Holomon

Final Document

Team: May1737
Client: Iowa State University/Jim Lathrop
Adviser: Jim Lathrop
Team Members/Roles
Ryan Krause/Team Leader
Tyler Thompson/Webmaster
Robert Young/Communication Leader
Zachary Koehn/Concept Holder
Team Email: may1737@iastate.edu
Team Website: http://may1737.sd.ece.iastate.edu/

Revised: 04-22-17

1 Introduction

1.1 Project statement

We want to create a location-based mixed reality game for the Microsoft Hololens, where the player explores their environment looking for holographic monsters. This project takes inspiration from Pokemon Go, but we plan to make it a deeper and more robust gaming experience.

1.2 purpose

This project is exploring a brand new technology that very few people have worked with. We will be contributing to the ongoing development of the Hololens as a whole, as not much has been done with this new technology. Much of the current work our peers have done has not been documented. Our work will be adding to the relatively small zeitgeist of Hololens development. Finally, many of us want to go into the video game industry, and making fully-fledged games is the most important part of entering into the industry.

1.3 Goals

We would like to gain experience designing and developing video games while also exploring the possibilities of the Hololens. By the end of the project, we aim to have a building-scale game developed for the Hololens. This game will include as much of the Hololens functionality as it can, including displaying holograms, hologram interaction, voice recognition for battling with your monsters, and mixed-reality videos and trailers. We will have a variety of monsters to collect and train as well as a way to battle with them, both against AI and with other players using a Hololens.

2 Deliverables

Gameplay

- Holomon
    - Ten different Holomon to collect
        - Battle wild Holomon to weaken them before collecting them
    - Various levels, stats, and attack moves for each Holomon
        - Attack, Defense, Speed, Experience, Level, etc.
    - Can be found by searching around the building
- Battling
    - Initiate battles with wild Holomon, AI trainers, and other Holomon players
    - One-on-one, battles occur in a turn-based manner
    - Each Holomon has various strengths of moves, each with their own purpose depending on the situation
    - Experience system for leveling up Holomon
    - Buff and debuff items
        - Potions, speed-ups, increased defence, etc
- Items
    - Earned at kiosks located around the building
    - Usable in battle

Technological

- Control battles using Hololens hand gestures or voice commands
- Building-scale exploration with Holomon appearing in different places throughout the building
- Multiple Hololenses that can communicate for multiplayer Holomon battles
- Image recognition to determine where kiosks can be found

User Interface/User Experience

- The player may interact with the menus using their voice or using gestures
- Users should not feel nauseous or otherwise uncomfortable while playing our game
- The menus shall be “tagalongs” (a Hololens term for a type of menu that stays in front of the user) so as not to disorient the player

Documentation

- Tips & tricks that we’ve found during development that can help the Hololens community grow to make better games.

3 Design

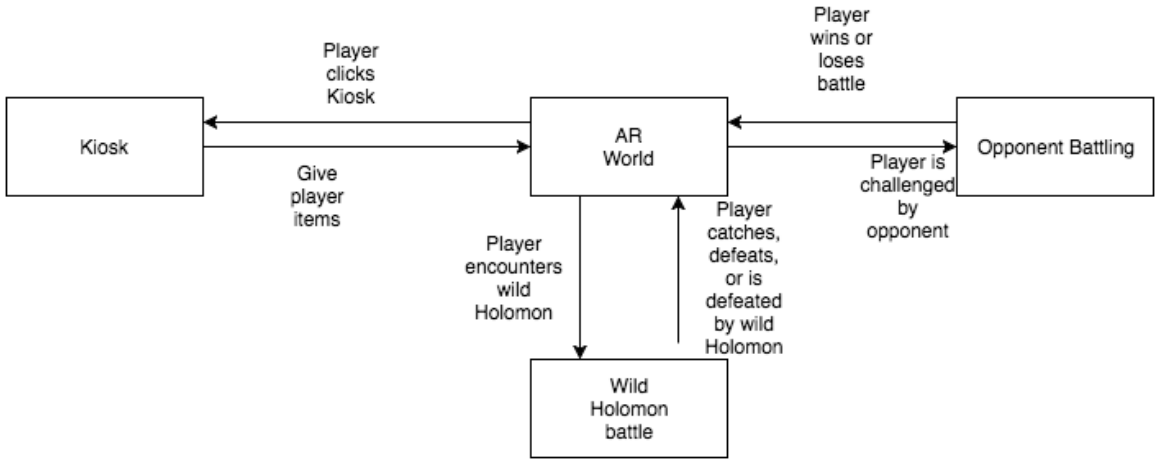
3.1 Previous work/literature

This project is very similar to Pokémon Go. The player explores the area to find different Holomon, in the same way Pokemon Go does. Pokemon Go also have a radar system that we are looking to build off of and improve. We've played Pokemon Go and we've seen/played the battle mode that they have, which we would also like to improve.

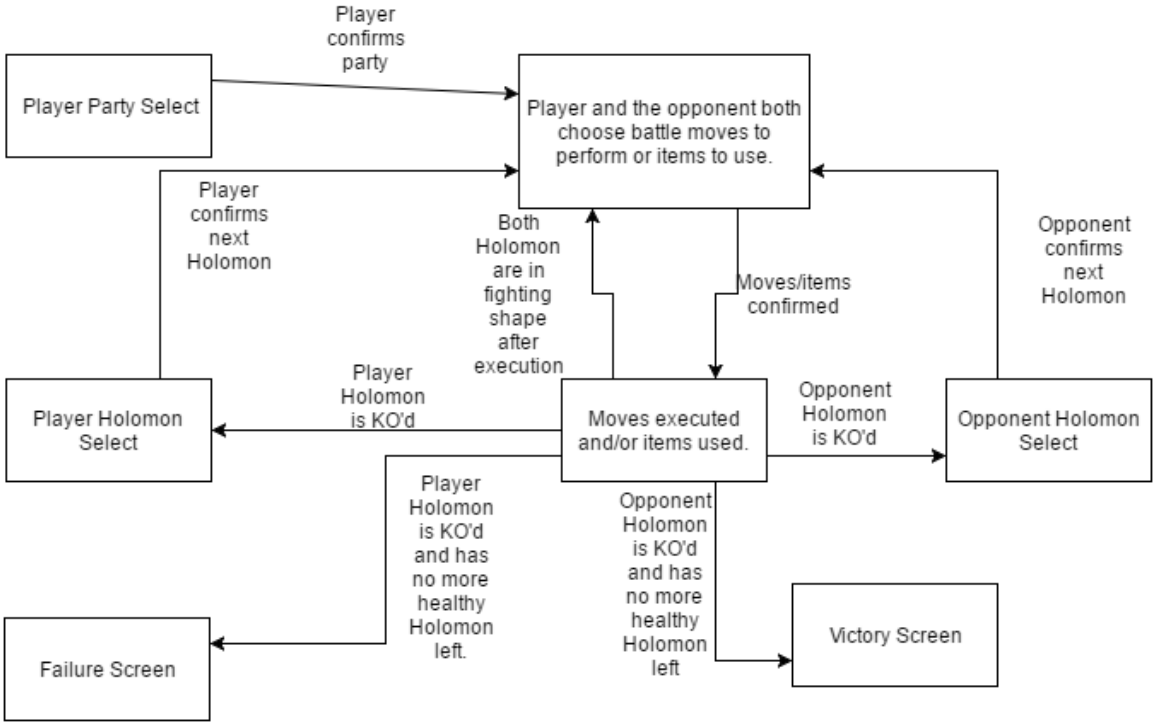
We're using Microsoft's Hololens Academy to get us up-to-speed on Hololens programming basics. There's also a lot of projects on the Hololens that we can play and understand how the general feel is supposed to be.

