



| **CONSTRUCT**

Student ID:

**38511351**

# Statement of Intent

For my Major Work I intend to create a video game, a website to display some basic information about the game and serve as promotional material for it. The video game will be a low poly style game and will involve some form of logistics gameplay mechanic. I will write all the code and create all the 3D models myself.

I will make the website myself and will use HTML, Tailwind CSS and Javascript.

## Reasons & Motives

I have chosen to create this project because it requires a large range of skills which will help me achieve a higher mark. I specifically wanted to make a video game as it allows me to combine my skills in coding and development with 3D models, graphic design and video creating/editing.

## Target Audience

My target audience for my major project will be people who enjoy playing logistical video games which involve a decent amount of thinking and problem solving. However I intend for the game to have some elements of randomness & chaos to make the game more interesting.

## Software & Equipment

To complete my project I will need a powerful computer to handle all of the software that I need.

I will need a game engine, 3D modelling software, code editor/IDE for the game and website, graphic design software for logos/graphics in-game.

## Time Frame

The timeframe for my project will be from **24/4/23** to **10/8/23**. This is a reasonable amount of time to complete my project and providing that I can manage my expectations of how the game turns out, and that I'm able to effectively time-manage I should be able to complete my project in time.

# SWOT Analysis

## Strengths

The strengths of my project are that it involves a lot of different skills which will allow me to achieve a higher mark if completed successfully.

My strengths are that I am already experienced and have all of the skills that the project requires, meaning that I don't have to spend time learning any new skills in order to complete it.

## Weaknesses

My project doesn't really have any weaknesses apart from the fact that a large amount of the work is coding, which takes time and doesn't cover a lot of skills. Another possible weakness is that there may be bugs I don't find in the code or that the game may not be fun to play.

My biggest weakness is time management and my documentation. I intend to set aside some time each week in order to work on my portfolio and document my progress. I will also need to ensure that I don't shoot for the stars with what I want to achieve and keep it realistic.

## Opportunities

My project provides a lot of opportunities. I want to work in Game/software development and this project will look great on a portfolio if completed to its fullest potential. It will also provide me with a chance to further develop and improve my skills in coding, 3D modelling and video editing.

## Threats

As mentioned before, a big threat to my project will be time management and having to document everything correctly. Hopefully this won't be a major problem but I will have to keep on top of the project and not get too distracted or busy with other schoolwork, sport, hobbies or my job.

### Evaluation

I am happy with my statement of intent. I think that my plan for my major will work out well and that I will be able to use and showcase a range of different skills, such as development, 3D modelling, web design, graphic design and video editing.

I will need to keep on top of my time management and ensure that I am completing everything in a timely manner and don't fall behind on the project.

## Research - Materials

### Coding Languages



A coding language is a system of notation and language used for writing computer programs. Most coding languages are text-based and will have a unique syntax and semantics. There are a wide range of coding languages, each having its own unique purposes and points of strength and weaknesses.

Coding languages are classified into "levels"; High level coding languages and low level coding languages.

High level coding languages require a compiler to "translate " their code into machine code. Low level languages only require an assembler to directly translate the code instructions into machine language.

Whether a coding language is high level or low level will directly affect how you develop using it.

Feature	High Level Language	Low Level Language
Overview	These languages are programmer-friendly that are manageable, easy to understand, read and write, debug, and are widely used in modern applications.	These are machine-friendly languages that are very difficult to understand, read and write but are easy to interpret by machines.
Memory Efficiency	Not very efficient; often provides the developer with no control over the memory or only control over the garbage collector	Extremely efficient if coded correctly; however mistakes will lead to memory leaks and extremely slow execution/processing times.
Portability	Can be used on many devices without much modification to the code	Can not be used on different device
Comprehensibility	Human-friendly and easy to read. Uses easy to read and understand syntax and keywords such as `for, if, else, foreach`	Machine-Friendly and therefore very difficult for humans to understand.
Debugging	These languages are generally very easy to debug as the compiler will print out any errors and provide details such as line and character errors.	These languages are typically very hard to debug as there is no initial error check meaning that if you make a mistake anywhere in the code you will have to go back through everything you wrote in order to find the error.
Maintenance	These coding languages have a simple and comprehensive maintenance technique	It can be incredibly complex to maintain these languages.

## C

C is a general purpose computer programming language which was made in the 1970s. It is a level coding language and does not have support for object oriented programming or other higher level languages. Because of this; code written in C is generally hard to read and understand; meaning that it is not ideal for game development. C is no longer used to write modern languages, however a lot of languages that are used in modern development (such as C++ or Python) have been built on C.

## C++

C++ is a cross-platform coding language that can be used to create very efficient applications. This can be done because C++ gives the programmers a lot of control over the system's memory and resources.

C++ is one of the most populating programming languages and is used to create most operating systems. It is an object-oriented language which helps to keep the code clean and readable.

C++ is considered a low-mid level programming language because of the amount of control that is given to the developer over system resources, while maintaining some high-level features such as classes and object oriented programming.

## C#




C# is a typesafe, object oriented programming language that runs on the .NET architecture. This language is very similar to Java and has a very similar syntax. C# is widely used in software development as it is a high level programming language and provides lots of tools, plugins and extensions to make the development process easier and faster.

Software	Pros	Cons
C	<ul style="list-style-type: none"><li>• Powerful and efficient if coded correctly.</li></ul>	<ul style="list-style-type: none"><li>• I have no experience with it</li><li>• Not supported by any game engines. Provides no game development libraries.</li></ul>
C++	<ul style="list-style-type: none"><li>• I have some experience with it</li><li>• Runs efficiently</li><li>• Lots of control</li></ul>	<ul style="list-style-type: none"><li>• Not easy to read.</li><li>• Contains boilerplate and takes longer to code</li></ul>
C#	<ul style="list-style-type: none"><li>• I have lots of experience with it</li><li>• Object-Oriented</li><li>• Easy to read</li></ul>	<ul style="list-style-type: none"><li>• Does not run as efficiently as other low-level languages</li></ul>

## Game Engine

A game engine is a software environment/development environment used primarily to develop video games. Game engines usually have features such as 3D Rendering and environment control, materials, shaders, lighting components, animation tools, artificial intelligence, physics and collision engines, audio engines and much more.

The tools provided by a game engine allow a developer to develop games without having to write a physics engine or 3D renderer from scratch.

Software	Pros	Cons
Unity 3D 	<ul style="list-style-type: none"><li>• I have experience with it.</li><li>• I have experience with C#.</li></ul>	<ul style="list-style-type: none"><li>• Not great for realistic graphics</li></ul>
Unreal Engine  <b>UNREAL ENGINE</b>	<ul style="list-style-type: none"><li>• Great for realistic graphics</li><li>• Blueprints (Codeless system)</li><li>• Supports C++.</li></ul>	<ul style="list-style-type: none"><li>• I don't have any experience with it.</li><li>• I have limited experience with C++</li></ul>
Godot  <b>GODOT</b> Game engine	<ul style="list-style-type: none"><li>• Low learning curve</li><li>• Intuitive</li><li>• Great for 2D games</li></ul>	<ul style="list-style-type: none"><li>• Is based around it's own coding language, GDScript</li><li>• Limited 3D Functionality</li><li>• I don't have any experience with it</li></ul>



I will use **Unity** as my game engine for this project. I have chosen Unity because it is the most versatile and suited to what I want my game to be. It is perfect for creating simple graphics (Such as low poly/voxel) but has a very large library of components that I can use to speed up the development process.

One of the main features that I will use in Unity is its powerful Input System. This system was recently updated and overhauled the way that game developers can handle user input. Rather than having to make calls back and forth checking if certain keys are pressed down; making it incredibly convoluted for games with many different actions, it allows you to define all of your keybinds in a single place and allow the necessary classes to “listen” to these actions that you have defined.

I will use **C#** as my coding language for Unity. I have been forced into this choice as C# is the only language that Unity actively supports. In 2017/18 versions of Unity there was a language called UnityScript (an adaptation of JavaScript) which has since been deprecated and can no longer be compiled, and the third party plugins that you could use to run Python with Unity have also since been discontinued.

Despite being forced into this choice, I think that using C# as my language will be perfect for what I need to do. It is extremely similar to Java which I have a lot of experience with and I have previously developed games and applications using C#, meaning that while completing my project I won't run into any issues of not knowing how to code something or not being sure how the language's syntax works.

### Evaluation

I am extremely glad that I chose to use Unity and C# as my game engine and coding language. I believe that if I had chosen to use a language that I wasn't already familiar with; I wouldn't have been able to complete my project on time as I would have had to spend hours learning how to make even the most basic game mechanics and elements.

