**Cubed**

This is the documentation for our game that was created called cubed. In this documentation, you will find all you need to know about our game, how it was made and how the code is structured. We all contributed equally to the game and would like the mark to be a group mark.

**Game Summary:**

**Objectives:**

When making our cube based isometric survival game we had three core objectives, these were; the game was easy to play; it got progressively more difficult and that the game had some way to progress the player as the game went on (to help counteract increasing difficulty).

We believe that these objectives have all been achieved. This is because the game only has 4 controls, the directional keys (with being WASD optional allowing for both left and right handed people to play easily) and clicking (to fire bullets) thus keeping input simple. We also added in a control and rules section of the menu for players to view how to play the game, this allows the game to be understood and easy to play for everyone prior to the start of the game, this achieves the first objective.

The game is also designed in a way so that it gets progressively difficult. For example, we have built a framework that allows us to increase the enemies speed and the number of enemies as the waves increase. We can control exactly how the speed increments, where the enemies spawn and how many spawn. This allows us to completely modify the difficulty of the game as it progresses and even test the difficulty making sure that later levels are not impossible and can be completed even if they are extremely hard.

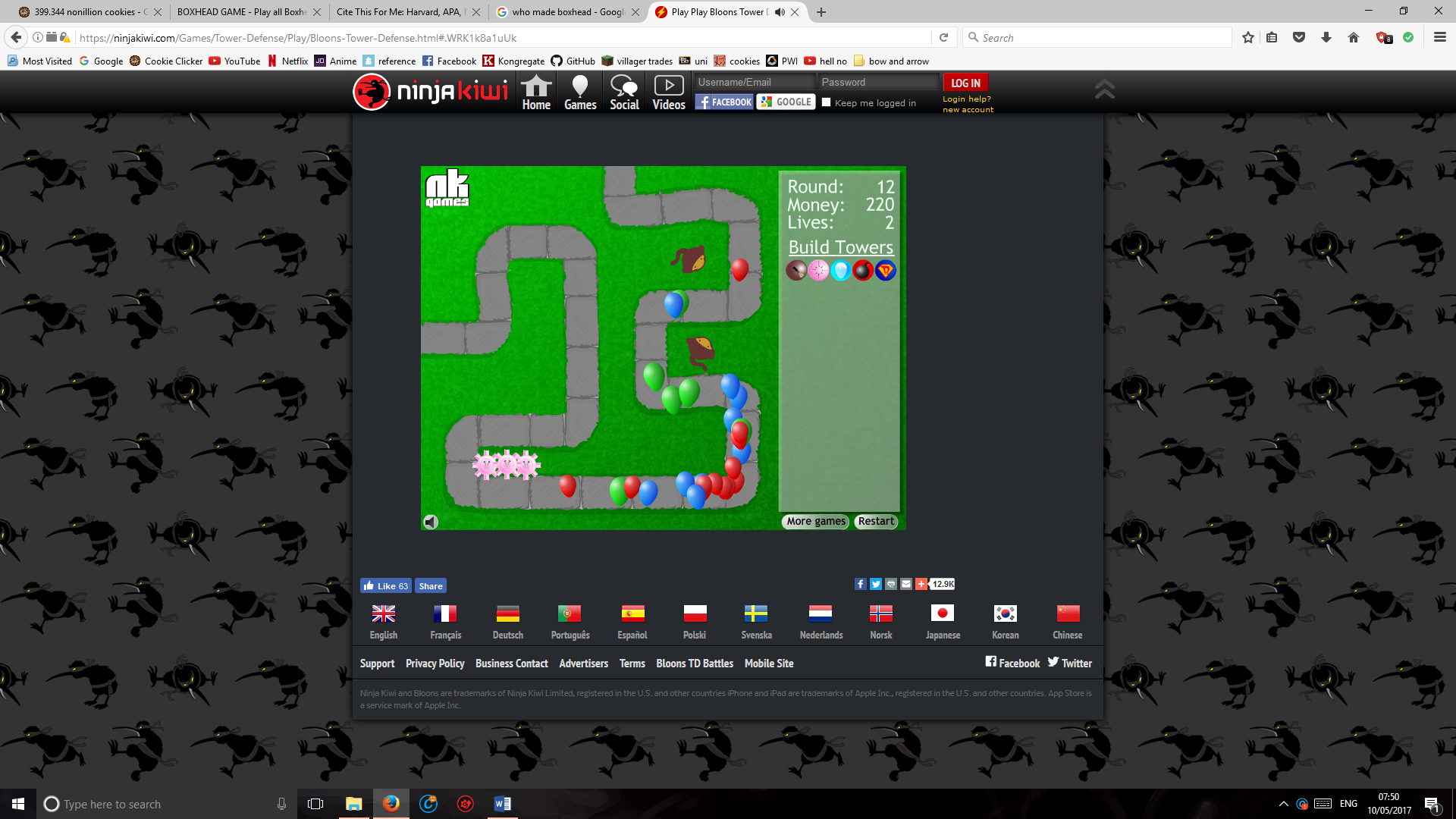
Finally, as the game progresses you can buy more items (power-ups) from the shop making you stronger. Each upgrade is unique and has a special effect on the gameplay, the upgrades include; increasing the bullet speed and player speed; increasing the score and money multipliers; buy a new bullet with unique properties and finally nuke all the enemies on the screen. This achieves our final objective as it adds a form of progression.

**Rules:**

This is Galactos’s last final stand against the enemy darts, the player Galactos must destroy all the enemy darts by shooting them. Darts will continuously spawn regardless if the current waves enemies have been defeated, you will want to make sure you don’t miss (the better you play on the easier waves the less difficult later waves become as enemies don’t build up). Galactos can collect scraps (money) from the darts he kills to upgrade himself to last longer however, these upgrades are capped and once reached can no longer be upgraded. Galactos can move anywhere on the platform and dies once any enemy dart collide with the player.

**Gameplay**

In this game, you can move around with the WASD or the arrow keys (depending on whether on if you are left or right handed) moving the charter around a square isometric grid. Clicking on the screen allows the player to shoot a bullet in that direction. The player collects scraps by killing the enemy darts and can open the shop in the middle of the game and can purchase items to upgrade themselves. As stated earlier in the objectives there are 6 different upgrades, the way these upgrades are used dramatically affects the gameplay. For example, if you are more focused on getting a high-score you might be more inclined to spend your money on maxing the score multiplier upgrade, however if you want to survive to the later (more difficult waves) you may feel that purchasing upgrades that directly affect the player will be more beneficial such as player speed or bullet speed. The players own objectives will affect how the game is played in turn the gameplay.

**Research:**

One of the first games we looked at when planning our game was balloons tower defence. We originally looked at this game for a tower defence game that we wanted to make. However, over time we decided we wanted to change our game to a survival/horde game to make it more original in comparison to all the other tower typed games proposed. Although the research was for a different genre of game, this game had great examples and implementation of progression. For example, the player can get stronger over time by upgrading towers to counteract the increasing difficulty of the game. We also really liked the idea of having a shop with purchasable items that could be upgraded. This game directly affected two of our game objectives with these being easy to play and player progression.

Another game that inspired us was blockhead, this is a wave survival game where the player must fight waves of zombies and slowly unlock new attacks and score more points the further they go. As you can see that there is a score multiplier this inspired us to make our own in our shop allowing your score to grow exponentially. The implementation of this multiplier cause quite a few issues, this is because we could allow players to either purchase this as an upgrade (currently Implemented) or implement a score multiplier that gradually increases as you kill more enemies within a given time frame. For example, if you kill one enemy you then have two seconds to kill another and if this is true then your score multiplier increases, otherwise it simply resets. We decided not to go for this implementation because we thought by allowing players to purchase it would mean the player would have seriously consider each upgrade and its effect on their survivability for current and future waves.

