Space Invaders

**Task 1 – P1.1**

***Research the current state of the art with respect to game engines and outline what the most common game engines in use today are. Write a short paragraph explaining at least three interactive game engines with one example of a game from each.***

There are various game engines that are being developed and used nowadays since there are consoles, hand held devices, touch screen monitors and so much more to come in the future. Thus, companies are always developing game engines for users to bring them a more realistic and fun experience. Various companies develop their own game engine as some on the market don’t have what it takes to provide their users the ultimate gaming experience. Some game engines worth mentioning are the following:

***Anvil Engine***



Ubisoft’s Assassin’s Creed II

If Ubisoft hadn’t developed this game engine, probably the Assassin’s Creed games, one of the world’s most popular games wouldn’t have been a success. Ubisoft had developed this engine from scratch as they needed something new and fresh to give gamers detailed environments, realistic cloth mechanics, a variation of AI algorithms as well as lights, reflections and texturing. This all helped the company’s success in making Assassin’s Creed I and II and Prince of Persia a must have and the engine a top priority in the market.

***Unreal Engine***

Probably one of the engines that has been used in a lot of games for a long number of years and has been tested, developed and enhanced to greater heights. The company that created this engine, Epic Games, have come a long way in always maintaining this engine for such a long time as well as keeping up with the times and technology. Although it started out in 1998, it’s still being used today on the world’s leading consoles and PC games. The biggest advantage of this engine is the landscapes. It is known for being user friendly when it comes in generating such items as one can see in games like Borderlands, Mass Effect and BioShock.



Epic Games’ Mass Effect

***EGO Engine***

If you’re interested in creating cars and racing them, then this is the engine for you. Codemaster’s EGO Engine, formerly known as NEON, is one of the engines that are on top of the market for cars, racing and terrain, high details from far and near, smoke, lights and speed. Its impressive AI is also renowned amongst gamers that thrive and have played DiRT and DiRT2.

Codemaster’s DiRT 2

**Task 2 – P1.2**

***Explain how different screen sizes are catered for using the Unity game engine, with reference to the ScreenToWorldPoint function.***

The ScreenToWorldPoint function helps developers and programmers to create games for any sort of device from a PC or a game console to a hand held device. How is this possible? It is because the default screen setup of Unity is that at the bottom left corner the co-ordinate is 0, 0. Using the ScreenToWorldPoint, the developers can input their appropriate variables and needs for the game and then Unity will calculate how it should be Built and Exported depending on the devices.

**Task 3 – P2.1**

***Explain how a simple sound effect is implemented in Unity, with reference to audio listeners.***

A sound effect is implemented by having an audio listener on the main camera. If there won’t be such a listener, then the audio from the other audio sources won’t be audible. Each and every element that requires a sound needs to have the Audio Source component added to it and the appropriate clip dragged onto the said component.

**Task 4 – P2.2**

***Record and generate an effective laser sound for your game. To do this you need to present the following three .mp3 format sounds: the original sound recorded, a description of the effects applied to the sound and the final modified sound.***

To record the sounds needed for the game I found a virtual synthesizer on [www.audiosauna.com/studio](http://www.audiosauna.com/studio). It had various presets of lasers and spaceships thus I arranged a couple of settings and experimented until I found a couple of sounds that I liked and deemed appropriate for the game. I later converted them to MP3 and applied the effects in Adobe Audition CS6 as shown on the next page.

|  |  |  |
| --- | --- | --- |
| **Original Sound Name** | **Laser Sounds\_RAW.wma** | Virtual Synthesizer found on www.audiosauna.com/studio |
| **Effects Applied** | Converted to MP3  Removed Noise  Saved each laser noise in separate files  Added Echo and Adjusted Reverb d ‘Lost In Space’ |
| **Final Sound** | Spaceship Laser Sound\_Final.mp3 |



Selecting each laser sound separately and saving them as individual files

**Task 5 – P3.1**

***Explain how to change materials of different objects based on in-game events, with reference to the OnTriggerEnter and OnTriggerExit functions.***

The materials can be changed by first creating an array that will keep all the relative materials. This can then be called by the OnTriggerEnter and OnTriggerExit functions. An example of this is when the spaceship gets hit.

