Personal Appraisal

Alastair Campbell

May 7, 2014

1 Transferable Skills

I believe that the most important transferable skill in my course is programming. It is the primary reason I am on the course and what I will be doing once I am employed. Employers will look to programming ability as the main metric by which to hire a programmer - it's in the job description, no less. I have definitely used programming in my computing project coursework. The game engine has thousands of lines of code, some of it complex. I have written a good amount of object-oriented C# code in the project.

However, writing code isn't useful if the project is not well planned. I believe that planning is also a vital transferable skill in this course and for employment. If a project is not well-planned, it will never succeed, regardless of the budget or how many people are working on it. Employers are more likely to want employees capable of planning projects such as the the one we planned for our Computing Project. We have definitely used planning quite intensively during our project. We planned out each element of our game long before we thought about implementing it. We drew diagrams of what we wanted our levels, characters and user interface to look like. We also planned out how we should manage the project.

Project management is also a very important transferable skill. Employers will be looking for candidates whom have a good background of successfully managing projects, especially for project manager roles or for consultancy. I believe that a project, no matter how small, requires at least some management. We have definitely used project management as part of our project - our project was planned vigorously using Microsoft Project, establishing a time-line, Gantt chart and a critical path for our project. However, even the best planned and well-managed project will fail if the project team doesn't work together.

I believe that teamwork is a significant transferable skill which is often overlooked. Teamwork is one of the "soft-skills" which are so well-desired among technical positions such as programming, due to the fact that people that work together in a team get more done than one person can do alone. We have definitely needed to use teamwork in our project. To work together and decide who should work on what is the most important part of working in a team. We could not have accomplished what we did in the same timespan if our team did not work together to get things done. Good teamwork, however, requires good communication.

Communication is another noteworthy transferable skill which is highly valued by employers. Good communication is required to perform well in an interview for a position as well as succeeding in that position once you have it. Software developers always have to communicate with their customers or clients to learn what they need, and communication is key to establishing a good relationship.

We have definitely required a good deal of communication during our project, since without it we wouldn't have done any work at all.

2 Challenges

As a group, we encountered several problems while working on our project. Firstly, we had issues with the XNA 4.0 Framework. Support for this was dropped by Microsoft and it has not been updated to be fully compatible with Visual Studio 2013, which we needed to use for compatibility with our `git` repository. There were a few solutions for this. Firstly, we could switch our code to use MonoGame, an open-source implementation of the XNA Framework, but this was not without flaws itself, and compatibility with Visual Studio 2013 was only a beta feature. Secondly, we could switch our version control system to Team Foundation Server, which would make managing code in Visual Studio easier and compatible with Visual Studio 2010, the last version of the software compatible with XNA Framework 4.0. As a resolution, we decided to switch to the MonoGame framework, which worked better with Visual Studio 2013 and everything seemed to work well. It was also far easier to setup with the rest of our team. However, MonoGame itself led to other problems...

MonoGame seemed to work well with our code for a long time. However, when we tried to introduce text to our game, we could not create the SpriteFont files using MonoGame (it just crashed when you tried to start compiling the font). We found a content compiler standalone application that needed to be built on a computer with XNA 4.0 installed, and luckily my desktop PC still had XNA 4.0 on it and we were able to compile the binary and use it to convert fonts into sprite files. This allowed us to continue working with MonoGame.

While working with `git` we had a few issues because there was a huge zip file (over 500MiB) which had to be downloaded every time the repo was cloned. This caused hours of delays when a fresh copy of the `git` repository had to be pulled, even after the zip was removed from the repository. We looked through the `git` command list and found one to remove all files from the history of the entire repository to remove it.

3 Working as a group

3.1 Effective groupwork practice

In a group, working together is highly important and often underestimated. Maximizing group work efficiency is time well spent. With a view to this, I have compiled a list of what I believe are some of the key do's and don't's of groupwork.

3.2 Do's

Communicate with your team Communicating with your fellow teammates is very important because not only do you get an idea of what everyone is thinking but your team's experience and knowledge will always outweigh each members' individual experience, no matter what. Team members may have experience with common issues or "gotchas" that you may not be aware of, and working together you may come up with some solutions that you would have never come up with individually.

Keep members up to date with your work If all of your team members know that you are working on implementing sound handling into a game engine, no one else is going to duplicate that work and instead they may work on an unrelated task. If everyone just works on whatever they feel like doing not only will work be duplicated but no one will have an idea of how the project is developing as a whole. Worse still, this can start competition between supposedly co-operative groupwork.

Let your team know if you solved a problem Other members of your team may have the same, or similar issue, or your insight may prove valuable. In fact, your team members may come across an identical issue in the future and remember how you helped them to solve it.

Establish a group work repository Don't keep duplicated copies of work or use a flash drive or similar device to share work. Use a web-based version control system so that members aren't afraid of losing data or messing up your work. Faulty revisions can be undone at the press of a key and there's never any risk of work loss. It also prevents issues with emailing each other older versions of the work or being stopped from working because you don't have the right revision of a piece of code.

Give the project a codename It gives the team a sense of community and cohesion, as well as a personality.

3.3 Don't's

Neglect to support your teammates Starting work on something can be daunting. It is definitely worth motivating your other team members to get them excited about the project and trying to get the best results possible. This will work in a cycle - you motivating other members will result in them motivating you when your morale is low also.

4 Project Management

Project Management, as the title suggests, is an important aspect of a project. Managing how the project is going and how it should progress can make or break a project, however doing so can be more complex than it immediately appears.

I have compiled a list of "do's" and "don't's" regarding working with managing projects.

4.1 Do's

Plan out what tasks must be done These don't need to be too specific, as long as everything is "caught" without requiring an "other" category it should be okay. Generic tasks such as "write game engine" is acceptable as long as enough time is dedicated to it, however they could be improved with subtasks.

Roughly estimate the time for each task Is a task going to take an hour or a week? This also helps to show if a project is likely to become overdue and whether or not it needs to have more time invested into it or needs to be re-evaluated for credibility.

Decide what must be done for other tasks to begin Establishing a task hierarchy shows you what needs to be done first so you know what tasks can be worked on independently of each other. This allows multiple team members to work on separate tasks.

Consider an agile methodology The agile manifesto states that collaboration and working software are more important than comprehensive documentation and contract negotiation. This is something to consider. Responding to change rather than following a plan strictly allows your team to react to change rapidly and remain more agile.

4.2 Don't's

Spend too much time planning The core of the project is the deliverable. Having a highly detailed plan but nothing to show for it gets you nowhere. If you spend a day working on a plan, that is a day you could have spent implementing what you have already planned.

Be too restrictive in your planning Sometimes an over-engineered plan can cost more time than it saves. A plan, especially in software, should be flexible. If your plan is too heavy then you may not be adaptive or responsive enough to react to changes or issues.

Use the project plan as an excuse for not working Just because you aren't scheduled to be working on a certain thing and need to wait for a component to be finished before starting work on another module doesn't mean that you can't do anything at all. You could begin working on some component that doesn't rely on what's unfinished or you could document what you've already done to make it easier to maintain.