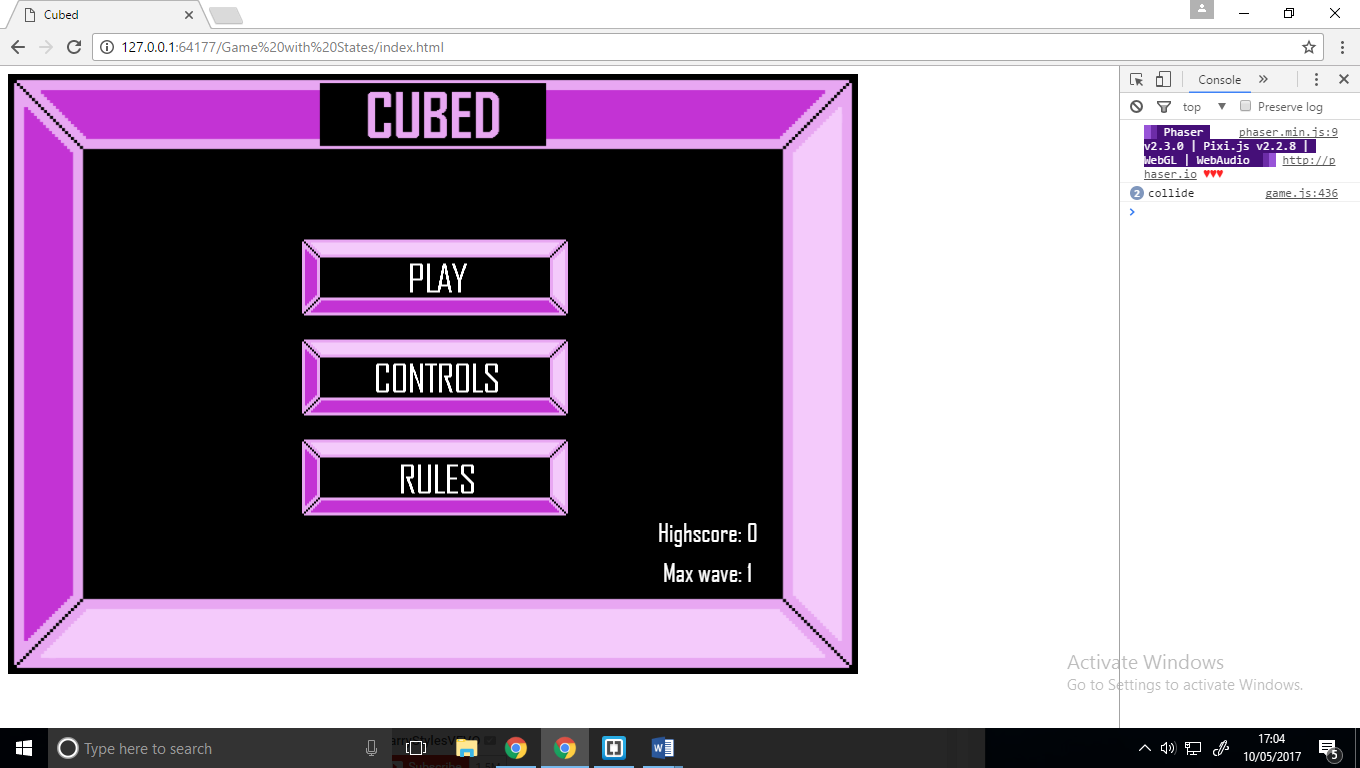


Finally, a game that inspired us was the hordes mode (a minigame) on Gears of War. In this minigame you also must fight waves and buy upgrades as you go along. This game was far less user friendly, meaning you couldn’t just play it and do well, it took you many replays of the game to figure everything out. This emphasised the idea that we can’t go wrong creating a game with easy controls, it allows new players to do well without a massive time investment, people who done well in the Gears of Wars horde minigame were usually veterans of the main game.

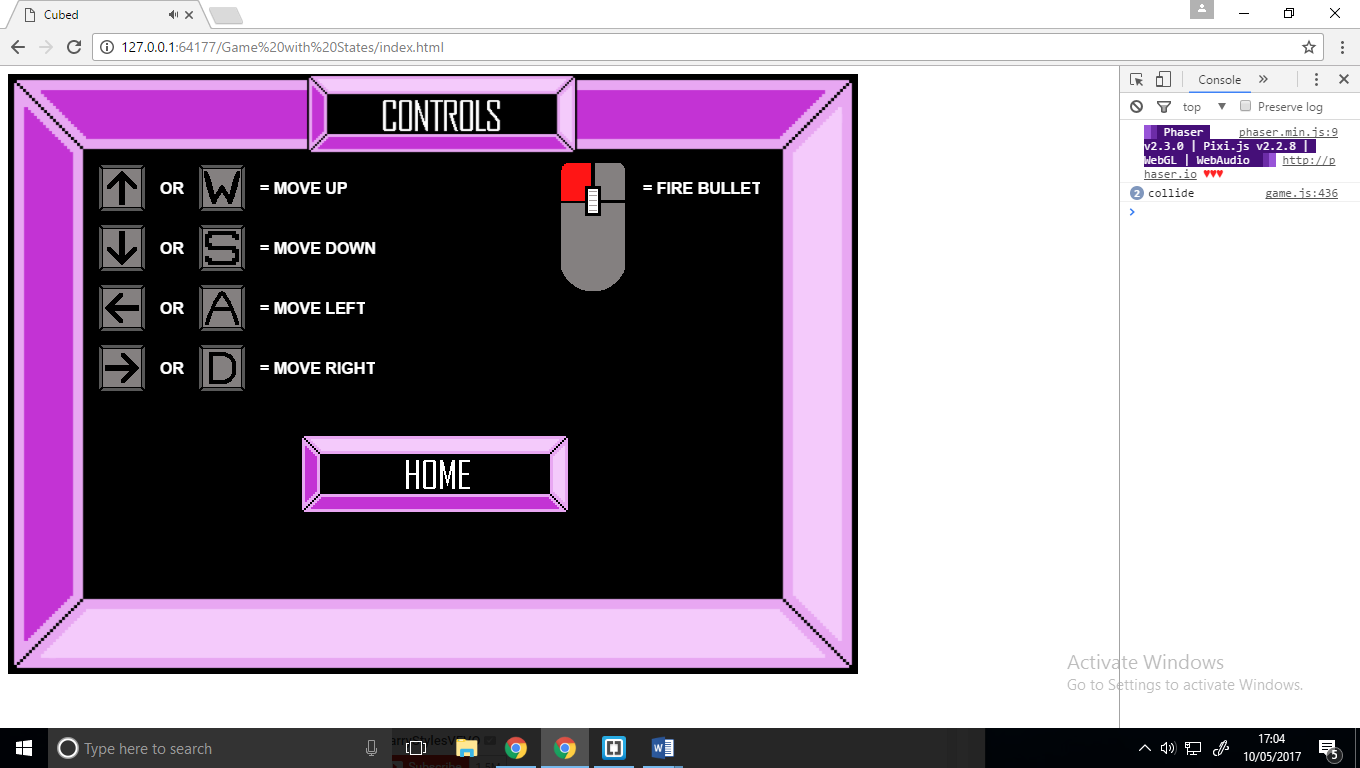
We were inspired by many aspects of each of these games and incorporated key aspects of their design into our game. We decided to keep the isometric view from our original game as we felt it was unique and had good originality, it gave the game a completely unique feel in comparison to a 2d top down game.