## IDE (Game Development)

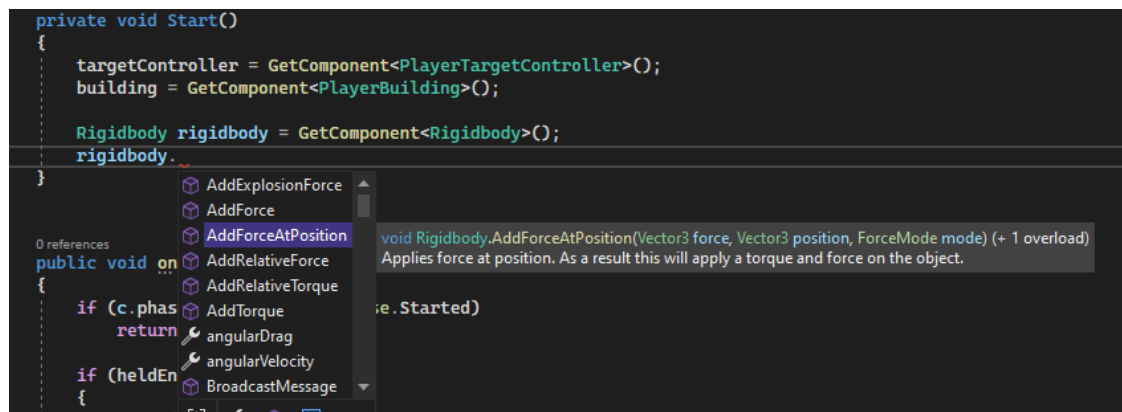


An Integrated Development Environment (IDE for short) is a piece of software that helps programmers develop software efficiently. For game development it needs to integrate with my Game Engine and be able to use IntelliSense.

“IntelliSense” is a general term for code editing features including: code completion, parameter info, quick info and member lists.




One of the main uses for IntelliSense is code completion. Code completion is primarily used to see different methods available in a given class. This saves developers time by not having to go and look at the code documentation and search for what they are looking for. Most IDEs can also incorporate a language's documentation into their IntelliSense features; like with Unity:

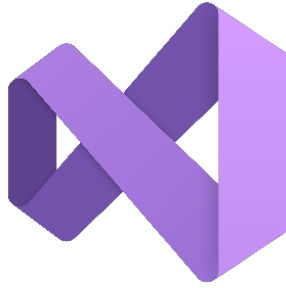




In the above example I have referenced a Rigidbody class and IntelliSense is providing me with a list of different methods that I can call on a Rigidbody, and for each method it is directly quoting the code documentation and telling me what variables I can pass in as well as what the method does.

Unity uses Visual Studio's C# compiler to compile the scripts that you write to be able to run them in the Unity editor. It also allows you to display error details of any errors in your script in both Visual Studio and the Unity editor.

Software	Pros	Cons
Visual Studio 	<ul style="list-style-type: none"> <li>• Supports C, C#, C++ and most other coding languages.</li> <li>• Extremely powerful</li> <li>• Works well with Unity and it's documentation</li> <li>• Free</li> <li>• I have experience with it</li> </ul>	<ul style="list-style-type: none"> <li>• Resource intensive application</li> <li>• Requires plugins and/or extensions to be installed in order to use some languages.</li> </ul>
IntelliJ Rider 	<ul style="list-style-type: none"> <li>• Works well for .NET Development languages (C#, Web Development, etc)</li> <li>• Supports C++</li> <li>• Integrates Unity Editor controls into the UI (Play/Pause/Errors)</li> <li>• Automatically configures to be used with Unity</li> </ul>	<ul style="list-style-type: none"> <li>• Can be a resource intensive application</li> <li>• Costs \$16.90/month</li> </ul>
CLion 	<ul style="list-style-type: none"> <li>• Support C &amp; C++</li> <li>• Supports Unreal Engine</li> </ul>	<ul style="list-style-type: none"> <li>• Does not support Unity</li> <li>• Does not support C#</li> <li>• Lots of unnecessary features for game development (primarily used for software development)</li> </ul>



For my project's Game Development IDE I will use Visual Studio. I have chosen Visual Studio as I have plenty of experience using it and I know how to get around and quirks or bugs that may occur when using it. I opted for Visual Studio (despite the small issues that it has) over another suitable IDE such as IntelliJ Rider as if any issues arise with IntelliJ Rider, I will have no idea how to fix them and will have to spend time trying to search and find what the issue is, and then possibly even more time trying to resolve it.



### Evaluation

I think that using Visual Studio was a good decision for my project, however I did briefly have an error occur. This error meant that everytime I opened up my project in Unity, I would have to relink Visual Studio in order for IntelliSense to work. However this issue was eventually resolved when I updated to a newer version of Unity.


## 3D Modeling Software (Game Development)



3D modelling software includes programs/software that allow a user to design three-dimensional models of objects, characters, landscapes and really anything that you can imagine. Most 3D modelling software comes with a variety of different tools, controls and features; including but not limited to: texturing, materials, HDRis, animation, rendering, sculpting and prototyping. 3D modelling software can be used to produce standalone image or video renders, or export your models for use in another piece of software; such as a game engine.

Software	Pros	Cons
Blender 	<ul style="list-style-type: none"><li>• I have experience with it</li><li>• Free and open source</li><li>• Decently fast rendering time</li><li>• Able to use realtime rendering</li></ul>	<ul style="list-style-type: none"><li>• Limited ability to create realistic graphics</li></ul>
Cinema 4D 	<ul style="list-style-type: none"><li>• Extremely good at creating realistic graphics.</li><li>• Very powerful modelling and material creation tools.</li></ul>	<ul style="list-style-type: none"><li>• Very slow rendering time</li><li>• Costs \$60 a month</li></ul>



Maya 	<ul style="list-style-type: none"> <li>• Powerful animation/ motion capture &amp; tracking</li> <li>• Free-form approach to 3D modelling (Doesn't limit you to only using modifiers)</li> <li>• High quality rendering</li> </ul>	<ul style="list-style-type: none"> <li>• Cost \$1,470/year</li> <li>• I have no experience with it and would have to learn how to use it.</li> </ul>
---	---	--

## 3D Modeling Software



I will use Blender as my 3D modelling software for my project. I have chosen to use Blender because it is completely free and there is an abundance of free resources, tutorials and documentation on the internet that I can use to aid me when modelling. I have a decent amount of experience with using Blender and am comfortable enough with it in order to create what I need for this project.

### Evaluation



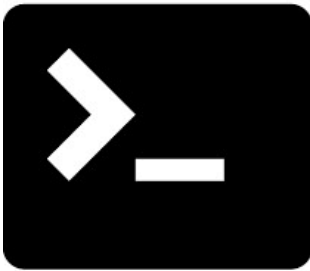
I think that using Visual Studio was a good decision for my project, however I did briefly have an error occur. This error meant that everytime I opened up my project in Unity, I would have to relink Visual Studio in order for IntelliSense to work. However this issue was eventually resolved when I updated to a newer version of Unity.

## Version Control



Version control (sometimes referred to as source control) is the practice of tracking and managing changes to a code base; whether that be a website, piece of software or a game. Modern version control solutions will allow users to define a "commit" (An update to their code base) which can then be pushed to their repository. Each commit will allow you to specify a name and description so that you know what has been changed in each update.

Systems such as GitHub or GitLab allow you to make a repository public; allowing other developers to see and use your code. They also tend to have support for different branches. These 'branches' are typically used when you have published a product. You would likely have 3 different branches 'production' (Used to hold the code that your users are currently using), 'development' (Used to hold the code that your developers are currently working on) and 'test' (Used to hold the code that needs to have tests run on it before it can be promoted to the production branch). This system ensures a smooth transition from a new update in your code to actually releasing it.

Software	Pros	Cons
GitHub Desktop 	<ul style="list-style-type: none"> <li>• I have experience with it</li> <li>• Higher availability</li> <li>• Stronger and faster infrastructure</li> </ul>	<ul style="list-style-type: none"> <li>• Severely limits the file size that I can upload to their servers. Meaning some large 3D models cannot be backed-up.</li> </ul>
GitLab Desktop 	<ul style="list-style-type: none"> <li>• Offers more features</li> <li>• Works especially well for web development</li> </ul>	<ul style="list-style-type: none"> <li>• I do not have any experience using it.</li> </ul>
Command Line 	<ul style="list-style-type: none"> <li>• Lots of control over what branch different commits go to</li> </ul>	<ul style="list-style-type: none"> <li>• Complicated and requires me to remember a lot of commands.</li> <li>• If I make a mistake I may ruin my repository.</li> </ul>

## Version Control



For my project's Version Control I will use Github and Github Desktop. I have chosen these 2 applications because they are simple, straightforward to use and I have experience with both of them.

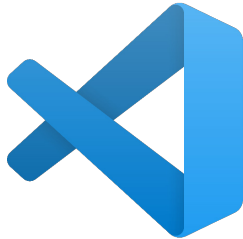
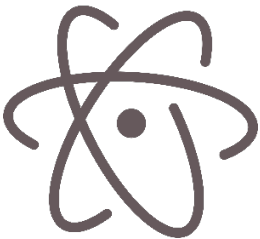
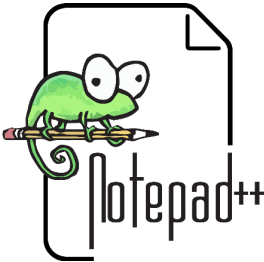
I definitely considered using Gitlab for this project as I would like to learn more about Gitlab and how I can use it for my other projects and take advantage of some of the extra features that it has, however I decided against having to learn a new git control system and wanted to stick with what I knew would work and that I wouldn't have any issues with.

I will use Github to manage both my game's files and also my website. I will create separate repositories for each in order to create a clear separation between the 2 parts of my project.