3.2 Proposed System Block diagram

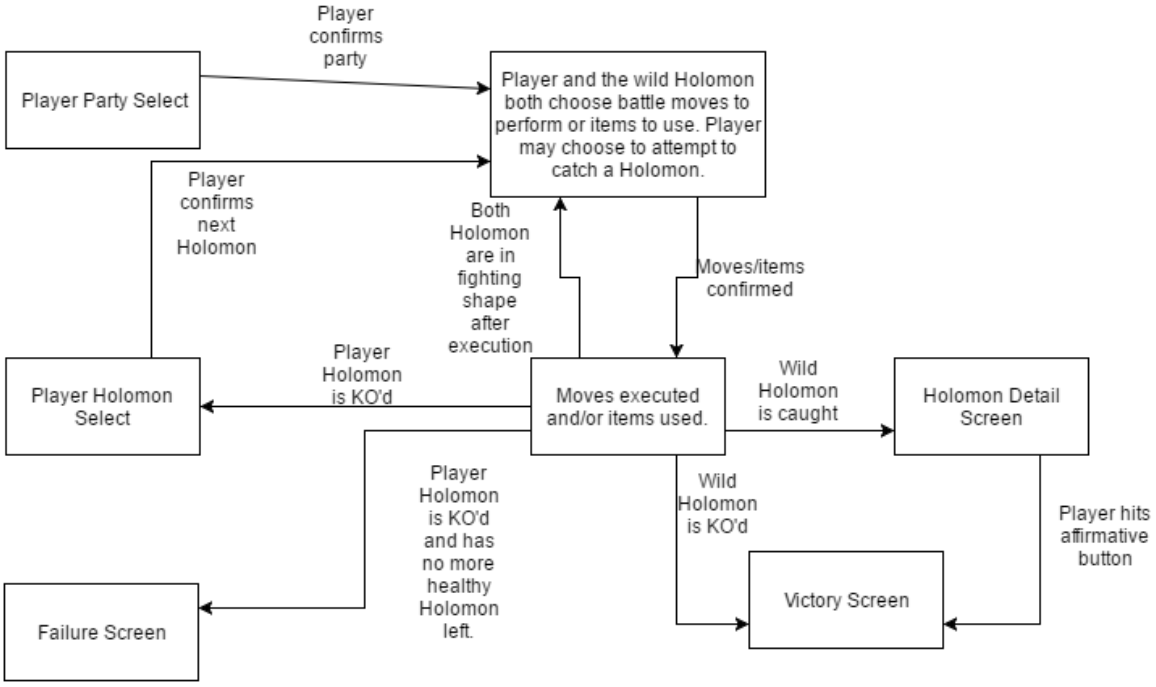
Basic Game Flow



Player vs. AI Holomon Catcher/Player vs. Player Flow



Wild Holomon Battle Flow



3.3 Assessment of Proposed methods

When creating a Hololens app, there's only two ways to create: DirectX 11 and Unity.

DirectX 11 - This method is not recommended by Microsoft and would result in too much work for a ~7 month project.

Unity - This method is recommended by Microsoft and will be much faster to develop with. This is the method we've chosen, as we've all had experience with Unity, and understand how the pipeline process works.

3.4 Validation

Because we're using an Agile/Incremental development, we will be testing new features as they are added. All new features will be examined by other members of the team, making sure they hold true to the original plan on the design doc, or the design doc has been modified to fit this new functionality. Tests will include the average cases that a user will experience, with edge cases that we see fit to test along the way.

Also, because most of our project will be using the Hololens, which depends on numerous variables (real world position/rotation, room shape, etc.), our different requirements will be tested on the Hololens, in different rooms, in different areas. We will go through and execute different tests in different areas to make sure that our project is functioning correctly. For example, "The player will be able to engage in combat with these wild monsters" can be tested by interacting with numerous spawn points located around the building.

4 Project Requirements/Specifications

4.1 FUNCTIONAL

List and explain the functional requirements of the project. This would include all the technical requirements you fulfil during your senior design project.

The player will be able to:

- Walk around our projected environment with the Hololens
- See wild monsters in this environment
- Engage in combat with these wild monsters
- Capture these wild monsters
- View what monsters they have captured while playing
- View monster stats of each of their captured monsters
- Interact with an augmented-reality kiosk in the real world to gain virtual items
- Use these items to help capture/heal monsters
- Use a radar system to determine where these monsters are
- Interact with and battle holographic AI
- See and battle other players

4.2 NON-FUNCTIONAL

List and explain the non-functional requirements of the project. This is where you would enlist non-technical requirements. This may still be a fundamental deliverable that your client needs at the end of the semester.

- The environment the player walks around in will be Coover Hall, and can go to any of the rooms.
- There will be 10 unique monsters to find and capture.
- All battles will be a turn-based system in which the player attacks first.
- Players can capture up to 50 monsters.
- Players can hold up to 50 items.
- Players can receive items from Kiosks every three minutes.
- Monsters will have an HP, Attack, Defense, Speed, Stamina, Level stat.
- Monsters will have three moves to use to attack.
- The radar system is refreshed every 3 seconds.
- There will be 4 AI's to interact/battle with.
- PvP battles will not last longer than 10 minutes.

