Tools

We have both used the ADT Bundle

Felix :
I have had nothing but problems with Eclipse. Problems include, not being able to create new Activities from the new File menu. The GUI editor did not work at all. And Eclipse crashed extremely frequent. This resulted in that I only used it for actually running the program and doing all the editing of files in a normal text editor. This is weird since I do not have any problems with my normal Eclipse Kepler without the Android tools. I did try reinstalling it several times/ downloading but to no avail. This was on my lubuntu system, on my old Windows 7 I did not have these problems, but were unable to connect a android device that I could run the application on. I did not use the virtual device since our app utilized OpenGL ES and the virtual device is simply to slow to run it. I cannot test the game when it is running so much slower than intended.

Andreas :
I've been using eclipse ADT bundle under windows, it has been running fine for me only running into some minor problems. While editing the layout with the graphical editor I couldn't place text inputs, if I did the editor would "crash" and wouldn't update the live view anymore. I also had some problems while trying to get my phone hooked up to the device selection in ADT. This turned out to be caused by my phone not using the default drivers, Ihad to download HTC's custom drivers for the phone. I never ended up using any virtual devices simply because I thought it was too slow.

API:s

We used OpenGL ES 2.0 to create the visual interface of our games. The positive thing about it was that it is open and well documented. OpenGL performance is very good in comparison to other alternatives. It is also good to have such low level control as OpenGL offers since it is "close to the metal". This low level gives us the ability to control the smallest of details. Investing time in learning it will provide you with a very powerful Graphics tool. However this comes with a price since you need to understand a fair amount of how modern Graphics work to do very simple tasks, like drawing a set of triangles instead of just instantly drawing a complex shape. The bad thing about it was that Andreas had never used OpenGL before and it took up some time to get used to it.

Developer documentation

We started out with the idea of making a remote keyboard for desktop computers that would communicate over wifi. We didn’t realize when we picked this idea that there would be alot of trouble. The first trouble we ran into was that we forgot to enable networking of the application by giving it permissions to access the internet. This was easy to solve once we realized what we forgot though. The second problem we encountered was that the application kept crashing every time we actually tried to create a socket connection. The debug print was really of no help, but after some searching on the internet we found out that that the socket connection had to be run in a separate thread. When you think about it, it’s kind of obvious since the only thing you have to interact with on a cellphone is its graphical interface and if you use the same thread to wait for a socket connection, the screen would just freeze and become unresponsive. We found that there were multiple ways to do this but we thought that this project was a bit too much for us to handle at the moment and Felix had already been experimenting with some other stuff (OpenGL) on the side.

We decided to change our project into a set of single player games based upon the classical game Pong. It ended up being three games, an impossible pong where the goal is to survive as long as possible. A breakout game where you can win by destroying all the bricks and a game where you are supposed to dodge balls coming at you at an increasing speed.

During the development of this project we have suffered a lot from technical difficulties, mainly that eclipse crashes constantly on Felix system (lubuntu). He has therefore mainly been working with a simple text editor for editing and eclipse only for running and debugging. Strange errors like not being able to create new Activities in eclipse forced me to use and old computer with a Windows 7 OS. Here eclipse was stable, but I was unable to run the project on a physical device. This all resulted in a very tedious and slow workflow where I had to switch computer for what I needed to do. I would estimate that at least 75% of my time spent on the project was wasted on technical problems instead of actually programming.
We also had some problems with irrelevant files causing conflicts when committing from different systems to our github repository, we do not have a deep enough knowledge about git to figure out how to exclude these irrelevant files from the commits. Making us unable to work at the same time from different computers.

In this project we have learned the following.

- Creating buttons and make them useful.
- Editing xml and set up necessary things in the project.
- Switching activities.
- Accessing the view components of an Activity.
- Using GLSurfaceView to use OpenGL 2.0.
- The fundamentals for setting up and work with OpenGL 2.0 including.
- Creating vertex and fragment shaders
- Creating an OpenGL program
- Linking it all together
- Using vertex attributes to put data so that the shaders can access it.
- Learned to use uniforms variables.
- Drawing triangles with the help of vertices.
- Controlling the color of the fragments/pixels.
- Using an orthographic projection matrix so solve the different screen size problem.

Android platform

In general we did not really like how androids echo system was designed. We thought it was overly complicated to create a basic application even when the ADT bundle environment did most of the setup work for you. We would rather see that it was simpler to create a basic application by just creating one main class and running it. It felt weird to have all the strings in a different file, might be because we're just not used to working this way. We also reacted over the fact that you would have to typecast when accessing different components.