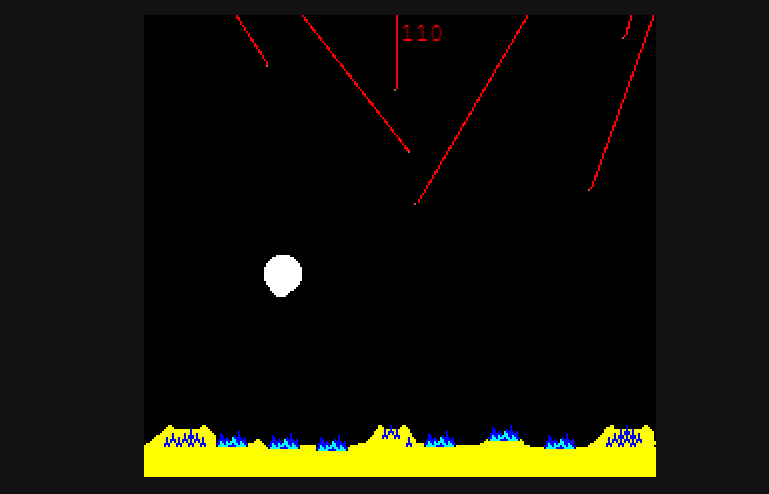
**Year 4 Game Design and Development Assignment 1**

**Kalum O’Toole X00121797**

<https://github.com/kalumot/GameDevCa1>

**Part 1**



The game I chose to focus on for this assignment is Missile Command, originally released in 1980 and developed by Atari. This is a single player, arcade shoot’em’up. The core idea of this game is that you are being attacked by missiles and must use your own missiles to defend against these to protect the cities on the ground.

This is a 2D game with very low-resolution graphics, as is expected for the era, being 256x231 pixels. Missiles are represented by a single pixel but are still visible as the resolution is so low to begin with. The graphics are nothing amazing, but it was 1980 and there wasn’t much ability to make improvements here at the time. There are also some basic sounds for explosions and such but no music.

The enemy missiles shoot from the sky and are randomly aimed at your 6 cities or 3 bases. If they hit one of these, they are destroyed. If you lose all 6 cities, your game is over. Your bases are where your own missiles are shot from. Each base has 10 missiles loaded. If your base is destroyed, you lose all remaining missiles stored in this base.

To defend against these enemy missiles, you must shoot your own. This is done by placing your cursor in a location and clicking/pressing. This was originally done with a joystick but works even better now with a mouse and left click. When clicked a missile will shoot from the closest base that has remaining missiles. Unlike most other games with shooting features, you’re not aiming directly for the enemies. The rocket explodes where you chose to shoot it and this explosion is what you want to hit the enemy with. Upon impact between an enemy and an explosion it will kill the enemy missile as well as creating another explosion which can create a chain reaction and let you destroy multiple enemies with a single shot. This creates an interesting gameplay mechanic as you must shoot in from of the missiles approaching and calculate how long t will take your missile to get there and how long it will take theirs. The explosions themselves add to this dynamic too as they grow, shrink and then disappear all in roughly 3 seconds so you must take this into consideration too. Each time you shoot a missile you lose it and you have limited ammo starting at 30 missiles.