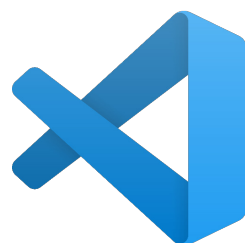
# Code Editor (Web Development)



A Code Editor is a piece of software that (like an IDE) allows a developer to write code. However what sets a code editor apart from an IDE is that it is usually extremely lightweight and does not have support for compiling and debugging code. This is why code editors are primarily used for web development as any debugging will appear on your website in the console; however some text editors such as Visual Studio Code provides plugins that can host your website locally and provide you with any information or details on issues that you encounter. Most code editors will have some form of IntelliSense and will allow you to install plugins to enhance IntelliSense and make it work with your frameworks.

Software	Pros	Cons
Visual Studio Code 	<ul style="list-style-type: none"><li>• Powerful IntelliSense</li><li>• Lots of plugins for different development frameworks.</li><li>• Completely Free</li><li>• Built in terminal.</li><li>• Provides syntax highlighting</li></ul>	<ul style="list-style-type: none"><li>• You need to install extensions in order to get IntelliSense for some frameworks.</li></ul>
Atom 	<ul style="list-style-type: none"><li>• Extremely clean UI</li><li>• Easy to navigate</li><li>• Supports git and version control in app</li><li>• Highly customisable</li><li>• Provides syntax highlighting</li></ul>	<ul style="list-style-type: none"><li>• No longer maintained (archived)</li><li>• Very little documentation</li><li>• Complicated keyboard shortcuts</li><li>• Prone to crashes</li></ul>
Notepad++ 	<ul style="list-style-type: none"><li>• Provides syntax highlighting</li></ul>	<ul style="list-style-type: none"><li>• No IntelliSense</li><li>• You have to manually upload to the server each time</li><li>• No built in terminal</li></ul>

## Version Control



I will use Visual Studio Code to create my website. I have chosen to use Visual Studio Code because it has all of the features that I will need for my project (and more!). In particular I chose Visual Studio Code because of its inbuilt terminal feature, which allows me to run console commands (such as npm (Nodejs) imports) directly from the editor rather than having to open File Explorer, console and then run my commands.

I also really like Visual Studio Code's syntax highlighting features which help to make the code easily readable, which when combined with its powerful IntelliSense helps you to write clean, simple and well documented code.

# Web Design Software



Web Design software is a piece of software which allows a designer to create, edit and prototype the front end of a web page. This is an extremely important step in the web development process as front end development can take a lot of time and be fiddly to go back and make any modifications if there is something you are not entirely happy with. Web Design software allows you to prototype and iterate your initial design until you have created something that you are happy with and you can then take that design into development.

Some web design tools such as Figma, Adobe Xd and Adobe Dreamweaver provide “prototyping” tools which allows you to create basic button actions, animations and web flows in order to fully plan out your website before you begin development.

Software	Pros	Cons
Figma	<ul style="list-style-type: none"><li>• Professional Grade software</li><li>• Free for up to 3 projects</li><li>• Ability to import either official or community made components.</li><li>• Prototyping features are powerful and intuitive.</li></ul>	<ul style="list-style-type: none"><li>• For any good looking components you need to download them.</li><li>• Layers/UI can be confusing to navigate at times.</li></ul>
Adobe Xd	<ul style="list-style-type: none"><li>• Lots of in-built, official components from Google (Material Theme 2), Apple and more.</li></ul>	<ul style="list-style-type: none"><li>• Costs money</li><li>• Prototyping features are extremely limited.</li></ul>
Photoshop	<ul style="list-style-type: none"><li>• It's simple to use layers, drop shadows and other image effects.</li><li>• Easy to create guides to create even spacing between components</li></ul>	<ul style="list-style-type: none"><li>• Costs money</li><li>• Is not meant to be used for website design.</li><li>• Does not include any prototyping features</li></ul>

# Web Design Software



For my web design software I will use Figma. I've made this decision because in the past I have used Adobe Xd, and while it works great for designing extremely simple, one page websites; it really struggles with more complicated sites and has almost no prototyping tools for connecting different page.

Figma solves all of these problems and is a much more powerful tool that will be able to perform the tasks I want to do without any hassle.

# Front-End Framework (Web Development)



Frontend frameworks are frameworks used to develop the front end of your website (what the user sees) however some frameworks (Such as Nextjs and Vue) go the extra mile and will also make creating backend calls or Serverless applications (meaning that the frontend and backend are hosted on the same machine) much easier.

With modern frameworks, there is a lot of emphasis being placed on component based design rather than having to reuse code and therefore creating a lot of unnecessary boilerplate.




An Example:

In a standard HTML/CSS project, if you wanted to create a header class you would have to write that in your `index.html` file. However if you then created another page in your website `signup.html` you would have to copy & paste the exact same code into the new page.

This is alright (Creates a mess in your class but isn't too bad). But if you ever want to change something in your header, say add another link; you would have to change it in every single class that uses your header code.

Component based frameworks such as React allow you to write your code once in a class, e.g `Header.tsx` and then you can reference this anywhere in your react code as `<Header />`.

If you ever want to make a change to your header you only need to make this change in the `Header.tsx` class and it will apply everywhere.



Software	Pros	Cons
HTML/CSS 	<ul style="list-style-type: none"><li>• Widely used</li><li>• Simple</li><li>• Does not require any node js installations or plugin</li></ul>	<ul style="list-style-type: none"><li>• Limited functionality.</li><li>• Does not have a component based design leading to lots of boilerplate code</li><li>• CSS can get extremely messy very quickly</li></ul>
React 	<ul style="list-style-type: none"><li>• Component based design (Write once, use everywhere)</li><li>• Page routes</li></ul>	<ul style="list-style-type: none"><li>• Requires node js installations to run</li></ul>
Nextjs 	<ul style="list-style-type: none"><li>• Implements flawlessly with React</li><li>• Create by Vercel (A web host)</li><li>• Basic page routes functionality</li><li>• Easy to use API routes for backend calls.</li></ul>	<ul style="list-style-type: none"><li>• Limited functionality in the page routes.</li></ul>

## Languages (Web Development cont.)



Javascript and Typescript are both web development coding languages and are primarily used for front-end development. (Handling button presses, backend API calls, user input etc).

There is very little difference between the 2 languages however Typescript is a typesafe language. Meaning that unlike javascript; where (in development) it would let you multiply a string by an integer, Typescript will throw an error and let you know that your types aren't compatible with each other.

Software	Pros	Cons
Javascript 	<ul style="list-style-type: none"><li>Is a “developer knows best” framework meaning that it allows you a lot more freedom and won't restrict stop you from writing unconventional code.</li></ul>	<ul style="list-style-type: none"><li>Is not typesafe, meaning that you may have to wait until production or testing to discover bugs or issues in your code</li></ul>
Typescript 	<ul style="list-style-type: none"><li>Typesafe; extremely useful for writing complicated code</li><li>More widely used</li><li>Industry standard</li></ul>	<ul style="list-style-type: none"><li>Sometimes makes you jump through hoops in order to do what you want</li></ul>

## Web Framework/Language



For my project's front end, I will use React as I intend on taking advantage of its routing features, as well as its ability to write reactive and real time components. I also like that it makes your HTML code clean and readable and massively reduces boilerplate code by allowing you to create Components rather than having to rewrite massive chunks of code.

I will also use Typescript in my project to ensure that my code is type safe; which should decrease the number of errors and bugs that I encounter during the development process






## CSS Framework (Web Development)



A CSS framework is a tool used by UI developers to make their job of developing the front end of a website easier. Rather than having to start from scratch every time; define colour palettes, grids, flex boxes etc; a framework gives developers the tools to quickly make a UI that can be tweaked and iterated on constantly. Frameworks are also incredibly useful in large teams and companies that need a theme that can be used more than once in a single project; or across multiple projects.

In its simplest form, a CSS framework is a collection of CSS stylesheets that can be used or combined with other CSS classes. They're mainly designed for use in common situations, such as setting up navbars, buttons or text elements.

Software	Pros	Cons
<div>Tailwindcss</div> 	<ul style="list-style-type: none"><li>• I have experience with it</li><li>• Able to create custom colours and modify the framework</li><li>• Powerful for responsive design</li></ul>	<ul style="list-style-type: none"><li>• It can be complicated to create your own colours or customise it</li><li>• Does not come with any pre-made components.</li></ul>
<div>Bootstrap</div> 	<ul style="list-style-type: none"><li>• Simple to use library</li></ul>	<ul style="list-style-type: none"><li>• Does not support responsive design</li><li>• Limited components</li><li>• Limited support for creating custom components</li><li>• </li></ul>
<div>Material UI</div> 	<ul style="list-style-type: none"><li>• Supports React</li><li>• Components look really clean</li><li>• Lot's of prebuilt; professional components</li></ul>	<ul style="list-style-type: none"><li>• Some components are behind a paywall</li><li>• Limited customisation options</li></ul>

## CSS Framework (Web Development)



For my website I will use the tailwindcss framework. I have chosen this framework as it will give me the most control over my website and the different components/colours while still speeding up my development process. I have also chosen tailwind because of the features that it offers in creating responsive design.


