COS30031 Games Programming

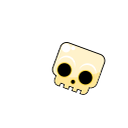
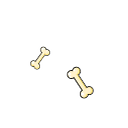
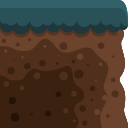
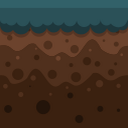
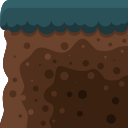
Learning Summary Report

Ford Killeen (9731822)

Introduction

Braaaaains

Gaaaaames



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 learning outcomes, and a reflection on my learning.

Self-Assessment Details

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Adequate (Pass) | Good (Credit) | Outstanding (Distinction) | Exemplary (High Distinction) |
| Self-Assessment (please tick) |  |  | ✓ |  |

*Self-assessment Statement*

|  |  |
| --- | --- |
|  | Included? (tick) |
| Learning Summary Report | ✓ |
| Lab Test 1 and 2 + feedback + responses | ✓ |
| Complete “core” spike work | ✓ |

*Minimum Pass Checklist*

|  |  |
| --- | --- |
|  | Included? (tick) |
| Additional non-core spike work (or equivalent) | ✓ |
| One or more *non-code* pieces | ✓ |

*Minimum Credit Checklist, in addition to Pass Checklist*

|  |  |
| --- | --- |
|  | Included? (tick) |
| Code for non-trivial program(s) of own design | ✓ |
| Document for the program(s) included (structure chart etc) | ✓ |

*Minimum Distinction Checklist, in addition to Credit Checklist*

|  |  |
| --- | --- |
|  | Included? (tick) |
| Research report, and associated pieces |  |

*Minimum High Distinction Checklist, in addition to Distinction Checklist*

# Overview of Pieces Included

This section outlines the pieces that I have included in my portfolio. All of the pieces listed below were included because each of them reflects the knowledge and understanding I have gained from completing this subject in one way or another. Whether it be for one ILO or many, each included piece will display skills I have learnt or improved upon during this unit.

The table below gives a quick and easy view for each of the pieces and what ILOs it relates to. A more in depth breakdown of this can be seen on the following page in the *Coverage of the Intended Learning Outcomes* section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Piece Title** | **ILO 1 (D)** | **ILO 2 (I)** | **ILO 3 (P)** | **ILO 4 (M)** |
| Lab Test 1 with Feedback | ✓ |  |  |  |
| Lab Test 2 with Feedback | ✓ |  | ✓ |  |
| Spike 01 – Simple Game Loop | ✓ | ✓ |  |  |
| Spike 02 – Intro to Visual Studio |  |  |  | ✓ |
| Visual Studio Report |  |  |  | ✓ |
| Spike 03 – Debugger Use |  |  | ✓ | ✓ |
| Spike 04 – Non-Blocking Game Loop |  | ✓ |  | ✓ |
| Spike 05 – Game State Management | ✓ | ✓ |  | ✓ |
| Spike 06 – Basic Game Data Structures | ✓ | ✓ | ✓ | ✓ |
| Data Structures for Games Report | ✓ | ✓ | ✓ | ✓ |
| Spike 07 – Graphs | ✓ | ✓ |  | ✓ |
| Spike 08 – Command Pattern |  | ✓ |  | ✓ |
| Spike 09 – Composite Pattern |  | ✓ |  | ✓ |
| Spike 10 – Component Pattern |  | ✓ |  | ✓ |
| Spike 11 – Messaging |  | ✓ |  | ✓ |
| Spike 12 – Announcements & Blackboards |  | ✓ |  | ✓ |
| Messaging Specification |  |  |  | ✓ |
| Spike 13 – Unit Tests |  | ✓ |  | ✓ |
| Spike 14 – Unreal Engine Familiarisation | ✓ | ✓ |  |  |
| Spike 15 – Create a Simple Scene | ✓ | ✓ |  |  |
| Spike 16 – Create a Simple Blueprint in C++ | ✓ | ✓ |  |  |
| Spike 17 – Input Handling | ✓ | ✓ |  |  |
| Spike 18 – Sounds | ✓ | ✓ | ✓ |  |
| Spike 19 – Collisions | ✓ | ✓ |  |  |
| Spike 20 – Measuring Performance | ✓ | ✓ | ✓ |  |
| Using UE4 Performance Profiler Report |  |  | ✓ |  |
| Distinction Project & Report | ✓ | ✓ | ✓ | ✓ |

Table 1 – ILO and piece breakdown

# Repo

All of my spikes and reports (with the exception of the UE4 projects) are available on the following repo. A copy of them has also been provided on USB for convenience.

<https://github.com/fordkilleenuni/GamesProgrammingSpikes>

# 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.*

With relation to the above Intended Learning Outcome and the use of different game engine components, I believe I have included sufficient evidence in my portfolio to show a deep understanding.