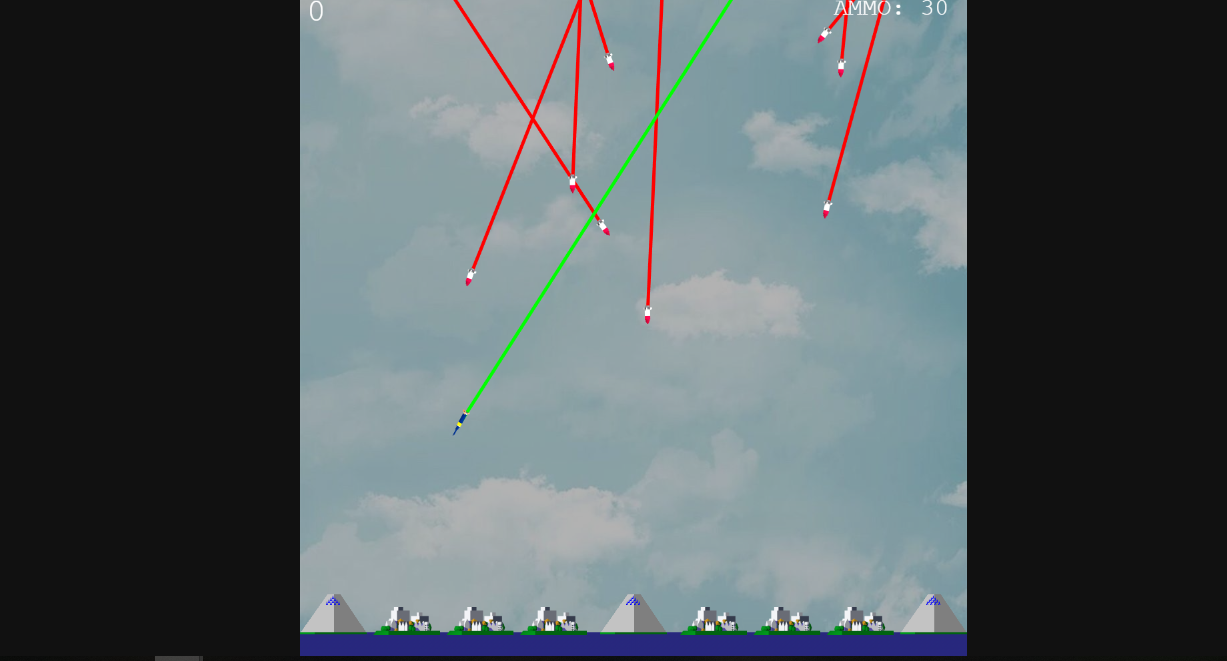
The main goal of this game is survival. Its to get as far as you can before all 6 cities are destroyed. The game uses waves. Each wave the rockets get faster and there’s more of them. At the start of each wave all your missiles and bases are restored but your cities remain missing. Points are also awarded for shooting missiles and surviving with ammo to spare and cities.

Like the majority games in the 80s there is no permanent progress in this game. This is due to it being in arcades so different people would be playing each time. This means no progressive story or saving or unlocks.

There are also no respawns in this game. If you lose your cities, its game over and you must start from the beginning again. This was very common for arcade games back then. Nowadays it’s referred to as permadeath and is a mechanic that has been making a resurgence in the last few years, especially with indie games. This can be seen in games such as Binding of Isaac and Dead Cells. It’s a good way to add replayability without having to add a ton more content as users can play the same content again and again trying to get further than the last time. When using this though there must be some randomness as otherwise it would get monotonous. This is done today with such things as procedurally generated level structures and in Missile Command it is done to an extent with the random nature of the attacking missiles.

By todays standards the game obviously isn’t amazing but for a game from the 80’s it’s very fun and contains lots of ideas that would inspire later great games.

**Part 2**



If this game was released today the first thing everyone would notice would obviously be the graphics. With some games now supporting 4k, a game which is 256x232 will look prehistoric. Some games nowadays still use low resolutions going for a pixel art style. The difference however is in the hardware. In the 80’s the resolutions had to be small but also the developers had to be clever with how they made the games as to not waste a ton of memory as it was very limited in these games. This mean that there couldn’t be a ton of textures saved so different looking game objects were limited. An example of this is in Mario where the bushes and clouds are actually the same texture, just coloured differently.

The audio is also very basic compared to modern day games. Mainly just in the quality. Most games nowadays will also have a soundtrack though too. Even small indie ones. Missile Command does not.

Another major difference between this game and modern-day games is the lack of depth. Its very straightforward and doesn’t offer much variety. There’s no powerups or variety in levels. There’s no unlockables or story. This is very common for classic arcade games but today there tends to be a lot more depth to even small mobile games.

Framerates have obviously increased over the years. The original missile command runs at a fairly slow framerate. Most games now will meet the industry standard of 60fps if the machine they’re running on is up to date. The occasional games will cap at 30fps but these are big 3D games with high quality textures. Missile command today would run on pretty much any machine without ever dipping below 60fps.

With the assignment only being to create one level I decided to not put a limit on the level so that the player can keep going as in missile command the levels are over fairly quick and the score would be nearly pointless if its only representing a few seconds of survival.

When it came to the improved game then I took the single level survival as the concept aand improved on that.

