Technische Dokumentation

Thursday, July 20, 2017 3:20 PM

1. Spielprinzip

Das Spiel "QuickQuizCopter" (kurz QQC) wurde im Rahmen der Veranstaltungen ITS und MOSY von den Medientechnikern Tjarko Slomski (2182553) und Nico Flach (2175367) und den Media-Systems-Studentinnen Mona Röttger (2220740) und Corinna Bohnenberger (2217555) entwickelt. Es wurde beim Rundgang Finkenau am 13.07.2017 ausgestellt.

Das Spiel ist für zwei Spieler gedacht, die auf ihrem eigenen Smartphone ein Quiz (ähnlich wie Quizduell) gegeneinander spielen können. Zu Spielbeginn drücken die beiden Spieler auf Start und verbinden sich dadurch mit dem Server. Haben beide das Spiel gestartet fliegt das Flugobjekt in die Mitte der Konstruktion. Der Hintergrund der Konstruktion besteht aus einer zweifarbigen Skala, wobei jede Farbe einem Spieler zugeordnet ist. Ist das Flugobjekt in der Ausgangsposition angekommen wird die erste Frage angezeigt. Je nachdem wer die Frage richtig beantwortet hat bzw. die Frage schneller richtig beantwortet hat, fliegt das Flugobjekt entweder ein Stück nach oben oder nach unten und die nächste Frage wird angezeigt. Je nachdem, ob sich das Flugobjekt nach 10 Fragen im oberen oder unteren Bereich befindet, hat Spieler 1 oder Spieler 2 gewonnen. Das Spiel kann auch vorzeitig beendet werden, indem das Objekt ganz oben oder ganz unten ankommt, wodurch auch eine Konfettikanone ausgelöst wird.

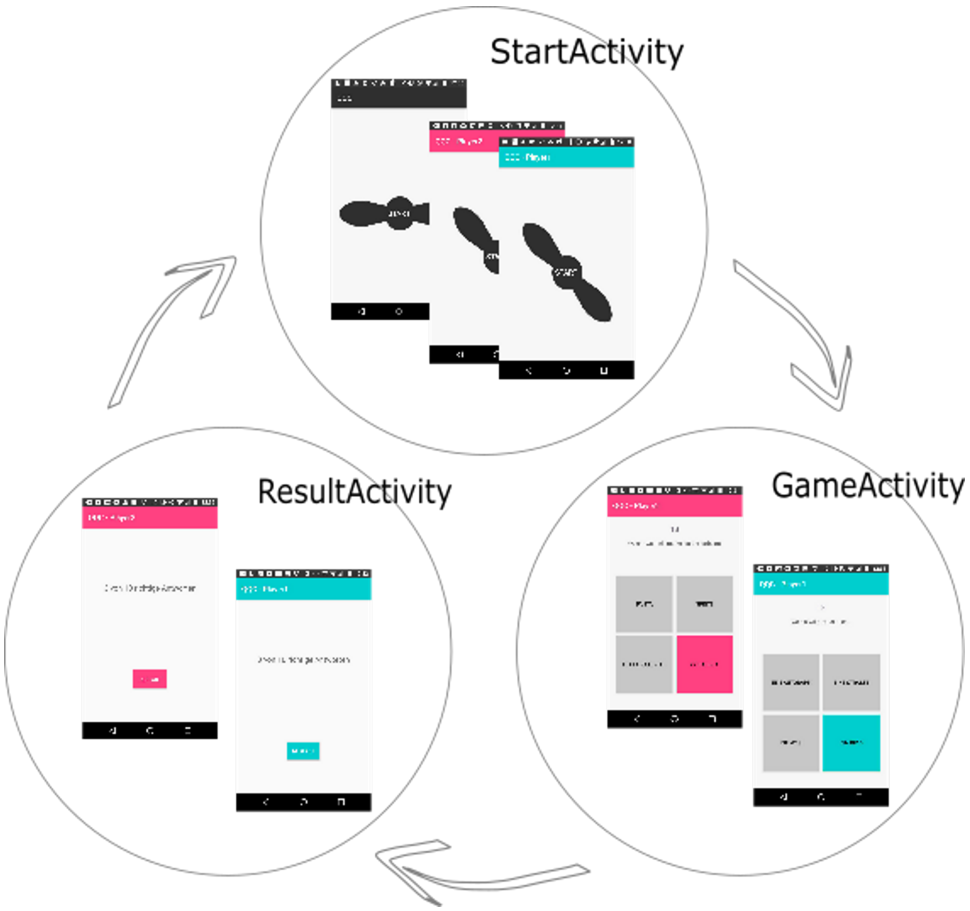
2. Software

2.1 App

Die App zum Spiel QQC wurde als native Android App mit Android Studio entwickelt. Sie besteht aus 3 Activities, der StartActivity, GameActivity und ResultActivity.