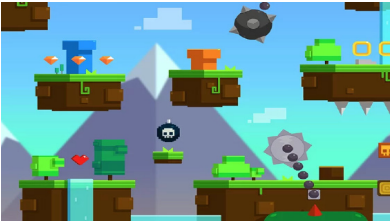


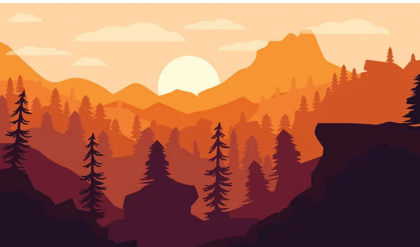
I chose this over something like Bootstrap because I want to still be able to make my website feel and look like my own; whereas other frameworks like Bootstrap predefine the looks for nearly all components (buttons, divs, modals, etc) and it can be incredibly complicated to change them.

# Research - Processes


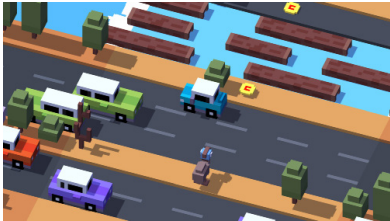


## Art Style (Game Development)



### 2D

Style	Pros	Cons
<p>Pixel Art</p> 	<ul style="list-style-type: none"><li>• Simple to create</li><li>• Easy to look at</li></ul>	<ul style="list-style-type: none"><li>• It can be difficult if you want to try to create high resolution pixel art that looks good</li><li>• Shading can be difficult</li><li>• Lack of details</li></ul>
<p>Vector Art</p> 	<ul style="list-style-type: none"><li>• Simple to create</li><li>• Can be scaled infinitely for different screen resolutions.</li></ul>	<ul style="list-style-type: none"><li>• Lack of details</li><li>• Often looks flat and boring</li><li>• Looks childish and boring</li></ul>
<p>Cutout Art</p> 	<ul style="list-style-type: none"><li>• Interesting to look at</li><li>• Lots of details</li><li>• Look's playful and like a cartoon.</li></ul>	<ul style="list-style-type: none"><li>• Difficult to create</li><li>• May be difficult to animate in Unity.</li></ul>
<p>Monochromatic Art</p> 	<ul style="list-style-type: none"><li>• Don't have to worry about colours (Only 2)</li><li>• Reasonably easy to create</li></ul>	<ul style="list-style-type: none"><li>• Lack of details</li><li>• Makes the game look boring and dull</li></ul>
<p>Flat Art</p> 	<ul style="list-style-type: none"><li>• Lot's of artistic freedom</li><li>• Simple, flat shapes used to represent objects, characters and settings.</li><li>• Makes good use of colour</li></ul>	<ul style="list-style-type: none"><li>• Lack of depth and volume</li></ul>

# 3D

Style	Pros	Cons
Low Poly 	<ul style="list-style-type: none"> <li>Simple to create 3D models and assets</li> <li>Doesn't require a lot of detail for each model (e.g a rock may just be a deformed sphere)</li> </ul>	<ul style="list-style-type: none"> <li>Can be difficult to accurately represent complicated shapes</li> </ul>
Voxel 	<ul style="list-style-type: none"> <li>Simple to create basic shapes</li> </ul>	<ul style="list-style-type: none"> <li>Can be difficult to accurately represent complicated shapes</li> <li>Lack of depth/shading in the models</li> </ul>
Realism 	<ul style="list-style-type: none"> <li>Looks incredible if done correctly</li> </ul>	<ul style="list-style-type: none"> <li>Extremely difficult to create</li> <li>Requires accurate lighting in game scenes</li> <li>Requires accurate materials for all objects.</li> <li>Materials will require depth, normal and texture maps.</li> </ul>
Cartoon 	<ul style="list-style-type: none"> <li>More detail than similar art styles like Low poly</li> <li>Smoother shapes</li> </ul>	<ul style="list-style-type: none"> <li>Difficult to create smooth models in blender.</li> </ul>

## Game Art Style



I have chosen to create my game using a 3D Lowpoly art style. I have chosen this art style because it is a good balance between simplicity, ease of creating and aesthetic appeal. Creating everything in this art style ensures that I will be able to quickly create and iterate over the different design concepts that I may have with an object without having to fuss over finding materials, textures and node graphs.

## Ongoing Evaluation

So far my project and research is progressing well. I have started testing different game mechanics and I am starting to form an idea of what I want the final game to look like.

Choosing 3D lowpoly art was a great decision as I am able to quickly prototype and iterate 3D models while still making them look great.



## 3D Model Files

3D model files were first created in order to allow 3D models to be moved around from software to software; e.g being able to take a model made in blender and put it into Maya to apply textures and animation or vice-versa.

3D model files store information about 3D models. This information can include geometry (it's shape), appearance (it's colour, texture and/or materials), scene (position of lights, cameras and other objects) and animations.

3D model files will store this information as either plain text or binary data.

File Type	Pros	Cons
<b>Fbx</b> (Flexible Binary Exchange Format)	<ul style="list-style-type: none"><li>• Industry Standard</li><li>• Contains all the data about the model (Materials, textures, animations)</li><li>• Supported by Unity</li></ul>	<ul style="list-style-type: none"><li>• Often produces a large file size</li></ul>
<b>Obj</b> (or interchange)	<ul style="list-style-type: none"><li>• Smaller file size</li><li>• Supported by Unity</li></ul>	<ul style="list-style-type: none"><li>• Does not include any texture maps, materials or other properties.</li><li>• Only includes the geometry of a model.</li></ul>
<b>Dae</b> (or Collada)	<ul style="list-style-type: none"><li>• Designed primarily for storing animation data.</li><li>• Supports all other data</li><li>• Supported by Unity</li></ul>	<ul style="list-style-type: none"><li>• Outdated and no longer maintained</li><li>• Not widely supported</li></ul>
<b>Stl</b> (Stereolithography)	<ul style="list-style-type: none"><li>• Most popular for 3D printing</li></ul>	<ul style="list-style-type: none"><li>• Not supported by Unity</li><li>• Only contains the geometry data of a model</li></ul>
<b>Skp</b> (SketchUp Pro)	<ul style="list-style-type: none"><li>• Supported by Unity</li></ul>	<ul style="list-style-type: none"><li>• Can only be created using the SketchUp Pro 3D modeller</li><li>• No textures or UV mapping tools</li></ul>

## 3D Model Files

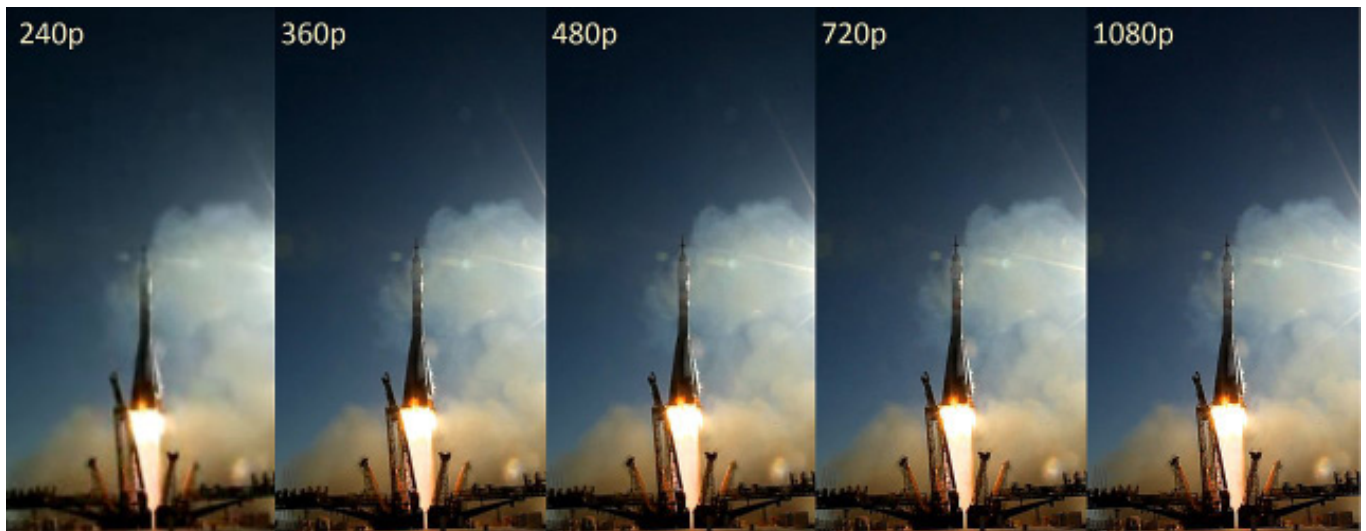


I have chosen to use **FBX (Flexible Binary Exchange Format)** files as they will allow me to transfer files from Blender directly into Unity without having to worry about separately importing materials or textures.

## Image/Video Resolution



The resolution of an image is how many pixels (a point of data containing a colour) there are in an image; usually represented as **Width In Pixels x Height In Pixels**. The higher an image's resolution, the sharper, more detailed and clearer the image will be.



