Prototype Process Journal

Tyson Moll (Jan-April 2020)

This document is intended to accompany my thesis *Multimodalities in Metadata: Gaia Gate* as appendix material detailing the design process for the prototype. Citations included in this paper are in reference to the bibliography of that document.

Journal 1 documents the proof-of-concept prototype, whereas Journal 2 covers Gaia Gate, the demonstrative prototype for External Contextual Metadata in application development.

JOURNAL 1: Proof-of-Concept Prototype

June 30th, 2019 - August 1st, 2019

2019-06-30: [Proto1] Reflection on Personal Prototyping Process

My personal process of making generally begins with developing an overarching understanding of the concepts and players involved in a project: who is it for? What is to be achieved? What are my milestones and goalposts? These questions help me determine a timeline and frame to encapsulate my process and how each milestone can provide myself or a supervisor some sort of value. I provide myself room to make adjustments through the phases of development, which may result in achieving deadlines earlier to move creative experimentation forward or rearrangement of priorities and goals.

I am influenced by architectural design modularity and software engineering project structures where much of the work is modularized and functions under a series of systems that work hand-in-hand but still have foundational elements and a sense of hierarchy. I see these wickets as very reminiscent of Agile practices: common tech industry workplace structures that often prioritize rapid production of minimum viable products and low-risk stages of development. The framing of this prototyping structure can lead to a ‘tunnel vision’ bias where much of the focus is spent on smaller details within the frames as opposed to a larger restructuring of the modulus of the project timeline. It also can result in many small, unpolished elements as opposed to a simpler, whole composition that may be less technically nuanced but result in a more complete experience.

2019-07-02: [Proto1] Process Reflection / Digital Photos

We had a group discussion of our personal processes of making in class today. During our talks, a classmate made an interesting analogy to a pendulum, where focus bounces back between different elements of a project beginning with an initial strong force that reaches

equilibrium as the project progresses (not necessarily to completion). The imagery was suggestive of several metaphors:

1) Bouncing between multiple elements of a prototyping process instead of divulging a more focused attention on any particular elements to completion
2) the process of reaching equilibrium relating both to the gradual sieving of the different directions of the prototyping process blurring together into the final product, and
3) the gradual loss of momentum / focus during the bouncing between different project focuses

The lessons in the metaphors are coincidental, but each concept brings its own insights into facets of the design process, the labour of creating.

I did a sort of self-directed brainstorming activity to explore the potential answers to a thesis question I have been pondering… "what can photos do digitally?"

- Exchangeable as token/currency
- become printed in 2d/3d
- tell a history
- comparative analysis (between images, or with reality modern)
- gamify spaces
- data seed for game
- animate
- immersion
- smell / feel the space (temp, humidity, glare)
- speak with photo subjects
- decay through time (digital archivalism)

My thesis depends on determining ways to manifest a captured moment in rendered media… Maybe I need to explore some design fictions for these angles?

2019-07-04: [Proto1] Low- and High- Fidelity Prototypes

Low fidelity prototypes are easier to assemble and fail gracefully; an opportunity to try things and experiment. Lots to learn from the process of working directly with the theoretical concepts explored in a thesis through process. Care to not be naive of the problems that arise in low-fidelity prototypes; could be due to low level of development or hidden problems.

Agile as a frame seems effective for keeping objectives on track and regularly reflecting on them through the process. I feel that my current frame of working through objectives is similar, but I should consider emphasizing reflection e.g. this journal.

2019-07-09: [Proto1] Memory, Portraits, Ethics

Very interested in exploring triggers for recalling memory; the photos I capture do not have to be limited to being recreations of the memories of the photographer, but can trigger related memories for those who engage with them.

Classmates made suggestions asking about how the project was to be put together as a platform and how the renderings would be created. The most curious comment had to do with the topic of colonialism; I think colonialism links most with tourism and the concept of 'collecting' souvenirs from travels but beyond being an interesting discussion point on the activity of photography I don't feel that it's a major area for concern in terms of relevancy to the project and research. Still, worth exploring in my thesis when I discuss the topic.

