CPD Report

COMP240

1607804

May 6, 2018

When starting this course, almost two years ago now, I wanted to make games. My career goals were to become a programmer who made games. After the game making opportunities given to me, I have found that my career goals have changed. Initially, the small challenges and problems that came with creating and implementing standard game mechanics were interesting and unique. Same too for the implementation of UI components. However, these tasks have started to seem repetitive, more like a grind than an interesting problem to solve. Modules such as the MicroRTS bot have allowed me to realise the novelty and joy I can find when using my programming knowledge to solve new and engaging problems. When the opportunity arose to work on PocketPals a new app that will use GeoLocation services, I knew the challenge and uniqueness of the problem presented would prove enjoyable. I was not wrong. My career goals involve finding a job that will stretch my creativity and challenge my knowledge forcing me to learn new things. Whether it's in the games industry or in IT it matters not, provided these criteria are met.

1 Affective Domain: Letting my frustration towards poorly designed systems influence my actions.

Often, in the industry professionals are required to start working on existing systems. Whether you're trying to fix bugs or implement something new it can be frustrating especially when the quality of the code base is sub-par. The MicroRTS game while fun was a terribly designed system. I soon found myself frustrated with the lack of clarity. As a result, I delete large chunks of my work when it didn't run correctly. In addition, I found myself reluctant to start working on it.

S. Spend at least two hours investigating and trying to understand a system before trying to implement anything.

M. I will keep track of the hours spent working out how a system works. Even if I feel I have a good grasp on a system, I will exhaust the full two hours.

A. Starting with a better understanding of a system will make me less likely to make large implementation errors. Which in turn makes me less likely to become frustrated.

R. It might be difficult to study a trivial system for two hours. With larger systems more time will probably be needed.

T. I will practice over the summer, by using mod kits for games like ARK to test if this action will lower my frustration.

2 Interpersonal Domain: To better communicate to colleagues what I can deliver and what I can not.

Unsuccessfully communicating what I can and cannot deliver will negatively affect my interpersonal relationships with my colleagues. Doing this in the games industry could give me a reputation for being unreliable. This semester I had agreed to produce a game component for a colleague. Because of my inability to communicate what I could do within the time frame, they were expecting a more polished component than I could deliver.

S. Increase colleague satisfaction to 95% on any deliverable I produce.

M. I will count the number of times what I deliver satisfies and fails what my colleague believes to be the agreed deliverable.

A. I will draw rough Gantt charts of the workflow to communicate to my colleagues what can be done. This way we I can more effectively communicate to my colleague what I will and will not be able to deliver. Reducing the chance of unachievable expectation from my colleagues.

R. By having a solid percentage I can see how good I am at communicating with my colleagues. If I consistently fail to meet their expectation, I know I need to reduce the scope of my projects.

T. Over the summer I will be working on a milestone basis with PocketPals. By the end of summer, I will hopefully have a 95% or more delivery ratio.

3 Dispositional Domain: To effectively evaluate when investing more time into to a problem is not worth the impact the solved problem will have.

In the Games industry, the ability to order my user stories based on how much effort they will be and how much impact they will have on the project will be useful. It will allow the game to progress at a faster rate, as the highest impact and lowest effort tasks should be completed first. Often low effort tasks become high effort when unforeseen circumstances arise. I believe being able to evaluate a cutoff point, where anymore invested time to a singular problem has little to no return. This semester, I found myself tweaking parameters on my AI bot to improve its performance at a certain map. I spent a lot of time, for little to no impact. This time would have been better spent refactoring or commenting.

S. To become more proficient at evaluating how much effort/impact a task will take.

M. I will set a reminder at the start of each working day. I measure the number of user stories I can complete during a given week.

A. By reordering at the start of each day, A problem that I initially thought of as low effort but turned out to be high effort can be pushed to the back of the backlog. This means I can re-pick a task at the start of each day if the previous days' task was more effort than expected

R. Its hard to know actually how much effort/impact competition of a task will have. To mitigate this uncertainty revaluating each day should allow a more dynamic backlog.

T. I hope to become better at evaluating the effort and impact a task will have by the end of my time at University.

4 Cognitive Domain: To learn how C++ and C# can be used together to write multiplatform plugins

Both indie and AAA game studios use Unity and Unreal game engines. Being able to write one application that can be used in two of the biggest game engines in the industry at the moment, will make me very employable. I am adequate at writing code in both C# and C++ programming languages, I do not know how to use the various tools available to write multilanguage applications. For my dissertation project next year I plan on writing a plugin for both Unity and Unreal, scope permitting.

S. To build 3 simple algorithms that can be used in both Unreal Engine and Unity Engine.

M. I will measure my success by the complexity of the algorithms and whether or not they work as intended.

A. To achieve this goal I will use various tutorials online and such as: https://ericeastwood.com/blog/17/unity-and-dlls-c-managed-and-c-unmanaged

R. From a small bit of research, it seems easy to write programs in c++ for c#. I believe the most challenging part will come when dealing with game engine integration.

T. I will need to be comfortable with the relative technologies before I start my third year of university.

5 Procedural Domain: Increase my proficiency in using UML to express system architecture

When explaining my code structure I often find it difficult to articulate the architecture of my code. This makes it much harder for people to understand my code. Which in turn makes it harder for them to work with me. This semester, I have had two different opportunities to express my code solution in the form of a UML diagram. One of these times I passed up the opportunity to make one. The other I made a mess of it, making my work only slightly easier to understand.

S. Create two full UML diagrams both for non-trivial systems. Including a UML that considers both structural and dynamic behaviours. In addition, the overall object hierarchy.

M. I will count the UML diagrams I complete. I will give these diagrams to peers to get general feedback on how helpful they are.

A. This will be a learn by doing approach. Hopefully, after a few detailed UML diagrams, the quicker draft versions will come more easily to me.

R. UML is perfectly achievable, I just need to spend some time looking at examples. When I become more practiced a more open approach could be adopted.

T. By the end of the summer, I will have created two different UML diagrams of systems I implement for the game PocketPals.