Zombie Rampage!

You awake in a sleepy town with no one around, and not a sound can be heard from any of the many houses that surround you. Suddenly you see a figure appear off in the distance. You approach this figure and start to realize that he is walking toward you. Then you see that he is disfigured and looks like a walking, decaying corpse. You assume he is hostile and pull out a gun. As you fire, you notice that more of them are coming from all directions. You realize that perhaps you will never survive this horde of hostile undead, but nonetheless you begin your rampage!

**Obstacles**

Image Loading:

Though the path that was suggested for loading multiple images into the game was to create a single image file with several images placed next to each other in a square, I chose to pursue a method that allowed for much more efficient way to dynamically add new textures or remove old ones. This obstacle was probably one of the toughest I encountered since the area was not discussed in classed since there were more important things to cover, so I took it upon myself to search the internet for ways to implement a WebGL program that successfully loads and uses multiple image files. After analyzing the code Dr. Mihail provided for loading a single image file, I was able to search each method in the documentation and discover what each piece did. Furthermore I discovered that WebGL utilizes callback functions for image loading, since images are not loaded synchronously but rather the process of loading the image is only started before the program continues to run other commands. So, in order to load multiple images, I had to implement a solution that loads all images before binding each of them to texture objects and performing the necessary operations on them, after which I would call a callback which should be the render method (or another method that has a callback to render).

Using textures in Chrome:

Google Chrome does not allow for loaded pages (or their resources) to load images from a source that isn’t a server. So to combat this issue, rather than downloading Firefox (since I like Chrome better, especially for its error messages when debugging), I wrote a simple server in Node.js that delivers the necessary image files for my WebGL program.

Making bullets:

In order to have bullets, I needed to not only store the bullet coordinates, but also the bullet direction vectors. I also needed to make sure that when moving and rotating I also updated both the bullet coordinates and their direction vectors, as well as moving the bullets in the direction of their direction vectors each frame. One key detail that took a while for me to figure out was that only the bullets’ coordinates need to be translated, whereas the bullets’ direction vectors only need to be rotated.