Design
Wird die App gestartet, befinden sich die Spieler jeweils in der **StartActivity**. Es ist nur die ActionBar mit dem Spielkürzel "QQC" und ein großer Start-Button in Form einen Propellers in den schlichten Farben weiß und dunkelgrau zu sehen. Drücken die Spieler den Start-Button fängt der Propeller an sich zu drehen (GIF) und die ActionBar wird in der jeweiligen Farbe des Spielers eingefärbt (blau=Spieler1, pink=Spieler2). Für das GIF wurden extra 4 PNG's mit verschiedenen Propeller-Rotationen erstellt. Der drehende Propeller ersetzt den ProgressSpinner und verdeutlicht den Spielern, dass ihre Eingaben erkannt und weiterverarbeitet werden. Sind beide Smartphones mit dem Server verbunden, startet der Motor und steigt bis zu Mitte der Hintergrundskala. Erst jetzt hört der Propeller auf sich zu drehen und die GameActivity wird geöffnet.
Wird die App gestartet, befinden sich die Spieler jeweils in der **StartActivity**. Es ist nur die ActionBar mit dem Spielkürzel "QQC" und ein großer Start-Button in Form einen Propellers in den schlichten Farben weiß und dunkelgrau zu sehen. Drücken die Spieler den Start-Button fängt der Propeller an sich zu drehen (GIF) und die ActionBar wird in der jeweiligen Farbe des Spielers eingefärbt (blau=Spieler1, pink=Spieler2). Für das GIF wurden extra 4 PNG's mit verschiedenen Propeller-Rotationen erstellt. Der drehende Propeller ersetzt den ProgressSpinner und verdeutlicht den Spielern, dass ihre Eingaben erkannt und weiterverarbeitet werden. Sind beide Smartphones mit dem Server verbunden, startet der Motor und steigt bis zu Mitte der Hintergrundskala. Erst jetzt hört der Propeller auf sich zu drehen und die GameActivity wird geöffnet.
Die **GameActivity** besteht hauptsächlich aus der aktuellen Frage und den dazugehörigen vier Antwortmöglichkeiten. Auch hier ist die ActionBar in der zugewiesenen Farbe des Spielers gesetzt. Die eingefärbte ActionBar soll es dem Spieler erleichtern, seinen aktuellen Spielstand an der Konstruktion schnell abzulesen. Die Antwort-Buttons sind in jeweils in sich verschachtelten Linear Layouts angebracht, da diese sonst in anderer Anordnung in den restlichen Layouts wegen der

verschieden Antwort-Längen verrutscht sind. Wird eine Antwort gewählt, färbt sich dieser Button ebenfalls in dieser Farbe, sodass der Spieler ein Feedback bekommt. Über dem ganzen Spielfeld wird dem Spieler die noch zu verbleibende Zeit eingeblendet (20s). Wurden alle Fragen der aktuellen Runde beantwortet und der Gewinner ermittelt, geht es über in die ResultActivity.
Die **ResultActivity** ist die letzte Activity des Spiels. Hier werden den Spielern die jeweilige Anzahl der richtig beantworteten Fragen angezeigt. Weiter unten gibt es einen Restart-Button, mit dem eine neue Quizrunde gestartet werden kann. Der Button sowie die ActionBar sind passend zum Spieler eingefärbt. Somit lässt sich ohne Probleme der Endstand an der Konstruktion dem Gewinner und Verlierer zuordnen.



