HTWG Konstanz

Tetris für Android

Lange Programmiernächte

Urs Rust (282257), Pascal Laier (282234), Sebastián Guillén ()
29.07.2012

Inhaltsverzeichnis

Einleitung

Zweck

Dieses Dokument beschreibt ein Softwareprojekt für das Fach „Lange Programmiernächte“ an der HTWG Konstanz.

Umfang

Das Softwareprojekt besteht aus diesem Dokument, einer Benutzerdokumentation, dem Programm selbst sowohl als Sourcecode als auch als ausführbare Version und dem Testprojekt mit welchem dieses getestet wurde.

Übersicht

Im Folgenden wird zuerst das Programm allgemein beschrieben, danach folgen die Anforderungen, eingeteilt in Nutzungsanforderungen und Systemanforderungen und die Systemarchitektur. Dies wird von einer Beschreibung der Implementierung ergänzt.

Allgemeine Beschreibung

Problem- und Aufgabenstellung

Im Rahmen dieses Softwareprojekts soll ein bereits existierendes Tetris Spiel, welches in der Programmiersprache Java geschrieben ist, dahingehend modifiziert werden, dass es auf mobilen Geräten mit dem Android Betriebssystem von Google verwendet werden kann. Des weiteren soll ein Export des im Spiel erzielten Highscores zu einem entfernten Server via Internetverbindung und Import von diesem hinzugefügt werden.

Einschränkungen

Da diese Software auf einem bereits existierenden Spiel basiert kann die Planung der Architektur und der Implementierung nicht von Grund auf erfolgen sondern muss an die Gegebenheiten angepasst werden.

Abhängigkeiten

Voraussetzung für die Verwendung dieses Programms ist ein Endgerät mit Googles Android Betriebssystem mit Version 2.3.3 oder höher. Für die Verwendung des Highscore Exports/Imports muss eine Internetverbindung vorhanden sein.

Verwendete Software

Dieses Projekt wurde unter Einsatz von Eclipse Indigo Version 3.7.2, dem Android SDK Version 20.0.1 und dem zugehörigen Android Eclipse PlugIn ADT erstellt.

Zur Versionsverwaltung wurde Git verwendet, das Repository ist unter https://github.com/Pascal88/langeNaechteApp zu finden.

Zum Testen wurde neben den von ADT angebotenen UnitTests die Testbibliothek Robotium (1) verwendet.

Anforderungen

Stakeholder

Stakeholder sind bei diesem Projekt die Entwickler, der Nutzer und die bewertende Person der HTWG Konstanz.

Nutzungsanforderungen

Nutzungsanforderungen sind außer den bei einem Spiel auf einem mobilen Endgerät üblichen keine besonderen gegeben. Diese sind:

Der Nutzer muss das Spiel starten können.

Der Nutzer muss das Spiel jederzeit pausieren und beenden können.

Der Nutzer muss seine erzielte Leistung abspeichern und mit seinem Namen verknüpfen können.

Der Nutzer muss das Spiel spielen können, das heisst es muss Steuerungsmöglichkeiten geben.

Der Nutzer muss seine erzielte Punktzahl während des Spiels jederzeit sehen können.

Systemanforderungen

1. Die Anwendung muss auf den mobilen Endgeräten der Zielplattform (Google Android 2.3.3 oder höher) installierbar und verwendbar sein.
2. Die Anwendung muss nach Start ein Menu anzeigen in welchem der Start des Spiels, das Anzeigen des Highscore, Abgeben von Feedback und Beenden der Anwendung mittels Berühren des Bildschirms ausgelöst werden kann.
3. Bei Auswahl von „Start“ muss ein Spielbildschirm angezeigt werden im welchem das Spiel abläuft. Dieser muss desweiteren folgende zusätzliche Elemente besitzen:
   - Steuerungsschaltflächen um die Tetrisspielelemente nach links oder rechts zu bewegen, zu rotieren und beschleunigt am unteren Rand abzusetzen
   - Eine Anzeige der aktuell erzielten Punkte
   - Eine Vorschau auf das nächste erscheinende Spielelement.
4. Steuerung des Spiels muss neben den angezeigten Schaltflächen auch mittels Touchscreen möglich sein.
5. Bei Auswahl von „Start“ muss jedes Mal ein neues Spiel erstellt werden, es darf kein altes weiterverwendet werden.
6. Bei Ende des Spiels nach den Tetrisregeln muss das Spiel beendet werden.
7. Bei Ende des Spiels muss der erzielte Highscore zum Server exportiert werden, dabei muss es eine Möglichkeit geben diesen mit einem einzugebenden Namen zu verknüpfen.
8. Bei Betätigen der in Android vorgeschriebenen „Back“ Schaltfläche muss eine Abfrage stattfinden ob das Spiel neugestartet werden soll oder ob zurück zum Menubildschirm gewechselt werden soll.
9. Bei Betätigen der in Android vorgeschriebenen „Home“ Schaltfläche muss das Spiel beendet und bei Neustart der Anwendung an gleicher Stelle fortgesetzt werden.
10. Bei Auswahl von „Feedback“ muss eine Möglichkeit gegeben sein mit den Entwicklern Kontakt aufzunehmen.
11. Bei Auswahl von „Exit“ muss die Anwendung beendet werden.