**Screen/ Level Map Design:**

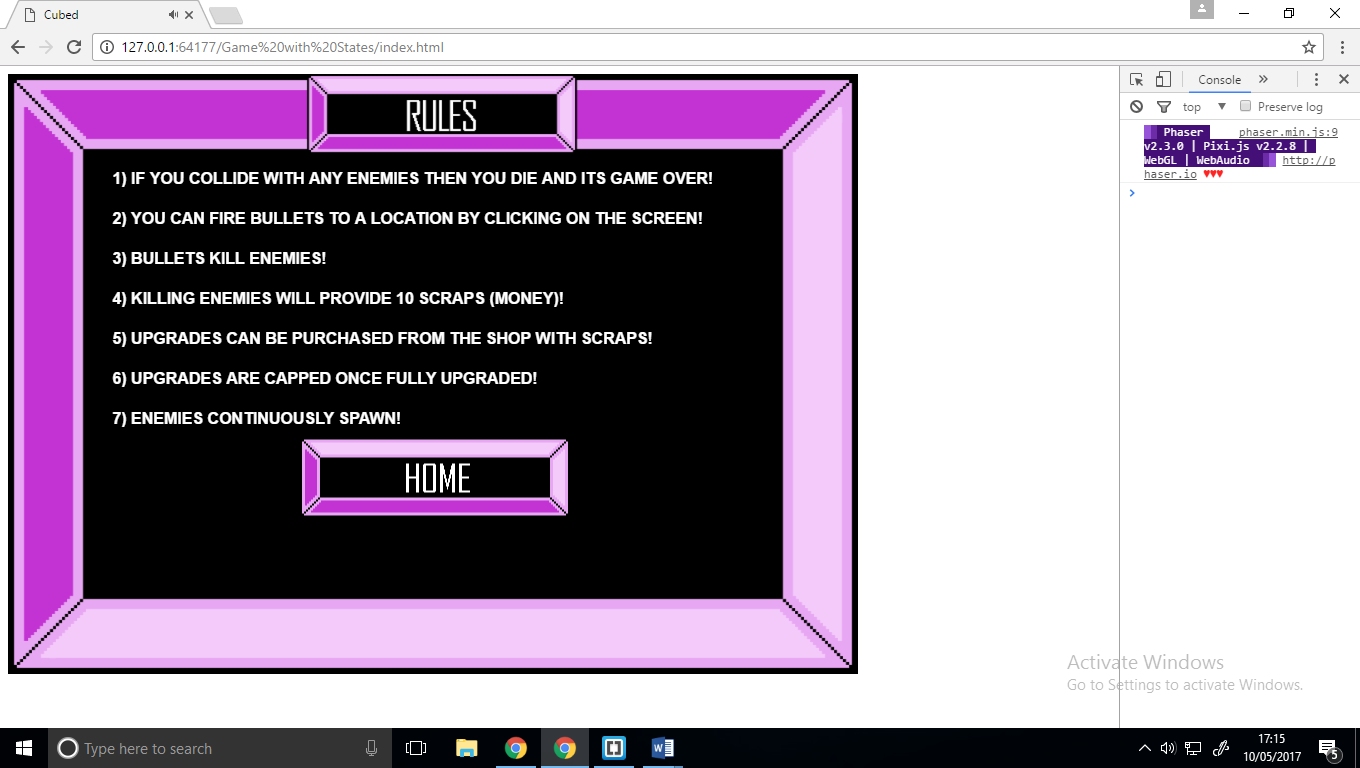
All assets in this game, including both sound and artwork was created by the group and it is all original with no 3rd parties used.



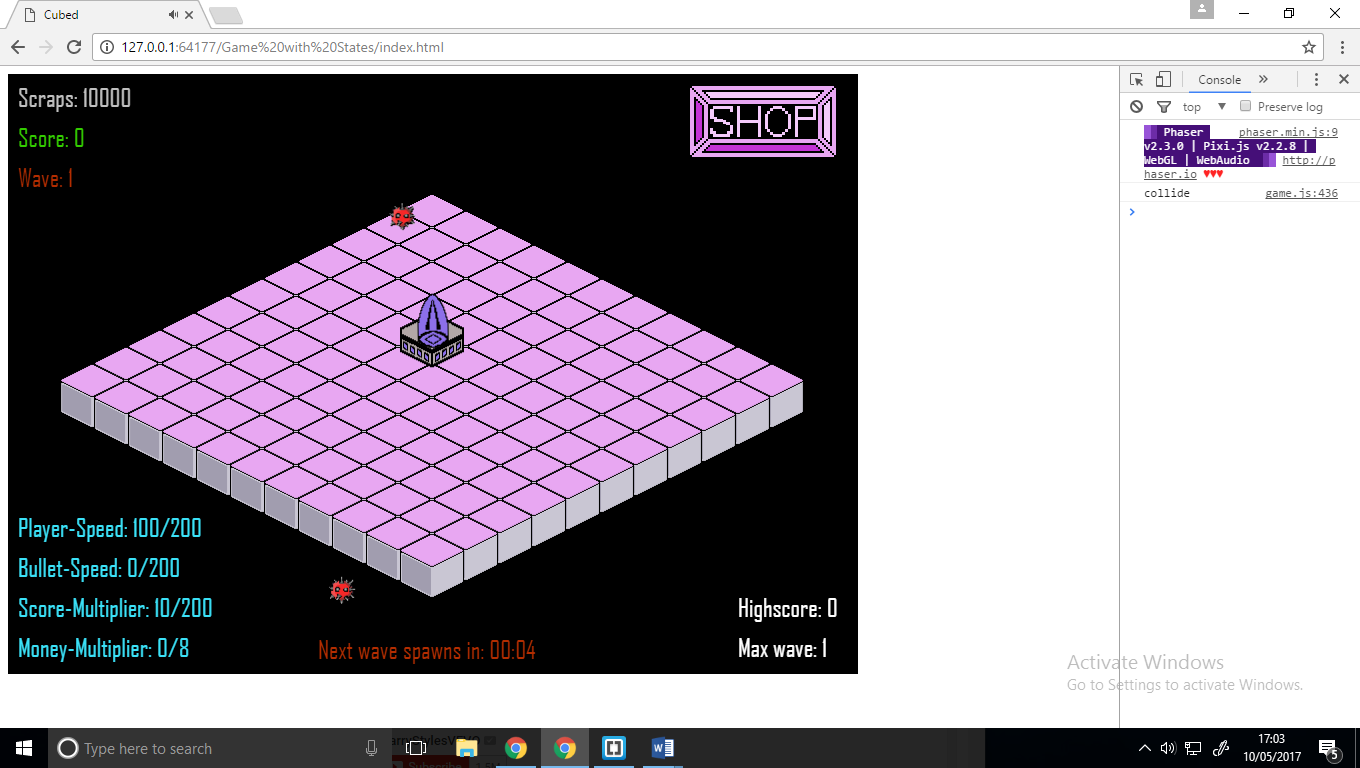
**Home Screen:** This is the home screen that is loaded when the game is opened. It has three interactive buttons that all have individual functionality. These buttons take you to different pages of the game, the controls page, the rules page or start the game. The home page also records your high score and maximum wave achieved.



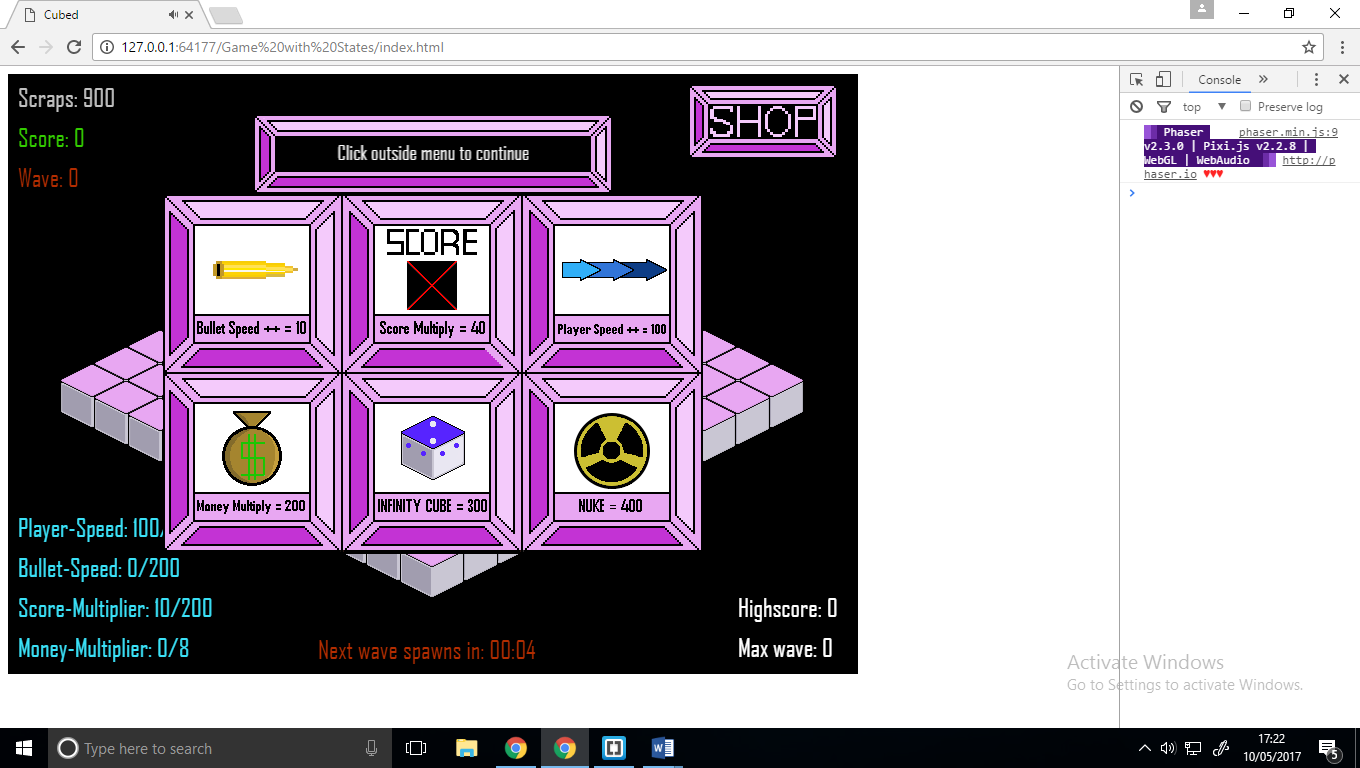
**Controls Screen:** This is the controls screen that is loaded once the controls button is clicked, it displays the controls for the game visually and verbally, it has a home button allowing the user to return home**.**