5 Challenges

One difficulty we will have is that the Hololens is a brand new piece of technology and there's not a lot of documentation on it. A lot of our project will be researching the Hololens, scanning through all of the documentation online, and looking through different tutorials. Since these are brand new devices, there aren't many cases where people have successfully created a large project for it or documented it well. On top of this, the infancy of this device means that the forums will be fairly barren and only starting to fill out.

Another potential problem will be checking out the Hololens for developmental purposes. Only having two to check out from the college could mean that they are checked out during crucial testing times. It also means that we won't have access to the device at any time that we may need it, which could slow down our progress. On top of this, if another user happens to break or ruin one of the devices, we'll be even more crunched on time when trying to test our project. However, there is a Hololens emulator we can use, and there are some cases in which we won't need a Hololens to test out new features.

6 Timeline

List of Tasks (*These tasks are referred to by number in our assignments and charts):

1. Have a building scan of Coover Hall and be able to load it onto the Hololens
    a. Only hallways and select rooms will be scanned
2. Have the Hololens understand where it is located in the building scan from task 1.
3. Create/find 10 monster models that are fully animated. These models should be able to be viewed on the Hololens.
4. Have a spawn point that will spawn a monster at that point. The monster will then wander around that point with their walking/idle animation.
5. Place spawn points around Coover. Monsters will not be visible until the player gets within range.
6. Create the monster class, which includes the battle stats every monster has. These stats are randomly generated when the monster object is created. A master movelist also needs to be created for all monsters.
7. Create the battle system, i.e. how turns work, how damage is calculated, how moves work, etc.
8. Give the player their first monster to start the game with.
9. Allow the player to interact with a monster in Coover to start the battle.
10. Create the UI system for battling with a monster. Player will need to decide which move to attack with.
11. Add sound effects to all interactives to give feedback to the player so they know that they tapped something (A menu, Holomon, etc.)
12. Add voice recognition to battle mode, so players can either tap their UI or use voice commands to execute attacks.
13. Finalize battle logic, so that a battle ends once a Holomon runs out of health. If the wild Holomon runs out of health first, it will be removed from Coover.
14. Add a party list that shows all of the monsters the player has caught. A monster can also be selected from this list to show stats about the monster, and the monster can enter the “real world” and run around the player.
15. Create capture items that allow the player to capture wild Holomon. This includes the base class, all models, and different “tiers” of items (with better percentages), and animations.
16. Create an inventory system that holds all of the items the player has gotten so far. This includes a UI to see what they have currently. Also give the player 10 or so items at the start to use.
17. Extend the UI system during battle to allow players to capture wild Holomon. Capture rates depend on the tier of item used and the Holomon’s current health. If a catch is successful, the monster is added to the player’s party list, and disappears from Coover.
18. Modify the start of a battle so that players select their starting Holomon. A UI is needed for this.
19. Add the option to permanently remove a Holomon from the player’s party list, so that they can free up space if needed. This will extend from the player’s party list UI.

20. Create a radar system that lets players know where to look for a Holomon. Using this radar system, players will know which direction to travel in Coover.
21. Create the Kiosk object, which grants the player a random assortment of items every 3 minutes. These items are added to the player's inventory, and used up in battle. This include the UI on the Kiosk, the model of the Kiosk, and the placement of the Kiosks in Coover.
22. Add a new battle type which is AI controlled. These battles will have a set number of Holomon to face, and the AI will be smart enough to know when to use what attack.
23. Place this AI in Coover somewhere and add a UI to engage with the AI in battle. Also on this UI the player will select their Holomon they want to battle with.
24. Create 3 more AI's, each with scaling intelligence and Holomon strength. Add these new AI's to Coover to be battled.
25. Add network support so that the player can see other players who are also playing Holomon nearby. Other players will have a visual effect to distinguish vs. other normal people.
26. Add the option to challenge other Holomon players to a Player vs. Player battle. A UI is needed to challenge, accept, and select monsters for the battle. Once both players have accepted, a turn-based battle will begin.
27. Add player profiles so that stats such as "Holomon caught", "battles won", etc. can be viewed by the player or other players.
28. Add player achievements that are awarded on certain milestones in the game, such as defeating all AI, catching all Holomon, etc.
29. Trading between players (stretch goal)
30. Finish up the game by adding credits. :)

You can view our Gantt Chart below as fig. 1

You can also view our Critical Path diagram below as fig. 2

Total Assignment of Tasks:

Zach: 3, 5, 9, 10, 12, 15, 20, 23, 27 (110 days)

Ryan: 1, 5, 22, 23, 24, 25, 26 (135 days)

Tyler: 2, 7, 11, 13, 14, 17, 19, 21, 28, 29 (108 days)

Robert: 1, 4, 6, 8, 13, 14, 16, 18, 19 (79 days)

6.1 First Semester

Tasks 1-6 are expected to be completed by the end of first semester, with 7-21 being finished by the start of second semester.

6.2 Second Semester

Tasks 22-30 will be completed over the course of the semester, according to our timeline chart (fig. 1).

7 Conclusions

Our overall project will be creating a game for the Microsoft Hololens that is similar to Pokémon GO. Pokemon GO has some missing features or features that were poorly executed. For example, we're looking to add turn-based battling, while improving the radar system. In our game, players will be able to search for Holomon, battle other AI and players, and explore their environment. Along the way we will be adding to the communal knowledgebase of Hololens development by finding our own best practices for both back-end code and UI design.

8 References

Microsoft Hololens Site/Tutorials:
https://developer.microsoft.com/en-us/windows/holographic

Unity API - UnityEngine.VR.WSA:
https://docs.unity3d.com/550/Documentation/ScriptReference/VR.WSA.Input.GestureRecognizer.html

Pokemon GO Overview/Review
http://www.ign.com/articles/2016/07/13/pokemon-go-review

9 Appendices

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. You may also include your Gantt chart over here.