I will be focusing on photographs related to landscapes, whether urban or natural, for the sake of my prototype. As the goal of my prototype is to recreate physical aspects of these photos from metadata to ephemeral renderings, they most accurately reflect the intention of this exploration. The project could be used with pictures of people, but I feel that would distract too much from the project objectives and open up issues with research ethics down the line. I'm looking forward to having this prototype effectively support future explorations in photographic metadata beyond what an Exif schema can describe.

2019-07-11: [Proto1] Structuring the Prototype

Today's the day to properly sort out how I'm structuring my prototype. Need to make a schedule… My goal is to make a photo app that can not only capture new photos and get fresh metadata for them, but take existing photos and render them. Unfortunately API services that offer historical weather data are costly, so I will need to create dummy data for the pictures that I wish to present, using the saving and loading processes that will be incorporated into my project. Somehow, I also need to come up with a way of making the interface attractive for an android device.

On a related note, I managed to get my late grandfather's Polaroid camera working. Funny how finicky it is… and how expensive it is to operate! $5 a photo! But there is a real attraction to the immediateness of the photographs it captures and its stylization. It encourages a culture of capturing special moments and provides the technology for people to instantly engage with the capture. Instagram has been set on recreating this experience for mobile devices, in part by restricting access to the platform to mobiles only. With the minor annoyance of having to collect this API data manually, I feel like my project will benefit from its concept of instantaneous capture.

2019-07-16: Weather Assets

Headed off to Quebec for the week with the intention of somehow pulling together my prototype while on vacation! We're at a cottage in Trois-Pistoles adjacent to the St. Lawrence river. It shifts between high tide and low tide depending on the time of day. Wonder if there's an API for that? It's another indicator of changing spaces… I suppose the movement of animals qualifies as another factor, seeing the seals pop up upon the rocks on the water periodically. There's a great deal of periodical data that remains undocumented.

I've purchased the Enviro Unity Extension Asset today to represent differing weather conditions. It has several built-in systems for rain, clouds, sun and fog, but I'm hoping that I can pull together more interesting ones in the future. Still, with the ability to control the time of day (and hopefully the angle of the sun?) I imagine I can create representative demonstrations for the limited set of photos I will be incorporating into the demo for next week.

2019-07-20: [Proto1] Prototyping: Draft

I've put off the bulk of the work until today, but I now have a visual of how it will look in the context of the application. I decided to go with having the images and camera view represented in a 'plane' object. On the programming side, I'm trying to design the app to be flexible to whatever operating system it runs on and however many images are loaded on it. Still, I've yet to properly test everything working together as well as Android compatibility. A bit nervous about having everything together on time for a presentation among other deadlines, but I believe the system that I've set up will work nicely for my goals. I might use the 'Gaia' Unity Extension Asset which could create some interesting landforms[1], but for now I am going to focus on assembling this bare minimum of weather representation for images.

In summary, my goal is to have a camera applet that captures an image and the present weather and GPS information, storing it into the file via steganography instead of traditional metadata. I will be using a custom-made format with a header identifying it as a 'PhotoStego' file. When booted up and after saving an image, the applet will identify all the .png files currently stored in its snapshot folder and allow the user to navigate between them in photo view mode. Whenever an image is loaded, it will overwrite the currently displayed features. Once this app is properly developed, I should be able to develop and use it alongside any project I work on through the course of my thesis, adding new API sources and features as I go and utilizing it to gather resource images and data to use.

[1] I did not end up doing this.

