Design decisions

Freedom

After much deliberation we came to the conclusion that we wanted to create as much freedom as possible within the app, as to best resemble a physical deck of cards. This meant choosing not to focus on any actual rule aspects for particular card games, instead leaving the responsibility of rule making and enforcing up to the users.

This meant that for the best user experience the players should be able to have physical contact with each other in order to keep score,enforce fair play etc. This is something that we wanted to push for in order to keep the social aspect of playing card games with a physical deck.

The way we chose to keep this freedom was by realizing that most, if not all, card games can be described as a collection of 'piles' of cards. At first we had planned to have different views for the table, the common piles and every users individual 'hand'. But not all card games have the need for a specific hand, for example solitaire, and some have the need for more than one hand so by choosing to treat every hand as a pile we could just create a table view and a pile view to inspect the pile of choice.

This in combination with an option to 'protect' piles gives the same effect as having an individual 'My Hand' view but with the added bonus of the ability for users to have more than one 'hand' and the ability for several users to play on the same device. It also made development easier because it removed any special cases . The drawback of this is that you might lose some convenience in games where you want quick access to a hand that should only be viewable by one player.

Choosing to forego everything that has to do with rules also gave a lot of flexibility to the app in the sense that you can play a wide range of games because all card games are essentially just a combination of piles with face down or face up cards or a mix of both. This has the downside that you might lose some useability because of course it enables cheating. But users will be less likely to cheat when in the same room as other users than otherwise.

We also chose to break down all card handling to small and easy card operations (flip, move and peek) in order to quickly have a working application to build upon. This gave us the advantage of being able to delegate work to a larger extent because it removed a lot of dependencies we would otherwise have created. The thinking was that it would be easier to implement and reduce redundancies when creating more complex operations later on. All complex card handling can be constructed either by repeating an operation or by linking any of the basic card operations together.

We contemplated making the application turn-based but quickly decided against this in order to keep the air of freedom. This means that users can join or leave existing games at any point in time, much like a real life game table. And because when a users leaves a game their protected piles are reverted back to unprotected, the other players can easily continue playing. But this also gave testing and proper developing even more importance because the ability for users to perform card or pile operations simultaneously can easily create bugs.

Architecture

We decided that one device would act as the host when connecting devices instead of using a dedicated server or Bluetooth etc. Mostly because it meant we did not have to find and run a dedicated server but it also gave us the advantages of having all the code in the same project and written in Java. And it made the implementation easier, we did not have to look into how to pair phones for example. The negative aspect to this choice is that the game session is dependent on the host device staying alive.

In conjunction with this we decided to make it so that whenever a new game is created it is set up to run via network but the hosts actions are sent via it's loopback interface to localhost. Effectively this means that local game play is equal to hosting a network session. This realization made it much easier to implement the multi-player aspects by removing all special cases. Now any changes made will be reflected in both single and multi-player which removes redundant code and simplifies maintenance and testing which saved us a lot of time. We probably lost some performance in local play with this decision but since the amount of data sent is so small the loss is negligible.

Although at first we decided to not put any focus on the network aspect and chose to get the Gui- and Gamecontroller up and running to have a solid base to work from. This made feature implementation easier because we got a grasp of how the GUI worked and knew what had to be sent between them. However, this meant that we had no network support during the first two sprints and had to refactor the code to implement it. This in turn left us with less time to test and find bugs on the network side.

The GUI-controller was made singleton since it should be restricted to one instance per device and this was a convenient way of achieving that. The advantages of this decision are that the GUIcontroller does not have to be passed around the different devices and there is no possibility of creating more than one instance of it by accident. The downside is that it has to be passed objects via methods before it is completely initiated.

We chose to send Java-objects instead of strings over the network. This means that the application is dependent on Java based clients but decreases the risk for bugs and making it easier to test while making it more secure. It also produces cleaner code and easier implementation.

The desired way of implementing card operations in the pile view was to be able to perform gestures on the cards to perform different operations. While this feature was implemented we chose to leave it out of the final release because it did not match our expected level of stability and we did not want to draw focus from bug fixes and polishes during the final week.
We felt that the lost functionality did not outweigh the need for a stable product.

We chose to target API-level 15 in order to reach as big a market as possible, while maintaining most of the features of newer Android versions. With level 15 we also get the big graphical update that is Android 4.0, which brings plenty of changes that we could have needed.