When it does, the spaceship will change its colour for a small second and will go back to its original colour. This method however, was later removed to make way for the 4 frame animation of the spaceship. The code would be like this:

**Var spaceshipColours: Material [];**

**Function OnTriggerEnter()**

**{**

**if(other.gameObject.tag=="enemylaser")**

**{**

**health--;**

**this.render.material = spaceshipColours[1];**

**}**

**}**

**Function OnTriggerExit()**

**{**

**this.render.material = spaceshipColours[0];**

**}**

**Task 6 – P3.2**

***Draw a sketch of three screens of your game:***

***Main Menu***

SPACE INVADERS

New Game

HELP

EXIT

***High Score***

Main Menu

HIGH SCORE

Matthew 100

Jack 90

R2-D2 65

***Game Screen***

Name: Matthew Score: 25 Health: 85 Shots Hit: 10 Shots Missed: 20 Shots Fired: 30 Aliens: 8

**SPACESHIP**

**INVADERS**

**Task 7 – P4.1**

***Conceptualize an interactive system by explaining how to interrupt or control a sequence of events using Unity, with particular reference to the start(), update() and WaitForSeconds() methods.***

Unity has various techniques and systems to control, process and stop the program that the user is running. Some processes can happen with certain conditions while others can be executed after a couple of seconds.

The in-built functions Start(), Update() and WaitForSeconds() are such tools that the programmer can use in Unity so as to execute desired results. The code shown in the examples below can be found in ***SpaceshipController.js***

***Function Start()***

This function is usually used for assigning numbers, texts or Booleans for the variables needed in the current script. If the script is called on multiple times, it will always start from this function and execute whatever code it has in it, not just assigning variables.

The example below shows variables being assigned with particular numbers to start with as well as not to destroy the Spaceship upon going from one scene to the next.:

**function Start ()**

**{**

**health = 100;**

**shotsfired = 0;**

**shotshit = 0;**

**score = 0;**

**totalscore = 0;**

**checkpowerup = false;**

**DontDestroyOnLoad(this.gameObject);**

**}**

***Function Update()***

This function is one that is constantly called by the program. Its job is to constantly check conditions and inputs from the user so as to execute the appropriate commands at the appropriate time. The example below shows that it’s always checking if the player’s health is less than or equal to 0. If it is then it will destroy the spaceship from the screen and load the appropriate level. If not, the program will continue running accordingly. The else statement is applied so as the health will never exceed 100 should the player collect a health boost when the health is at 96 or higher.

**Function Update()**

**{**

**if (health <= 0)**

**{**

**Destroy(GameObject.FindGameObjectWithTag("spaceship"));**

**Application.LoadLevel(8);**

**}**

**else**

**{**

**if (health > 100)**

**{**

**health = 100;**

**}**

**}**

**}**

***WaitForSeconds()***

This function will stop the game object from whatever it is doing and will restart it after the number of brackets the programmer has assigned to it. In the example below, the program will wait 5 seconds before executing any other lines of code.

**if(other.gameObject.tag=="speedboost")**

**{**

**Destroy(GameObject.FindGameObjectWithTag("speedboost"));**

**checkpowerup = true;**

**speed = 100;**

**yield WaitForSeconds(5);**

**checkpowerup = false;**

**speed = 15;**

**score += 2;**

**}**

**Task 9 – P4.3**

***Write a short paragraph explaining what improvements you would have implemented in the game if you had more time.***

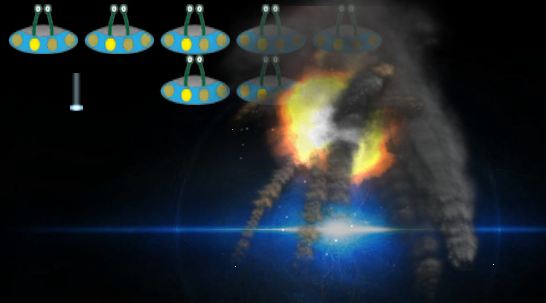
If more time and knowledge was given for this assignment I would have added the following improvements:

