Reflection Journal

The final project is the largest programming project (and certainly most open-ended) I have worked on to date. Due to this fact I wanted to write a reflection *journal* as opposed to a simple short reflection of the whole experience after the fact. This will be beneficial by allowing me to organize my thoughts for the next steps in a meaningful fashion and more thoroughly express my experience at each phase of the project. **If you're only interested in the synopsis, please see the entry dated 12/3/2016**.

Important Side Note: You may notice that my derived classes are from "Tile" and the pointers to locations are in my "Location" class instead of having them be part of the same class. I did get this approved by the professor as it still remains within the spirit of the project (polymorphism and pointers).

11/25/2016

I have now made some decent progress for the final project. As this is the first entry in the final project journal, I will provide a summary of the last several days and discuss some challenges that I have overcome. As soon as the project was posted I was elated at the concept of designing a game (this is what I want to do after all), but I needed to make sure the Lab and Test #4 were complete before I could do anything to make sure that there were as few disruptions as possible during the process. Between Monday and Wednesday I had been tossing around various ideas on what I could do with the project, and I decided to settle on the (admittedly difficult) concept of a MUD-style game. Planning this was immensely difficult.

I had several ideas as to what the game *could* be based on my past experience (first- and second-hand) and videos I've seen; but I found myself constantly needing to whittle down my ideas to very basic core concepts. It became very easy for me to mentally inflate my project with loads of unnecessary and completely impossible features without a several-month deadline and a team of people to work with. Eventually I was able to pare down my ideas to meet *just* the basic criteria, and coming to terms with the fact that, while I want to always create great and entertaining games, this project isn't about that. It's about making sure I understand the programming concepts in this class. I know that there was a warning on the project description about this but there are no amount of warnings that will ever get me to settle that in my heart from the beginning.

Now that I had my core concepts mentally settled I needed to plan things out so I turned to my trusty whiteboard (if anyone is reading this – I recommend you invest in a 3x4 or larger whiteboard ASAP, they're lifesavers) and started in on mapping out potential classes, things that need to be done, how actions can be classified, etc. I struggled moving much further than concepts though. I knew that I wanted to create a game where the player moves around on the map, but I had no frame of reference for the level of difficulty that this would entail. Trying to figure out how long this would take, or if this was even feasible at my skill level, was becoming an arduous task. This was a huge turning point for this project. This is where the project moved from concept to reality. I decided that if I couldn't properly plan it without experiencing it first, then I may as well dive in and see if I can achieve my goals.

I started out a grade above very simple. I knew that each room would need two types of (what I now call) Tiles – at the very least it would need walls and floors (depicted as '#' and '.' respectively). I began writing some functions that would create a dynamic 2D array based on parameters I've fed it, then conditionally instantiate either a wall or floor based on the array position. This took a very short amount of time and it eased quite a large amount of troubles in my mind. I'll admit that, though it's cheesy, I smiled when I saw this work. I finally thought, "Wow! I can do this!"

At this point I had dispensed with the traditional method of planning out the project in a document. The issue I've taken with this approach is that it's slow, clunky, hinders productive thinking,

and is all around limited in its capabilities. My whiteboard is my thinking space and it's where I find I best plan out the phases of a project. Since my whiteboard is obviously not large enough to contain this entire project, I decided to take pictures of it during the process in case I erased my work and needed to reference it later (these images will be provided). Returning to my whiteboard I began planning out the basics of how these maps could be structured, how things would interconnect, and thus began the basic form of this project.

Testing this actually took several forms (only one of which will be written in my test plan for simplicity) but I found that the most difficult part of the testing for the map creation section was not in trying to get the walls/floors/doors laid out properly on each map; but rather, it was in managing the memory. I walked into this with the idea that I could use static memory since the maps are static – very few things will actually "move," but I was wrong to think this. My desire from the outset was, as I stated earlier, to have a single function that would be fed parameters and generate a map based on those parameters. This is in direct contradiction to my wanting the maps to be pre-generated.

Once I settled on this idea I began planning out how I would handle all of the dynamic memory, but experience was the best teacher for this. I knew that, with this many dynamic arrays and dynamic memory for each item in the dynamic arrays, I could easily be overwhelmed if I waited until the project was complete to handle memory leaks. Approaching this in pieces was definitely the better way to go. I didn't run into many memory leaks, but one of the issues I encountered was indeed *very* difficult to locate. The problem was feigning that it dealt with how I instantiated my maps as the number of leaks directly correlated with however many rows my 2D array had. The issue wasn't here, however; it was in my not deleting the row itself after all of the contents were deleted. This took approximately 2 hours to troubleshoot and set my production schedule back quite a bit as I wanted to use this time to make sure I could properly generate all of my maps. Once this was fixed I was able to address the next steps of the process (making sure all of the maps generated properly, all of the memory was deleted, etc.). While there were no critical errors that I ran into, this process did take several hours more than anticipated.