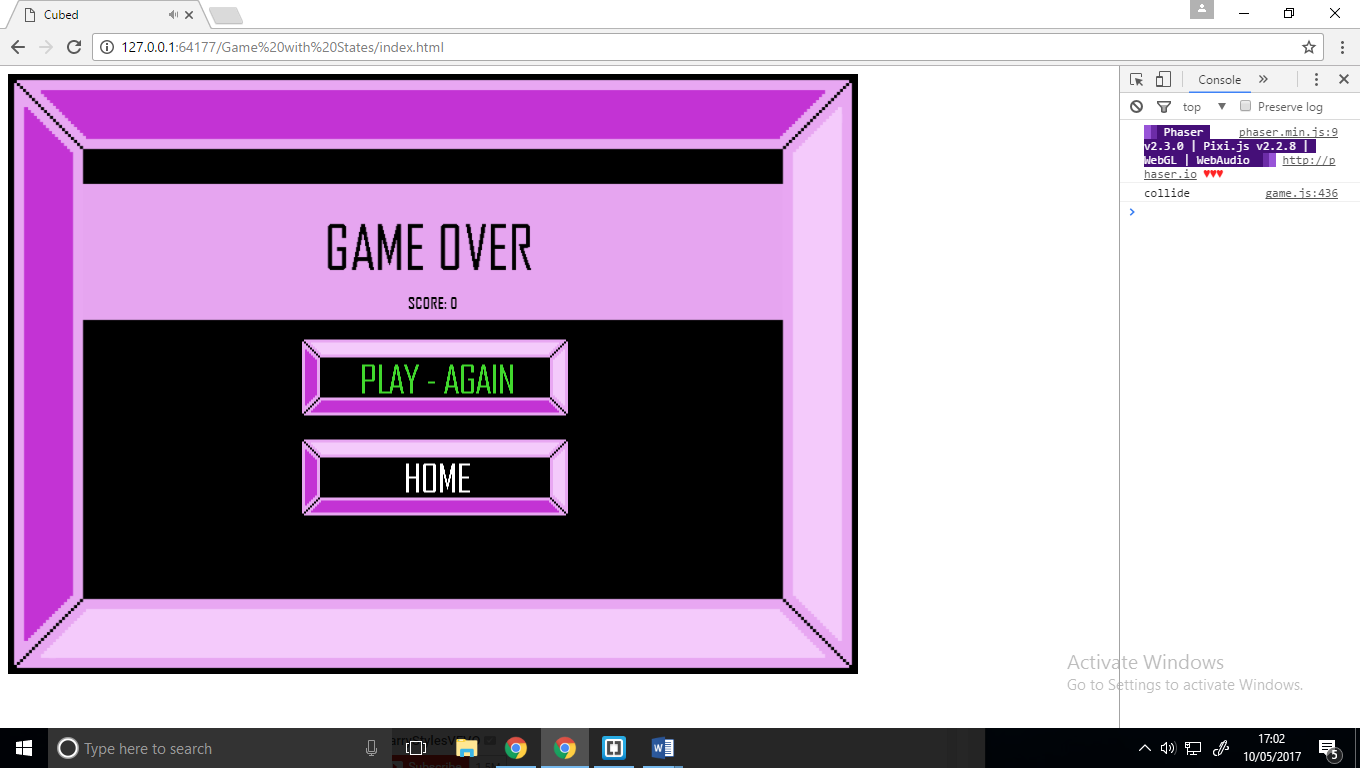
**Rules Screen:** This screen (state) simply explains the rules of the game (which is not many) and then provides a button allowing the user to return home.



**Game Screen:** This is the game, it’s the first thing the player see’s once the game is started. In the top left the important functionality is shown, so the current score, current money and wave. The bottom left of the screen deals with showing the user their current power-ups, in the bottom centre of the screen the wave timer is shown which counts down the next wave spawn. The bottom right states simply the current high score and current highest wave achieved. Finally, there is a button in the top right that gives the player access to the upgrade menu (shop).



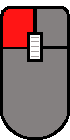
**Shop Screen**: This is the upgrade menu or shop, when opened the game is paused and there are six option (upgrades) for the player to pick from. The bar above the shop prints out responses depending on what is trying to be purchased. For example, “Not enough scraps”, or “Maximum player speed reached”. Clicking outside the shop will close the menu and un-pause the game.



**Game Over Screen:** This is the screen that is used when the player dies, it prints out the current score and allows the player to either play again or return home to view high scores.

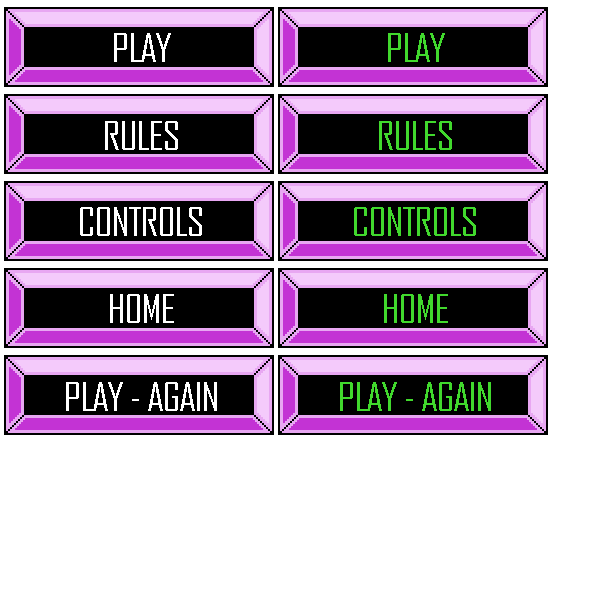
**Other Assets:**

C:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\left.pngC:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\right.pngC:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\down.pngC:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\up.png

