CTRLZ

COS 426 Final Project

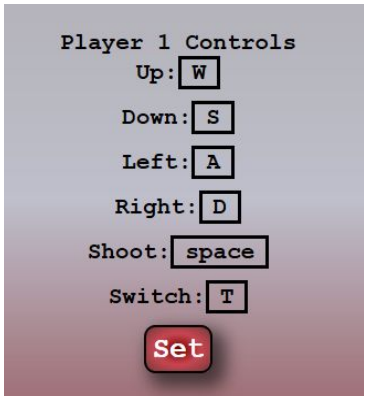
Quinn Donohue (qdonohue), Cody Sedillo(csedillo), Victor Riveros (vriveros)

Find our project here.

Introduction:

The goal of our final project is to build a system incorporating one or more ideas that have been learned in class. We decided to create a game called CTRL-Z where two players compete against each other in a closed arena. CTRL-Z features a base building system where players can place different types of blocks as part of their defensive strategy. Some blocks are simply static, but others actively fire projectiles and contribute to the combat. The fort building happens over time - while a player's first several blocks of any type are placed down immediately, the remaining blocks will appear over time at a constant rate. If a player is hit by a projectile, they get sent back in time to a previous position they occupied relative to the damage amount. Visually, these positions are indicated by footprints that are left behind on the floor. If a player is sent back in time before a block was placed, the block will be hidden. In order to win, a player can either fully destroy the opponent's base or instead send them back in time far enough that they have turned to the starting platform. We hope that the players will enjoy developing a strategy that successfully utilizes this mechanic - for example, by shooting where the player *was* rather than where they are prior to being hit by a structure's projectile.

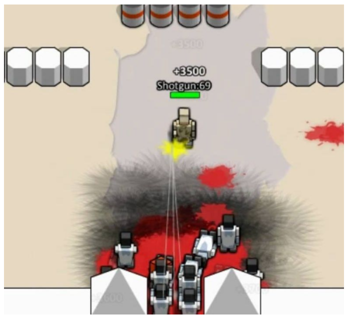
Our minimum deliverables were to make a two player game where players can build their own forts, destroy the other player's forts, and have a time travel mechanic in lieu of player damage. Beyond this, we wanted to add in multiple block types, different weapon types, and better graphics. One of the features we're most proud of is the option for the players to remap their keys at the start of the game to a control scheme they are more comfortable with if needed.


Player 1 Controls
Up: W
Down: S
Left: A
Right: D
Shoot: space
Switch: T
Set

League of Legends: Ekko

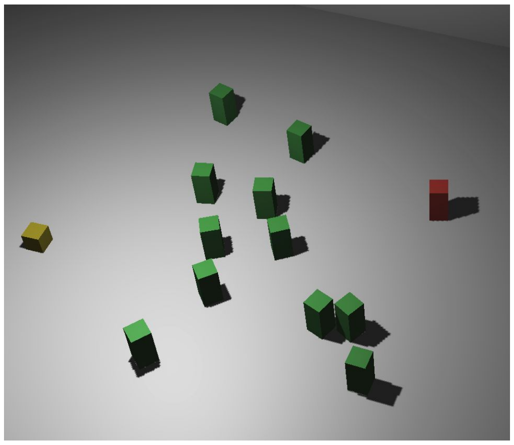
Ekko, from League of Legends, was one of our inspirations for the time travel mechanics. Ekko is constantly followed by a "ghost" that reflects his prior position - his ultimate allows him to travel back to this prior location. While we ended up taking a different direction with time travel (we had time represent health), we discussed the character during our planning phases.

BoxHead:



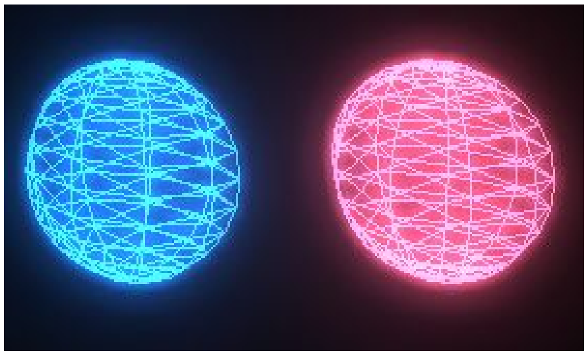
One of the inspirations for our work was the game Boxhead - which is an isometric two player shooter. While we chose a different perspective, Boxhead's boxy look inspired the eventual look of our game.

