Unity

Benefits

- Contains a built in tile map system, allowing the games city grid and grid tiles to be set up relatively efficiently with a lot of work already done
- Uses the C# language, which strongly follows OOP principles which are important for game programming
- Linked to Unities asset store making adding/finding new assets very quick and efficient
- Has a built in rendering pipeline (URP) which supports 2d rendering
- Works for multiple devices if later the choice is made to port to multiple devices
- Free tier available for students use

Drawbacks

- Not the most efficient for creating UIs
- No built in system for saving and loading
- Massive amounts of objects could cause slowdown

Godot

Benefits

- Engine built specifically for 2d games(may render somewhat faster than Unity)
- Contains a built in tile map system, allowing the games city grid and grid tiles to be set up relatively efficiently with a lot of work already done
- Very fast to setup and get going
- Open source- therefore is completely free to use
- More flexible when it comes to creating UI than Unity

Drawbacks

- Uses its own, specific language (called GDScript, which is similar to python) which would therefore slow development. Using its own language means less support material available
- Less ready made plugins/assetts than unity
- May struggle with performance when the game gets bigger

Game maker studio

Benefits

- Engine built specifically for 2d games(may render somewhat faster than Unity)
- Uses a drag and drop interface for designing logic which would greatly assist designing speed

- Contains a built in tile map system, allowing the games city grid and grid tiles to be set up relatively efficiently with a lot of work already done
- Built in support for organising layers (e.g sprites background and UI elements) meaning the interface can be designed more easily inside the game engine itself
- Able to be used for many platforms if a port is decided on later
- Free for the intended use, as it is non commercial

Drawbacks

- Uses its own language called GameMaker Langauge. Which is likened to javascript and c languages. Using its own language means less support material available
- May struggle when the simulation gets bigger and more complex
- Smaller than unity, and uses its own language, meaning there is a much lower amount of support for external code packages

Chosen engine: Unity

I have decided unity is the best decision for this project for a variety of reasons. One of those reasons being c#, as it is the most mainstream of the 3 languages and there is a the most plugins and support out of the 3 engines it means I am able to get the project going and spend more time coding the complex aspects of the game rather than getting bogged down coding more simple parts
The attached asset store also is a part of this decision, as once again it allows me to sort less complex issues (like assets) quickly and spend more time focusing on programming the more complex code.
In addition I believe Unity is the best choice as, from my research, it will struggle the least when it gets to the large complex simulation I am aiming for. White early on it would appear Game maker and Godot would run better, once more parts get added to the equation unity will start to overtake them in performance