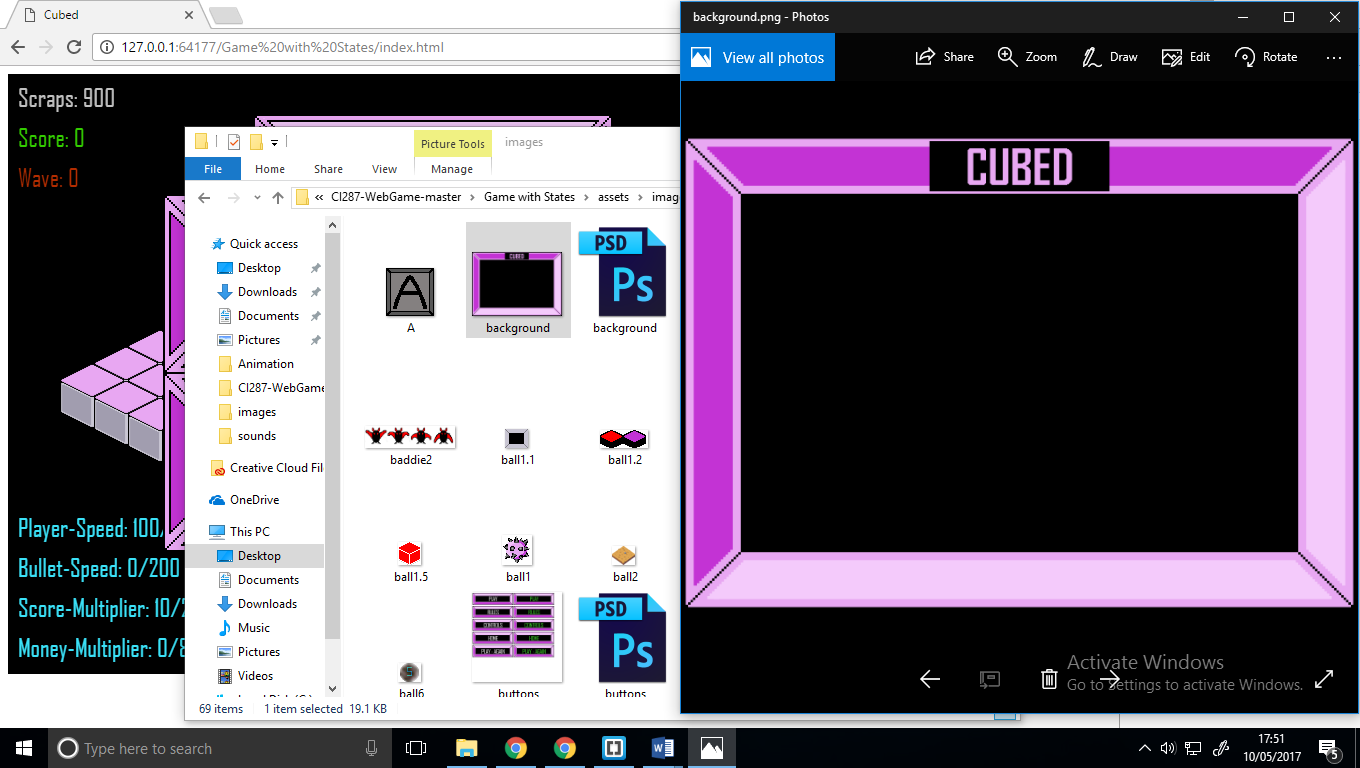
**Arrow keys:** These keys were used on the controls screen to provide a visual for how the player moves.

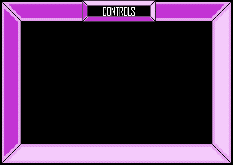
C:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\W.PNGC:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\A.PNGC:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s.pngC:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\D.PNG**WASD keys:** These keys were used on the controls screen to provide a visual for how the player moves.

**Mouse:** This mouse was used on the controls s screen to provide a visual for how the player shoots.



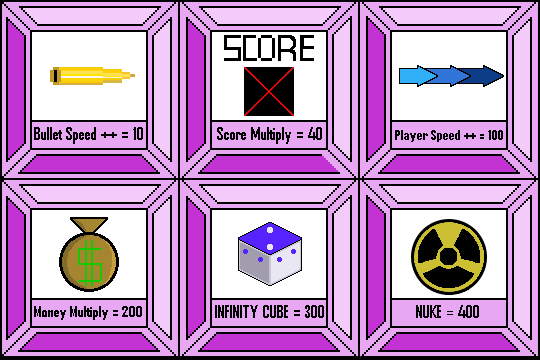
**Buttons Spritesheet:** We created a spritesheet for the button to provide interactivity for the user. These buttons are used in almost all the game screens.

**Home Screen background:** This is background image used for the main menu.

**Control Screen background:** This is background image used for the control menu.



**Rule Screen background:** This is background image used for the rules menu.

****

**Shop Display:** this is the shop display its used for the player to see what upgrades to buy and how much they cost. Above the shop image is the sprite used for printing out the shop result when items are purchased.



**Shop icon**: this is the button the player clicks to access the above shop.



**Floor sprites**: these are two sprites that we created to make up the floor (base) of the isometric board.

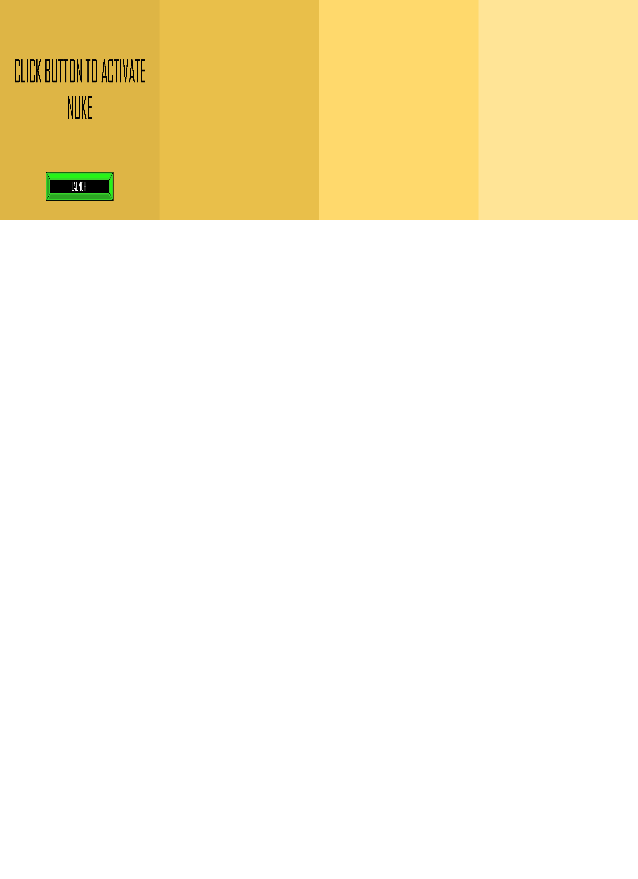
**Enemy Sprite**: This is the sprite we created for our enemies, it was more complex than a regular circle and came out well when playing the game.

C:\Users\samuel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\turret1.2.png

**Player sprites**: these are the three sprites that we created to make up the player. It is broken into 3 parts, the player turret, the players core (heart) and the player base



**Bullet sprites**: these are the three sprites that we created as our player’s bullets.



**Nuke spritesheet**: this is the spritesheet that is called when the player buys a nuke from the shop.

**Why the change?**

When creating our initial design, we planned on making an isometric tower defense game. However, when we did our initial presentation we saw that multiple other groups had also decided to create tower defense games. Because of this, we decided to take our base concept and redesign it to a horde mode game, keeping only the isometric view and shop concepts. This challenged us to be more original.

**Functions**

**The Preload function**: In this function, we load in all the sprites that we are going to use in the game, we also load in spritesheet’s that we want to use for animation and music.

**The Create function**: In this function, we create all the sprites by calling the assets in the preload function, these include the player, enemies, bullets, heart, animations, sounds, text, cursor inputs and pause functionality. The whole use of this function is to create all our necessary sprites, animations and music that is updated constantly in the update function

**The Update function:** is what we use to handle all game updates. This function contains many other functions that are regularly called in the game. For example, it deals with collisions, movement and AI.

**Balls.forEach function:** This function is called when the enemies spawn, it deals with the enemy movement the enemy speed and how many enemies are spawned.

**DestroyBalls function:** This function is used for checking if the bullets and balls (enemies/darts) collide. It also updates the player’s money, score and deals with playing the sounds that deal with the collision. If the bullet and enemy collides it destroys both unless the player has purchased the infinity cube power-up.

**Killplayer function:** This function deals with the collision between the player and enemies. If they collide both are killed and the music stops, it then starts the game-over state and calls the updatehighscore function.

**SpawnTiles function**: This is used for drawing the base tilemap.

**Render function:** This is used to monitor how long till the next wave spawns and test to see if it the enemies have spawned.

**Reset timer function:** Will reset the games timer once the timer reaches zero. It also increases the waves, creates more enemies and updates the ‘max wave achieved’ variable.

**FormatTime function**: This function is used to calculate a minute and a second. It is called in the render function as each wave spawns after 5 seconds.

**Fire function:** Handles creation of bullets when the player fires a bullet, it then checks to see if the bullet is dead (destroyed) if true is resets the bullets position. It also it used to play the bullet sound.

