COS30031 Games Programming

Learning Summary Report

Mitchell Wright (100595153)

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (P)	Credit (C)	Distinction (D)	High Distinction (Low HD) | (High HD)	
Self-Assessment (please tick)	X				

Self-assessment Statement

	Included? (tick)
Learning Summary Report	X
Complete Pass (“core”) task work	X

Minimum Pass Checklist

	Included? (tick)
Additional non-core task work (or equivalent) in a private repository and accessible to staff account.	X
Spike Extension Report (for spike extensions)	
Custom Project plan (for D and/or low HD), and/or High HD Research Plan document (optional)	

Credit Checklist, in addition to Pass Checklist

	Included? (tick)
Custom Project Distinction Plan document, approved	
All associated work (code, data etc.) available to staff (private repository), for non-trivial custom program(s) of own design	
Custom Project “D” level documents , to document the program(s) (structure chart etc) including links to repository areas	

Distinction Checklist, in addition to Credit Checklist

	Included? (tick)
Custom Project “HD” level documents , to document the program(s) (structure chart etc) including links to repository areas	

Low High Distinction Checklist, in addition to Distinction Checklist

	Included? (tick)
High Distinction Plan document, approved	
High Distinction Report document, , which includes links to repository assets	
All associated work (code, data etc.) available to staff (private repository) for your research work	

High High Distinction (Research) Checklist, in addition to D/Low HD Checklist

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: *Mitchell Wright*

Introduction

This report summarises what I learnt in COS30031 Games Programming. It includes a self-assessment against the criteria described in the unit outline, a justification of the pieces included, details of the coverage of the unit's intended learning outcomes, and a reflection on my learning.

Overview of Pieces Included

This section outlines the pieces that I have included in my portfolio…

Task 4.C – Multithreaded GridWorld: Demonstrates my understanding of a basic Input, Update, Render game loop and additionally, multithreading.

Task 5.P – Debugging: Demonstrates my ability to identify and solve performance issues and bugs in existing code.

Task 7.P – Performance Measurement: Demonstrates my understanding of code performance and efficiency, as well as my ability to meaningfully measure and improve code performance.

Task 8.P – Game State Management: Demonstrates my understanding of the state pattern, as well as my ability to implement game loops and stack structures, maintaining states for later reuse.

Task 9.P – Game Data Structures: Demonstrates my ability to identify and implement efficient data structures with necessary functionality and modularity.

Task 11.P – Game Graphs from Data: Demonstrates my understanding of file IO, graph structures, and data formatting in order to create an extensible game world with various connection and entity types.

Task 12.P – Command Pattern: Demonstrates my understanding of patterns and my ability to implement code based on key programming concepts, as well as my understanding of modularity and extensibility when implementing it.

Task 13.P – Composite & Component Patterns: Demonstrates my understanding of inheritance, class structuring, and the Composite and Component patterns.

Task 16.P – Sound Board: Demonstrates my ability to learn and work within media frameworks, as well as making use of official documentation, and audio files.

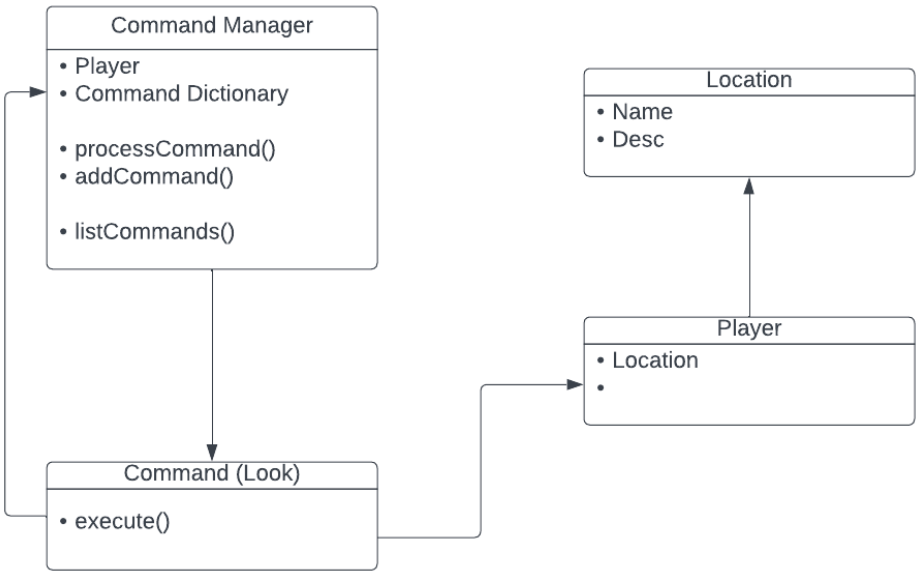
Task 22.P – Collisions: Further demonstrates my ability to learn from official documentation, as well as an understanding of built-in functions and collision types. Additionally demonstrates my understanding of how to work with sprites, rectangles and circles.