12. Das System muss ausreichend performant ausgelegt sein um das Spiel nicht zu behindern. Dies bedeutet, dass mindestens jede Bewegung eines Spieleelements angezeigt werden muss.
13. Die Reaktionszeit muss gering genug sein um spätestens beim nächsten Update des Spielebildschirms Benutzereingaben anzuzeigen.

Implementierung

Architektur

Die Architektur der Android Tetris App orientiert sich streng am MVC (Model View Controller) Pattern, es wurde ebenso viel Wert auf Flexibilität und Abstraktion gelegt weshalb jede Komponente ein Interface hat und von anderen Komponenten auch nur die Interfaces kennt. Ebenso wurde eine Schichtarchitektur umgesetzt, das heisst jedes Package ist ebenso ein Layer welcher nur mit sich selbst und den Layern darunter kommunizieren darf. Dies zu realisieren ist leider nicht immer möglich weswegen zusätzlich das Observer Pattern verwendet wurde um die Abhängigkeit zu höheren Layern umzukehren, dies trifft auf das Tetris Spiel und die Android Gui zu.

Für den Server, welcher in PHP geschrieben ist, wurde ein einfaches MVC Pattern umgesetzt.

Die beschriebene Architektur ist in Abbildung 1 zu sehen.

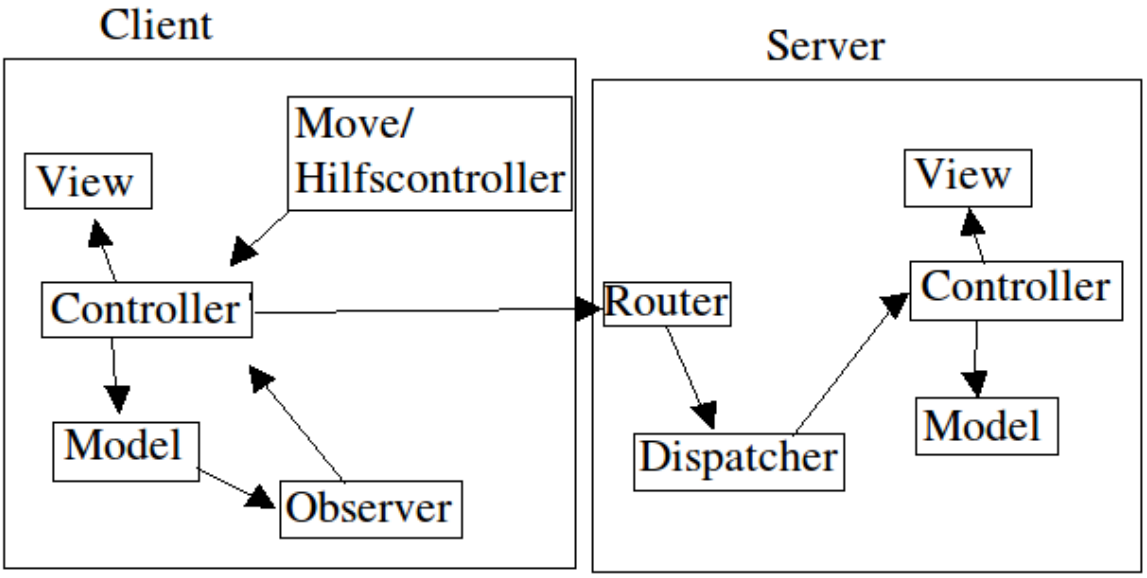

Abbildung 1 – Architektur

Klassen

Im beigefügten Klassendiagramm sind sämtliche Klassen des Programms zu sehen, im Folgenden wird hauptsächlich nur die für dieses Projekt bearbeiteten oder neu hinzugefügten eingegangen.

