COS30031 Spike03 Engine report – GODOT (not edited for ease of task 30 review)

## Features

* Specialised 2D workflow for games and apps
  + Real 2D and pixel-based unit system (no worrying about z axis)
  + Tile map editor
  + Flexible GUI system (The editor itself is an example)

A screenshot of a computer

Description automatically generated

* Simple yet powerful 3D engine
  + Supports high and low end devices (mid level androids to desktops)
  + Robust importing pipline, allowing for animated models and objects to be easy imported and viewd
  + Animated videos and prerendered cutscenes allowed with new 4.0 update.
* Lots of Code support for different languages
  + Has its own, built-in scripting language, GDScript that is inspired by Python and some other languages.
  + Has .NET support as a dedicated engine executable.
  + Due to being open source, there are a number of community supported languages.
* Nodal structure
  + Many things in Godot are created from these nodes, such as characters, scenes, items, making them all easy to link up and structure.

Godots biggest feature is that it is open source. This is a large advantage over Unity and Unreal for 2 significant reasons:

* Its completely free.
* Its far more customizable.

Pricing (I am going to compare it to Unreal over unity as Unreal has the better pricing)

There is 0 cost to developing in Godot, unlike Unreal’s eventual royalties (5% after 1mil gross revenue) allowing for small developers to profit more from the games they make, assuming they do well.

Customizability

With the open source nature, far more community created extensions and features have been created for Godot than would have been in the same time for Unity or Unreal.

There is also the fact that Godot is significantly lighter weight than Unity is, with the editor launching noticeably quicker and having a lower impact on performance.

## Platforms

* Linux
* MacOS
* Windows
* Android
* iOS
* HTML5, WebAssembly
* VR headsets

There is no difference between the engine on different platforms, it all runs the same once complied and exported. Godot has inbuilt support for all these platforms, the only tricky ones are the 2 mobile ones, with android needing the SDK and iOS needing to be routed through a macOS running computer with Xcode (these are necessary for basically all apps anyway).

## Games

Almost all games that have used Godot so far have been indie games. Examples include:

* Brotato (excellent roguelike/bullet heaven)
* Luck be a Landlord (roguelike slot machine, not actual gambling)
* Buckshot Roulette
* Halls of torment
* Cassette Beasts (pokemon like)

Spike 30 report expansion/revisit

## Intro

Looking at the previous report and what I now know about game design, I covered the basics/overall features well and it is still a good/the best choice. Further analysis into the details of the engine, to see what smaller features it has that will benefit game development will be this report.

## Language support/style

* Godot does directly support C and C++ through its GDExtensions
* And GDscript is object oriented.
* The scripts (code files) directly extend the nodes making integration very easy.

## The node system

With more experience with game graphs and development, a lot of the objects being nodes really makes a difference with how a game is laid out, how easy it is to duplicate/make a lot of similar objects that are easy to access and code against, especially since nodes can be made up of nodes such as:

* A CharacterBody2D named Player with nodes
  + Camera2D
  + Sprite2D
  + CollisionShape2D

This is an example of the Component Pattern, like Spike 14, and is the core method of creating games in Godot, which is a fantastic feature as it makes high level development far easier.

## Rendering (2D)

* It allows for sprite, polygon and line rendering which is a great set for flexibility, which would make creating applications similar to spike’s 18 and 22 easier, or at least a similar low level of complexity (mainly 21 and its moving circler).
* It also allows for both CPU and GPU based particles, which is a really useful feature, as it can split the processing load nicely.

## Collisions (2D)

It has a built in collision detection system, with shapes:

line, box, circle, capsule and world boundary (infinite plane).

And it has collision polygons that can be generated from sprites (or drawn manually) for those objects that cannot be a simple shape.

Spike 22 would have been a bit simpler with this.

## Input mapping

* Has a similar level of mapping allowed to SDL but allows for more simultaneous inputs (like multiple controller, up to 8).

## Other Unit observations

I did Mobile development this semester (COS30017) which had a big focus on UI which transfers to game development quite nicely, and Godot has an excellent and robust GUI system, that is an improvement over the android system I was using during that unit

I also did AI for Games last semester (COS30002) and it has a number of navigation features, including an inbuilt implementation of A\*, that would make ai creation simpler (to a degree).

## Conclusion

Looking further into Godot with experience in this unit and others, I am confident that it is a great choice, the only downside (and it’s a minor person preference) is that its format is very close to python, more so than I originally thought, and I am not the biggest fan of pythons layout (have spent most of my time coding in C#/C++) but that is something I can easily work around.