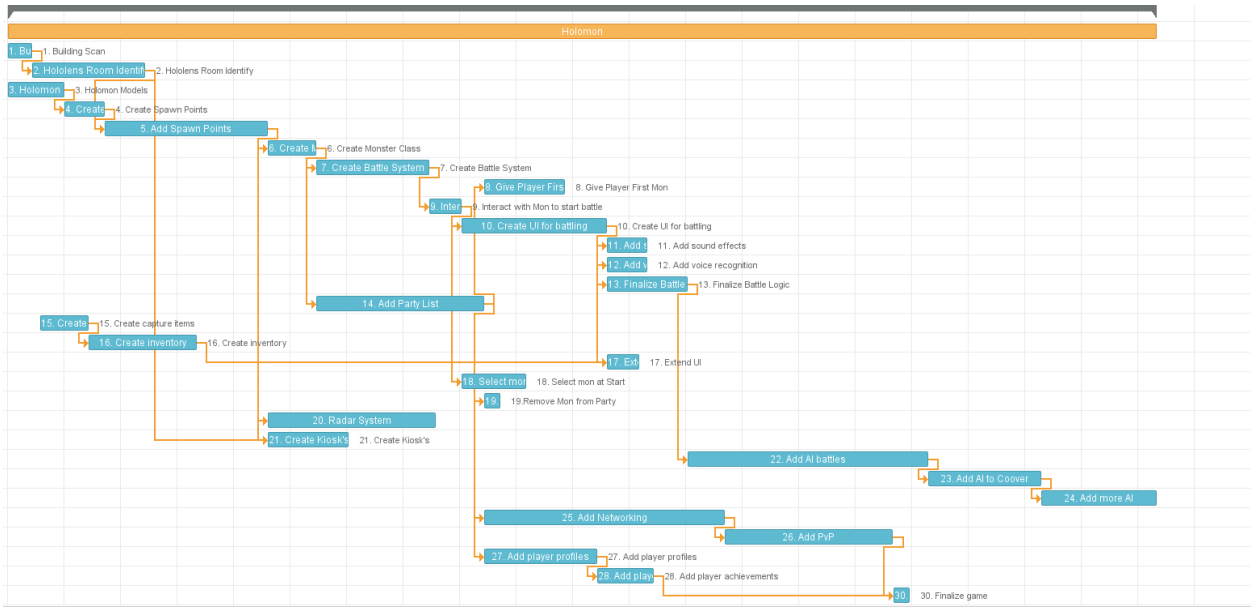


Fig. 1 Gantt Chart

(Link)

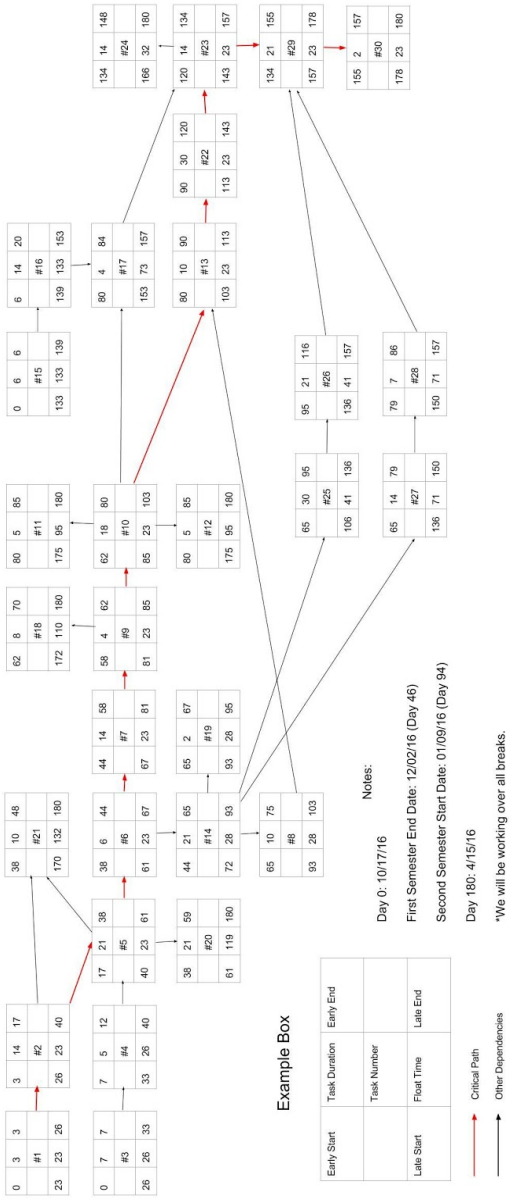


Fig. 2 Critical Path

Implementation Details

Overview

We created a lot of different features for the game Holomon. Some include: Holomon (creatures), Spawn Points, Turn-Based Battles, Items, Kiosks, and Multiplayer. We tried a few different ways to implement each feature, but this is how we decided to create them.

Holomon (creatures)

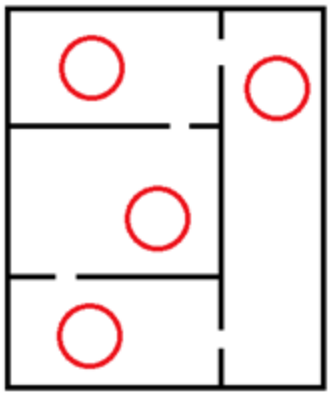
Every Holomon consists of a model and a script. This Holomon script keeps track of each individual Holomon's stats and is used in battle for attacking and defending. We decided to store values for every Holomon in a JSON file, and this file is read when a Holomon is created in the game. The values in this file describe a range of numbers each stat for a Holomon can have, so each Holomon is unique from others from the same species. The model that is to be shown to the player is decided from this JSON file as well. The benefit to this method of saving data, is that it's easy to read from the file, and we can add/edit/remove the file as needed.

Spawn Points

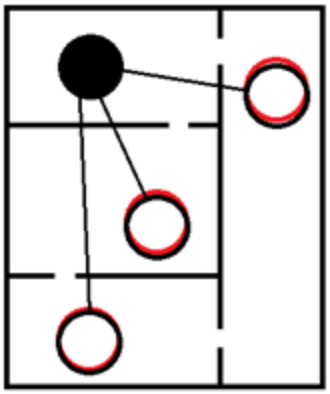
Spawns Points are placed around the building by an admin user, so that Holomon can spawn around the building. This was the feature that went through the most iterations to get correct, and we finally achieved this goal using anchor points that are built into the Hololens. The major benefit to this method, is that the game can be molded to fit any play area, and players don't have to scan the entire building to enjoy the game.

