CS242: Advanced Programming Concepts in Java

Beat Blocks - Final Project - Fall 2015

Hunter Quant, Cameron Icso

1. Introduction

For our project we chose to develop an original idea. After much group deliberation we decided on a rhythm-puzzle game for the Android platform. The main idea of the game is to beat your personal high score by making matches on the game board all while moving blocks to the beat of the song playing in the background. We eventually decided on the name of Beat Blocks as it seemed fitting and catchy. We wanted a game that was easy to pick up and play when you have a minute or two to spare. We also wanted the game to be addictive like retro arcade games, which is achieved by pursuing the high score.

2. Problem Description

The core mechanics of the game is to make matches of three or more blocks on the game board while matching the beat of a song that is playing in the background. The player can move the blocks in any direction of their choosing, but if they fail to move to the beat 3 times it's game over. The goal is to achieve the high score before reaching game over. One tactic to help the player achieve a high score is making large combos, which drastically increases the amount of points received. When the player is going for a high score and making large combos the game has a hidden mechanic of block organization, which can be satisfyingly challenging to accomplish while focusing on the rhythm mechanic. For example if the player only focuses on the minimal combo of three blocks they will only receive 90 points for that combos versus the 1690 points they can receive if they make the largest combo of thirteen blocks. After the player hits game over their high score is saved for them to beat during their next play session. In the beginning we had a completely different vision for the game, but it was based on the same mechanics. The final decisions which are stated above were based on our personal user experiences.

3. Methodology

There were three main back end features which needed to be developed. First, we had the core component, the game board. The board consists of three small algorithms that handle the task of: board population, match detection, and match removal.
Second, we had to draw the blocks to the screen and provide an unobtrusive, but visually pleasing animation for the removal of blocks after a match has been made. We draw the blocks to the screen in a grid and draw the bitmap corresponding to the value in the board at that position. When a match is made the board populates after a certain time period has passed and draws an empty block in the removed blocks place. This adds a simple animation that is both unintrusive and aesthetically pleasing. Third, we had to have a way of displaying the beats in a song and verifying that when a move is made it's in the accepting region of the beat map. To do this we have a collection of beats, which are animated across the screen into the accepting region. The pacing and distance between each beat is based on defined constants and song length.

4. Implementation

Tasks

In the beginning we had multiple meetings to discuss the vision of the game and each team member brought their own insight into the project. The tasks were broken up based on team member interest rather than assigning someone a job to complete. We figured by working on an area of interest that development would be more productive as well as enjoyable for each team member. In the beginning the task were broken up into the following assignments. Hunter was in charge of the game board and when the time came working with others to have the board communicate with the other components of the game, and also writing the final project report. Cameron was in charge of the rhythm system and working with Hunter to have the beat map communicate with the board. Our other teammate was in charge of the user interface and working with Hunter and Cameron on displaying information from the board and beat map. Midway through the project timeline our teammate withdrew from the course leaving only 2 members. We had to restructure the project with Hunter picking up the GUI aspect of the project and an overall downscale of project scope.

Tools

The development tools used by the team played a critical role in productivity, communication, and education. We used Android Studio 1.3.2 for application development which is vastly important as it has all the conveniences of a typical IDE, a visual GUI creation tool using XML, and it translates the written Java code into ART bytecode. We used Git 2.5.3

as our distributed version control system, which is immensely useful for collaborating in teams as we can push our code to a hosted repository and retrieve each others code with little to no hassle. We used the campus hosted GitLab service to host our project repository. GitLab provided multiple features for communication such as issue management and a wiki. Lastly, we used the SimpleUMLCE 0.01 plugin for Android Studio for generating the UML class diagram of our project.