Resolution Name	Resolution	Use
HD 720p	1280 x 720	<ul style="list-style-type: none"><li>• Low quality videos</li><li>• Videos/Images displayed on smaller screens (such as phones/ipads)</li></ul>
Full HD 1080p	1920 x 1080	<ul style="list-style-type: none"><li>• Quality Videos or images</li><li>• Some TV shows/movies</li></ul>
4K Ultra HD	3840 x 2160	<ul style="list-style-type: none"><li>• High Quality Videos or Images</li><li>• Some TV shows/movies</li><li>• Sharper image</li></ul>
8K Ultra HD	7680 x 4320	<ul style="list-style-type: none"><li>• Very high quality videos</li><li>• Some TV shows/movies</li><li>• Provides an incredibly sharp and detailed picture</li></ul>

# Image/Video Resolution



The resolution for my in-game assets will be 1080p. This will ensure that they will appear high quality when viewing in the middle of a match, this is because any time that these assets are used they will likely be scaled down and placed in the scene on the side of a box, crate, pallet or in other locations where the detail of the model is not important.

However for some of my menu assets (Buttons, game title/logo, Icons) I will either use 4K Ultra HD images or I will use Vector Image File Formats (Such as .TIFF). This is because in the menu, not only does it have to support all screen resolutions but the images also need to appear incredibly high quality with smooth, pixel-less edges and curves in order to make the game appear as professional as possible.

## Evaluation

While I am definitely glad that I opted for using high quality images with a large resolution in order to give my game a level of polish; it did not come without problems during development. While I was creating the fading effect between images on the game's loading screen, I initially had **8K Ultra HD** images. However loading and modifying an image this large every frame caused Unity to form a memory leak (Taking upwards of 16GB of RAM) and eventually crashing. This definitely made me realise that while higher resolution images are always going to be best for crisp, clear imagery; sometimes they are not the best option overall.

## Image File Types



A digital image is just a series of pixels (A data point containing colour, usually in the RGB format) arranged in a rectangular array. The more pixels an image has, the clearer that it appears and the larger the image is.

An image file format is the format in which these pixels are stored and compressed. Different file formats have different use cases, benefits and disadvantages.

For my project I will need to use images in a number of different places. I will need to store any of my in-game 2D assets (menu's, icons, etc) as an image so it will be important to choose an image file type that will be suitable for what I need to do.



Resolution Name	Pros	Cons
<b>PNG</b>	<ul style="list-style-type: none"> <li>• Lossless Compression</li> <li>• Allows transparency</li> <li>• Widely supported</li> </ul>	<ul style="list-style-type: none"> <li>• Can have large file sizes</li> </ul>
<b>JPG</b>	<ul style="list-style-type: none"> <li>• Small File Type</li> <li>• Choose the amount of compression (1-100%)</li> <li>• Widely supported</li> </ul>	<ul style="list-style-type: none"> <li>• Lossy Compression</li> <li>• Does not allow for transparency</li> <li>• Generally lower quality</li> </ul>
<b>TIFF</b>	<ul style="list-style-type: none"> <li>• Lossless Images</li> <li>• Extremely high quality</li> <li>• Allows transparency</li> </ul>	<ul style="list-style-type: none"> <li>• Extremely large file sizes</li> <li>• Blender cannot render/export in this format</li> </ul>
<b>BMP</b>	<ul style="list-style-type: none"> <li>• Lossless Image</li> </ul>	<ul style="list-style-type: none"> <li>• Extremely large file sizes</li> <li>• Outdated when compared to formats like TIFF</li> </ul>
<b>GIF</b>	<ul style="list-style-type: none"> <li>• Lossless Compression</li> <li>• Allows Transparency</li> <li>• Can be used to create animations</li> </ul>	<ul style="list-style-type: none"> <li>• Limited to 256 colours</li> <li>• Not widely used/supported</li> </ul>

## Image File Type



I have decided to use PNG files throughout my project; and especially for UI sprite elements. I have made this decision because PNGs are the only viable option for my project. While using .tiff file types would be ideal; because the majority of my sprites are going to be rendered images of 3D models, I unfortunately cannot use .tiff as blender cannot export in a vector file format.

I then looked at using JPG files, however, also due to the majority of my sprites being rendered 3D models to be displayed in UIs, I needed the renders to have a transparent background; which JPGs do not offer.

This left PNG as the only viable option for my project and for my render export format.

### Evaluation

While I didn't have any options other than PNG for my sprites and render exports, I had no problems with using the format and creating sprites with transparent backgrounds inside of Unity.

I would have liked to have spent some time attempting to use .tiff files for some aspects of my project (such as the brick texture that becomes pixelated on large walls) however overall the PNG has worked perfectly.

# Compression



Compression is the process of reducing the size of a file by removing or reworking data and optimising it for storage/use.



	Lossy	Lossless
What it does	Permanently removes data.	Restores and changes compressed data
When it's Use	When quality loss is okay	When quality loss is not okay.
Where it's used	Images, video, audio	Text, images, audio
Common File Types	JPEG, MPEG, AVC, HEVC, MP3, AAC	RAW, BMP, PNG, ZIP, WAV, FLAC

	Lossy	Lossless
Pros	<ul style="list-style-type: none"><li>• Small File sizes.</li></ul>	<ul style="list-style-type: none"><li>• No loss in quality</li><li>• Slight decrease in file size</li></ul>
Cons	<ul style="list-style-type: none"><li>• Lower quality at high rates of compression</li></ul>	<ul style="list-style-type: none"><li>• Files are larger than lossy files</li></ul>



For my project I will exclusively use Lossless compression. I have made this decision because I want to ensure that I am keeping the quality and professional look of my project high in any way that I possibly can; and using lossless file compression doesn't take any extra work; yet can drastically improve the quality of images at smaller file sizes.

# Development of Ideas

While I am still happy with my original idea for the game; I feel that in some aspects of the game I need to simplify the game mechanics, not only so that I will be able to get the project done on time but also so that the game is easier to play and does not require the user to learn a lot of new controls, etc.

One of the main systems that I decided to simplify is the building system. In testing I found that it could be quite complicated to use and while I did my best to make the control scheme user-friendly, it could still be confusing to place the objects exactly where you wanted them to be.

To resolve this issue, rather than having the game be about the players constructing a production line/factory and then having automated workers moving resources around, the production line and factory will already be setup and it will be player's job to keep it running (providing resources to machines, ensuring that they are ordering enough materials, etc). To create more of a challenge I could set up levels in a way where the players have to work together to run the factory (e.g conveyor belt splitting the room down the middle and the player's have to "throw" materials to each other or somehow transfer the items across to each other. I could also add an extra challenge by having machines/conveyor belt's "breakdown" after a certain amount of time.

## Ongoing Evaluation

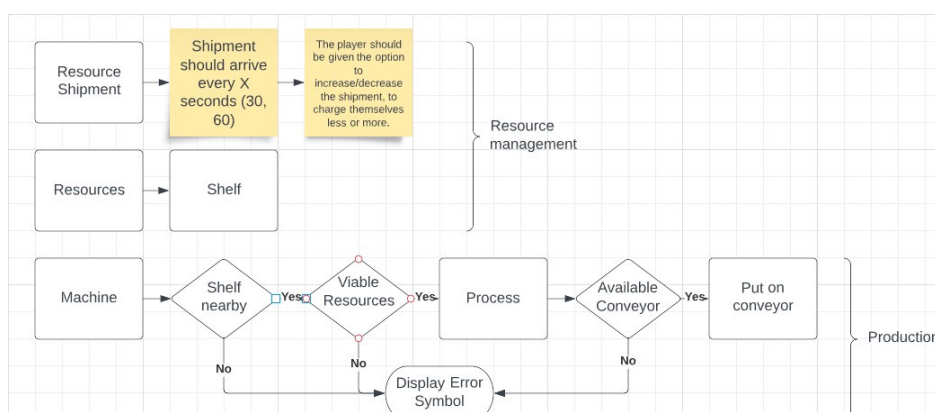
I am happy with how my project is progressing, I have got a decent amount of coding and development done and I am happy with the amount of planning that I have got done and I think that the adjustments that I have made to my initial game design will make the game more enjoyable and easier to play.

## Evaluation

My project ended up being quite different to the game described in my early planning. As development of the game progressed and I started to add more and more game mechanics I realised that if I wanted to go ahead with my original idea of players being able to create their own factory and placing down conveyor belts and machines, the game would get extremely complicated and would be incredibly hard to understand. I also felt that the original idea wouldn't make use of having 2 players as much as I wanted it to.

## Game Mechanics Planning

Throughout my project I am going to make use of flow charts to both plan and communicate the logic behind the different scripts and systems in my project.



This plan requires a lot of player involvement in the resource management of the game; with the player having to actually manage the quantities of each resource in the incoming shipments and then having to ensure that the items are moved away from the incoming delivery to a storage shelf.

I also want the machines to automatically handle the production of different resources and items, with the player only having to ensure that items are placed on a shelf nearby to the machine in order for the machine to be able to access the items, take & process them and then add the output item onto the export conveyor belt.

### **Design Modification**

My original concept for the machines changed a lot, and very early in the creation of my project. I realised that a system in which the machines would have to look for nearby shelves, get the correct item, process it and then export onto a conveyor belt for the item to take away would become extremely complicated to code, and wouldn't add value to the gameplay.

I also ended up simplifying the pallet orders and restocking materials/resource management aspect of the game. Instead of requiring the player to specify how many of each item to order, and when to order; I wanted to create a system for the player to be able to press a "restock" button and the game will calculate how many of each item type to order.

### **Evaluation**

I am extremely glad that I made this change in how to resource management and machines worked in my game. Not only did requiring the players to directly input items into the machine and take the export item make the code for the machines simpler, it also made sure that even with a small level with simple game mechanics, the players constantly had things to attend to and to interact with in order to keep the factory running smoothly.