Funktionsweise
In der StartActivity wird zunächst der Server angefragt, welchen Status das Spiel hat:

- Ob ein neues Spiel gestartet werden kann
- Bereits ein Spieler darauf wartet, dass ein zweiter Spieler dem Spiel beitritt
- Ob bereits ein Spiel zwischen zwei Spielern läuft und somit ein Dritter warten muss

Bei Klick auf den Start-Button wird die Verbindung zum Server aufgebaut und je nachdem wechselt das Spiel in einen der genannten Zustände. Problematisch ist hierbei, wenn zwei Spieler gleichzeitig auf den Start-Button drücken, da in dem Fall das "Auf den zweiten Spieler warten"-Signal noch nicht gesendet wurde. Es ist momentan erforderlich, beide Apps komplett neu zu starten. Eine mögliche Verbesserung wäre, nach Klick auf den Button eine zufällig lange Zeit abzuwarten bis das Signal gesendet wird um so die Fehleranfälligkeit zu verringern.
Derjenige, der zuerst den Start-Button klickt, ist Spieler 1. Über ihn laufen alle Berechnungen, die beide Spieler betreffen.
Zum Beispiel wird hier auch die Rundenzahl abgeglichen und an Spieler 2 weitergeschickt, damit beide Spieler die gleichen Fragen gestellt bekommen. Es gibt 8 Runden mit jeweils 10 Fragen. Diese wurden ausgewählt aus Themengebieten wie HAW, Hamburg, Disney, Märchen, etc... Danach beginnt es wieder bei Runde 1. So konnten wir vermeiden, dass wenn pro Runde zufällig Fragen ausgewählt würden, Spieler, die zwei oder mehr Runden hintereinander spielen, Fragen doppelt gestellt bekommen würden. 8 Runden hintereinander schienen uns ausreichend. Durch das Senden der Rundennummer werden entsprechend Fragen in die SQLite-Datenbank eingetragen, aus denen dann je Runde die Fragen ausgelesen werden. Dies ist keine sehr schöne Lösung, doch leider war es

nicht anders machbar, da es sonst zu Fehlern kam und teilweise Fragen nicht angezeigt wurden. Den Fehler konnten wir bisher leider nicht beheben.

In der GameActivity wird die jeweilige Frage aus der Datenbank gelesen und angezeigt. Die Antwortmöglichkeiten werden dabei in zufälliger Reihenfolge angezeigt, um zu verhindern, dass sich die richtige Antwort immer an der gleichen Stelle befindet. Bei Klick auf einen der Antwort-Button, wird diese mit der Richtigen aus der Datenbank verglichen. Außerdem wird die benötigte Zeit zur Beantwortung an den Server gesendet, die durch einen zu Beginn jeder Frage gestarteten Timer gestoppt wird. Wenn die Frage falsch beantwortet wurde, wird diese auf 0 gesetzt. In der TimerHandler-Klasse wird bei Spieler 1 berechnet, welcher der beiden Spieler schneller geantwortet hat und entsprechend ein Signal an den Server gesendet, sodass sich der Propeller nach oben bzw. nach unten bewegt. Bei einem Gleichstand bleibt er an der Stelle und es wird den Spielern durch einen Toast "Gleichstand" angezeigt. Zusätzlich wird dem Spieler direkt durch einen Toast angezeigt, ob er die Frage richtig beantwortet hat und falls nicht, die richtige Antwort eingeblendet.
Durch ein Signal vom Raspberry Pi wird die nächste Frage eingeblendet. Hier sollte während des Wartens ein ProgressSpinner erscheinen, dies hat allerdings nicht wie gewollt funktioniert.

Nachdem alle 10 Fragen gestellt wurden bzw. ein Spieler frühzeitig das Spiel beendet hat, wird die ResultActivity angezeigt. Hier sieht man, wie viele Fragen man richtig beantwortet hat und kann durch den Restart-Button eine neue Runde starten. Zudem sollte hier noch der Gewinner genannt werden und wie viele Fragen schneller beantwortet wurden als vom Gegner. Dies hat allerdings nicht funktioniert, da die Berechnungsergebnisse durch Player 1 zu spät geschickt wurden, um sie über den Observer (s. weitere Informationen bei Networking) der GameActivity abzufangen und zu früh, um sie in der über den Observer der ResultActiviy abzufangen. Da das Ergebnis allerdings vorrangig über die Position des Flugobjekts abzulesen ist, war dies nicht weiter schlimm.

