte Spieler keine Möglichkeit hat, auch alle nötigen Züge durchzuführen. Hier muss entschieden werden, wie damit umgegangen werden soll. Es wäre möglich, den Zurück-Button zu entfernen, oder man könnte das Spiel so anpassen, dass das endgültige Ergebnis erst dann angezeigt wird, wenn alle Spieler alle Züge gemacht haben, egal wann sie im Spiel eingestiegen sind. Dies bedarf aber größerer Anpassungen.

Ein weiterer Fehler betrifft das Schließen der App. Bei der Activity ScoreGame ist ein Button angelegt, der das kontrollierte Schließen der App ermöglichen soll. Im Moment führt das Klicken des Buttons jedoch zu einem Absturz der App. Auch hier könnte man sich entscheiden, ob der Button entfernt werden soll (da keine wichtigen Daten eingegeben wurden und der Spielstand nicht gespeichert wird). Oder eine funktionierendes Schließen der App wird bei Knopfdruck ausgelöst.

Im Moment kommt es zu einer kurzen Verzögerung beim Abspielen des Würfel-Sounds. Zudem werden die Würfelergebnisse sofort angezeigt. Dies kann behoben werden, in dem das Würfeln animiert wird, und das Ergebnis der Würfel erst am Ende des Sounds angezeigt wird.

4. Ausblick

Neben der Behebung bestehender Fehler sollte bei der nächsten Version noch einige Verbesserungen sowohl für den Nutzer als auch für die Übersicht des Codes umgesetzt werden.

Dazu gehört eine Fehlerbehandlung durch Try-Catch: im Moment ist nur eine rudimentäre Fehlerbehandlung umgesetzt. Als nächstes sollte der auszuführende Code in einen Try-Block gesetzt werden, und dort sollten Fehler und Exceptions aufgeworfen werden. In einem dazugehörigen Catch-Block würden die Fehler dann behandelt werden, so dass es nicht zu einem (unkontrollierten) Absturz der App kommt. Einige Methoden könnten noch verschlankt werden, in dem sie zusammengefasst werden, beispielsweise treten **PlayGame**.*getCurrPlayer.().setScores(…)* und **PlayGame**.*getCurrTurn.setScored()* immer paarweise auf. Durch das zusammenführen solcher Methodenpaarungen kann der Code in den Activities übersichtlicher gestaltet werden. Zudem könnte eine vollständige Umsetzung des MVC erfolgen, indem die View auch vom Controller getrennt wird. Dies erleichtert das Testen und das Weiterentwickeln der App.

Mögliche Weiterentwicklungen werden auch das Nutzererlebnis verbessern. So könnte man das Würfeln auch durch das Schütteln des Gerätes auslösen. Zudem kann der Gewinner in der Activity ScoreGame benannt, und animiert werden, z.B. durch ein Hervorheben des Avatars. Im Moment ist die ScoreGame-Activity für drei Spieler ausgelegt. Dies sollte beliebig erweiterbar sein, so dass auch mehr als drei Spieler spielen können (das können sie schon jetzt), und auch ihr Ergebnis sehen. Für die Einzelspielervariante bietet es sich an, vorhergehende Highscores zu speichern. In der Activity ScoreGame kann dann ausgewertet werden, wie gut man abgeschnitten hat, oder ob man sogar einen neuen Highscore erreicht hat.