Task 24.P – Measuring Performance & Optimisations: Further demonstrates my understanding of code efficiency and factors measurement of performance.

Coverage of the Intended Learning Outcomes

This section outlines how the pieces I have included demonstrate the depth of my understanding in relation to each of the unit's intended learning outcomes.

ILO 1: Design



Each of the GridWorld and Zorkish tasks demonstrate my ability and understanding of designing a functional game architecture, constructed from modules/components for ease of scalability and functionality. Task 4.C shows that I am able to design a basic game loop and simple objects, with tasks 8.P through 13.P showing that I can build up a program from specific components, such as a world graph, game objects, managers, and so forth. This approach is necessary in order to create a readable and functional code base that while being able to be expanded upon, is also built well to begin with.

ILO 2: Implementation

Tasks 4.C and 10.P though to 22.P all show my ability to implement game engine components, such as a modular collision system in task 22.P. In that task, different types of collisions are able to be subbed in and out as needed. Additionally, in the Zorkish tasks, I managed to implement various patterns and concepts, such as the command, component and composite patterns. 16.P's SDL2_mixer API was also implemented effectively and demonstrated my ability to recognise when additional components were necessary for further functionality, as with base SDL2, programming the soundboard was far more complex and likely would have been much less efficient.

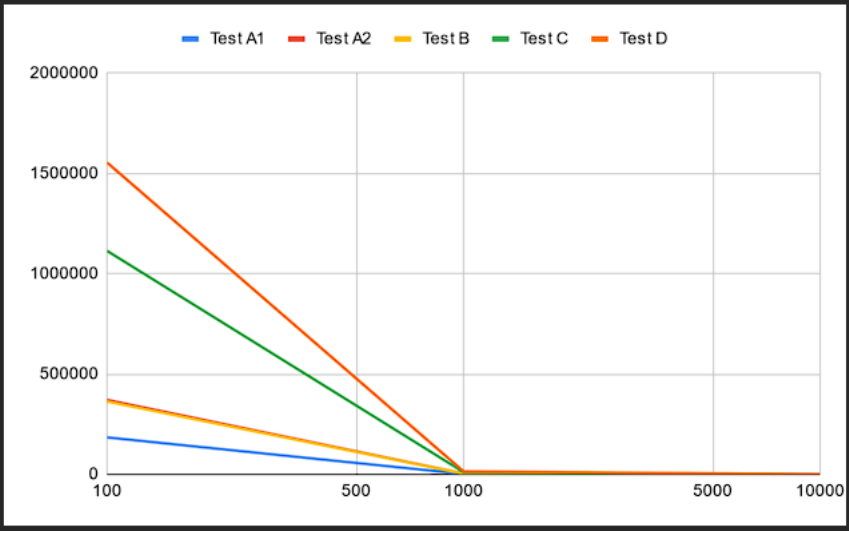
Create games that utilise and demonstrate game engine component functionality, including the implementation of components that encapsulate specific low-level APIs.

ILO 3: Performance

The primary tasks that demonstrate my understanding of code performance are tasks 7.P and 24.P. In these tasks, I spent time measuring and graphing out differing functions, using a methodical approach. In task 24.P specifically, I demonstrated my ability to understand a code base, improve upon it and show those improvements via relevant and readable metrics. This is important in programming games, as an efficient code base allows for better performance and a greater degree of flexibility in the hardware that said

games are able to run on. While within the industry, this will likely be done using specific tools, understanding the basics of measuring and comparing code is a necessity.

Number of Boxes	Test A1	Test A2	Test B	Test C	Test D
100	183992	370592	363515	1114120	1554522
1000	1758	3561	3456	10686	14281
10000	18	36	35	97	123



ILO 4: Maintenance

Task 10.P demonstrates my understanding of data structures and my ability to effectively rationalise them and determine their usefulness for a given situation. In the task, I selected a vector due to efficiency, though at the cost of ease of use and described my reasoning behind that. Tasks 4.C, 11.P, 12.P and 13.P also show my understanding of patterns and loops and the appropriate uses for them. Patterns, as in 12.P and 13.P are great ways of structuring code, due to their scalability and maintainability, with the command pattern in particular being developed and extended over both tasks. The command pattern is as well, useful for situations and scenarios where a limited number of inputs need to be processed, with a broad range of interaction, and seemingly, no time constraints. Additionally, messaging systems are able to allow for interaction between game objects and ensure functionality is possible without direct explicit interaction between objects.