As an admin is placing the spawn points around the building, the Hololens is creating something called world anchors for each object. World anchors are a way to tie an in-game object to a real-world location. These world anchors are stored on the Hololens, but can also be transferred to other devices. Using this method, we can have the admin place all spawn points, then transmit all spawn point anchors to other players.

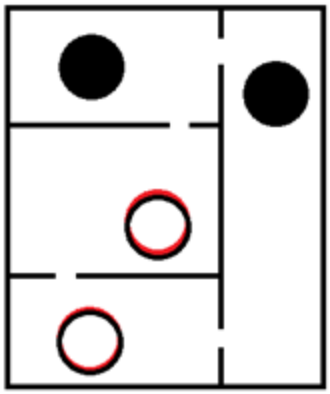
However, a world anchor is only recognized when the surrounding area has been scanned by the Hololens. This led to an issue. The player would have to scan every area around every spawn point to get it to register, which is the problem we were looking to avoid. We found a solution that worked well though. Because we know the layout of all spawn points, once we find one spawn point (or world anchor) all other points can be updated instantly too. This can be viewed in the diagram below.



The planned floorplan with spatial anchors



The player found a spatial anchor (black), and the "assumed" locations of others were placed



The player finds another spatial anchor, and it jumps to the correct location

Turn-Based Battles

When in battle, every Holomon is offered the choice of three moves, to be decided by the user. Each attack deals different damage depending on the attack chosen. However, attacks require stamina to use, where the strongest attacks require the most stamina. Stamina is replenished every turn, but the Holomon's stamina will drain if only using the strongest attack. The benefit here is that it takes some strategy to win a battle, as well as having a strong Holomon with high stats.

The battle continues after both battlers have chosen a move. Instead of polling every frame to check if moves have been selected, the check only occurs when a battler chooses a move. Despite the battling not continuing until both players have selected a move, there is still an order to the way moves are executed. Each Holomon has a speed stat that determines the order. The faster Holomon executes its move first, and, if the slower Holomon is still alive, it executes its move.

Items

To increase strategy, items have been added to buff and heal Holomon. Items can be used in place of moves, once per turn, provided that the player has that item in their inventory. For example, depending on the situation, the player might find it better to buff their strength this turn instead of attacking.

Items have been implemented with a baseItem class in mind, so that each Holomon can use a generic baseItem object. Every health, buff, and capture item extend this class to implement specific items. Within each item, values are stored, so it's possible to have different tiers of each item, to provide a means of progression to the player. We did not have enough time to implement these tiers of items, but the groundwork has been laid down to meet this goal.

Kiosks

Kiosks were designed to dispense items to the player to help them on their journey. A kiosk will only dispense items every three minutes, so the player will have to explore around for more kiosks if they need more items and can't wait. Kiosks are guaranteed to dispense health items, then a random number of other items.

Kiosks are placed around the building using Vuforia, an image recognition software. Using Vuforia, players can look at images on paper on walls or on tables to view kiosks. Kiosks are then displayed only when a player is looking at them, and disappear when not. Kiosk timers are saved in a dictionary using each Kiosk as the key. When kiosks reappear, they get the remaining time from this dictionary, and they save to the dictionary when they disappear. We decided to use Vuforia for this feature, because we wanted to compare different options for saving hologram positions. Because kiosks are only needed when the player is using them, Vuforia was a great choice to implement kiosks.

Multiplayer

Having multiplayer in our game was an interesting challenge. Microsoft has a tutorial that helped us learn how to send position of each player, as well as sending commands when a player does something, like send a battle challenge, or use a move. Using a shared world anchor, positions of each player are synced and sent in real time, with little to no lag. Each Hololens is connected to a virtual server located on a computer on the same network as the Hololens, and information is sent and received in this way.

UI

Ring Menu

One of the most interesting challenges that we faced in designing Holomon was creating an interaction pattern for functionality that doesn't have any physical representation (e.g. a settings menu or an option to look at your Holomon party). We wanted to create a pattern that could be applied in many different situations and feels natural to the player. This lead to the creation of a ring that follows the player at their feet, allowing them quick and easy access to a sort of menu system just by looking down at their feet. When a player looks at their feet, they see a ring of bubbles with icons representative of what they do when clicked. When clicked, they perform whatever action they are design to do.

The three parts of a ring menu are the MenuSystem, menu_ring, and MenuItems. The MenuSystem exists as a parent object of the menus and it's primary function is to serve as the container that follows the camera (and thus the player) around. Below that are the ring and the items. The ring acts as a base and is in charge of keeping it and its children firmly planted to the ground. Finally, we have the menu item itself, which is the actual interactive part of the system. Each item contains two scripts: the general one that takes care of swapping materials on the icon and such and a unique one that implements the IMenuAction

interface. This script is where all the special code code goes for what the button is actually supposed to do. It must implement the action() command that gets called by the general script when selected.

Popups

Popups are one of the simpler ideas to understand because they were developed from an existing pattern you may be familiar with: computer windows. On most GUI-based operating systems, the content and interactivity exist within a window that can be moved around and interacted with and menus popups are very similar. In our case, a popup is an object that is designed to, well, *pop up* when a user needs it to move on in our game. Our best example of this is the Holomon select popup which appears at the start of a player battle. When a player interacts with an item that has an interaction orb, anything that is spawned to give information or further interaction is also a popup.

Interaction Orb

The interaction orb came out of a need to have more interactible options for items within our virtual world. We wanted to create a non-obtrusive way to let the player know that they can interact with something in the world and let them know what they need to do to interact with it. We took inspiration from Google's design guide for Android devices and figured that an orb containing three periods inside it worked as a way to tell the player that more can be done with this item.

This idea was designed and implemented as a prototype, but is not used in our game.

Testing Process

When it comes to games, testing can be a challenge, due to the number of variables that can change. When it comes to the Hololens, even more variables are introduced. Depending on the position of the player, look direction, room size, and room layout, the game could encounter bugs. As a result, we came up with a cycle for doing tests. When a new feature was implemented, it was tested in Unity to get rid of the major issues. Then it would be tested on the Hololens emulator in order to see if the hardware could run it without issues. After that, it would be deployed to a Hololens device, and be subject to a multiple of tests, involving different people and rooms.