**Updatehighscore function**: updates the high score if a new one has been reached. This is called in the killplayer function.

**Unpause function:** Handles the shop in the game and will deal with purchasing new upgrades and power-ups, this function makes calls to changeWeapon5. This function allows to purchase items in the shop and it changes the sprites mid game to use the newly brought power-ups. The shop constantly updates and check global variables, informs the player with what have and have not purchased, then decrements the player’s scraps (money).

**File Structure:**

**Assets**

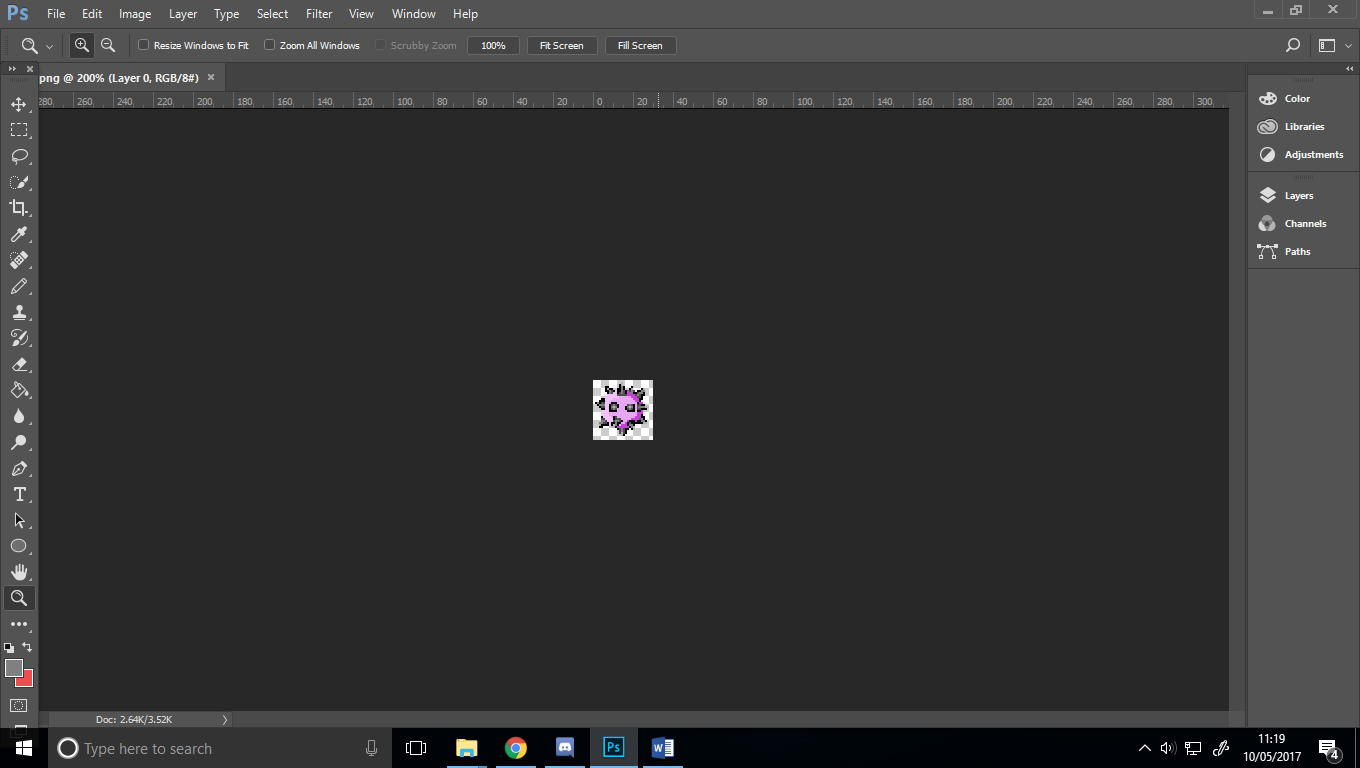
**Images All Images files(player.png)**

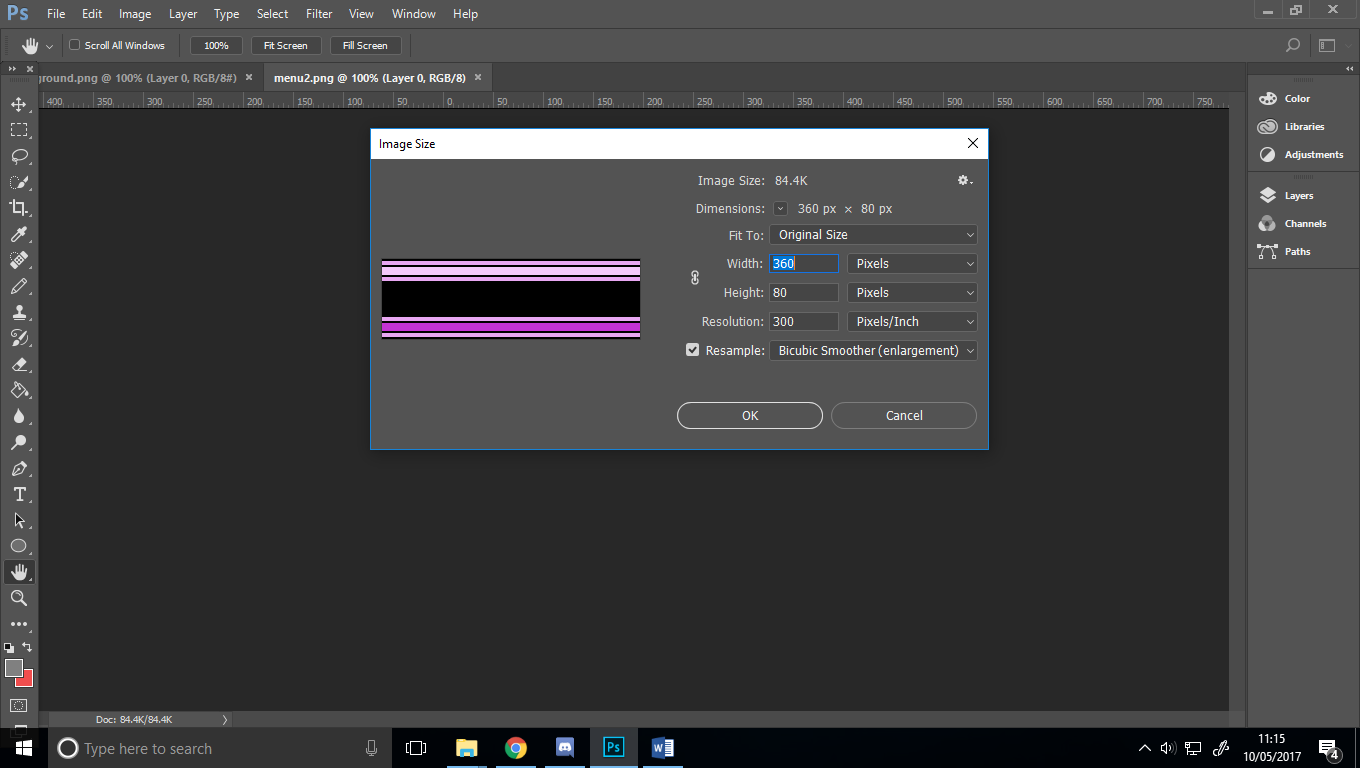
**JS All JS files (game.js)**

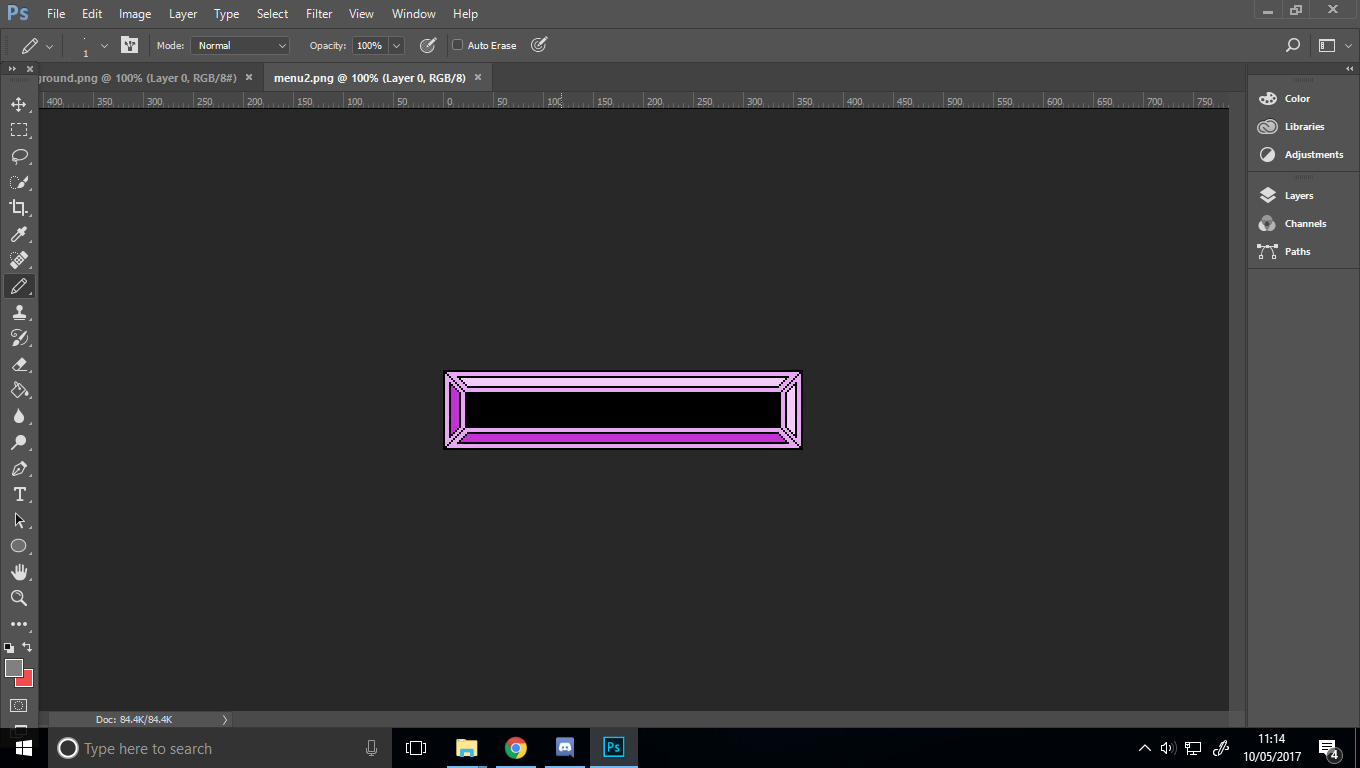
**Sounds All Sound files (bullet.mp3)**

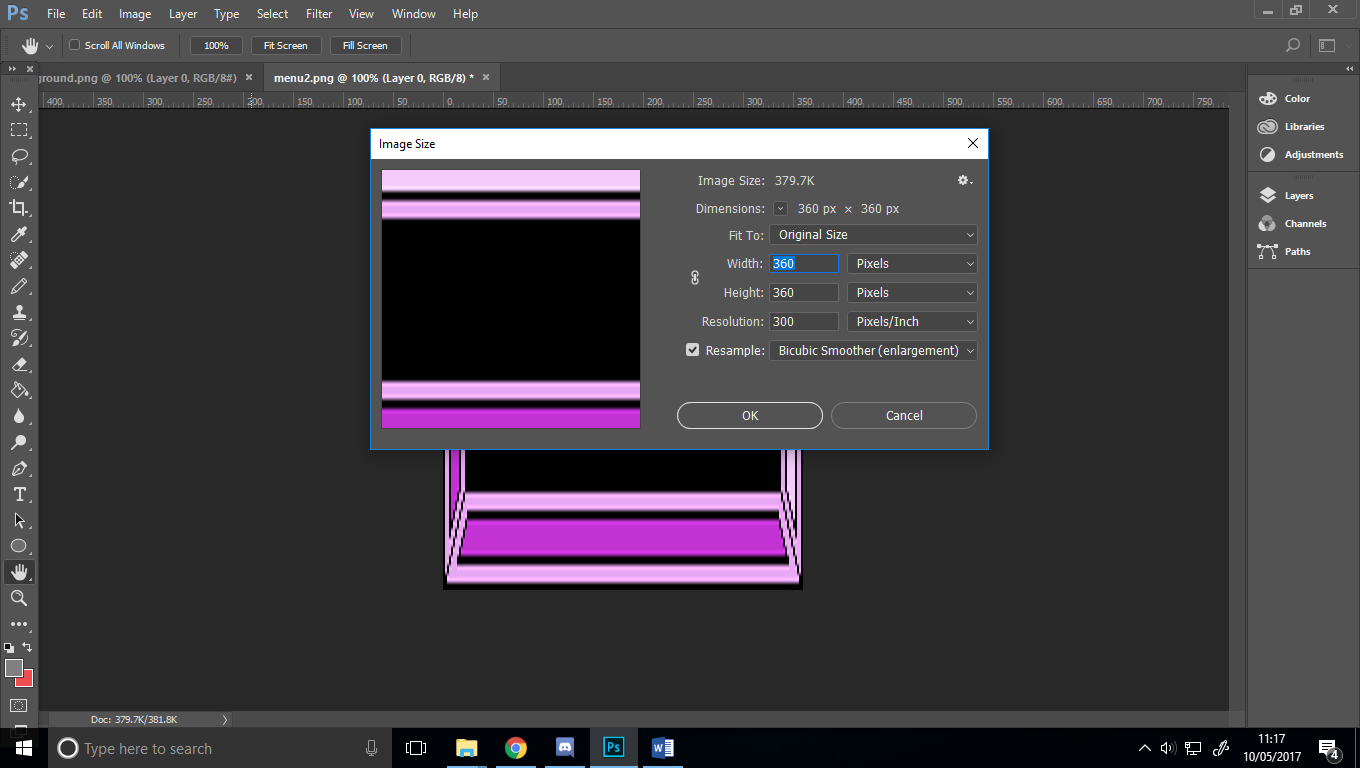
**Index.html**

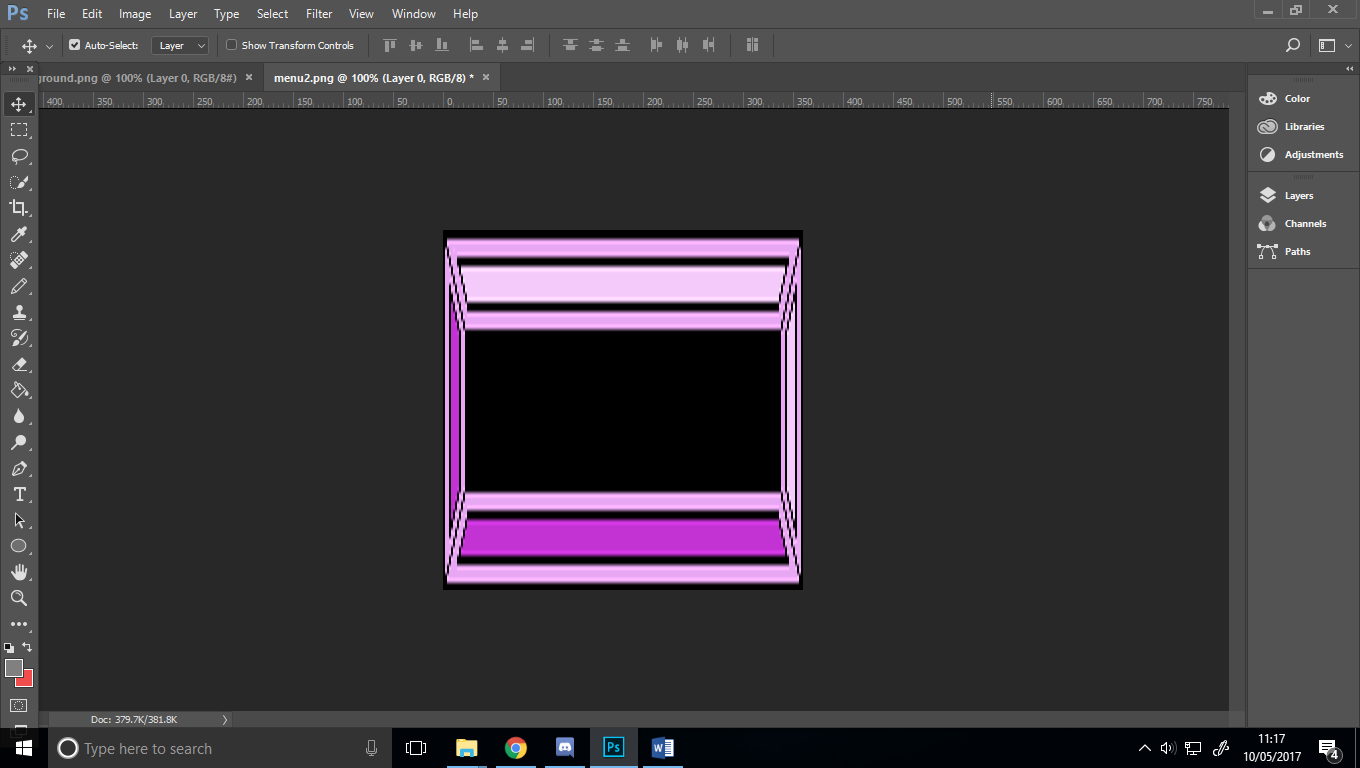
**Assets:**

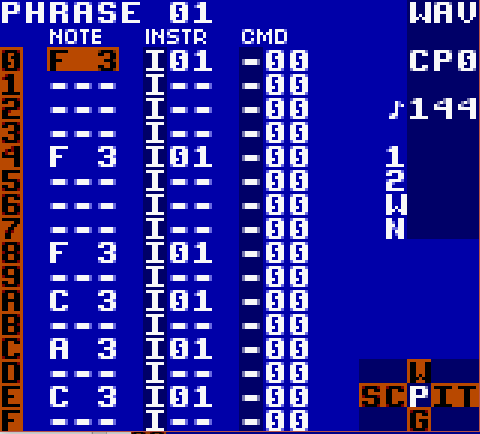
As mentioned earlier all the Sprite assets were created by the group, these were all made in photoshop. The reason we picked photoshop is because it allowed us to created sprites down to 1 pixel accuracy. Although some of our sprites look similar for example the menu screen background and the buttons all assets that are not the same size had to be redrawn.



For example, this asset to the left is the sprite used in the shop menu, left of it are the dimensions of the sprite, 360 pixels by 80 pixels.



Here is the same sprite however we have now set the height to 360 pixels and as you can see the sprite is more pixelated and it does not look right for how an image of this size following the same style of our game. Creating the sprite was a long process but a necessary one



**Music Assets:**

To create the music and sounds for a game we used a variety of methods. For example, for the unused purchase sound we added a handful off coins into a cola can and shook it in the right way to make a metallic shake, then pitch shifted it in post to create a metallic change sound.

For the other sounds, we used a program called Little Sound DJ (http://www.littlesounddj.com/lsd/). This is a game boy ROM hack which hacks the game boy sound card and enables you to use it as a MIDI controller. The program gives you a lot of control over the composition you create, enabling you to adjust the wave type, time signature and much more.