I am going to relate this specifically to the Unreal Engine 4 spikes as I completed all of them and the majority of their extensions as well as my distinction game *Lunch Break*. There are many different game engines and game engine components and choosing when and where to use the right ones is an important skill to have. For my game Lunch Break, we decided on using the Unity game engine as it was the most familiar to us and seemed like the easiest option to pick up and run with. This was largely due to the component based system that Unity uses for its game objects. By making every game object capable of having multiple components, when designing and implementing our code, we only needed to write scripts specific to a certain bit of functionality and then attach it to a game object or prefab.

The underlying technologies for the Unreal Engine seemed to be based on inheritance much more than Unity and had less focus on component based objects. Objects in UE4 were created by selecting a base class with which to inherit from (eg. Actor) and then build upon its functionality that way using the visual scripting/blueprinting or by writing your own C++ blueprint classes.

## 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 relation to this Intended Learning Outcome and my ability to implement games by utilising game engine functions and APIs, I have completed the following items which I think prove my capabilities.

● I completed and implemented working Grid World and Zorkish games.

● I completed a scene in Unreal Engine 4 which contained multiple actors, lighting and I used the UE4 visual scripting system to make one of my actors rotate on the spot.

● I created a scene in Unreal Engine 4 which was based off of the FPS template project. This scene contained multiple actors that were either standard meshes or blueprint objects. I modified this project multiple times for different spikes, including using the in-built UE4 collision events to alter the texture of meshes when they collided with another actor. I also used the in-built UE4 sound APIs to play and modify sounds within the scene, including a constant sound that played on the player blueprint and another sound which played at a static objects position, giving off a sound that died off over a set distance.

● I worked on the game Lunch Break and used many of the in-built Unity APIs to achieve certain features within the scripts I had written. Some of these include using the LERP functions to move the camera around while bouncing, using the transform and position values when moving the camera at a constant rate down the map and using colliders to detect collisions among objects.

## ILO 3: Performance

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

With regards to performance and optimisation I used multiple tools and techniques to optimise or look into the performance of my projects.

The first time this can be seen in my work is during spike 3, as the Visual Studio debugging tool offers a little insight into the performance of your project. It shows memory usage, CPU usage, call times and more, all of which allowed me to find any major performance bugs and fix them.

When completing the Unreal Engine 4 spikes (specifically the sound and performance ones) I used the in-built performance profiler to record and look into how the project was running. This tool took a little getting used to but once I figured it out it offered a great amount of detail and the ability to pinpoint any performance bottlenecks by using statistics such as frames per second, memory usage, game and rendering thread usage and much more.

Performance was also something we took into consideration for the Lunch Break game. We tried not to think too much on the topic as premature optimisation can be a bad thing and we didn’t want to jeopardise out script and functionality, but when we did think about it, it was because of obvious reasons that were easily avoided by changing the way we coded. An example of this is by calling any GetComponent (or similar) functions once at the beginning of an objects initialisation and storing the result, as opposed to running that function every update/tick. This resulted in a performance increase and required little mucking around or risk to achieve.

## 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.*

Data structures and patterns within games are quite important as they act as guidelines to achieve certain functionalities or actions within specific game scenarios.

They are very useful as long as they are used in the right way, but can be confusing and worthless if they do not work for the scenario in focus. An example of this could be the use of the Command pattern within my Lunch Break game. This game is designed to be played with a controller and has no text input whatsoever. Because of this, the use of the Command pattern would not be advised or even make sense as there are no instances where a command would be sent to the engine or game needing to be processed.

In my Zorkish spikes I have shown an understanding of the command pattern, component pattern and composite pattern, along with blocking and non-blocking game loops and messaging systems. I also showed good use of data structures by always choosing the one that best suits the situation, for example a map to store my locations or a vector to act as my underlying inventory.

This was also shown in my Lunch Break code, for example using a List to store menu items and similar object instances. Without the use of the correct data structure or pattern, coding a game can become much more difficult and take much longer to achieve certain functions as opposed to planning and using the structure/pattern that works for you in that situation.

# Reflection

## The most important things I learnt:

The most important things I learn (or re-learnt) were C++, Unreal Engine 4, many of the design patterns and unit testing. These are some of the most important things I learned during the unit because simpler things like using Visual Studio, debugging and software patterns were all things I knew about to a good degree. I included C++ because I hadn’t touched it in about two years prior to this unit and getting that knowledge back I found pretty handy. Unreal Engine 4 was a great tool to learn how to use and is also in my most important list as before that I had only ever used Unity to develop games in. Learning how to use UE4 opened me up to different tools and ways of developing games (eg. visual scripting). Unit testing and some of the software patterns learnt were also quite important and I think they will help me greatly in the future.

## The things that helped me most were:

My tutor and the internet. Early on in the unit I got stuck many times getting back into the flow of C++, but with the aid of my tutor and the internet I was able to overcome all of the challenges I faced.

## I found the following topics particularly challenging:

Definitely towards the start of the unit I found getting back into C++ the hardest, with much confusion surrounding pointers and all those nice C++ things. Besides that I think the only other really tricky part of the unit was the Component pattern and possibly the loading of entities from a text file to fill an environment, as the text files were just getting messier and more confusing.