Reflection

The most important things I leant:

I think the most important piece of knowledge I obtained during the unit was the idea of patterns, such as the command and composite patterns, and how to effectively implement them when programming a game. Prior to this unit, I had very little proper exposure to these concepts and thus had a sizable knowledge around game data structures. The idea of components and their implementation has helped me gain a better understanding of game engines as well, such as Unity, which has a base GameObject class type, as well as components for each game object. These components can be called upon in much that same way as was implemented in task 13.P.

Additionally, due to the sudden mess at Unity, becoming more familiar with C++ has been useful in expanding my career options, should Unity implode even further. As Unreal utilises C++, rather than C#, more familiarity with the language is definitely not a bad thing.

The things that helped me most were:

It goes without saying that my tutor, James, was incredibly helpful with giving feedback and advice for my submissions. Additionally, I found that official documentation, Git Wikis and first party sources were far more useful to me in this unit. I put this mostly down to me having a much more solid grasp of the language in this unit, as opposed to others. I was much more easily able to understand and implement things such as the SDL2 projects, just with the use of documentation.

I found the following topics particularly challenging:

Generally, the topics involving patterns were the most challenging for me. As previously mentioned, prior to this unit, I didn't have all that much experience with implementing patterns, so they did take a while for me to understand. I spent far too long grappling with the command pattern implementation before it finally clicked, which probably wasted enough time that I could have aimed for a high C grade had I gotten it done at the same pace as the tasks previous to that. The component pattern stumped me for quite a while as well, though I eventually figured out an implementation that worked. I will put this down more so to the growing complexity of the Zorkish program at the time, giving me a lot of problems as I tried to refactor the code.

I found the following topics particularly interesting:

It's difficult to narrow this down to anything other than all of it. And without trying to sound like a broken record, patterns were definitely high up on the list of things that interested me during this unit. After understanding how to implement them properly, it opened up a lot of possibilities for how I could use them in ideas I've had for a while. Components and commands would be useful for a digital card game, for example. SDL2 game me a solid idea of how to work with frameworks as well. A lot of the implementation and setup of those projects was interesting to see. I had previously only ever really thought about using full engines such as Unity or Unreal, but knowing SDL2 does open up a few more possibilities.

I feel I learnt these topics, concepts, and/or tools really well:

Towards the end, I found that I learnt SDL2 reasonably well, and was able to implement SDL2_mixer for task 16.P without much issue. Additionally, being able to locate appropriate documentation was a strength as well. My ability to code clean and generally

readable C++ should also be evident across most of my tasks, as I tried to be consistent with function and variable names, spacing, etc.

I still need to work on the following areas:

Frankly, everything. I have a long way to go with my basic coding skills in C++, as evidenced in tasks 12 onwards, as my code base for Zorkish got progressively messier. My ability to comprehend and quickly implement ideas that I've learnt is still lacking as well, though this is something that will likely come with more experience. T12 and T13 took too long to complete due to this.

My progress in this unit was …:

Slower than expected. The earlier half of the semester is when I should have been churning through tasks, I didn't get as much done as I would have ideally managed. This led to a lot of stress during the latter half of the semester when I was struggling to understand the pattern tasks for a while and ended up with tasks still to do going into week 12. I believe a mid to high Credit would have been possible had I managed to push through the early weeks' tasks a bit faster and just ask for help when I was stuck. I think the main takeaway from this unit in terms of time management I can bring into my remaining capstone units is to really keep a consistent level of work, even rostering time to complete tasks if need be, rather than leaving it all to a last minute panic.

This unit will help me in the future:

I've touched on this a few times already, but more knowledge and experience with C++ is always a good thing. Additionally, making use of patterns is something that will be key in making games going forward. The ability to structure an object efficiently and effectively is going to help me regardless of the programming language. As well as that, being able to find and absorb documentation is a skill I was sorely lacking in, that I will be able to use to properly learn different engines, languages, tools and methods, meaning I don't need to rely on poorly explained chunks of code from AI or StackOverflow.

If I did this unit again I would do the following things differently:

I'd probably allow myself more time earlier in the semester in order to power through the Zorkish tasks. They took so much more time than I anticipated, and as such, I fell behind during the latter half of the semester. Moving on to SDL2 when I got stuck would have also helped progress through the tasks faster, and had I finished T13 in week 9 or 10, I would likely have had much more time to finish the Credit tasks. Getting more feedback from my tutor or checking into the Discord server more often would have definitely helped me out as well, but due to stubbornness, I tried to figure everything out myself.

Conclusion

In summary, I believe that I have clearly demonstrate that my portfolio is sufficient to be awarded a Pass grade.