Testing Results

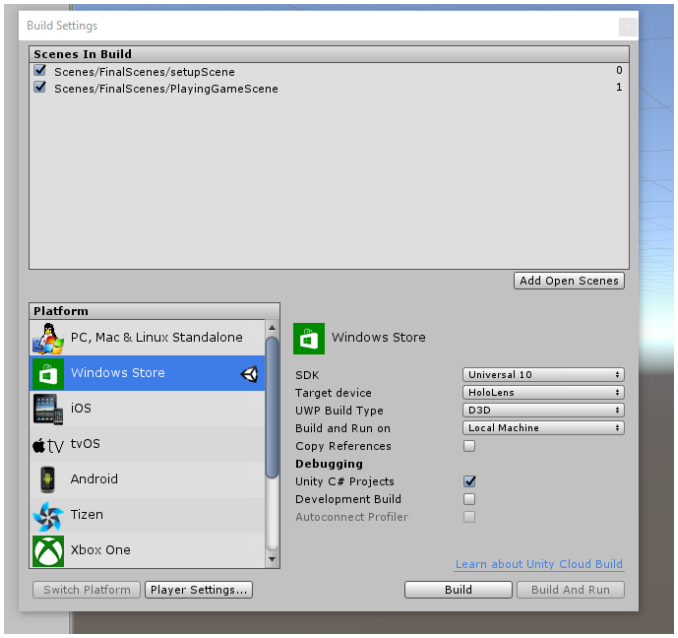
The results of the tests varied from feature to feature. The hardest feature to test was the spawn points, as that needed real-world reference points to work correctly. Then, we needed to test it on two devices, to make sure points were being transferred correctly. When a bug or issue was found within a feature, it was written up on a bug list, organized by each feature within the project. In most cases however, testing came up as a success, and we would continue adding to the project, still testing along the way.

Appendix I - Operations Manual

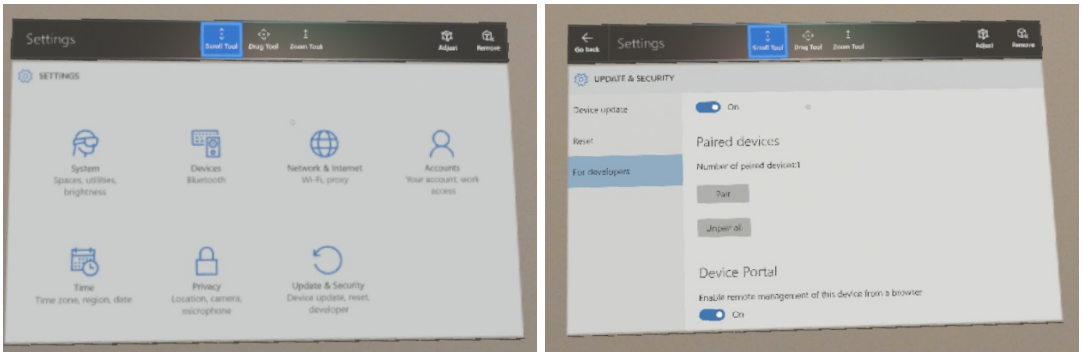
In order to turn Holomon into a playable game, you'll need the following:

- One (or more) Hololens devices
- Unity Game Engine
- The Holomon Unity project source code
- Internet access

1. Open up the project in Unity. In the build settings, make sure that the scene "Setup Scene" is checked, and is first in the list of scenes. Make sure the target platform is Windows Store, and you are creating a D3D Hololens application.

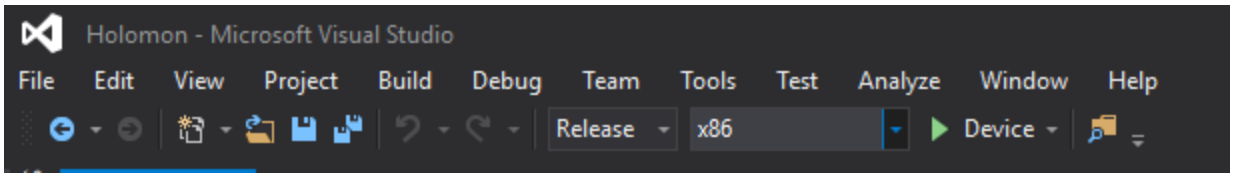


2. On the Hololens, turn on "Developer Mode" in settings. While there, you'll also want to turn on the device portal option, as you'll need that later.

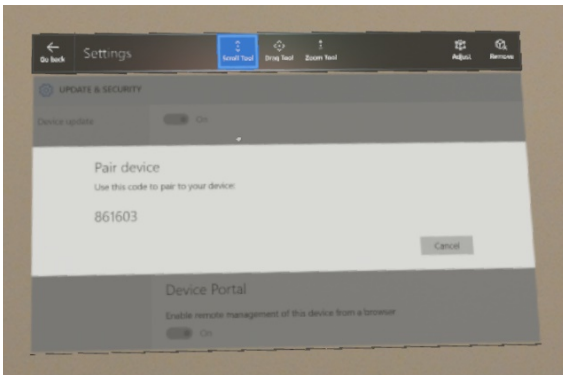


Developer Settings are found in the Update & Security section

3. Once built, open up the solution in Visual Studio. Begin deployment to the Hololens by plugging it in, and selecting "Device." You may also use the "Remote Device" option to deploy over Wi-Fi, but know that this is slower.

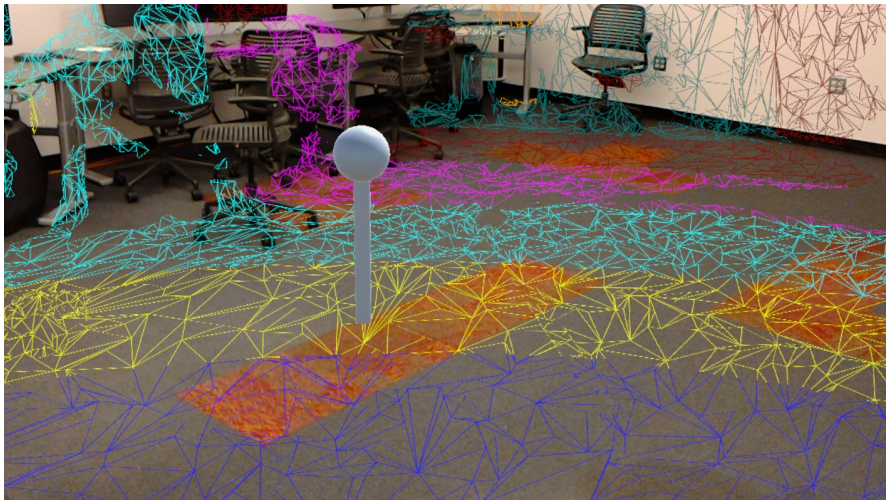


4. Visual Studio will prompt you for a pin. In the settings area where you enabled Developer Mode, there is a "Pair" button that gives you the pin needed. Enter this pin in Visual Studio.