## **Pallet Mechanics Planning**

In my project, one of the ways that the players will be able to receive, store and export will be pallets. There will be 2 key pallets on the factory floor, the import/incoming pallet and the export pallet. The incoming pallet is where all the items will appear when a player's order is complete and submitted and the export pallet is where players will be able to place finalised products to be exported. Once an item is exported the player should be compensated with some form of currency.

### **Design Modification**

The concept of a pallet stayed roughly the same throughout my project however there were some modifications done to the import pallet.

The export pallet functions exactly as I planned it to, taking items that players place on them, destroying the physical item and then rewarding the player with currency.

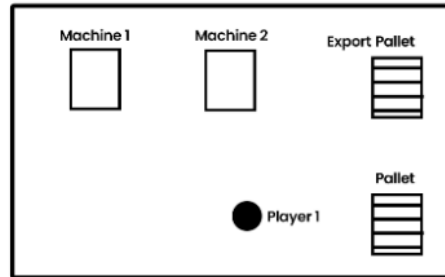
However during play testing with one of my classmates, he suggested that instead of having all of the items appear on a single pallet (only on one half of the map, only giving a single player access to raw materials); that each player gets 2 raw resource types on their half, forcing the players to work even closer together. I tried implementing this by creating 4 separate pallets, one for each item type and then placed 2 on each half of the map. After some more play testing, we decided that having to pass even more items across halves made the game more challenging, but also a lot more interesting so I decided to keep the change.



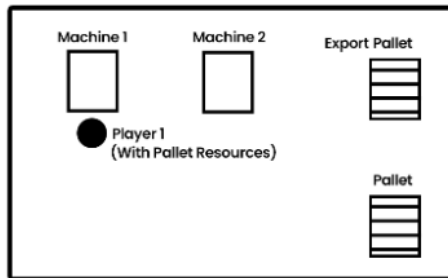
# Storyboards



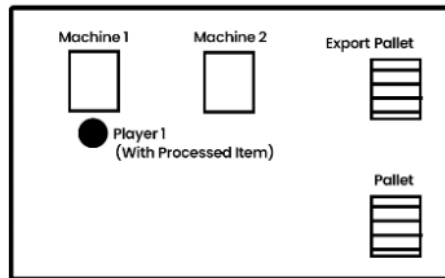
Menu Screen



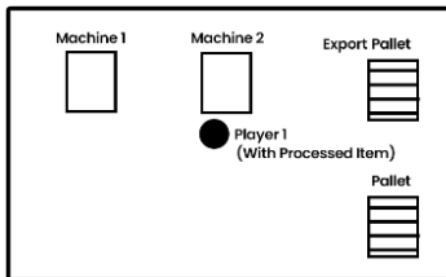
Level 1/Intro/Tutorial (Top Down)



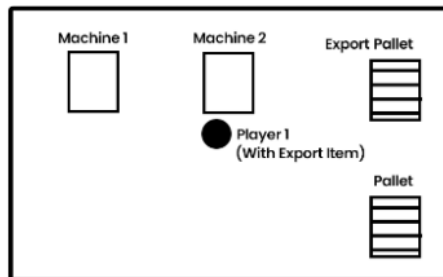
Player takes resources from pallet to Machine 1



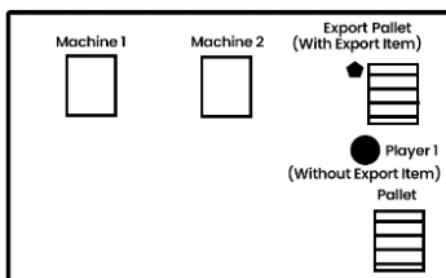
Machine 1 "processes" the resources and provides the player with another item



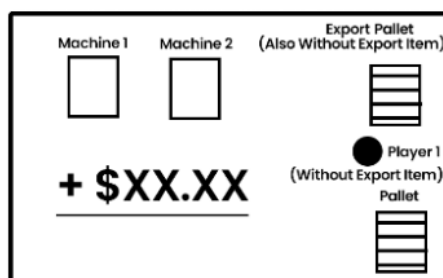
Player takes item from machine 1 to Machine 2



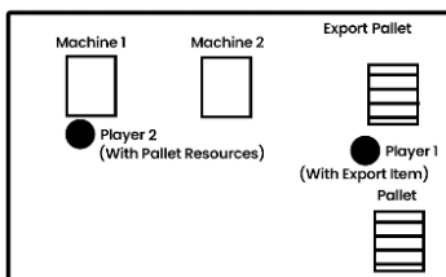
Machine 2 "processes" the item and provides the player with another item (In this case; the



Player can place the item on the export pallet



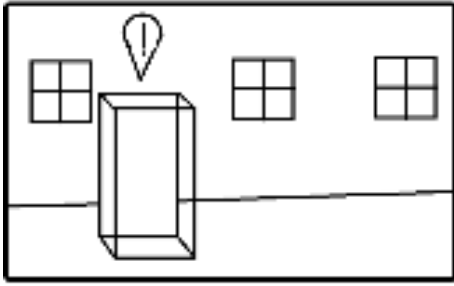
The item is then sold from the export pallet on the next buy/sell cycle.



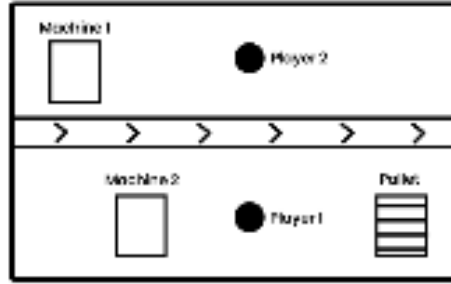
This gameloop could also function with 2 or more players



# Challenges / Level Design



Machine breaking down



Level Split by Conveyor (Top Down)

## Ongoing Evaluation

So far my storyboarding and planning has helped me to build out the basic game loop and mechanics and ensure that the game will actually play smoothly. One of my classmates brought up the idea that the game could support couch co-op (2 or more players) so I have added some boards to my storyboards exploring how multiple players could work.

## Website Storyboard - Home/About Page

For my website's homepage I want it to clearly display where the user can get the game from (As if this was an actual game I would want to drive sales towards its Steam/Epic Game page). I intend on doing this by making these buttons really stand out when looking at the webpage. I have done this on the home page by giving the button it's own border in the Navbar and I have tilted the "Get The Game" banner slightly to one side to immediately draw the user's attention to it.

At the top of the page I will include a short, looping video displaying some gameplay footage.

## Ongoing Evaluation

I am planning on using Figma to create my Website Storyboard. I'll make use of very simple shapes and placeholder objects in order to just plan out where each component should go on the page without actually adding in any colour or final images.



## ABOUT

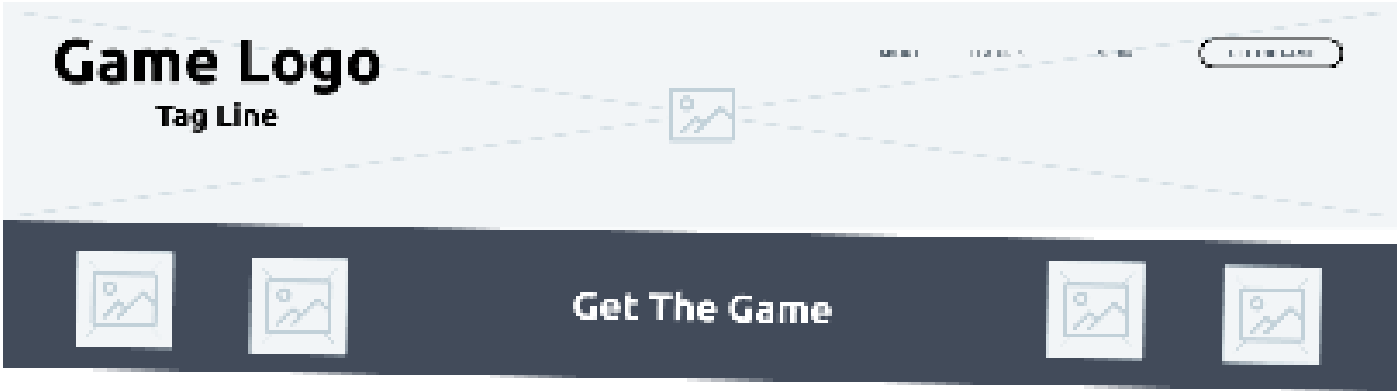
About...



## Website Storyboard - Features/Roadmap

For my website's Features/Roadmap page, I want to once again include the same navbar, header and "Get The Game" banner. The page will consist of a list of the currently included features and "upcoming" features in the games. For the purpose of this project, the "upcoming" features will be features that I think would be a great addition to the game; but were not within the scope or timeframe of this project.

The roadmap will be similar to the upcoming features section and will go more in depth into what could be added to the game.



# Main Features

## Features (Released)

- ★ Feature
- ★ Feature
- ★ Feature
- ★ Feature
- ★ Feature

## Features (Upcoming)

- ★ Feature
- ★ Feature
- ★ Feature
- ★ Feature
- ★ Feature

# Roadmap

- 1 "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
- 2 "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
- 3 "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
- 4 "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute

## Website Storyboard - Media

The media page on my website will be used primarily to display my 6 minute video, but will also display additional images and videos of the development process/the finished product.