The next phase that I am now working on is player movement. I am planning this out on my whiteboard, but I am not sure how to properly implement this just yet. The main ideas I have thought of are somewhat hacky workarounds (which may have to do if I spin my tires for too long on this). I'm finding that when one idea crops up, I start planning it out and I find that it's not a scalable idea. It will only work in simple scenarios. I will obviously overcome this challenge, but it is (so far) the most burdensome to tackle. As a brief summary of why this is difficult: I do not want the Player class to be forced to know its coordinates on the map as I have a Controller class that is intended to handle this. If the Tile (and children) class, Player class, and Location class are thought of as body parts, the Controller class is the brain. It unifies everything and makes it a cohesive, interconnected whole. The problem is I don't currently have a setup that allows for the player to move because once it is initially placed, the coordinates are essentially lost. Whatever method I decide on, I need to make sure that I remember this player will be crossing into various rooms as well, so it needs to "cross" maps – resetting any coordinates that it may initially have. I will continue to ponder this and return with an updated entry later.

11/26/2016

After working for an additional ~5 hours on this project, I have successfully implemented the player movement all across the map. I wasn't sure yesterday what kind of implementation was necessary for this to work; but after working out the logic on my trusty whiteboard I was able to plan this portion out with relative ease. I did run into difficulty when trying to plan the "cross-map" feature that allows the player to move from one room to the next – this took approximately 30 minutes – 1 hour of planning to fully conceptualize. I was able to return to coding when I came to the conclusion that all of my doors (and probably any that I would later implement if I desired changes) would reside on the

edges of each map (people don't traditionally go through a door in the middle of a room and end up in a new room). This "Aha!" moment led me to implement a movement condition that told the player to move to the "X" direction room if they were on a specific edge and used a specific movement key that would move them out of bounds.

In order to implement everything I had to make a few minor changes, namely to the Door class. What I ended up needing was a means of storing (somewhere) the coordinates for the next room over. Since this was specific to the door that was being used I created some integers that hold the door coordinates for the player to appear at when they venture into the next room. This solution is not my ideal situation as it presents potentially error-prone areas. The major issue I can see on the surface is that if the room sizes were to change, I would need to make sure that the coordinates for the Doors change as well. Thinking on this briefly I decided that this was worth the potential problems as creating logic that would try to detect the room's length/width and assign the door appropriately would require more coding than I was comfortable doing for a project of this scale. This would *definitely* require a change should the project's scale change to be akin to the size of a multi-level dungeon crawler – it would be too much work to reassign a large number of doors manually if you found out that you made a small mistake in the room sizes.

Testing for this phase included making sure that the player could traverse into and out of each room *and* appeared where they should in each room. I had a few doors that were locked to test that they would be prevented from moving to that location (which did work) but I found that I had an issue with the player sometimes appearing in the wrong room. Earlier bugs included the game crashing due to segmentation faults. The latter of these was due to my copy-pasting swaths of code and forgetting to make the minor (but highly necessary) changes to the conditional statements. Copy-paste is both a wonderful friend and terrible, fickle enemy. The former issue was due to my accidentally setting the doors in the wrong spots. Once this was fixed I made sure that Valgrind gave me the greenlight – meaning that no memory leaks were detected. Another weight was off of my shoulders.

The next phase I will be working on is interactivity. While the player is able to navigate through the level, there is currently no engagement in the game. The player simply walks through multiple rooms and does nothing. I will be planning out item pickups, item storage, special room-objects that the player can interact with, showing the player where items are, and providing the player with menu options that will allow them to interact with their environment. This is the meatiest part of the project, but it is the second-to-last layer of the metaphorical pyramid. Once this is done, I will this design to construct a short (but hopefully engaging) story for the player to walk through. After I have completed the story sections, I will then implement the game-end mechanic and the winning condition. I am choosing to implement this last as I want to theme this appropriately with the story (though I am fairly certain that I know how this will operate). I will report back once I have made some progress in these areas.

