TK3 – Exercise 3
Group O

Thorsten Hollstein - Uwe Mueller - Frederik Peukert - Tom Schons - Daniel Wegemer
The entire project can be found on GitHub via: https://github.com/UweM/Splitris

Documentation for “libsplitscreen”

Problem Statement:
When programming applications for Android, it is by default not easily possible to create Java applications capable of sharing a single screen of a device (or a server) over multiple Android smartphones. Even when using modern state of the art Android smartphones (e.g. having the possibility to use the Android 5.1 libraries) there is no predefined library to do so. We therefore wanted to change this situation by creating a generic framework (java library) capable of sharing a screen of a device (or a network server) as a split screen amongst other Android devices.

Additionally we will provide 2 demo applications, which make use of this library and include such split screen features in a very intuitive way.

Our solution:

1. Server side:
The server side of the library is divided into multiple sub parts. Firstly, our virtualscreen class is containing and managing the overall image of the later so called split screen. This class provides the abstraction layer necessary to handle all kind of underlying image technologies. The virtualscreen class basically inherits its properties from the canvas, meaning that the entire area of the virtualscreen class can be considered as a huge drawing map. The virtualscreen class contains a render method, which is responsible for rendering the image by running through all the viewports. Secondly, our viewport class can be considered to represent the view of a single camera in the overall screen image. For example a single viewport can represent the camera in the most left corner of the overall image, thus being related to the most left smartphone display. The input data for the virtualscreen class is the overall screen image. The virtualscreen class then cuts out his own view (e.g. the current camera view) out of the entire screen image and displays exactly this part of the overall image on the screen of the smartphone, resulting in the split screen functionality. The server functionality also includes filters to fade out the left most and right most parts of the local image to avoid overlapping effects across multiple screens. Lastly the current view is transformed into a bitmap and the saved locally for displaying purposes. Lastly the blockscreen class contains information on which smartphone actually controls which part of the overall screen. Later on this single block (of the overall image) is translated into the corresponding pixels on the smartphone display.

2. Client side:
The client side of the library is by far not as complex as the server side, as most of the calculations and the network operations are made (partially in advance) by the server side.
The client side of the library is handling the local view of the smartphone display. It therefore uses the screenview class of the library, where the calculations for the local displaying of the shared split screen are performed.
To begin using the library on the client side, the library must solely be included into the source code of the application. To do so, one can for example easily call the method initialize in the mainactivity of the application.

Due to logically splitting the library into a client and a server side, we can benefit from a few advantages. Firstly, we reduce the necessary network overhead in an active session, as the server is in the sole responsibility for rendering the complete virtual screen for all of the clients. Therefore each

of the clients only gets his own part of the overall image, resulting in way less network traffic (compared to a situation where we would need to transfer the entire image to all of the clients). Secondly, only the server has to be more or less powerful in order to render the images. All of the connected clients can operate on absolute low end hardware, as they only have to display the received image and exchange messages with the server. The only additional feature the client has to manage is the actual user input, in order to realize the game play (left/right/up/down buttons or swipes).
Additionally the server can easily be used for multiple different setups, as one can freely decide what to display on which display. For example, a device containing two virtualscreens can easily receive and also display two different views on two separate virtualscreens.

Technologies:
Using common smartphones having stock Android (version 4.4 and 5)
Using Java as programming language

Implementation:
See attached source code which is properly documented and also available on GitHub via:
https://github.com/UweM/Splitris

Documentation for “Splitris”

Problem Statement:
Tetris is way too easy and also boring for a single person on a single screen. We try to improve this disastrous situation by creating an enhanced version of Tetris, by enabling a multiplayer mode across multiple Android smartphones.

Our idea:

1. Basic Idea:
The basic idea we had was implementing a multiplayer version of the old-school classic mind game "Tetris". Our multiplayer version of Tetris is to be played cooperatively across multiple Android smartphones. Given the case that 2 or more players are playing a cooperative game, the smartphone displays of all currently connected users is used as one huge field of play. This means that all players share one common game area and try to cooperatively fill a line on the screen.

2. Game play:
The first participant will host the game. All following players meet in a lobby at the beginning of the game. The common playfield is extended horizontally to all phones (on which side could be chosen by the host). Players just have one "Tetriminos" (a block in Tetris terminology) to position on his playfield. The player on whose screen the Tetriminos currently is can control it via touch gestures. A new Tetriminos is dropped on a randomly chosen screen of one of the players. The Tetriminos then drops to the bottom like in normal Tetris, but in the multiplayer version the players can send the block to another player by flipping it out of its screen with a touch gesture. To score a point the players have to fill a line over their shared playfield to get points. The game is lost, when the Tetriminos reaches the top.

Technologies:
Using common smartphones having stock Android (version 4.4 and 5)
Using Java as programming language

Our solution:
For this modification of the classic game Tetris, we use our aforementioned library “libsplitscreen” to easily share the Tetris screen amongst multiple smartphones.

For controlling the Tetriminos currently showing up on the local screen of the running Splitris game we included a more fancy control than standard left/right controls:

All left/right/up/down controls can also be controlled by using the swipe technology. Therefore a left swipe results in a left move of the Tetrimino, a right swipe results in a right move, a swipe to the top results in turning the current Tetrimino by 90 degrees, and lastly the swipe to bottom move results in a fast drop down of the current Tetrimino.

Additionally we use the virtualscreen feature of our libsplitscreen to display two separate and completely independent views on the same screen. Both views are rendered by the server, transmitted via the network, and then displayed on the local Android device. For our Splitris implementation, we use a huge virtualscreen as game field, and a smaller virtualscreen located below the game field for displaying the next Tetrimino available in the game.

Implementation:
See attached source code which is properly documented and also available on GitHub via:
https://github.com/UweM/Splitris

Documentation for “SplitImg”

Problem Statement:
It is not so easy to share a single image over multiple Android smartphones, as it is not foreseen by default in the Android libraries. Therefore it would be a great idea to split a single huge image into multiple smaller images, being able to be displayed on multiple single smartphone displays. This feature can then be used to create a puzzle application, where the users must rebuild the original image.

Our idea:
The basic idea we had was implementing a multiscreen version of a classic offline puzzle. The application loads a huge image, or rescales a given image to the necessary size, and then displays it over multiple Android smartphones. The idea behind this is that “SplitImg” can then split up the entire huge image into multiple smaller images, fitting on the specific displays of the participating smartphones, and then randomize the displayed order. The users then have to coordinate amongst each other by physically relocating their smartphones in order the recreate the original image.

Technologies:
Using common smartphones having stock Android (version 4.4 and 5)
Using Java as programming language

Our solution:
For this modification of a classic offline puzzle, we use our aforementioned library “libsplitscreen” to easily share the screen amongst multiple smartphones.

Implementation:
See attached source code which is properly documented and also available on GitHub via:
https://github.com/UweM/Splitris