Server:

- Model
  - Für jede Datenbank Tabelle wir eine Bean benötigt, welche die Informationen und Namen der Datenbank Tabelle enthält.
  - Desweiten wird ein Datenbank Mapper benötigt, in diesem wird die Datenbank erstellt und an das Datenbank Objekt weitergeleitet.
  - Zusätzlich wird ein Datenbank Table Objekt benötigt welches den tatsächlichen Namen der Tabelle enthält

- View
    - Die View ist eine phtml Datei, welche denselben Namen trägt wie die zuvor ausgeführte Aktion des Controllers. Sie muss in einem Subfolder der View Folder gelegt werden welche den Namen des Controllers haben muss. Sie enthält nur ein html Fragment, kein komplettes Dokument.
- Controller
    - Werden vom Dispatcher aufgerufen und bekommen ein Request Objekt übergeben
- Router
    - Der Router verpackt den Http Request in ein Request Objekt.
- Dispatcher
    - Bekommt das zuvor vom Router generierte Request Objekt und ruft dann die vom User gewünschte Aktion am richtigen Controller per Reflection auf.

Client:

Da das Tetris Spiel vorher für ein anderes Projekt geschrieben wurde und für dieses Projekt deshalb nur eine neue Gui nötig war wird in diesem Abschnitt nur kurz auf dessen Architektur eingegangen.

- View/Gui
    - Die Eula Activity zeigt dem Nutzer die Nutzungsbedingungen an falls er sie noch nicht akzeptiert hat. Falls diese abgelehnt werden wird die App wieder geschlossen. Akzeptiert er diese wird er zur nächsten Activity , der Menu Activity, weitergeleitet. Dabei wird in den sharedprefences gespeichert, dass sie akzeptiert wurden, so dass der Nutzer diese nicht erneut akzeptieren muss.
    - Die Splash Activity erstellt eine initiale Instanz des HighscoreController Singletons.
    - Die Menu Activity zeichnet das Menu und leitet zu den vom Nutzer gewünschten Activities weiter.
    - Die Highscore Activity zeigt die Highscore Liste an, berechnet für wieviele Einträge Platz auf dem Display ist und lädt exakt so viele dann vom Server.
    - Die Game Activity zeichnet den GameView, die Buttons und den aktuellen Score des Nutzers. Sie leitet die Nutzereingaben von den Buttons oder der Touchevents an die Controller weiter. Ausserdem startet/pausiert/beendet sie das Spiel, speichert es temporär falls der Nutzer eine Notification vom System bekommt oder falls er auf den Home Button betätigt. Nach Ende eines Spiels bietet sie dem Nutzer die Möglichkeit seinen erzielten Punktewert auf dem Server zu teilen und erledigt dies falls gewünscht.
    - Die Game View zeichnet das eigentliche Spielfeld und den Stein der als nächstes erscheinen wird. Dies wurde alles mit einem Canvas Element realisiert.
    - Die MyApp Activity kommuniziert mit dem HighscoreController um Highscores vom Server zu laden.
- Controller (nur der HighscoreController ist neu hinzugekommen)
    - Der HighscoreController ist ein Singleton, er reicht die Daten an das/vom Model weiter.

    - Der GameController kontrolliert das Tetris Spiel und leitet die Nutzereingaben an die Hilfscontroller (Move Package) weiter. Stößt das generieren von neuen zufälligen Spielsteinen an.
    - Der GamemechanikController steuert die Gameloop.
    - Der TetrisController kontrolliert grundsätzlichere Angelegenheiten des Spiels wie etwas den Highscore und kann das Spiel starten/neustarten/beenden.
    - Die Move/HilfsController setzen die gewünschten Bewegungen des Spielsteins auf der logischen Matrix des Spiels um.
- Observer
    - Bietet Klassen und Interfaces für Observer und Subject an um die Schichtarchitektur einhalten zu können, dreht die Abhängigkeit der Layers um.
- Model
    - Enthält die Spielsteine(Element), die Logik Matrix(GameArray) und Kommunikation zum Server (Highscore).