# Game Logo

Tag Line

ABOUT

FEATURES

MEDIA

GET THE GAME



Get The Game



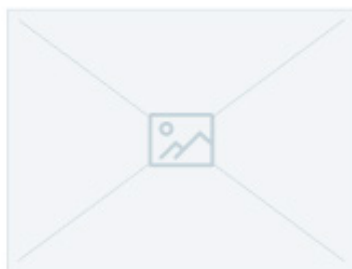
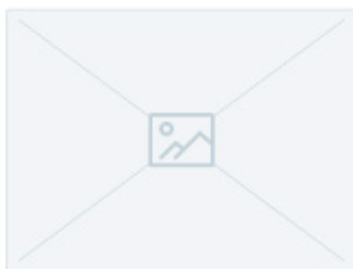
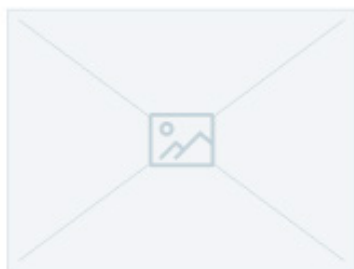
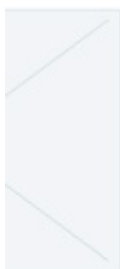
## Media

6 Minute Video



## Image gallery

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim.



## Ongoing Evaluation

While my web design is definitely rough, I think that I have clearly laid out where the content of my website will be placed, and gives me a rough idea of what I can include. I will have to refine this design before I begin the development process however I am happy with the overall look and spacing of web components.

## Time Planning

Task	Week															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>Game</b>																
Player Movement/Input																
Camera																
Building System																
Conveyor Belts																
Inventory System																
Items																
Pallet Deliveries																
Machines																
Sale Point																
Game Loop																
Levels																
Sound Design																
Menu																
Polish																
<b>Website</b>																
Design																
Creating of Assets																
Development																
Polish																
<b>3D Modelling</b>																
Shelves																
Conveyor Belts																
Crate																
Pallet																
Item Models/Renders																
Machines																
Character																
<b>Portfolio</b>																
Research																
Documentation																
Finalisation																
Create in InDesign																
Polish																

## Ongoing Evaluation

So far the timeline for my project looks good and I believe that I will realistically be able to get everything done in the time I have set for myself. I hope to complete the bulk of my project 2 weeks early in order to use those last 2 weeks to make everything as polished as possible and make any small tweaks/adjustments that may be necessary to improve my project or my portfolio.

# Finance Planning

I don't intend on spending any money on this project and will make use of technology and software I already own, so my budget will be \$0.

Item	Proposed Cost	Actual Cost
<b>Home PC</b>	\$5,905	\$0 (Already purchased)
Gigabyte GeForce RTX 4080 AERO OC 16GB GDDR6X Video Card	\$2,299.00	\$0 (Already purchased)
Intel Core i9 13900K 24 Core LGA 1700 3GHz Unlocked CPU Processor	\$999.00	\$0 (Already purchased)
Gigabyte Z790 Aero G DDR5 Motherboard	\$569.00	\$0 (Already purchased)
Corsair RM850 Gold Modular 850W Power Supply White V2	\$219.00	\$0 (Already purchased)
Samsung 980 Pro 2TB PCIe 4.0 NVMe M.2 SSD - MZ-V8P2T0BW	\$388.00	\$0 (Already purchased)
Corsair Vengeance RGB 32GB (2x 16GB) DDR5 6000MHz C40 Memory - White	\$369.00	\$0 (Already purchased)
Corsair iCUE H150i ELITE CAPELLIX Liquid CPU Cooler - White	\$329.00	\$0 (Already purchased)
Seagate ST8000DM004 8TB BarraCuda 3.5" SATA3 Desktop Hard Drive	\$189.00	\$0 (Already purchased)
Samsung 970 EVO Plus 1TB NVMe 1.3 M.2 (2280) 3-Bit V-NAND SSD - MZ-V7S1T0BW	\$175.00	\$0 (Already purchased)
iCUE 5000X RGB QL Edition Mid-Tower ATX Case	\$369.00	\$0 (Already purchased)
<b>School PC</b>	\$4394.9	\$0 (Provided by my school)
Gigabyte B760 GAMING X AX DDR5 LGA 1700 ATX Motherboard		\$0 (Provided by my school)
Microsoft Windows 11 Home 64-Bit USB Drive - Retail Box		\$0 (Provided by my school)
Corsair iCUE SP120 RGB ELITE 120mm PWM Case Fan		\$0 (Provided by my school)
Intel Core i9 13900K 24 Core LGA 1700 3GHz Unlocked CPU Processor		\$0 (Provided by my school)
Corsair iCUE H150i ELITE CAPELLIX XT 360mm Liquid CPU Cooler - Black		\$0 (Provided by my school)



Corsair Vengeance RGB 32GB (2x 16GB) DDR5 5200MHz CL40 Memory		\$0 (Provided by my school)
Samsung 980 Pro 2TB PCIe 4.0 NVMe M.2 SSD - MZ-V8P2T0BW		\$0 (Provided by my school)
Gigabyte GeForce RTX 4070 Ti GAMING OC 12GB Video Card		\$0 (Provided by my school)
Corsair iCUE 5000X RGB Tempered Glass Mid-Tower ATX Smart Case - Black		\$0 (Provided by my school)
Corsair RM850x 2021 850W 80 Plus Gold Fully Modular ATX Power Supply		\$0 (Provided by my school)
<b><u>Unity</u></b>	\$0 (free)	\$0 (free)
<b><u>Adobe Creative Cloud</u></b>	\$21.99/month (Student Plan) \$87.96 (4 Months)	\$0 (Provided by my school)
<b><u>Blender</u></b>	\$0 (free)	\$0 (free)
<b><u>Blue Yeti Microphone</u></b>	\$198.00	\$0 (Already purchased)

### **Evaluation**

For my project I had a budget of \$0 and I managed to stick to this budget. I made use of free, yet powerful software such as Blender, Unity and Visual Studio.

Sticking to this budget was definitely made easier because I had already purchased anything that I would need that wasn't free/provided by my school, such as a microphone and my own computer.

# Evidence of WHS

## Screen Time

- Ensuring lighting is even without glare/reflections.
- To reduce glare, purchase a screen with a matte or light diffusing surface.
- Set screen brightness to a comfortable level.
- Look away for a few minutes every 10 minutes, if your eyes begin to hurt.
- Use negative contrast.

## Ergonomics

### Mouse

- Your mouse should not be too bulky and should allow the wrist to be at a comfortable angle while using it, it should also be a comfortable distance away from the user.

### Distances

- Height of desk: If Fixed: 680mm to 720mm above floor level.
- If Adjustable: 580mm - 730mm above floor level
- The screen should be 350mm-750mm away from your eyes, and 30-40mm below eye level.
- Volume of leg space. Minimums:
- Width: 800mm. Depth: 550mm. Height: 580mm
- Area of work surface. Width: 1500mm. Depth: 900mm Bench Thickness Over Leg Span: 25mm

### Keyboard

- The keyboard should be in a position so that the elbows are close to being horizontal and the wrists are kept straight.

### Chair

- Chairs should swivel, have five wheels for stability, have breathable fabric on the seat, a rounded front edge and have adjustable height and backrest for lumbar support.

## Cables

- Cables should be kept neat and held together, out of the way to minimise any risk of getting your foot caught on one while sitting down.
- Cables running across the floor should be taped down to reduce tripping hazard.

## Lighting


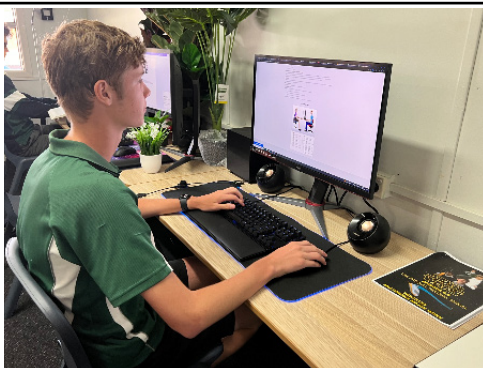
- VDUs (Visual Display Unit) should be placed to the side of light sources, not directly underneath.
- If possible VDUs should be placed between rows of lights
- If the lighting is fluorescent light strips, the sides of the desks should be parallel with the lights.
- Screens should not be placed near windows but if this is unavoidable, neither the screen nor the user should be facing the window.


## Sound

- Noise should be limited to an average of 85 dB over a period of 8 hours. Anything above this could cause temporary or permanent hearing loss

## Sound

- You should take a 15 minute break for every hour of work for tasks concentrated on a computer, or you can take a 15 minute break every 2 hours for less concentrated work.