**Change 1:**

When making the improved version of the game the first thing I thought to do was increase the resolution. This simple change would give me much more room for other improvements. This can be done in phaser by simply changing the config height and width. I changed it to 1000x1000 which more or less maintains the aspect ratio as I feel this plays a huge part in the gameplay. Most games now will have a widescreen rather than square or a mobile shape. These fit the screens better but it would completely change the gameplay so I kept it square. Many indie games still use the square screen though such as the Binding of Isaac. Originally when I made the first version I had pixel-art set to true in the config as the resolution as so small and I didn’t want the textures to be smoothed. With the increase in resolution I removed this as smoothing the hard pixel lines looks much nicer when its more zoomed out.

**Change 2:**

With the new resolution it opened the option for adding better textures. I found a few free ones online at <https://itch.io/jam/missile-command-throwback-/topic/178005/free-2d-and-3d-game-assets-for-missile-command-game-jam>. I used a few of these and edited some slightly as well as making some of my own to put together the new set for the game. In phaser I just had to call these images now instead. I then had to readjust the game to fit the new textures. This included increasing sizes, moving objects, and because of the new resolution I also had to increase the speeds that objects move to be proportional.

**Change 3:**

I felt the game was lacking some “juice” or “game feel”. I didn’t feel like I was getting much feedback outside of the sound effects. A common method used in games to give the user some feedback is screen shake. This can give the same feedback as an explosion sound effect or something similar and with both combined it makes for a very fun engaging experience. I added this in my game for when your cities are blown up. I tried adding it for when your rockets explode but with the amount of enemies your trying to keep track of on screen having screen shake this often just makes things too hard to follow. I was able to do this in phaser by using the shake() method in the collider for the enemy missiles and cities and bases.

**Change 4:**

I wanted to add more depth to the game. With it being only a single level for this assignment there was some problems. In the original game after each level your ammo is restored. With this being endless if you got good eventually youd reach a point where youre able to survive each time until your ammo runs out and then the game is over. I decided to add a new fun mechanic to the game to combat this problem. I came up with the idea to have a different kind of enemy missile that’s way more rare than the normal ones but if you can destroy it your ammo and bases will be restored as if your starting a new level. I also made the new missile faster than the others as to make it slightly harder to get than the others and to throw off the players timing ass you get good at timing your explosions for the enemy missiles to hit after a while. With some testing I found this added a great new dynamic to the game and often found myself being low on ammo trying to be selective with my shots in hope that an ammo missile would shoot down soon or making split second decisions on whether I should save one of the cities or hit the ammo missile.

I implemented it by creating a similar object to the other missiles with some differences such as the texture, the graphics for the trail and the speed. I made the colliders with explosions similar for both enemy missiles except with the ammo missile I added code to remove all bases and ammo that remain and then recreate all in their original full state.

For the normal enemy missiles there’s a 1/80 chance one will spawn each frame. For the ammo missile I made it 1/1200. This makes it 15 times more rare and with some testing I found this was a good number to settle on.

**Change 5:**

I decided to add some music to the game as it felt like it was missing that. Especially with the new graphics. I found a site that had royalty free music: <https://www.dl-sounds.com/royalty-free/category/game-film/video-game/>. I wanted one that had an arcade feel to it to match the game and also fast paced enough to add to the intense nature of the game when trying to move quickly. I found a piece call Defence Line which I felt fit perfectly and just also happened to have a fitting name as a bonus. I added this in as an audio object and just called it to play as soon as the game began in the create function. I also lowered the volume of it as it was too loud using (.volume = .5). I felt this tied the game up nicely.

**Change 6**

Finally, I made a small change in adding the ammo count to the top right corner. I felt it was important to add this to the UI now with the increased reliance on the ammo and it being hard to count the small displays for ammo on each base. Its good to know when you need to be selective with your shots to wait for ammo. I feel like this would also be a big improvement in the original game too for the same reason of it being difficult to tell how much you have left just from looking at the small blue sprites on the bases. I implemented this in the update function by adding a text object and using setText() to display the variable which is the count of the children of the group of ammo objects.

Along with these 6 changes I also made lots of small tweaks like changing speeds, size of explosions, placement of score, etc. When I play both versions of the game I definitely have a lot more fun and feel more engaged with the second so feel like I made some good decisions in the changes.

Both Versions of the game are in the GitHub Repo linked below:

Both are uploaded with node backed so if you run “npm start” from either project directory it should open the game on port 8080.

<https://github.com/kalumot/GameDevCa1>