The main screen of LSDJ shows the four types of audio waves you can use to create your sounds. The first two, PU1 and PU2, are pulses, and are your main melodic waves, used primarily to create music. The third column, WAV, is your wave file, this is the most manipulatable, and can be changed to anything from a bass line to a drum kit to your regular bullet firing sounds and thuds. The final column, NOI, is your white noise, and can be most easily described as the static that you can hear on an old TV. We used this column to add extra texture to the bullet sound and create the explosion sound for the enemy’s death. On the right side of the screen is the beats per minute of the project, this can be edited in the project menu.

The other four major menus in LSDJ are the chain, phrase, instrument and table screens, and can be easily navigated through the bottom right of the screen. Each phrase is linked to a chain and the instruments can be edited through the instrument tab

In the phrase menu, you can input your notation. In standard 4/4 notation each number represents a 16th note, and each note can have a different instrument. Finally, the instrument and wave tabs enable you to edit the sound that you are using in the phrase, everything from the speaker the sound will come out of to the wave length to the Attack, Sustain, Decay and Release can be edited here. This meant we could keep our music and sounds to the 8-bit blocky theme that we wanted to portray in the game.

**Implementation Evaluation:**

Strengths:

We have multiple strengths to our design. A big feature that we wanted to achieve was that we used all our own assets. As shown in the pages above, we used Photoshop to create all our own sprites and a mixture of home recording techniques and Little Sound DJ to create our sounds. We think that this has helped us achieve the view of what we wanted our game to feel like.

Secondly, we created states for each of the scenes in our game to keep our code clean and to be able to search through our code easier. We took this concept throughout our code, and neatly and efficiently laid our code out with different functions for each process, and grouping the variables into similar themed groups, for example listing the audio as a group of variables, and all score related variables as a group.

Finally, we have created a system in which the user can buy power-ups and be rewarded, this system changes game variables such as fire rate and speed of the player and generates a replay ability factor to the game, along with rewarding a player for getting further each time.

Weaknesses:

One of our biggest weaknesses is that we were only able to have one type of enemy with the same health and sprite variables. One way to fix this would be to create an array with different enemy types and let the program select which enemies from the array it would spawn. However, the easiest way to do this would have been to declare all the enemy properties in a JSON file. Unfortunately, JSON is not supported with the isometric Phaser plugin so this could not be done.

