**Assessment 2 – Listing Technologies.**

Collaborative Workspaces.

**Microsoft Teams.**

This is our primary method of communication. We hold all of our weekly group meetings and mentor meetings in this workspace, use it regularly for communication, and collaboratively work on documents required for our project.

Company: Microsoft.

Latest version: No version information available.

Cost: $0.

Link: <<https://www.microsoft.com/en-AU/microsoft-teams/group-chat-software>>

**GitHub.**

Our team performs regular uploads of our program code to GitHub to collaboratively work on our program. It allows us to dynamically accept and merge work done by our team mates, update others work to our local files, and revert to previous versions should we experience issues with any changes made to our build.

Company: GitHub.

Latest version: No version information available.

Cost: $0.

Link: <<https://github.com/>>

Project Repository: <<https://github.com/RossRRMIT/BITS_SP1_Group10_2DGame>>

**Eclipse IDE.**

We have chosen to use Eclipse IDE as the workspace that we will use to program our game in. We have chosen it becuase of our shared fimiliarity with its features, and because it is a proficient tool for what we require it for. It allows us to break our program into smaller classes that we can work on independently, then load them into GitHub to provide collaboratibility.

Company: Eclipse Foundation.

Latest version: 2021-03.

Cost: $0.

Link: <<https://www.eclipse.org/>>

**Trello.**

Trello is a task allocation and tracking collaboration tool that our team uses quite significantly. We use it to separate the tasks into our own personal areas, collaboratively track how each other is going, and upload assets and documents so they can be accessed and viewed by one another.

Company: Atlassian.

Latest version: No version information available.

Cost: $0.

Link: <[https://trello.com](https://trello.com/)>

Software.

**Java SE 16**.

We are using the latest Java release as the language to write our program in. It is a powerful, highly flexible language that allows us to create a game within our shared ability levels that is still complex and inidicative of our projects goals.

Company: Oracle.

Latest version: 8.

Cost: $0.

Link: <[https://www.java.com/en/](https://www.java.com/en/%22%20/t%20%22_blank)>

**LWJGL (Light-Weight Java Gaming Library).**

We are using the highly complex and powerful LWJGL to assist us in creating our project. It is a library that contains multiple API’s geared towards producing high-fidelity 3D games. Though we are not creating a 3D game, many of the features are still useful to us in order to create a functionally polished game.

Company: LWJGL.

Latest version: 3.2.3.

Cost: $0.

Link: <[https://www.lwjgl.org/](https://www.lwjgl.org/%22%20/t%20%22_blank)>

The main features we have chosen to use include:

* [OpenAL](http://openal.org/):

An audio library originally created for C/C++ that has been translated for use with Java. It handles loading audio data from file, storing the data, playing sounds dynamically in a 3D space, and managing memory.  

* [OpenGL](http://opengl.org/):

OpenGL is a popular library with a lean API that exposes functions allowing programmers to buffer data to the GPU for rendering, commonly this information will be in the form of “vectors”, populated with “floating-points” as GPU’s excel at floating-point computation. OpenGL supports a lot of modern day graphic technologies such as shaders and numerous other advancements. OpenGL supports both 2D and 3D vector graphics.  

* [NanoVG](https://github.com/memononen/nanovg):

A vectorized rasterization library for 2D graphics. This library is built around OpenGL and exposes an API for programmers to utilize, allowing them to easily create basic geometry shapes such as rectangles, circles, and triangles. This library rasterizes these shapes into vectors which OpenGL will interpret and buffer for the GPU to render.  

* [NanoSVG](http://nanosvg/):

A library that will parse a supplied SVG coordinate list and rasterize into geometry shapes that are supported by the SVG standard, much like NanoVG this library exposes an easy API for programmers, this library however does not depend on any external libraries such as OpenGL, instead this library simply parses supplied data, rasterizes and vectorizes it, and returns it as an array that a programmer can iterate over and implement in any way they see fit.  

SVG is a popular file format that describes “points” and “shapes” which forms “icons” or “images”, commonly SVG is used in the web space by front-end developers and provides theoretically infinite scalability, ideal to suit the technological push towards higher resolution displays and increased pixels-per-inch (PPI) or dots-per-inch (DPI).  