Tests

Zum Testen wurden die vom ADT PlugIn angebotenen Möglichkeiten genutzt um Unit Tests für die MenuActivity (in MenuTest.java) und die GameActivity (in GameButtonsTest.java) zu erstellen. Dabei wurde in beiden Klassen die Existenz der Buttons überprüft.

Ein Systemtest wurde mittels der externen Bibliothek Robotium (1) in MenuTests2.java durchgeführt, welcher die Funktionalität des Menus, das heisst der Aufruf der korrekten Activities, testet.

Des Weiteren wurden umfangreiche Tests des Spiels durch Spielen desselben durch die Entwickler durchgeführt.

Codemetriken

In Abbildung 2 - Komplexität ist die Komplexität der Klassen alphabetisch nach dem Namen ihrer Packages sortiert, man kann hier gut sehen, dass die Packages gui.activities und model die höchste Komplexität haben. Da eine durchschnittliche Komplexität von 8,2 pro Klasse recht gut ist, ist dies allerdings nicht weiter schlimm.

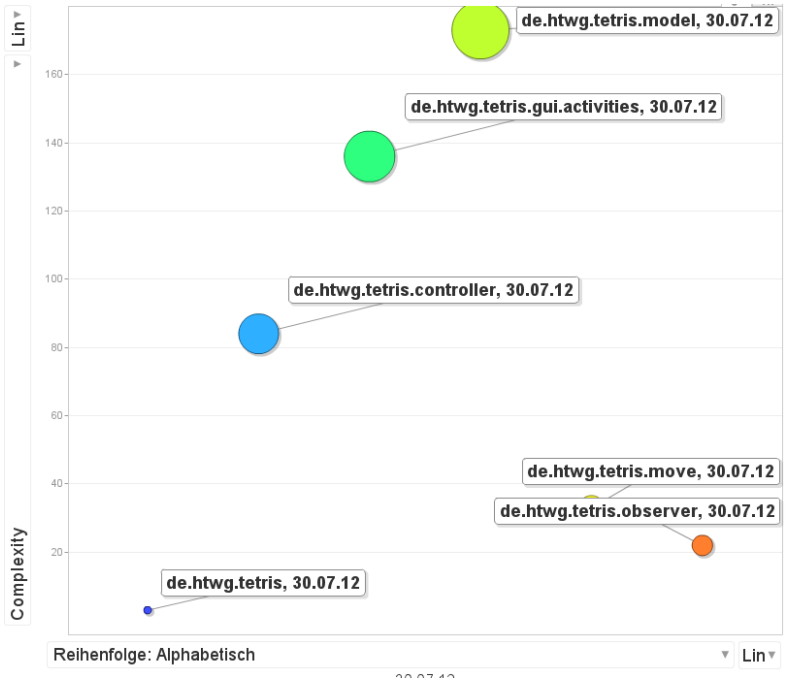

Abbildung 2 - Komplexität

In Abbildung 3 - Komplexität der Methoden ist die Komplexität der Methoden zu sehen. Gut erkennbar ist hier, dass abermals das Package gui.activities einen im Vergleich sehr hohen Wert aufweist, was darauf hindeutet, dass man die Komplexität dieses Packages vielleicht veringern sollte.