1. ***Keep a High-score*** – I tried to implement this, however with the lack of knowledge and not managing to find an appropriate and reliable source on the internet, the tests and code weren’t working appropriately.
2. ***Bonus Level*** – After every 2 levels I would have liked to implement a bonus level where the user will have a mini-challenge such as collecting as many stars as possible in a certain amount of time.
3. ***More Animations –*** The game would have been smoother if more animations would have been applied to it rather than just four frames. However, with time constraints, this was quite tedious and time consuming.
4. ***More Sounds –*** The game would have been more interesting if background music was applied to it as well as having more sound effects or for the user to choose what type of sound effects to listen to.

**Task 12 – M3.1**

***Use the appropriate structure and approach to creating appropriate visual cues in interactive contexts by adding explosions to the game.***

To create the explosions for the game I used a package called ***Detonator Explosion Framework*** from the following cited source (Technologies, 2012). With only a few lines of code, I managed to attach the package prefab ***Detonator-Insanity***. I then edited a few variables from the prefab such as size, colour, and how long it takes to explode and applied in game.



A screenshot of an Explosion in the game

**Task 14 – D1.1**

***Show that conclusions have been arrived at through synthesis of ideas and have been justified by creating a consistent visual theme for the entire game, including fonts and backgrounds. Justify the look of your game against examples of other similar games by including two screenshots.***

When creating any type of HUD or GUI, one needs to keep in mind that it should be consistent so the user will get used to where things are and what the appropriate colours are important on screen. Thus, I created an appropriate GUISkin so as to have this consistency regarding fonts, buttons, text fields and text colours. The backgrounds of the levels are quite realistic, in contrast to the characters which are rather simple and cute. However, comparing to two other Space Invader games, they also mix and match ideas. This can be so as to create a high contrast between the playable characters and static images.

By using the GUISkin, I managed to keep all of my layouts appropriate, neat and consistent from one scene to the other. The font chosen for my game was StarCraft, which was imported into Unity and then attached to the skin so as to be the main font of the game. Also, since most backgrounds were dark, I opted to go for a light green colour as it’s a common colour in various space games and films.

Above and Below: Two screenshots of modern Space Invader games



**Task 15 – D2.1**

***Show that the importance of interdependence has been recognized and achieved by recording at least three different and appropriate sound effects for your game. You must include the following for the three different sounds:***

* ***The original Sound***
* ***Any sound effects applied***
* ***The final sound***

Apart from the spaceship laser sound, I’ve also recorded the following:

* Game Intro Sound
* Alien Laser Sound
* Power-Up Sound

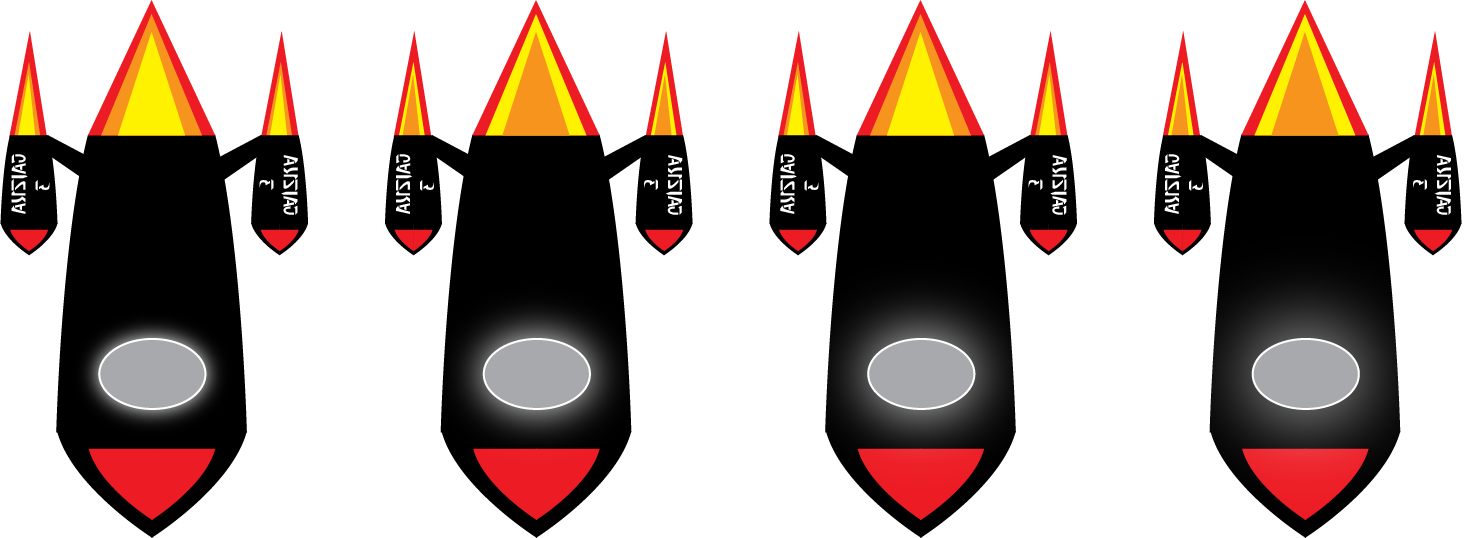
Using more or less the same process as described in ***Task 4*** I managed to create the sounds needed for my game to make it more appealing.