Figure 1: early iteration of the proof-of-concept prototype

2019-07-22: [Proto1] Coding Train

Taking advantage of the train ride home from Quebec to pull together what remains with this prototype. Lots of debugging, lots of case-proofing, lots of putting the pieces together. The steganography elements seem to work great now from both an encoding and decoding standpoint. Although I haven't had the chance to test it, I've made sure that the application should be flexible for use on other operating systems, mobile and desktop. I need to be conscious that there are several areas in the code that will need error handling methods once the application moves to a more public environment but I should be able to hold off on getting too nuanced with them considering that many aspects of the demo testing session will be held within a vacuum-environment. Still, the code is very flexible for future APIs and datasets; I'm looking forward to using it as a data-collection tool for my future work.

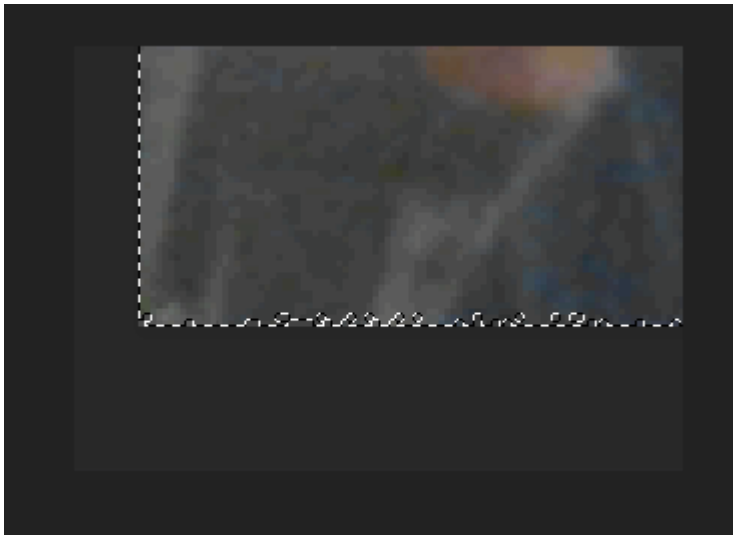

Figure 2: An image encoded using the PhotoStego algorithm (data highlighted by a Photoshop marquee)

2019-07-24: [Proto1] Thesis Prototype Presentations

I managed to put together the app in time. It still has some issues with lighting but the app ported over to my phone effectively, the steganography process works flawlessly and just 30 minutes before our class presentations were to begin I managed to code an injector to make some demonstration images for locations and times that I've visited in the past as proof of concept.

The presentation of the prototype to my class went very well, although the comments I received largely focused on the steganography aspect instead of what I really was hoping to get as feedback: suggestions for other rendering possibilities and talk about the potential affect of the images. Maybe I dwelled too much on the tech-talk; the prototype was intended to be a tech demonstration and platform for future work, but the program remains to be design-oriented. Some voiced concerns related to privacy, and there were suggestions towards its potential for more journalistic and social qualitative data points. For the time being, I would like to maintain the ethical standpoint that there is a need for users to know what information is being shared, and I want to avoid data that has personal associations with the user and neighbouring people. I was also approached afterwards about the potential to have the photographs have some sort of mark of uniqueness; unlike physical objects, digital artifacts are very easily duplicated. While it is possible to use watermarks to identify authorship and whether the work is modified, duplication remains an incredibly difficult matter to circumvent in networked data… something to ponder but perhaps not something to be concerned for as I shift my focus towards the use of the data that I now have access to.

2019-07-27: [Proto1] Post-Presentation feedback

I received some individual comments from my classmates intrigued by the extensibility of the prototype and how it circumvents the data scraping and disposal of metadata performed by social media platforms. Because the information is encrypted in this matter, it requires any platform wishing to use the data to be able to decrypt the messages included in it. I'm morbidly wondering how long it would take for such platforms to start scanning every pixel of every image uploaded to their platform for secret metadata if this application were to be successful, or whether the NSA would be infuriated by social media metadata clogging up pings as they search for covert steganographic communications. Who knows?