## I found the following topics particularly interesting:

The topics I found the most interesting were definitely Unreal Engine 4 as I had never used it before and probably unit testing. UE4 was new to me and therefore very interesting as the visual scripting and blueprinting was something I’d only ever seen before but never used. I enjoyed figuring out how this all worked and how easy it could be to throw in a few nodes and have something working in the scene. Unit testing was also interesting as I would like to get better at it and use it more in my professional workplace.

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

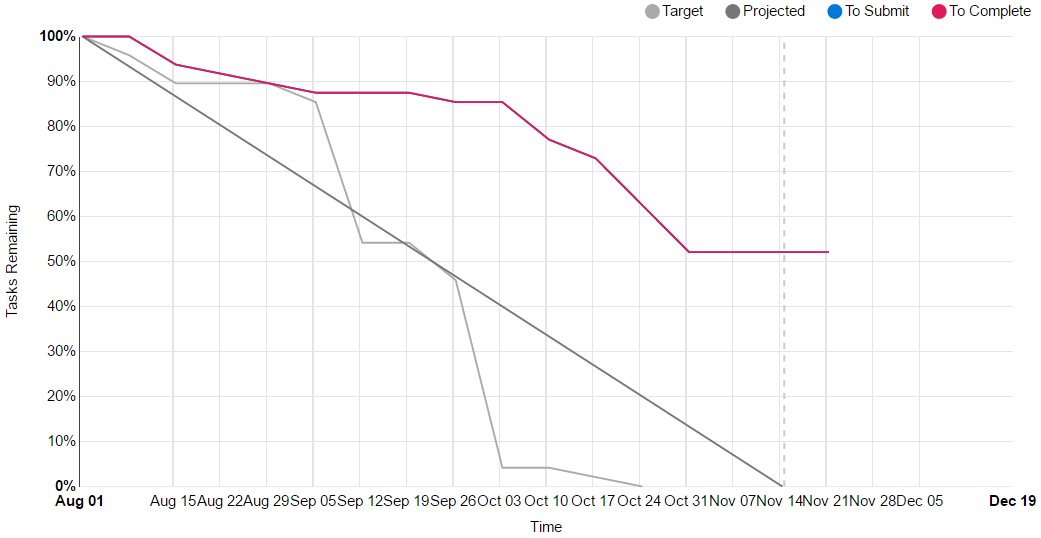
I believe I learnt majority of the software patterns fairly well, with many of them reinforcing pattern ideas and concepts in my mind. I also felt I picked up the Unreal Engine work pretty quickly as I had just spent the year using Unity for my capstone project.

## I still need to work on the following areas:

The areas I still need to work on would definitely be C++ and the component pattern. I understand the usefulness of the pattern but I struggled a little to effectively use it. I also need to work on using less singletons as I used a few throughout my Zorkish spikes. I tried to use as little as possible but needed to use them for the Adventure and Player instances as I found it the easiest way to reference important bits of information.

## My progress in this unit was …:

My progress for the unit overall was a little behind. I started off fairly well, keeping up with the spike work (though forgetting to submit a few reports on time), but after about week 4 or 5 I started to fall behind. This was mainly because I was still struggling a little with getting back into C++ but also because I was sometimes doing only one spike a week instead of the two or three I should have been doing. In the last few weeks of semester I picked up more and was able to smash through all of the spikes excluding the UE4 ones, which I was able to do in the few weeks after semester. Due to the late submission of the UE4 tasks and them not being marked off on Doubtfire, the end of the graph does not accurately represent my efforts and because of this I have added my own line of where I believe it should be (the orange line).



## This unit will help me in the future:

The things that will help me the most in the future would be some of the software design patterns that I didn’t previously know, unit testing and hopefully the Unreal Engine 4 work. This is because I can utilise the patterns and unit testing in many different work places but will hopefully be able to use UE4 in the future to develop something cool too.

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

If I were to do the unit again I would definitely manage my time better throughout the semester and aim to get more than one spike done per week on average. I would probably also try and attend every lecture if I did the unit again, as I was at the beginning of the semester but attended less and less as time went on. This didn’t help particularly near the end of semester as I had missed out on some vital explanations and ideas, especially relating to some of the test 2 questions, which some I was quite unsure about. Another thing I would do is definitely study more for the tests and try harder for a deeper explanation of my knowledge and understanding on the topics.

# Conclusion

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

I believe I have shown a good understanding of all of the key concepts for this unit and have been able to link them to the Intended Learning Outcomes. With all of the spike reports and lab tests there should be plenty of examples showcasing my knowledge.

I was able to complete every spike as well as nearly every spike extension and tried to show a little more knowledge and understanding where I could. For example adding in extra little things here and there or doing something a little differently because I felt it interesting and enjoyable to do.

I was part of a team that designed and developed our final year games project and though this was started a semester before starting this unit I believe I have shown a solid understanding of the concepts learnt in this unit and how they relate to things I had done for the game. I also linked what I did last semester with things I may have been able to do better had I completed this unit earlier.