Secondly, the enemies spawn randomly, and because of this there is a chance that the enemies will spawn right on top of the player, causing an instant death. A way we could have fixed this would be to create a function which considers the player position and set the enemies to spawn in a random position plus the variable of the player’s position.

Finally, when the game is paused we are unable to play animations or sounds. This is because when the game is paused all functions are also paused. Because of this, we were unable to use the sound we made for purchasing upgrades.

# Bibliography

Ang, C. (2017). *Rules, gameplay, and narratives in video games*. [online] researchgate. Available at: https://www.researchgate.net/publication/236870305\_Rules\_gameplay\_and\_narratives\_in\_video\_games [Accessed 9 May 2017].

Hernandez, P. (2017). *Gears Of War 4's Horde Mode Is A Blast*. [online] Kotaku.com.au. Available at: https://www.kotaku.com.au/2016/10/gears-of-war-4s-horde-mode-is-a-blast/ [Accessed 10 May 2017].

Cooper, S. (2017). *BOXHEAD GAME - Play all Boxhead Games, The Rooms, etc.*. [online] Boxhead-game.com. Available at: http://www.boxhead-game.com/ [Accessed 10 May 2017].

Ninjakiwi. (2017). *Bloons Tower Defense*. [online] Available at: https://ninjakiwi.com/Games/Tower-Defense/Play/Bloons-Tower-Defense.html [Accessed 10 May 2017].

Danny Markov. (2015). *Making Your First HTML5 Game With Phaser.* Available: http://tutorialzine.com/2015/06/making-your-first-html5-game-with-phaser/. Last accessed 09th Apr 2017.

xTYILIEIR. (2012). *BTD5 Bloons Tower Defense 5: All Tower Upgrades.* Available: https://www.youtube.com/watch?v=I2tpZE2I6BI. Last accessed 10th May 2017

N/A. (2011). *Gears of War 3 Horde mode footage and new multiplayer map screens.* Available: https://vegivo.wordpress.com/2011/06/08/gears-of-war-3-horde-mode-footage-and-new-multiplayer-map-screens/. Last accessed 10th May 2017.

**REFERENCES DIRECTLY LINKED TO CODE:**

Phaser. (N/A). *Tanks.* Available: https://phaser.io/examples/v2/games/tanks. Last accessed 05th Mar 2017.

Phaser. (N/A). *Accelerate To Pointer.* Available: https://phaser.io/examples/v2/arcade-physics/accelerate-to-pointer. Last accessed 05th Mar 2017.

Rotates. (N/A). *Pointer interaction.* Available: http://rotates.org/phaser/iso/examples/interaction.htm. Last accessed 05th Mar 2017.

Phaser. (N/A). *Pause Menu.* Available: https://phaser.io/examples/v2/misc/pause-menu. Last accessed 15th Apr 2017.