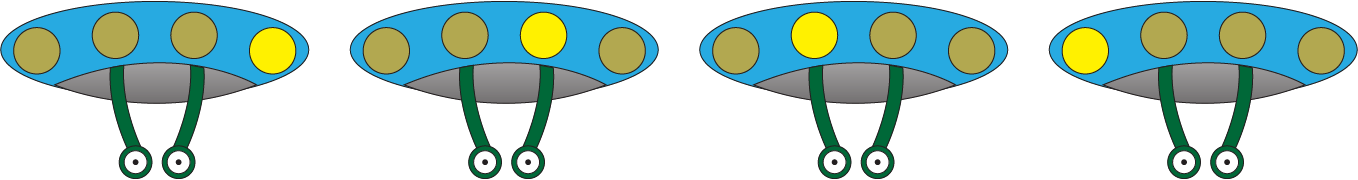
|  |  |  |  |
| --- | --- | --- | --- |
| **Original Sound Name** | **Spaceship\_RAW.wma** | **Laser Sounds\_RAW.wma** | **Spaceship\_RAW.wma** |
| **Effects Applied** | Converted to MP3  Removed Noise  Removed unnecessary length and sounds  Added Reverb and Pitch  Saved as Game Opening\_Edited.mp3 | Converted to MP3  Removed Noise  Saved each laser noise in separate files  Added Echo and Preset Reverb called ‘Lost In Space’ | Converted to MP3  Removed Noise  Saved each boost sound in separate files  Added Echo Preset called ‘Martian Echo’  Added Reverb and Pitch |
| **Final Sound** | Game Opening\_Final.mp3 | Alien Laser\_Final.mp3 | Boosts\_Final.mp3 |

These files can be found in the ***Sound*** folder where there are two sub-folders labeled ***Original*** and ***Edited***

**Task 16 – D3.1**

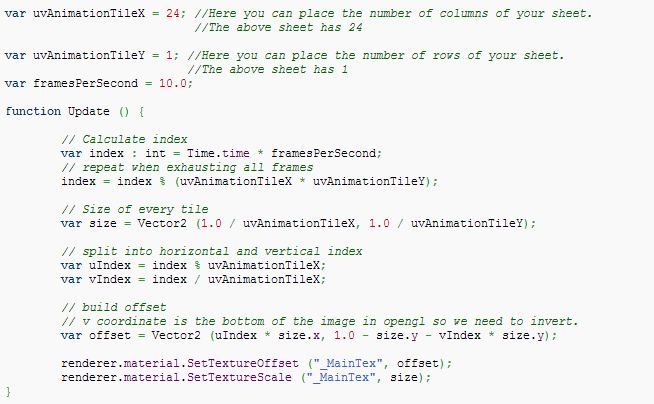
***Demonstrate that effective thinking has taken place in unfamiliar contexts by creating an animated material for your player character and your enemy characters. The animated material must contain at least 4 frames which change in sequence independent of game events.***

