Individual Report from the project “Computer Security Visualisation”

Albert “Errendir” Siddhartha Slawinski

May 13, 2013

Another story ends...

As a member of the Clockwork Dragon Team I’ve been working on the Security Visualisation project for over three months. In this time the five of us became the rock band of developers, and now our hit song is getting platinum. Three and a half thousands of lines of code is not much for a pro, but for me it might be more than I’ve ever written for a single cause in my life. There is no world in which I could have done this without the support and concentration of the team, errorless distribution of responsibilities and fully devoted fans. Now when we collect our award there is only one thing left to say: “We could have done better”. But also we could have done *less* better than other teams and that’s always the factor that decides the winner.

Two of the visualisations we presented are authored by me. Both of them are related and both put to use the same idea of hexagonal tesselation of the plane.

A road towards implementation

First rough implementation of the hexagonal map was commited to github on 16th of April and was far different from the final product. Had I known how I would end up stitching it all together I would have taken a completely different route. That is really what separates a great programmer from a merely good one —which I am neither. A great programmer refuses to write a single line of code or pseudocode without having a complete picture of the task. Had we been less of a rock band, jamming in the garage, and more of the responsible programmers, we would have done much better. But that is only clear in the hindsight.

In the weeks following my first commit I was busy trying to get a hold of OpenGL and trying to bring back the nostalgic memories of the 15 years old me drawing a single triangle with DirectX. I split my program into clearly separated subroutines, but completely ignored any knowledge about object oriented design I had. The first code I produced wouldn't meet any standards of any kind. But when the first rough version, with the first rough graphics was made I realized that I need to changer something...

Design? What is that?

It's always good to show your work to somebody who can distance himself/herself from it. Just like a rock band that need this one person that will always say what they truly think, I needed somebody from outside of the group. Somebody to asses the riffs I play and tell my why they don't harmonize. Our team had a great wisdom to split our work in a way that allows to 'mute' all the other instruments and just listen to one. With a basic network-packet logic we could have very easily develop indepentet visualisations. That made it really easy for me to separate "my piece" and show it to my brother who gave me a very important lesson about design:

- Be clear
- Lapidarity starts familiarity
- Lead the users to what you want them to see

Final Product

After making some hard design decisions and repackaging the code in the proper Object Oriented way I was close to finilising my work. Both the visualisations were working and after the whole night coding session in the basements of the CS department the tiling algorithm was ready. This meant that the hexagons I was so tidiously putting together could finally be grouped and placed on the dynamicaly scaling map.

We're live in 4... 3... 2...

Of course the most important part was still ahead of us. The Presentation. The Big Day. I am proud to say that when we came down to the CS department that day, we came down prepared. Suited up, with my very heavy but powerful PC and one hell of the gig to perform. What played the most important role in the process of impressing the judges was concentration. Concentration on the task of fully presenting our apps capabilities. All of us stood by our product and unlike other teams, we didn't wonder around looking for a shiny objects to look at. We explained our app to anyone who approached and we were honest disregarding who we talked to. Also the factor of "Actually having something to show" played a vital role. Quite a few of other teams were just showing mockup (second game - has anyone seen it?), not-working prototypes (mapping robot) or quickly duct-taped together park-assist devices.

It still was a shock when we won. We expected to do well, but not winningly well. As we were told later on it's the versatility of our app that really spoke to the judges. I believe that the semi-professional approach and a wonderful presentation were important factors as well.

Problems along the way

Of course it's quite a shame that so much work was just to make the OpenGL behave. Of course this knowledge will be useful, but there maybe we could have been better off using some already existing display/game engine.

Also, we could have tied our visualisations closer. Both of my map presentations were completely independent from any other visualisation. The great opportunity of having a one self-complementing product was missed.

Team assesment

Let me now give a brief assesment of every single member of my group:

James quickly became and unofficial manager of our band. Scheduling our meeting, writing to the sponsors, assiging tasks was his daily job. He presented great leadership skills and perfect overview of the project. Managing github issues and mile

Dominik was the one who always made sure that there is a decision made on a meeting. He did a great job writting down reports and specifications. He kept us alert and motivated.

Sergiu turned out to be the dark horse of our team. With great coding skills and brilliant ideas he contributed wonderful functionalities. He also had a really hard job of tying few visualisations together which he did perfectlty

Thomas gave a presentation as wonderful as the ones given by Steve Jobs. It's not too much to say that we owe him our victory. He also played a very important role in developing early stage logic, which allowed other team members to work in a stable and versitile environment.

Albert (that is of course me) presented nicely looking visualisations, and even though his work was always a bit "on the outter side of the project" his code found a place in the program and in the presentation and gave a non neglectable impression on the viewers.

Reflections

In the ideal team, does anyone learn anything? Truly understanding something means that you can do it by yourself and therefore you don't need a team.

This is not true in our case. A complex project consists of multiple scopes and I think each of us found his scope and learned something about it.

Crossroads of destiny?

What will happen to the project now? What path will any of us choose?

As it happens to almost every rock band, there comes a time to go solo. Will our separate work be as good as colective?

No way of telling...