Ergonomics	
Good	Bad
<ul style="list-style-type: none"><li>• Screen is a good brightness that doesn't hurt your eyes.</li><li>• Wrist rest with keyboard.</li><li>• Mouse keeps wrist straight w/out angle.</li></ul>	<ul style="list-style-type: none"><li>• The screen is slightly too high (Eye level is looking at the upper half of the screen, not over it)</li></ul>
	

Hazards	
Good	Bad
<ul style="list-style-type: none"><li>• Cables are out of the way of the walkway and are not a major tripping hazard.</li><li>• Cables on top of the desk are neat.</li></ul>	<ul style="list-style-type: none"><li>• Extremely bad cable management.</li><li>• My foot often gets caught on cables..</li></ul>
	

## **Evaluation**

My workstation is not very WHS compliant. The cables are a mess and the table is not at the right height for me, and I am unable to adjust it. This could lead to me having bad posture or aches while I am working. However the keyboard and mouse are very comfortable for me to use which allows me to work for a long time before my wrists start to hurt. When I am using the headphones I always have the volume at 50%, meaning that unsafe volume levels are not a major concern.

As I am at school, it is unlikely that I will be able to get an adjustable desk, or a more comfortable chair as that would be unrealistic.

However I could very easily purchase some cable ties or a Corsair cable kit and fix the cable management. All I would need to do is run the cables across the back of the desk and then down to the PC.

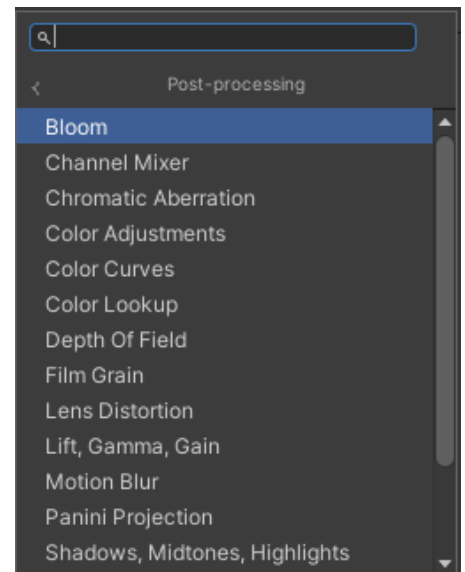
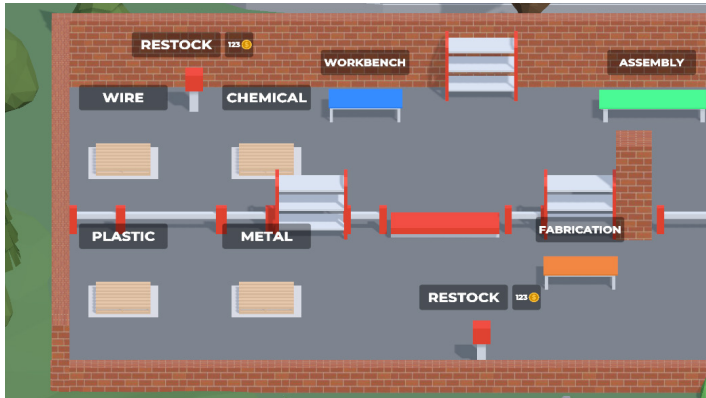
# Post Processing - Tone Mapping



Unity's build-in render pipeline, the Universal Render Pipeline (URP) supports adding post processing effects to enhance the visuals of your game. Unity comes with 14 built-in post processing effects.

In my project I didn't want to overuse post processing effects because I don't want someone viewing the project to be overwhelmed by too many effects that may not necessarily make the game look any better.

This is what the game's graphics look like without any post processing:



The first post processing effect that I added was Tone Mapping.

Tone mapping is the process of remapping HDR values of an image to a new range of values. In Unity there are 2 tone mapping "modes".

**Neutral** - Range remapping with minimal; impact on colour hue & saturation.

**ACES** - References ACES tonemapper, giving a very cinematic look with lots of contrast and has a large effect on the colour hues and saturation.

Tone Mapping Mode	Image
<b>None</b>	
<b>Neutral</b> Desaturated image, colours aren't as vibrant, low contrast between colours, easy to look at (colours aren't bright and doesn't overwhelm the viewer)	
<b>ACES</b> Blacks are much darker, colours are darker as well. The colours have a lot more contrast and they stand out.	

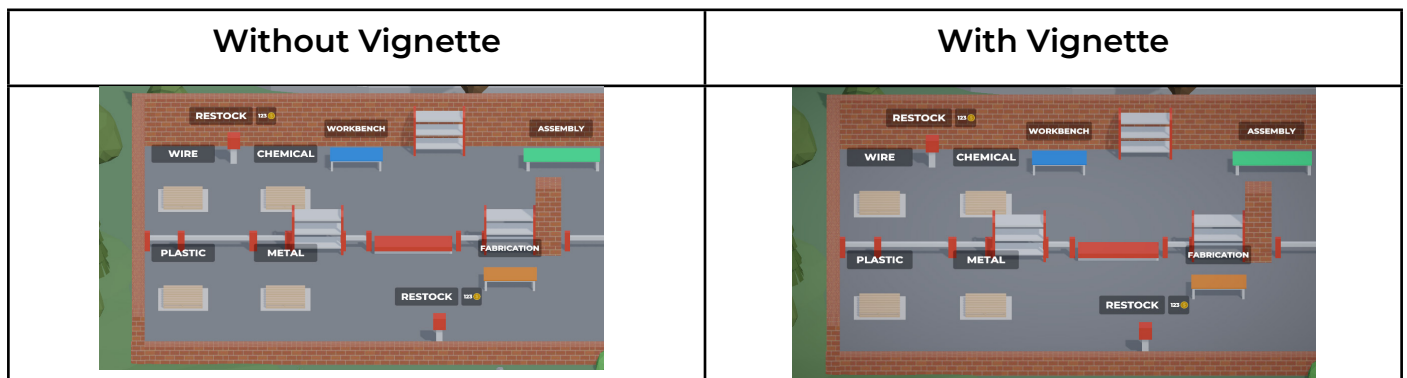
## Ongoing Evaluation

I have decided to use the neutral tone mapping option. While I don't like how much it desaturates the image, I think that it looks the best of the 3 options (None, Neutral and ACES) and is easy on the eyes to look at.

## Post Processing - Vignette

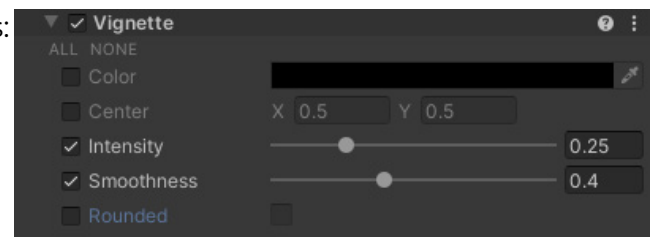


The next effect that I wanted to add to my game was a vignette. A vignette in Unity is an effect that darkens the edges of an image, leaving the center of the image brighter. A vignette is incredibly useful for drawing people's attention towards the center of an image, and because in my game the camera will always move to keep both players in the center of the screen, a vignette will really help make it clear where the players are so users won't get "lost" looking at other aspects of the map.



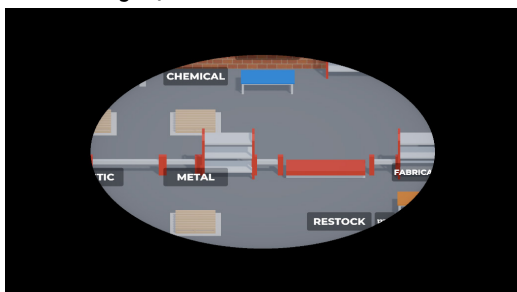
Unity also provides a range of settings for vignettes:

- Colour (Modifies the colour applied)
- Center (The position that the vignettes applies around)
- Intensity (How strong the vignette is)
- Smoothness (How smooth the vignette is)

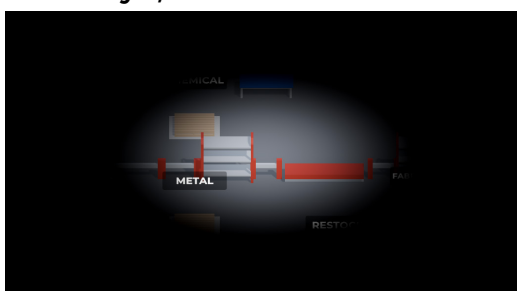


I only needed to modify the intensity and smoothness to get it to look good. I ended up using relatively low values as I didn't want the vignette to appear too strong as it wouldn't fit the art style of my game. A strong vignette is often used in horror games in order to not allow the user to see the corners of the screen creating even more suspense and shock when events happen.

### Intensity 1, Smoothness 0

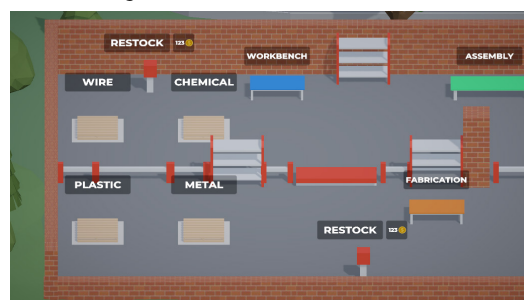


### Intensity 1, Smoothness 1



As the images on the left show, a vignette can be extremely harsh or take up too much space if the settings are configured incorrectly. I found that the ideal balance was:

### Intensity 0.25, Smoothness 0.4



These settings ensure that the vignette has the intended effect of focusing a user's view to the center of the screen, while the user may not even consciously recognising that a vignette has been applied.

## Post Processing - Bloom