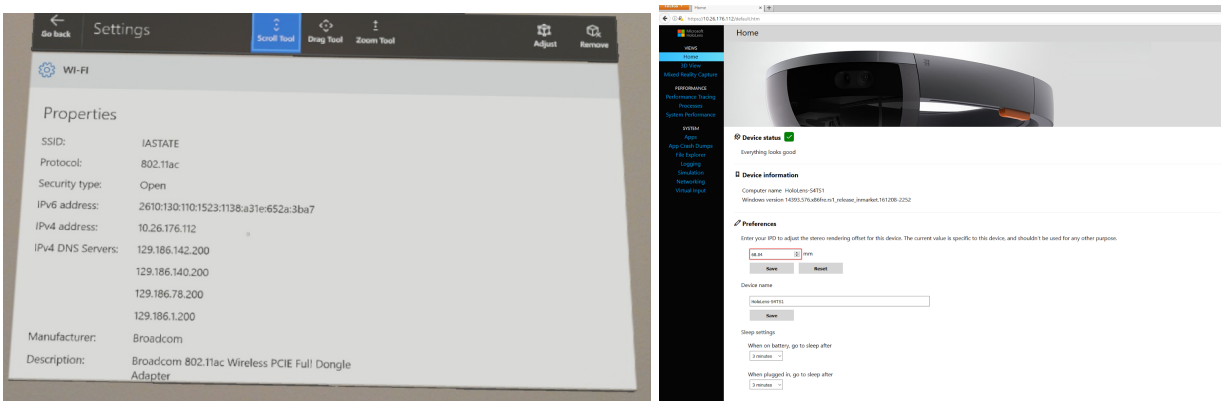
Number will vary

5. Once the game has been deployed, you should disconnect the cable if you used that option. Note you will have to restart the application on the Hololens.
6. On the Hololens, with the game running, you should be able to see a colored grid that shows the parts of the room that have been mapped. The Hololens will continuously be scanning while you are moving.
7. In a clear voice, say "Create Spawn" to create a spawn point. This spawn point will follow your gaze until you air-tap to place it. Repeat placing spawn points around the building until you are happy with the layout.



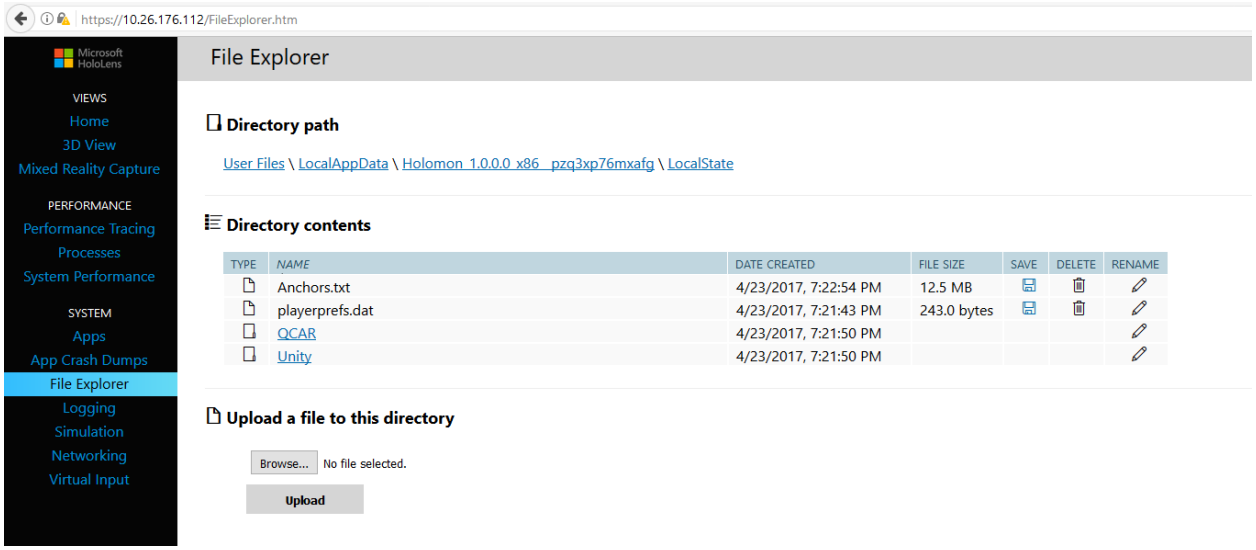
8. In a clear voice, say "Export Store" to save the building layout for other players. You will hear a chime sound when it has completed.

9. Close the application, and navigate to the online device portal for Hololens. It can be found by typing in the IP address of the Hololens on any computer that is on the network.