Verbindung zum Broker
Die Kommunikation zwischen der Applikation und dem Raspberry Pi läuft über einen MQTT-Server. In der Klasse **ClientHandler** geht es hauptsächlich um die Herstellung der Verbindung zwischen Server und App. Während des Quiz' wird genau ein Objekt der Klasse instanziiert, über welches die Verbindung zum Server überwacht wird und diverse Befehle zwischen den Activities und dem Raspberry Pi ausgetauscht werden.
Die Klasse erbt von dem Interface MqttCallback. Zusätzlich zu den dadurch bereits implementierten Methoden (connectionLost, deliveryComplete, messageArrived) wurden weitere Methoden hinzugefügt, um die Befehle benutzerdefiniert zu veröffentlichen (publish) und abzufangen (subscribe).
Dafür wird als erstes in der Methode connectWithServer() für den Client ein MqttAndroidClient-Objekt erzeugt. Über dieses Client-Objekt kann dann die Verbindung zum Server aufgerufen werden. Vorher haben wir noch Optionen für die Verbindung eingestellt (Username und Password). Der IMqttListener innerhalb dieser Methode informiert den Spieler über den Erfolg oder Fehlschlag der Verbindung.
Wurde die Verbindung erfolgreich hergestellt, können jetzt mithilfe der Methoden toPublish()/ toSubscribe() Messages veröffentlicht und empfangen werden. In der Methode toSubscribe() wird einmalig das Topic "haw/dmi/mt/its/ss17/qqc2" festgelegt, auf welches der Server hören soll. Die Methode toPublish() veröffentlicht die als Parameter übergebene Nachricht.
Wurde eine Nachricht erfolgreich veröffentlicht und übermittelt (deliveryComplete) wird diese in der Methode messageArrived() abgefangen. Durch if-Abfragen wird die Nachricht mit möglichen Eingaben verglichen und weiterverarbeitet z.B. wird vom Raspberry Pi ein "go" veröffentlicht, zeigt die App eine neue Frage an.
Um die Messages an die entsprechenden Klassen, in der sie verwendet werden, zu schicken, wurde ein Observer Pattern verwendet. Dadurch werden die GameActivity und der TimeHandler als Observer registriert und im ClientHandler bei bestimmten Messages benachrichtigt. In der updateMessage()-Methode der beiden Klassen, werden sie entsprechend behandelt. In der onStop()-Methode der GameActivity wird die Registrierung beider Klassen aufgehoben, sodass die Objekte der alten Runde nicht mehr in der neuen Runde registriert sind.

Ton
In der ResultActivity wird bei beiden Smartphones nach jeder erfolgreichen Quizrunde ein Applaus-Sound abgespielt. Den passenden Sound haben wir heruntergeladen (http://www.salamisound.de/) und für unsere Zwecke noch mit Audacity gekürzt. Die gekürzte Datei wurde in einem extra Ordner in Android Studio eingebunden und über ein MediaPlayer-Objekt abgespielt**.**

Vibration
Bei jedem Anzeigen einer neuen Frage wurde eine kurze Vibration eingebaut.
Die Aufmerksamkeit vom Spieler wird wieder auf das Quiz gelenkt, nach dem er sich den Punktestand an der Konstruktion anschauen konnte.

2.2 Raspberry Pi

Um die Kommunikation zwischen Elektronik und App herzustellen, dient ein Raspberry Pi 3 als Brücke, auf dem ein Programm in Python läuft. Hier werden die Befehle von der App umgewandelt in Signale an die Steuereinheit und Signale von einem Sensor umgewandelt in Befehle, die an die App geschickt werden.
Der Raspberry Pi wird zum Einen über das Netzwerk ans Internet angebunden, um den Broker zu erreichen und zum anderen an eine H-Brücke für die Motorsteuerung sowie einen Ultraschall Sensor für die Entfernungsmessung.