11/27/2016

I worked tirelessly through the night to make sure that appropriate progress was made in the department of interactivity. I believe I went to bed around 1:30-1:45AM, but the progress I made caught me up substantially in my desired production schedule. I planned out a couple of goals for myself to accomplish for the evening that would ensure that I was (at the very least) on the right track. The first goal was the most basic principle – if a floor tile holds an item, it should be set to show "!" instead of a "." as it normally would. The second goal required just a little more work: the player should be able to walk over to a tile and pick up the item that the floor contains. The last goal had more concept involved than firm thoughts so I perceived as a "get this started" type of goal. This was handling the player's "investigate" actions. I want the player to be able to look at the pieces that make

up their environment so they can have a more immersive experience. While this will be a lot of tedious work to fully implement, I feel that it is important from a game design perspective.

I met my first goal fairly quick. It was not difficult at all to implement a conditional symbol to be displayed. I did run into a few minor troubles, however, when I began implementing adding items to the player's inventory. The main issue I ran into was a simple fix, but took me a while to figure out due to simple oversight. When I attempted to pick up an item, the program would crash. I had a lot of trouble figuring out what would cause this. I went through and made sure that my "set item" function was working, the floor tile was working, etc. After a long bout of troubleshooting, I found that the error was in that while I wrote an "initialize item" function, I did not actually ever call it. This served as a humbling reminder that I needed to take more caution when working on these large scale projects as it becomes very easy to overlook minor details that will severely break the program.

Once I had the necessary functions written for the player to begin basic interactions with the world, I decided to move my "test" input functions from Main.cpp to my Control class. This move was easy, but I took it slow to make sure that my input validation worked the way I intended (clearing the buffer took a little bit of research) and that interactions operated as expected (this action began partially mixing with the investigate feature). Fortunately this area was bug free and required no additional testing/work.

With that feature now working, I found I was able to begin fleshing out the level design. I returned to my Notepad document to observe the rooms and find out how I could make them interesting and interactive. I began entering shapes in various areas to try and make them more visually appealing than the large, square, empty rooms they started out as. As I did this I discovered that I needed to create an additional document (which is titled "Room Planning"). This document would serve as my guide for what important objects would be in each room – both items and interactive tiles. During this process I discovered the joy of crafting story into the game by using "Show – Don't Tell" methodology (I give my thanks to the YouTube channel "Extra Credits" for reinforcing this idea). I have used some interactions with the environment as a means of unfolding corners of a larger story that unfolds more in the end. With that document finished I have a clearer idea of what I need to do in order to bring this project to completion. There are several more tiles to create, several more interactions to implement, and some story that is yet to be told. I will record another entry when I have made additional progress.

12/2/2016

I have been working hard over the last several days to make sure that this project reaches completion by the due date. There were a few roadblocks that I ran into that caused me quite a lot of trouble during this last hurdle so I will cover those in this entry. The first issue that caused me some trouble was extending the breadth of my interactions. I had previously implemented (without issue) basic interactions like picking up an item or investigating an object, but after this I was faced with the issue of creating interactive tiles (switches and terminals). I needed to do this in order to meet the specification of allowing the player to change the status of a room/tile. I needed these interactions to perform a variety of actions such as unlocks specific doors, accept only specific items as usable objects, and "move" things (air-quotes used because I didn't actually move a tile – I simply changed the symbol of them).

The complexity of this caused a massive slowness on my part because the necessary interactions were very specific to the situation that each room presented the player with. This, in addition to my desire for an easier-to-view display for the player, I needed to alter my pure virtual functions to return strings instead of using a void return. This actually proved quite useful because it allowed me to fully review the text that I would end up displaying to the player throughout the game whenever they interacted or investigated something. This was the largest hurdle I had over the last several days. It

didn't help that I had to work full time so I only had a couple of hours each night to make progress. Testing this became a bear to handle because I had so many different things I wanted to happen – but I needed to be cautious not to fall into the trap of "Just throw something in there to make it work" (though this did occur twice because I didn't see a viable workaround). Currently I am testing the game for memory leaks (found a consistent 37 blocks that are leaking) so I will be working on that until it is complete. After this I will be making sure that I have any necessary comments in my code, finishing the document write-up, and creating the walkthrough. My next entry will be my last for this project – summarizing my entire experience after briefly describing my memory leak testing.

12/3/2016 – FINAL ENTRY AND SYNOPSIS

I have completed all of the programming that is needed for this project so this will be my final entry in this journal. I will first cover my most recent issues, testing, etc. and then move into a synopsis of the project as a whole and how I feel about it. Almost all of the work I've performed since yesterday's entry was an attempt to fix a 37 block memory leak. I had a lot of trouble pinning down where this leak was coming from as valgrind isn't too keen on telling you what's causing the leak, simply that you have one. As I had ~2400 lines of code, this task was a bit daunting. Fortunately I could narrow this down to just the areas that I call "new," … or so I thought.

