COS30031 Games Programming

Learning Summary Report

Ryan Chessum (102564760)

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)	✓				

Self-assessment Statement

	Included? (tick)
Learning Summary Report	✓
Complete Pass (“core”) task work, approved in Doubtfire	✓

Minimum Pass Checklist

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

Credit Checklist, in addition to Pass Checklist

	Included? (tick)
Custom Project Distinction Plan document, approved in Doubtfire	
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 in Doubtfire, 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 in Doubtfire, 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 in Doubtfire	
High Distinction Report document, in Doubtfire, 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

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 01 - Lab - Bitbucket Setup

- Contains Bitbucket repository link.

Task 02 - Lab - C++ For Programmers

- Shows basic use of C++ features.

Task 03 - Spike – Gridworld

- Simple example of game loop architecture with input, update and render.

Task 05 - Lab – Debugging

- Shows use of IDE to debug code.

Task 06 - Lab - Data Structure Basics

- Shows basic understanding of different C++ data structures.

Task 07 - Spike - Performance Measurement

- Shows understanding of measuring and understanding code performance.

Task 08 - Spike - Game State Management

- Examples of game state or stage architecture.

Task 09 - Spike - Game Data Structures

- Simple inventory system for Zorkish that makes use of C++ data structures and a short report comparing several C++ data structures that show understanding on advantages and disadvantages of each.

Task 10 - Lab - File Input Output

- Shows knowledge of how to read and write to different file types including JSON.

Task 11 - Spike - Game Graphs from Data

- Shows graph data and loading game info from text file.

Task 12 - Spike - Command Pattern

- Basic Zorkish implementation with command processor.

Task 13 - Spike - Composite and Component Patterns

- Has basic examples of composite pattern with nested entities and components.

Task 15 - Lab - SDL2 Concepts

- Shows basic understanding of SDL2.

Task 16 - Spike - Sound Board

- Example of using SDL mixer to load sounds

Task 17 - Spike - Sprites & Graphics

- Example of using SDL image to load images.

Task 19 - Spike - Messaging: Announcements & Blackboards

- Example of a messaging system using a blackboard.

Task 22 - Spike - Collisions

- Shows understanding of different types of collision detection between gameobjects.

Task 24 - Spike - Profiling, Performance & Optimization

- Shows understanding of how to analyse and improve upon code 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

Discuss game engine components including architectures of components, selection of components for a particular game specification, the role and purpose of specific game engine components, and the relationship of components with underlying technologies.

In many tasks I learnt about, implemented and even designed some game engine components, architectures and more. In task 6 I learnt about different data structures in the C++ standard library and how to use them. In task 8 I Implemented a basic game stage architecture that could swap the game between stages like a main menu or gameplay. In task 12 I researched and designed a command processor for zorkish. In Task 15 I researched and learnt about SDL2, a development library that can be used to draw to a window. In Task 19 I looked at the role of messaging systems in games and how they can be used for interactions between entities.

ILO 2: Implementation

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

In many of the included tasks I implemented many different game components for use in games systems. In task 2 I looked using common programming ideas using C++. In task 3 I used gridworld to show a basic example of the classic gameloop. In Tasks 5 and 6 I looked at using basic C++ data structures. In task 9 I implemented a game stage manger to manage the updating and rendering of different gamestates using a state manager. In Task 9 I used C++ data structures to make an inventory system for the player that holds items. In task 10 I looked at reading files. In task 11 I looked at making graph structures and using a text file to load a graph into zorkish. In task 12 I implemented a command processor in zorkish. In task 13 I implemented composite entities and giving entities components. In tasks 15 game libraries to create 2d games. In task 16 I used SDL mixer to load sounds from file and use them for a program. In task 17 I used SDL image to load in images from file and render then to screen. In task 19 I created a Blackboard messaging system between entities. In task 22 I implemented collision detection in 2d space for rectangles and circles.

ILO 3: Performance

Identify performance bottlenecks by using profiling techniques and tool, and applying optimisation strategies to improve performance.

In task 5 I investigated the performance of different data structures and potential issues that can arise when using them. In task 7 I measured and analysed code performance. In task 24 I measured code performance again and used the data to understand, make improvements and optimisations for code.

ILO 4: Maintenance

Explain and illustrate the role of data structures and patterns in game programming, and rationalise the selection of these for the development of a specified game scenario.