* [STBVorbis](http://stbvorbis/):

STB Vorbis is a small library created by Sean Barrett, also known as “nothings”, it is originally a single-header library for the programming language C and Vorbis is designed to parse the .ogg file format and extract relevant information from the sound file format, examples being the number of channels, the format the audio is encoded in, frequency, length, and other numerous things. LWJGL has converted this single-header C library into Java and gives us the choice to include it in our configuration.

Tools.

**Lucid App (Lucid Chart).**

We have chosen Lucid Chart as our diagram production tool. All of the diagrams you see in this report are created in Lucid. It is a highly flexible diagram production tool with a wide array of features allowing us to create in-depth, explanatory visualtizations of our program structure. Our team has chosen UML as our diagram standard.

Company: Lucid Software Inc.

Latest version: No version information available.

Cost: $0 - $11/month.

Link: <<https://lucid.co/>>

**Figma.**

Figma is a design tool that is directed at creating wireframes and visual prototypes of digital applications. It has allowed us to create highly descriptive visual representations of actions within our games and share our ideas with one another. This helps us to maintain a clear vision of the requirements of our project and to not waste time on unnecessary activities.

Company: Figma.

Latest version: No version information available.

Cost: $0.

Link: <<https://www.figma.com/>>

**ShareX.**

ShareX is a screen capture tool that primarily allows for screen capture, be it screenshots or recordings, though is has many other functions including but not limited to image and video alteration, uploading to external repositories, debugging, gif creations and conversion, FFMPEG, and many other capabilities.

Company: ShareX Team.

Latest version: 13.4.0.

Cost: $0.

Link: <<https://getsharex.com/>>

**Photoshop.**

Photoshop is used as a digital editing software, using various tools, Pixel art can easily be made, by using grids, small canvas’ a frame of pixel art can then br created and made into an animation, that you can then edit the anime frame by frame, in addition to using photoshop, i came upon some script that i could run from within photoshop that takes the frames on an animation made, and then generates a Sprite Sheet upon those frames and placing them within a predetermined buffer that is decided upon creation

Company:  Adobe.

Latest version: No version information available.

Cost: $21.49/month.

Link <<https://www.photoshop.com/en>>

**Ableton Lite.**

Ableton produces a light version of their audio recording, mixing and mastering software – Ableton Lite. This will be the tool that we use to produce all of the sounds for the game. It is extremely powerful, even as a light version, and will enable us to create complex, specific audio relevant to the actions in game.

Company:  Adobe.

Latest version: 10.

Cost: $0.

Link <<https://www.ableton.com/en/products/live-lite/>>

Resources.

**Oracle documentation.**

The official Oracle Java documentation website. This provides our team with the information necessary to research and learn about Java functions not currently known to us, or a well explained version of concepts we are trying to convey to one another. It is also an absolute must when utilizing unusual imports and methods as often the parameters cannot be assumed and require some form of explanation.

Link: <<https://docs.oracle.com/en/java/javase/11/docs/api/index.html>>

**LWJGL documentation.**

The LWJGL documentation is listed on the Java API documentation website. As most of us are new to LWJGL, it has been a very commonly accessed website by us as we learn many new methods in an entirely unknown library to us.

Link: <<https://javadoc.lwjgl.org/overview-summary.html>>

**YouTube.**

YouTube has provided us with a wellspring of information as many of us learn new ways to produce our project. This isn’t entirely limited to LWJGL (which is convenient as there are very few current tutorials on LWJGL), but covers all of the tools and technologies covered by our project including but not limited to: LWJGL, OpenAL, OpenGL, NanoVG, NanoSVG, PhotoShop, Ableton Lite, Eclipse, Java, and all other aspects of our technical library.

Link: <<https://www.youtube.com/>>

**GitHub.**

Not only is GitHub an incredibly useful file version sharing tool, it also contains a massive range of documentation and demonstrations. We have used it mostly for information on LWJGL related API’s, but there are many other pages with swathes of useful information.

Link: <<https://github.com/>>

LWJGL: <<https://github.com/LWJGL/lwjgl3/tree/master/modules/samples/src/test/java/org/lwjgl/demo>>