Red Palito:



Red Palito is another zombie survival game, similar to BoxHead, but it is built using Three.js. We used it better understand how to handle basic aspects of our design including how to handle key input and how to setup the camera to follow the player's movement. No assets were used directly, but it helped us overcome the initial hurdle of getting something, anything at all, to display on the screen at the very beginning of the project. Our vision is quite different, but there are a few similarities to our current game product.

ColoRing:



This project written by a fellow COS 426 group saved us when it came to making split screen. The example on Three.js wasn't immediately helpful, but by looking at how they implemented it we were able to understand how to make things work (we changed things to a horizontal split screen and swapped around some other factors - but the general strategy we learned from them).

Our Approach:

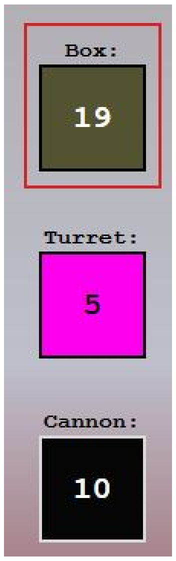
We decided to implement our game in Three.Js off a basic webpage. That way we can run a localhost instance of our index.html via the python -m http.server command. For board building, we used a grid made in html to represent possible block locations, and then fed that data into our Three.js initialization. We felt this approach would best allow us to leverage our existing skills, while still being flexible enough for us to make something novel and new. We also utilized jQuery to make DOM manipulation on our webpage easier.

Methodology:

For our methodology, we will be giving an overview of some of the most distinct elements of our game. The core gameplay loop is similar to most Three.js games, and things like hitbox detection are pretty common too since the geometry has built-in bounding boxes.

Fort Building:

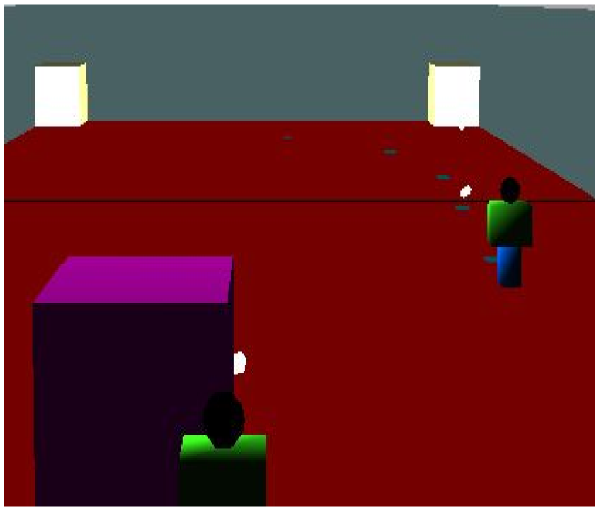
We let players build their forts before the game starts by selecting locations in html grid. We had considered allowing players to build in real time, but given the limited number of keys available when dividing the keyboard amongst two players, we were concerned about making adequate keybindings. We were also inexperienced with trying to manipulate the scene and track where blocks should be. Alternatively, we considered letting players build their forts from an overhead view into the Three.js scene - but looking at documentation, it seemed like it would be very challenging to track where mouse clicks happened. So instead we decided on using the html approach. Surprisingly, one of the biggest challenges in building the game was tracking block order placement during the fort building (in particular allowing for removals). We toyed around with a linked list, but ended up settling on a series of arrays.