******To do this particular tasked, I had to first create image sprites, which is one big image containing various different images. Sprites are useful for the web as well as for creating frame by frame animations in Unity. To create the 4 frame animation, I created the appropriate sprites in Adobe Illustrator as shown below:

******

Above and Below: Two Sprites used for the game so as to create a four frame animation in Unity

After finishing these sprites, I attached them to the appropriate prefabs in Unity so as they become the object’s material. I also had to change the setting of the texture from Diffuse to Unlit/Transparent so as to avoid any unnecessary white backgrounds. However, doing this wasn’t enough as the images remained static. Thus, I had to create a JavaScript named ***AnimationController.js*** where it contains the following script found from the cited source (Ante, 2013)



The basics of the above code is that it reads the user’s input of how many rows and columns there are in the sprites. Then, it’ll read the appropriate frame rate according to the user’s need. After the declaration, it will calculate how big is the sprite and divide each individual image and place them one after in each other on output.

# *Bibliography*

Ante, J., 2013. *Animating Tiled Texture.* [Online]   
Available at: http://wiki.unity3d.com/index.php?title=Animating\_Tiled\_texture  
[Accessed 11 May 2013].

Brauer, D., 2007. *Can I read and write text files using Javascript?.* [Online]   
Available at: http://forum.unity3d.com/threads/5084-Can-I-read-and-write-text-files-using-Javascript  
[Accessed 14 May 2013].

ChromeFXFilms, 2012. *Unity 3d Tutorials - Basics - #9 Importing Fonts.* [Online]   
Available at: http://www.youtube.com/watch?v=JFXrMqLCuqk  
[Accessed 9 May 2013].

Derrtyones, 2012. *Destroy box and show image.* [Online]   
Available at: http://answers.unity3d.com/questions/309428/image-to-appear-on-collision.html  
[Accessed 14 May 2013].

Eric5h5, 2010. *Help me about reading .TXT File.* [Online]   
Available at: http://answers.unity3d.com/questions/12598/help-me-about-reading-txt-file.html  
[Accessed 14 May 2013].

Microsoft, n.d. *System.IO Namespace.* [Online]   
Available at: http://msdn.microsoft.com/en-us/library/system.io.aspx  
[Accessed 14 May 2013].

RamboPanda, 2011. *How to keep Highscores?.* [Online]   
Available at: http://answers.unity3d.com/questions/161147/how-to-keep-highscores.html  
[Accessed 14 May 2013].

Stead, C., 2009. *The 10 Best Game Engines of This Generation. Without guts, there can be no glory: from Anvil to Unreal..* [Online]   
Available at: http://www.ign.com/articles/2009/07/15/the-10-best-game-engines-of-this-generation  
[Accessed 9 May 2013].

Technologies, U., 2012. *Detonator Explosion Framework.* [Online]   
Available at: http://u3d.as/content/unity-technologies/detonator-explosion-framework/1qK  
[Accessed 11 May 2013].

Unity Docs, n.d. *Camera.ScreenToWorldPoint.* [Online]   
Available at: http://docs.unity3d.com/Documentation/ScriptReference/Camera.ScreenToWorldPoint.html  
[Accessed 08 May 2013].

Unity Docs, n.d. *GUI.DrawTexture.* [Online]   
Available at: http://docs.unity3d.com/Documentation/ScriptReference/GUI.DrawTexture.html  
[Accessed 14 May 2013].

Unity Docs, n.d. *GUILayout.BeginArea.* [Online]   
Available at: http://docs.unity3d.com/Documentation/ScriptReference/GUILayout.BeginArea.html  
[Accessed 10 May 2013].

Unity Docs, n.d. *GUILayout.FlexibleSpace.* [Online]   
Available at: http://docs.unity3d.com/Documentation/ScriptReference/GUILayout.FlexibleSpace.html  
[Accessed 10 May 2013].

Ward, J., 2008. *What is a Game Engine?.* [Online]   
Available at: http://www.gamecareerguide.com/features/529/  
[Accessed 9 May 2013].