In Python wird der Broker abonniert. Dadurch hört der Raspberry Pi auf die Befehle der App und führt daraufhin Aktionen aus und sendet Rückmeldungen an die App.
Hier eine Übersicht der Befehle:

Befehle (Empfangen)	Start	Ende	Player1	Player2	Tie	-
Aktion	Einpegeln auf Startposition	Herunterfahren und Parameter Resetten	Ein Stück nach oben fliegen	Ein Stück nach unten fliegen	-	Konfetti ausgelöst
Befehle (Senden)	Go	resetOK	Go	Go	Go	Ende

Beim Starten wird die Aktion "Einpegeln" ausgeführt. Nun wird der Motor auf volle Geschwindigkeit beschleunigt (PWM = 100 (%)) und die Drehrichtung nicht verändert (DIR-Port auf "0"). Dann wird solange gemessen, bis der Motor die (mittlere) Startposition erreicht hat und daraufhin die Geschwindigkeit gedrosselt (PWM = 5 (%)).
Danach wird ein "go" gesendet, um das Spiel zu starten.
Wird nun ein "Player1" oder "Player2" empfangen, wird der Motor entweder beschleunigt mit der Funktion "fahrenHoch" ("Player1", PWM = 100 %), oder mit der Funktion "fahrenRunter" zuerst abgebremst, die Drehrichtung umgekehrt (DIR auf "1") und dann beschleunigt ("Player2", PWM = "100 %") und fliegt, unter ständigem Messen, von der aktuellen Höhe auf die neue Höhe von +- 10 cm, relativ zu dem vorherigen Wert. Bei erreichen der Höhe wird der Motor wieder gedrosselt und die Drehrichtung auf "normal" gesetzt (DIR auf "0", PWM auf 5 %).
Wird ein bestimmter (maximaler/minimaler) Wert über-/ unterschritten wird die Konfettikanone ausgelöst und das Spiel beendet indem ein "ende" gesendet wird. Daraufhin fährt der Motor runter (DIR auf "1" und volle Drehzahl/PWM = 100 %) und sobald ein Wert unterhalb von 30 cm (Der Schwung reicht dann, um auf die niedrigste Position zu fahren) fällt, bekommen alle Variablen einen Reset.

Die H-Brücke
Die H-Brücke dient als Bindeglied zwischen Motor und Raspberry und regelt die jeweilige Stromversorgung des Motors, der stärker ist, als der Steuerstrom des Raspberry's. Hier wird über einen Modellbautrafo die Versorgung für den Motor (7,2 V Nenn; 12,5 A bei P max), sowie die Pins für die Steuerung (+5V; GND; Motor; DIR) angeschlossen. Der DIR Pin wird auf 0 gesetzt, um den

Motor in "normaler" Richtung drehen zu lassen und auf 1, um die Drehrichtung umzukehren.
Der MOTOR Pin dient zur Steuerung der Geschwindigkeit über PWM. Die Pulsweite wird so direkt auf das Ausgangssignal übergeben.

3. Konstruktion

Die Konstruktion
Zur Visualisierung des Spielstandes dient ein Motor mit Propeller, der in einer Führung nach oben, bzw. unten fliegt.
Ursprünglich sollte diese Führung durch zwei Metallstangen realisiert werden, zwischen welchen der Motor montiert ist und mit Hilfe von Rohrstücken nach oben gleiten kann. Bei dieser Idee sollte der Strom über Schleifkontakte geführt werden.
Nach ersten Versuchen mit verschiedenen Materialen zeigte sich zum Einen, dass die Leitfähigkeit zur Nutzung als Schleifkontakt größtenteils unzureichend ist, zum Anderen, dass die Reibung der Rohre die Flugfähigkeit erheblich mindert. Dies ist auf den Radeffekt zurückzuführen: Der Motor entwickelt aufgrund des Luftwiderstandes eine Rotationskraft in Richtung der Rotordrehrichtung. Die Stangen, welche nur am Boden befestigt waren, wurden verzogen – die Rohre konnten somit nicht mehr frei gleiten.

Ein weiteres Problem stellte das Gewicht der Trägerkonstruktion dar: Die Auftriebskraft des Motors haben wir bei Volllast mit ca. 140g bemessen.

Um die Reibung zu verringern, planten wir zunächst mit Horizontalkugellagern, welche weiterhin auf Stangen gleiten sollten. Diese erste Idee musste aufgrund des Gewichts der Lager verworfen werden.

Stattdessen haben wir uns für eine 3D gedruckte Halterung mit handelsüblichen Kugellagern entschieden, welche in einer Schiene nach oben rollen können. Damit lösten wir gleichzeitig das Gewichtsproblem und verhinderten die Drehverzerrung der Führung, indem wir massive Holzlatten als Halterung für die Schienen verwendeten.