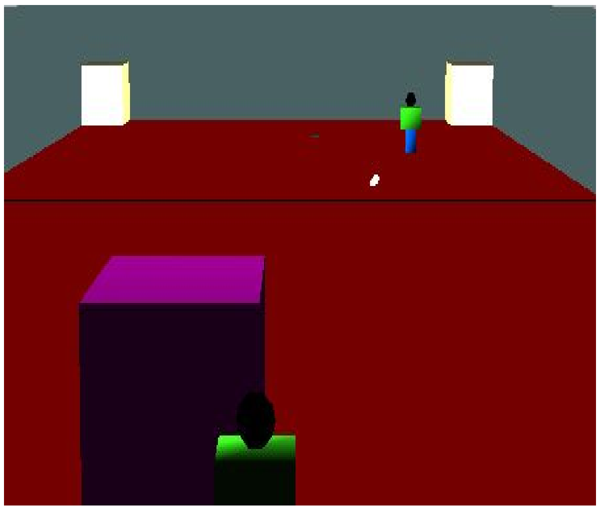


Remaining blocks to place

Time Management:

In a game such as ours, where we are sending players backwards through time, time management is a very real issue. We needed a system that would let us keep track of a player's prior position, send a player back to a previous position, keep track of which blocks of a player should be placed - and when those blocks need to return to a prior state. We managed this by having our player object store an array of all of their blocks and an array of positions they've been to. Every second, we had our "temporal manager" tell every player to update their position array. Players would then add a new position, and increment their internal clocks. Then, we could check whether or not block placement was “ahead” or “behind” by comparing the index in the block array we were at (which when multiplied by time between block placements should correspond to elapsed time) with the actual elapsed time. If we were behind, we would place blocks to catch up - if ahead (which happened when sending a player back in time), we would reset our index and hide “future” blocks. Given more time, we would’ve enhanced our array of past positions to include the direction we were facing as well - so that you would be where you were before and facing the direction you were facing.


Position before being hit


Position after being hit

Results:

We were successful in accomplishing all of our core goals and we made significant progress towards our vision. We added in a turret block type that targets the enemy player as well as a cannon block type which always shoots in a line towards the enemy base. In addition, we were able to add in different weapon types to the game; currently you can switch switch between an Uzi and a Pistol. We didn’t get a chance to add in AI opponents, and our core game is still pretty ugly in a lot of ways. As of this writing there is no sound, no animations, and very basic meshes for each object. We also still have some persistent bugs. For example, player 2’s camera position is set in a different perspective relative to player 1’s even though we created them both using the same process!

But all in all, we would call our progress as a success. We built a game that was more complex than we thought we would be able to, and when playing games against ourselves and friends we discovered a surprising amount of strategic nuance and range of tactics. Players would purposefully sacrifice themselves to protect their cannons (which is a risky move because it hurts them - but cannons are incredibly valuable to take down the opponent’s fort). Placing turrets and cannons later in the building queue caught opponents off guard, and there were some fascinating and close games.

We were able to demo our game on Piazza for feedback, and look forward to sharing the game with the class as a whole after we continue to make improvements and add polish.

Discussion:

Our approach worked well for us - given more time and familiarity with Three.js, we’d probably like to move away from static fort building to a more dynamic game, but given our experience with game-building we’re happy with what we ended up with. Three.js was able to render the scene as we’d like, and the WebGL renderer seemed to have no problem handling the number of meshes and various computations we had in our scene.

Follow up work should be to beautify the game - we could use a unique model for the cannon and turret’s to make them more visually interesting, and both need a damage model. There is a concern that different meshes for turrets and cannons will affect the hitbox and require increased complexity to remain accurate, so we held off on the change for them. Adding in a greater variety of weapons and weapon pickups would make the game more interesting and add more tactical options, and adding in sounds would help make the game more engrossing and provide instant feedback to the player.

Over the course of this project, we learned a tremendous amount about how to actually build games. We went from being terrified at the prospect of building a game, to confident we could make one as a personal project. We gained a lot of familiarity with building scenes in Three.js and handling the main renderer ourselves as opposed to building around the renderer loop like we did in other projects. Indeed, for future iterations of the class it might be nice to have a project where we build the core rendering loop ourselves. In addition, we became much more confident in writing JS code to interface with html, and writing JS code that dynamically generates html.

Conclusion:

Overall, we were successful in building the game we wanted, and had a fantastic experience doing so. We learned a lot, and while there are plenty of areas of polish left, we are proud of what were able to made in such a short timeframe. Before taking this class we never thought we could actually make something like this, and it feels incredible to watch an idea develop from the concept stage to something tangible that people can play and interact with. We hope you enjoy playing our game, and please feel free to leave constructive feedback so that we can continue to improve it!