In Tasks 5 and 6 I looked at basic C++ data structures and how they can be used. In task 8 I looked at the game state pattern to use to manage updating and rendering different game states. In task 9 I investigated the advantages and disadvantages of using different C++ data types and used the information to select a datatype to use for an Inventory system. In task 11 I made a graph structure for game worlds in zorkish. In task 12 I implemented a command processor to manage the handling of different commands for inputs in zorkish. In task 13 I looked at the advantages of using composite and component patterns for game entities. In task 19 I looked into messaging systems in games and how they can be used for interactions with objects without coupling them. In task 22 I looked at collision detection patterns and how to detect collisions with different objects.

Reflection

The most important things I leant:

There were a lot of important topics I learnt about this semester. First up was C++ memory. I've learnt about it before but I had forgotten a lot of it. It's an important concept I had to wrap my head around especially for optimisation. It's also just good to know how computers allocate memory. Other important things I learnt about were different game architectural patterns. This includes an overall state manager to manage the game states and stages. Another important pattern we composite and component patterns. They seem like they can get really extensive in more complex game systems and are really important for making said complex games. As well as that messaging system patterns seem extensive for the same reason. They could be used to define a very large number of interactions in games. Looking at the games I play I can already see how some of these patterns and systems are implemented and I look forward to figuring out how to implement my own.

The things that helped me most were:

The lectures on canvas were very helpful for getting a good introduction to a topic before I went and looked into it myself. They explained many of the base concepts well enough that I knew where to look next. Another great resource that was incredibly helpful for the SDL related tasks was lazyfoo (https://lazyfoo.net/tutorials/SDL/). The tutorials were really great for explaining how to do basic SDL stuff.

I found the following topics particularly challenging:

There's no topic that stands out to me as having been particularly challenging. Each topic had their own challenges that I had to figure out how to overcome. There would always be that one thing that wasn't working and I couldn't figure out why, only for it to take 3 hours to find the simple solution. One thing I did find particularly scary was the new keyword. I tried to avoid using it at all costs to avoid getting memory leaks. It was always safer for me to pass in scope pointer variables that would free themselves.

I found the following topics particularly interesting:

Messaging systems are an interesting topic for me. I've always coupled my objects together to make interactions in the past. After learning about messaging systems they seem like a great way to define a broad an complex interaction table between lots of entities and I cant wait to explore the topic more in my own time.

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

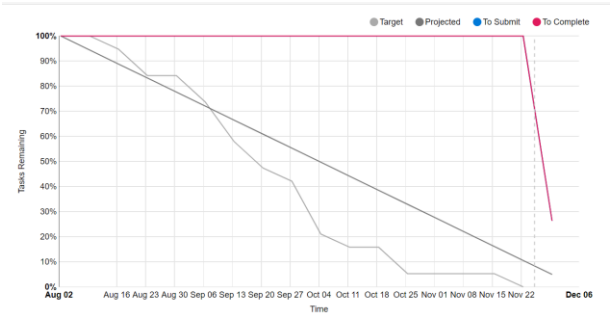
I've learnt visual studio really well this semester. I normally just use VS code but the need for a compiler meant I had to use the main visual studio instead. I've gotten a good feel for C++ programming which I hadn't done for a year and a half since I did DSP. I've gotten a lot more knowledge about how the compiler and memory works because of this.

I still need to work on the following areas:

There's definitely a lot more I can learn about collision detection. I only got to implement circle on circle and rectangle on rectangle collision in task 22. I know there's a lot more to learn like polygonal detection, circle on rectangle and like the examples in the lectures.

My progress in this unit was …:

My progression in this unit was hard. Due to health issues it was hard to focus on work for all of my subjects. Thankfully I was able to push through and get everything done in the end.



This unit will help me in the future:

This unit has really expanded my knowledge about games programming patterns. Games programming is something that I'm really interested. I feel I now have a greater understanding of base concepts I only roughly new about before like components, gamestates and more. There are many areas that I can explore and find out more about using the base knowledge I've obtained in this unit.

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

Ideally I would like to be able to take this unit without dealing with any of my health issues. Normally I try and go for a custom project but unfortunately this semester my health got in the way and it was hard to consistently get work done across all my subjects.

Conclusion

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