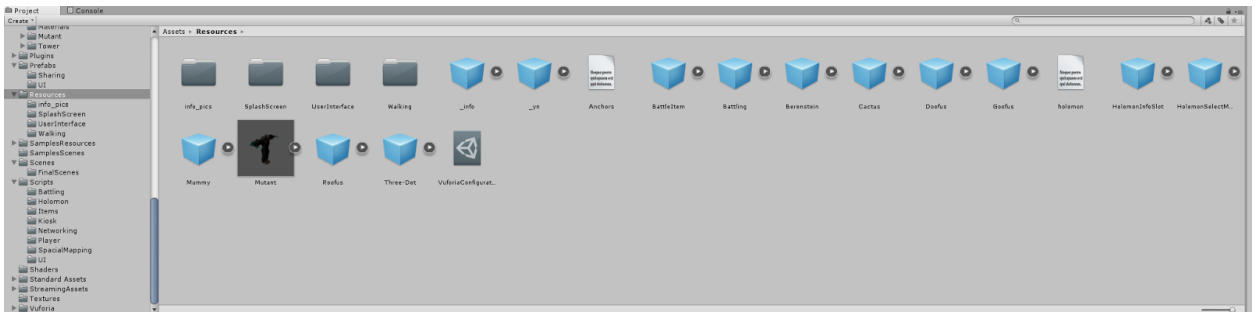


Use the Ipv4 address to log into the portal

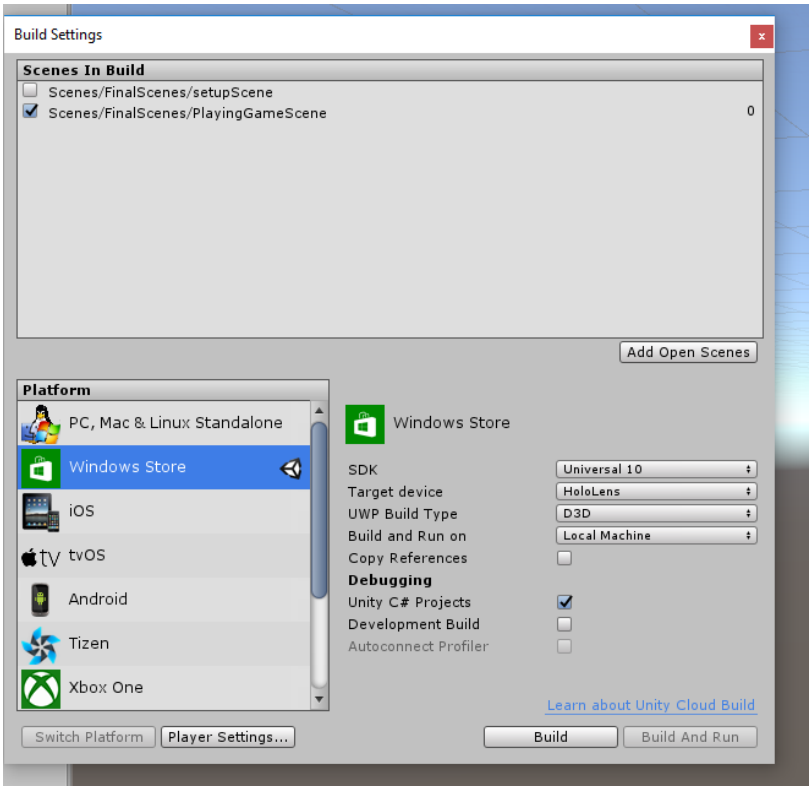
10. You can create an account for the Hololens portal if this is your first time using the portal.
11. In the File Explorer portion of the website, go to LocalAppData, Holomon, LocalState, and download the file that is called “Anchors.txt.”



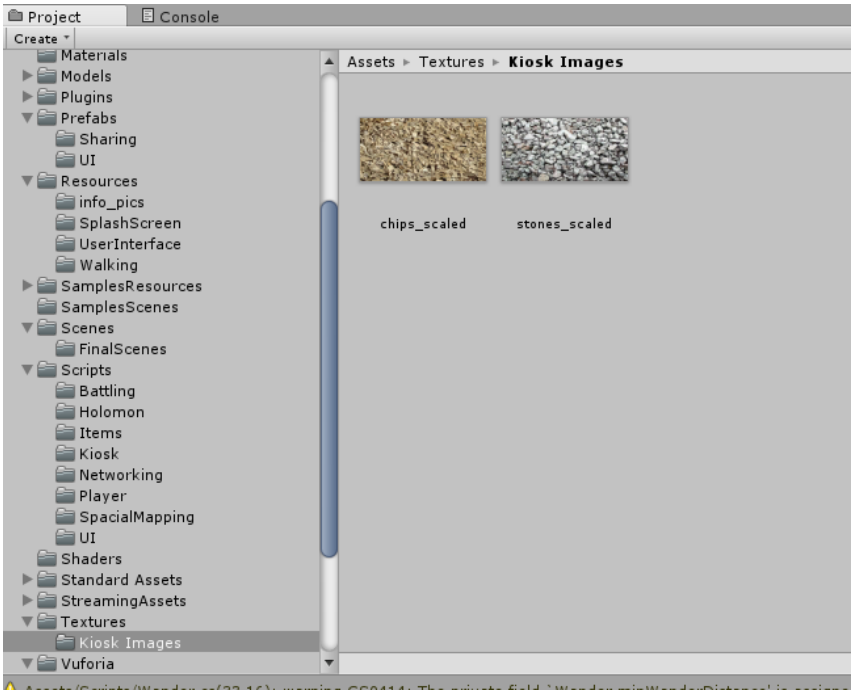
12. Place this file in the Resources folder in Unity.



13. In Unity, uncheck the “Setup Scene” scene in the build settings.



14. Export, build, and deploy the new version of Holomon to the device. You should know be able to find spawn points, see Holomon spawning, and engage in battles.
15. Kiosks can be placed around the building by using printed out images. These images are found in the Assets Folder -> Textures -> Kiosk Images.



Print these images on a normal piece of printer paper

Appendix II - Alternative Options

Some features had a range of options to choose for implementation. We spoke earlier about the spawn points, how that went through a lot of iterations to get to the current implementation.

The first idea that we had, was using the built in scanner of the Hololens to create an entire building mesh, that players would walk around in. By doing this, we would have the location of everything we needed while working in Unity. However, this led to a number of issues. The mesh of the building proved to be too big to download, as the memory of the Hololens wasn't big enough. Also, we would have needed to start the player at a set position and rotation in the building, in order to have the in-game mesh overlay the real building perfectly. Another idea we were toying with, before we learned about the download size issue, was using QR codes to sync the player position. So they could start at any position in the building, then sync the overlay mesh with the real world using this known position. This is what eventually led to using world anchors.

Multiplayer was going to use a 3rd party server system that one of our members has used in the past. It would have been responsible for sending the player positions and commands, what is now currently being handled by a Microsoft server that was designed for Hololens. The main issue with using the other system, was players would be spawned in a coordinate system on the server, and this coordinate system wouldn't always line up with the Unity coordinate system, and it definitely didn't line up with the real-world coordinate system. Basically, we would have had three coordinate systems overlaid each other, and that was too much for us to handle and sync up. By using the Microsoft sever, the Unity and "Online" coordinate system became the same thing, and we no longer needed the Unity coordinate system to interact with the real-world coordinate system.

Appendix III - Misc. Information

With our project being a research project, we have a lot of information to share about what we've learned. However, instead of putting it directly into this report, it is located within another document that is on our website. This document is written for other Hololens developers, giving them tips on creating an application, pitfalls to avoid, and general good practices we've found while developing.