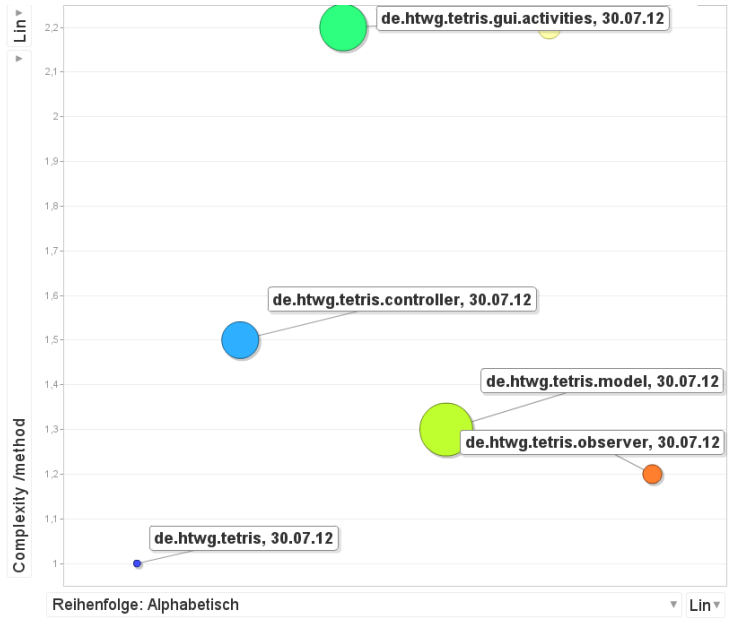

Abbildung 3 – Komplexität der Methoden

In Abbildung 4 - Lines of code ist die Anzahl der Code Zeilen pro Package zu sehen. Wie zu erwarten war haben die Packages der View und des Models die meisten Codezeilen.

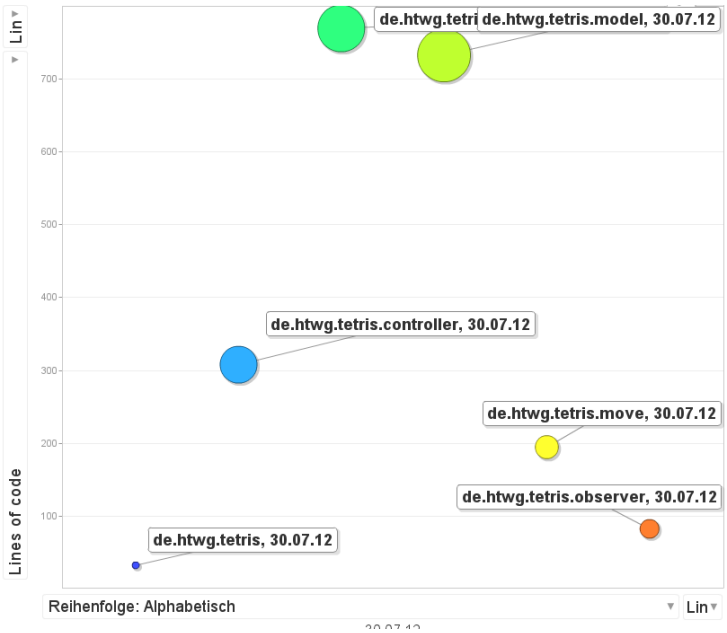

Abbildung 4 – Lines of code

Ergebnis

Obwohl dieses Programm auf einem bereits existenten Spiel basiert, welches in der Lehre eingesetzt werden soll und somit nicht für den reinen Zweck des Spielens sondern für Lehrzwecke optimiert ist, konnten die aufgelisteten Systemanforderungen alle eingehalten werden. Bei einem Entwurfsdesign von Grund auf hätte das unserem Projekt zugrundeliegende Spiel sicher einfacher und weniger komplex entworfen werden können, allerdings sind die Anforderungen an ein Spiel für mobile Endgeräte hiervon nur in Bezug auf Platzverbrauch und Performanz betroffen und die erzielte Geschwindigkeit ist ausreichend. Die den reinen Spielbetrieb betreffenden Anforderungen wie zum Beispiel Touchscreensteuerung sind dabei nicht von der erhöhten Komplexität des Spiels betroffen. Des Weiteren war das Ziel dieses Projekts das Portieren des Spiels zu Android, was hauptsächlich den Entwurf einer neuen GUI bedeutet und nicht eine tiefgehende Optimierung des existenten Codes, sofern nicht nötig.

Quellen

1: Robotium: http://code.google.com/p/robotium/

2: Die Android Referenz: http://developer.android.com/reference/packages.html

Hierbei insbesondere: http://developer.android.com/reference/android/graphics/Canvas.html

3: Die Unterlagen aus der Vorlesung lange Programmiernächte:

http://www.johner.org/studierende/lange-naechte/lange-android-nacht/skripts/

http://www.johner.org/studierende/lange-naechte/lange-android-nacht/uebungen/