Die erste gedruckte Halterung hatte ein Kugellager pro Schiene (welche aus auf das Holz aufgeklebten Kabelkanälen besteht). Beim Testen zeigte sich, dass wir den Radeffekt so kontrollieren können, stattdessen verdrehte sich die ganze Konstruktion aber in der Horizontalen.

In einem zweiten Prototyp musste also auch diese zweite Achse stabilisiert werden.

Dies geschieht durch zwei weitere Kugellager pro Schiene, welche an einem Ausleger einige Zentimeter unter den ersten Kugellagern sitzen.

Um einen gleichen Abstand der Schienen von oben bis unten zu gewährleisten, sind die Latten oben mit einer Gewindestange verbunden, über welche der Abstand mit Hilfe von Muttern exakt eingestellt werden kann.

Da eine Stromführung mit Schleifkontakten nicht mehr in Frage kam, wird der Motor über Kabel gespeist. Diese werden nach oben über zwei Rollen nach hinten abgeführt.

Der Motor
Die Stromversorgung selbst stellte sich zu Anfang als großes Problem dar: Der Motor hat einen Laststrom von 12,5 A bei 7,2 V. Diese Leistung konnten die Netzteile der Hochschule nicht liefern. Mit Modellbauakkus konnten wir den Strom für kurze Zeiten erreichen, ein Dauerbetrieb war aufgrund deren Kapazität allerdings nicht möglich. Wir entschieden uns daher, ein Modellbau Netzteil der Marke Graupner mit einem maximalen Ausgangsstrom von 20 A zu verwenden.

In der finalen Version sind die auf der Rückseite nach unten laufenden Kabel mit Gewichten beschwert. Der Grund hierfür ist, dass die Steifigkeit der Kabel bereits wesentlich die Flugfähigkeit einschränkt. Durch die Beschwerung kommt der Motor einfacher nach oben, gleichzeitig wird dafür gesorgt, dass es keinen "Kabelstau" gibt.

Die Elektronik wird durch eine Rückwand versteckt. Diese ist nach oben und unten in die zwei Farben der Spieler geteilt, sodass sie gleichzeitig anzeigt, in welche Richtung sich der Motor für den jeweiligen Spieler bewegt.

Der Sensor
Der Ultraschalsensor ist direkt an das Board angeschlossen und dient zur Entfernungsmessung zwischen Boden und Motor. Mithilfe dieser Messdaten kann die Höhe bestimmt und angepasst werden. Die besondere Herausforderung bei dem Sensor ist, die Messwerte halbwegs brauchbar zu erhalten. Hier kamen häufig fehlerhafte Messungen zustande, da die Messfläche der Konstruktion zu gering war. Die Fehlmessungen wurden durch eine breitere Applikation oberhalb des Motors abgefangen. Außerdem ist eine kleine Fehlererkennung implementiert, die Werte oberhalb der höchsten "Stufe" herabsetzt und auf den vorherigen Wert setzt. Eine bessere Fehlererkennung konnte aus Zeitgründen leider nicht mehr im Code implementiert werden.

Die Konfettikanone
Das letzte Highlight der Konstruktion ist eine Konfettikanone. Diese ist ebenso hinter der Rückwand im oberen Teil befestigt. Sie besteht aus einem herkömmlichen PC Lüfter, der mit einem Netz überspannt ist. Darauf kann das Konfetti abgelegt werden, ohne den Luftstrom wesentlich zu mindern. Die Konfettikanone wird ausgelöst, wenn der Wert 80 cm bzw. 20 cm über- bzw. unterschritten wird und die Frage von der führenden Partei richtig beantwortet wurde. Hierzu wird

lediglich ein Relais geschaltet, das einen Stromkreis schließt, der einen PC-Lüfter für 5 Sekunden antreibt. Dieser zerstreut dann wiederum Konfetti über einen einfachen Papiertrichter, der hinter der Konstruktion angebracht ist.

Plakat
Das Plakat wurde mit dem Programm Illustrator erstellt und für die Ausstellung extra mit einem kleinen Motor erweitert, der einen Propeller langsam dreht. Die Farben der Spieler werden im Projektnamen aufgegriffen.