



Project Proposal: Rekop Poker

James Scully (4304469)
psyjs20@nottingham.ac.uk
G400 Computer Science

Project Supervisor: Milena Radenkovic

Background and Motivation

Poker is a game loved by many around the world twice over and as such has found its way into mobile applications. Many of these do not give control to the players, and are much like in real life, focused on money, rather than a casual and relaxed game. Quite a number of these apps attempt to rectify this by occasionally handing out free chips to keep players using them, such as Zynga Poker.

A quick search of the Google Play Store reveals 250 poker applications, however most of these include some form of monetization whereby the player is likely to purchase chips in order to continue playing, when they have lost all of their chips. Many people prefer mobile games to be simple, easy to use and not plagued with offers for in-game products.

A solution to this would be an open-source approach, whereby the community can fix bugs that arise, add features to the official build; removing the need for transactions to support development.

As mentioned previously, many of these do not give much control to the user, such as customizing match settings. In existing apps, most only have presets and are at the hands of the developers. In regards to user experience, it is best to offer control to the user, as they can tailor the game to their likings. This effect can be seen in a variety of areas not limited to software; think having clothes fit to ones preferences, or the ability to make changes to a home.

Aims and Objectives

The aim of this project is to create a mobile, multi-player poker game as well as a server for people to host their lobbies.

With this in mind, we have some key objectives to consider:

- Players are able to view multiple statistics:
 - Previous hands dealt
 - Total win / losses
 - Accumulated wins
- Develop a server application which:
 - Handles disconnections / interruptions gracefully
 - Is open-source and can be used by the public for private servers
 - Is cross-platform, to ensure it is portable.
- Hands that are given out should not be repeated or predictable i.e. near-complete randomness (not time-based seeds)

Work plan

Research and Plan Documents

Firstly, research will need to be undertaken on areas to innovate upon and frameworks to be used within the app itself, for example if a game engine is going to be used - if so, what API level does it support? How will we host an official server for players to connect to? Next, we'll have to produce requirements; how we're going to meet our key objectives, and ideal features. This will be explained further with the vision and scope document which provides a general vision of a sufficient system and gives an idea of how many features need to be implemented (scope) to prevent feature creep.

Methodology and On-going processes

This project is going to be developed alongside other studies and as such will be obstructed by other pieces of coursework at some points. Because of this, the Agile methodology will be used as this allows for more flexible deadlines and progress. Other more "rigid" methodologies such as Waterfall are not suitable as game / app development is a **very** iterative process, not linear. This best fits Agile methods.

It is also imperative that a task management system is to be used such as GitHub issues; this will allow for the project to be easily picked up from a certain point and continued, as there will be a few phases where focus on the project will need to be diverted briefly.

A journal will be kept on the repository for documenting choices made, problems encountered to aid in developing the interim report and dissertation.

Testing and Evaluation

In similar fashion to the journal, development will include regular unit tests on the back-end and can be integrated into Continuous Integration (CI), as this will generate a pass or fail on the unit tests with each commit.

For the front-end, slightly less frequent testing will be done in the form of black/white box testing as automated unit tests are not suitable for user interfaces.

Evaluation towards the end of development will incorporate feedback from users, findings from quality reviews (both code and interface).

Bibliography

Gantt Chart