Several hours of tinkering (making minor edits, uploading to FLIP, compiling, and retesting the exit function) amounted to, what felt like 0% progress (though if you go with the Thomas Edison approach – I had learned that a lot of things were *not* the issue). Still, after seeing 37 blocks repeatedly pop up, I decided to go with the hammer approach and start commenting out large swathes of Location initializing and narrowing down my solution from there. This took me to a *very* interesting place. I found that the issue was partly coming from when I replace tiles with "named" Doors (meaning I use a string to name it). I didn't quite understand why this was the case because the constructor initialized everything the same as my other doors that weren't named. A couple more hours of tinkering and Google searching led to find out that the issue was with how my destructors worked. I do not yet know why (though researching this subject more in depth will undoubtedly inform me of the answer) but if I had a derived class that used a string in its constructor, this required my base class (Tile) to make use of a virtual destructor. I believe this is because my base class doesn't know about the string and therefore cannot destroy it, but I am unsure if this is correct.

The other portion of my time was finalizing my details in the project, ensuring that it was bug free, and all of my interactions worked as I wanted them to. I had my wife playtest this for me as she is a gamer and has some prior experience with playing text-based dungeon crawlers (my wife is awesome). I found some things that weren't displaying information the way I wanted them to so I quickly made some minor edits and had my wife finish her run-through. Having a fresh pair of eyes on it was a great boon as I received some input that I didn't think about because I was too close to the picture to see everything. Some of the feedback I was unable to implement because of the time-investment that would be required, but I was able to make corrections to several things brought up in this feedback. If it's not already recommended, I would suggest adding "Have someone playtest" as a recommended part of this process. It brings the programmer out of the hole that is the IDE and brings a fresh perspective on the project. I know this isn't about game design, but this can reveal glaring issues that may have been overlooked.

Now to the meat of the situation: how do I feel about this project as a whole? As I stated in the beginning, I didn't have a formal structure written down in a document prior to starting this project but instead I had more ambiguous concept that I was able to firm up as I designed the core code. After a large amount of work I was feeling like I was falling behind in my personal production schedule because I wasn't making as much visual progress as I wanted to be. I needed to lay out a lot of foundation for my plan to work and this included a lot of basic concepts like: generating 2D arrays, displaying 2D arrays,

developing player characteristics, implementing a control scheme, etc. Building this foundation felt a little bit like developing a level editor and then creating my levels using that editor.

Once I was at a satisfactory point in my development, I began fleshing out my levels on paper (Notepad really). I constructed a map of what I wanted the player to traverse through, came up with some interactive ideas as to how the player could do different things, and I began thinking of the different derived classes I would need in order to fulfill this level of programming. This flexibility allowed me to focus on the areas that I needed to in order to progress but came at the cost of the code not being organized in a way that I (eventually) wanted – which would have allowed for more, and easier, tile-to-tile interaction. This cost was only a small one, however, as taking the time to plan out everything ahead of time and implementing what I *wanted* to do would have taken me well past my allotted time for this project; so while the layout isn't the most ideal, it works for this project.

There were also several things I wanted to add in, but because of time, I needed to pick-and-choose what I wanted to accomplish (something that I hear is a frequent event in game development). Some things that I would have added if there were additional time for this include: colored tiles for visual appeal and ease of use (depending on my schedule I may still add this), items that cause special events, and combat. I have learned throughout my life that while I have large ambitions, I must learn where to make cuts and in this case I had to pare down quite a lot. This skill will only become more useful in life. I've spoken a lot about my testing experience in this journal so I will keep this next section brief.

In regards to testing I think that how I handled it (for the most part) was in ideal fashion. Since I was developing this in phases, I decided that testing at the end of each phase would also be appropriate. This served me well as I was able to detect several memory leaks early on that would have been extremely difficult to pinpoint later (e.g., the issue I had when deleting the 2D arrays). The virtual destructor issue I had could not have been found until I reached near-completion, though; so this, I'm afraid, was a necessary struggle for me to go through. In general, I found this project to be a very good learning experience. It required a lot of design decisions which in turn required a lot of research and resulted in a lot of confidence-building. I found out what I was capable of doing, learned about the importance of making informed design decisions and its long-reaching benefits/consequences (depending on how informed of a decision was made), and learned a few concepts that piqued my curiosity and may drive me to further my self-studies. It is my hope for this project that, while my documentation may be unorthodox, my game is perceived as high-quality in terms of general design, mechanics, and programmatically.