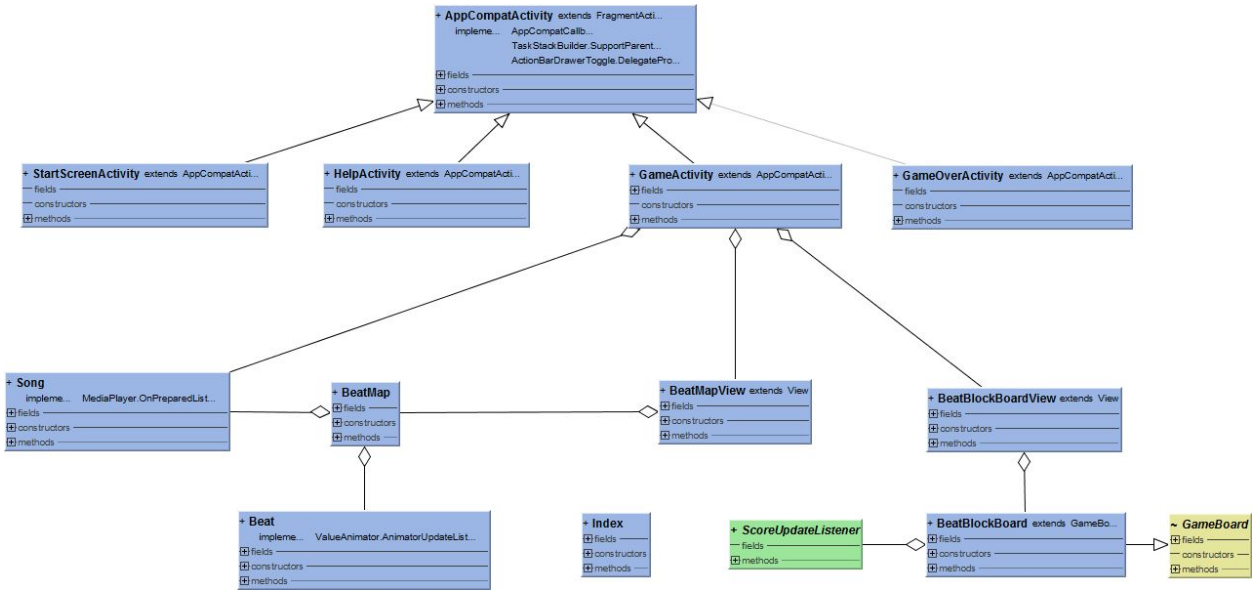
Code Layout

Our application consisted of 13 Java classes, 4 XML layout files, 3 XML resource files, 12 art assets, and 1 audio asset. The Java classes did the following. Beat represents an individual Beat to be used in the BeatMap. BeatBlockBoard serves as the backend framework for the interactive game board. BeatBlockBoardView handles the drawing of the BeatBlockBoard and updating the score information from the BeatBlockBoard. BeatMap defines all the specifications of the BeatMap to be displayed. BeatMapView defines the drawing of the BeatMap to the screen. GameActivity is the activity that is run to play Beat Blocks and handles the user input for interacting with the BeatBlockBoardView. GameBoard is an abstract class that serves as a template for various types of game boards. GameOverActivity is entered after reaching game over and gives the user the option to return to the StartScreenActivity. HelpActivity is the activity that is run from the start screen, which provides instructions on how to play the game. Index is a pair of integers x and y that serve as an index of the 2D array in the BeatBlockBoard and has a unique hashcode for any pair of x and y it can assume in the game. ScoreUpdateListener serves as an action listener for updating the game score. Song contains all the information about the song to be played as well as housing for the MediaPlayer. StartScreenActivity is the first activity that is displayed when starting the application and is used to navigate between the GameActivity and HelpActivity.

Testing

We tested our application using a few tools that are packaged along with Android Studio. We used Logcat for logging debug messages, which is useful for tracking down unknown errors and correcting them. We used the AVD emulator for testing the application on a virtual phone, which can be troublesome as it is very resource intensive and doesn't provide the best performance when playing the game. The last tool used for testing the application is an Android phone, which provides the optimal performance and quick application start times.

UML Class Diagram



5. Discussion

This project was a great learning experience, but with any learning experience it doesn't come without its fair share of challenges that must be overcome. One of the first challenges arose at the start of the project as most of our members have never worked in a team and have had no experience using version control. This problem was remedied swiftly through research and the use of online tutorials. We had very little merge issues, so the VCS process was very smooth and convenient for development. Another challenge we faced was learning all the tools for developing Android applications and the Android Java libraries. The Android tools and libraries are large in scale so they provided quite the hurdle to overcome. The Java and Android communities are filled to the brim with experienced developers who will gladly answer questions you may have, so many of our questions have already been asked in one form or another. One of the longest lasting challenges was the positioning of custom views in Android. When the problem initially presented itself we tried a number of methods, but none worked. We were trying to position it using Java rather than the XML tools provided by Android Studio. After little to no success using Java for the positioning we refactored the custom views so they would work properly with the XML layout tools and everything was in working order. This problem took a long time to fix, but after all was said and done we have a much better understanding of the Android system and how to use XML in

Android development. A problem that presented itself later on in development was verifying if a move takes place, while a beat is in the accepting region of the beat map. We had a working implementation, but it would make the GUI unresponsive so we had to go back to the drawing board. We met up and worked together to fix this problem, which led to a new implementation using concurrency to maintain the responsiveness of the GUI. The most disappointing challenge we faced was having a teammate drop the class mid project. This proved troublesome, because we had to reassign some tasks, take on more work, and downscale the project. The initial plan was for Hunter to work on the board and when he was finished shift over to helping Cameron work on the rhythm system. But when we lost the teammate Hunter had work on the GUI and scoring system. This led to a scaled back rhythm system. We wanted to give users the ability to use their own music and possibly edit their own beat maps, but with the scaled back project we were only able to achieve one song and no editable beat maps. If we had a team of three or four members we believe the project would be fully realized and publishable on the Google Play store. Overall the challenges faced were a great learning experience and provided valuable insight into problems that might arise in future development endeavors.