Through the process of creating this prototype I have gained a better idea of how extensible the project can be in terms of data and potential for representation.The steganographic method I am using seems to be capable of storing an immense amount of information within an image, so what remains to be included seems limited to my

imagination and the API / sensor services I can find. I'm excited to examine them further and broaden the range of expression available for designers using this tool. The tool, however, is still not quite perfect in terms of rendering the subject matter in a clean manner and would highly benefit from a privacy / extensibility feature that allows users to toggle which data elements to insert into the image as they choose, as well as a feature to indicate whether the data was successfully injected, certainly features I would like to include in the final revision of the prototype so that I can move on from the tool towards more creative projects.

2019-07-28: [Proto1] Non-technological perspectives and Automata

Spoke with a friend today about my project for some external critique. Her perspective is less technologically focused than others that I have spoken to in the past, so it was refreshing hearing a different sort of perspective on the prototype itself as well as how they understand the technology behind it. APIs for example may be common knowledge among programmers and those who work with IoT products frequently, but it is likely true that the average user knows very little about the technological process involved with regards to how their weather application transmits the GPS, weather, or other external API data from source to device. This seems like an important frame of reference to be considerate of, as I have largely been focused on promoting the findings of my research towards developers rather than consumers so far. Given that I am developing a consumer-driven technology that requires mass amounts of user-generated content to succeed, I think I should look more into becoming more acquainted with how the average smartphone user interacts with their device and networked technologies as well as the pipeline involved between capture, distribution, use, and death.

In terms of suggestions, one particular point that stood out for me was the notion that this sort of prototype does not have to be operated by a person, but might rather be operated remotely by a machine. For example, a weather station could be set up to share images of the city with information embedded concerning typical weather information as well as particulars such as tide levels, chance of dust storms, or current level of snow on the ground. In this sense, people from around the world could see somewhat real-time images captured from the station with relatively low bandwidth costs for the server versus real-time footage or detailed websites. The work of presenting the data can be offloaded to external applications in this way, freeing up lower income data resources from more financially demanding digital overheads.

Further, having regular postings from automated machines invites people to directly interact with locations that they may otherwise not have the opportunity to engage with. The possibilities are certainly more low-tech than VR experiences of these sites, but could

still be informative to a variety of communities looking for location-specific insights, such as surfers or fisherman debating whether the water conditions are desirable.

2019-08-01: [Proto1] Final Presentation Reflections

My final work on the proof-of-concept prototype was presented to my class and visiting professors. There was not much to show in this tech demo, as it visually demonstrates little more than a conventional digital camera at this time: what of the context that is captured in the image? What of the context external to it? What datasets do we consider worthy of including and which datasets do we choose to ignore? Are the memories captured exclusive to the photographer or are they common amongst viewers? Which frame of reference do we perceive the photos from? What value is intrinsic to each type of photograph, whether it be landscape or portraiture, documentary or artistic?

Some features that I would like to develop in future builds include privacy toggles, better visuals, additional data points, and a better file browser. Although I do intend to give users a high level of control over what data they capture and use, I believe a few default values can be established for the average user who may just disregard terms and conditions in order to get to the guts of the operation quickly; better to focus on seamless usage. For the sake of research I fully intend to ensure that the user is aware of the data they are contributing and give them the ability to opt out of the program at any time[2]; I am considering this from an extensibility viewpoint where the work takes a life beyond this project and ostensibly is used with a lighter regard for privacy similar to the millions who use Facebook today.

If one thing is clear from the conversations that were had, I cannot expect this to fully replace conventional photography unless it becomes incorporated as a standard, concealed element of digital photography in the way in which traditional metadata (Exif, etc) is hidden. Not all photographs need to be held in high regard or used in the expressive technologies that I imagine will result as outcomes of my research.

[2] This was written at a point in time when I was considering user testing and feedback for research purposes, which ended up not being an element of my research.

JOURNAL 2: GAIA GATE

January 13th, 2020 - March 30th, 2020

2020-01-13: [Proto2] Project Start:

Today, I planned out some preliminary sketches of the work I have in mind for the demonstrative prototype. These are somewhat idealized, but if it works out, all the better!

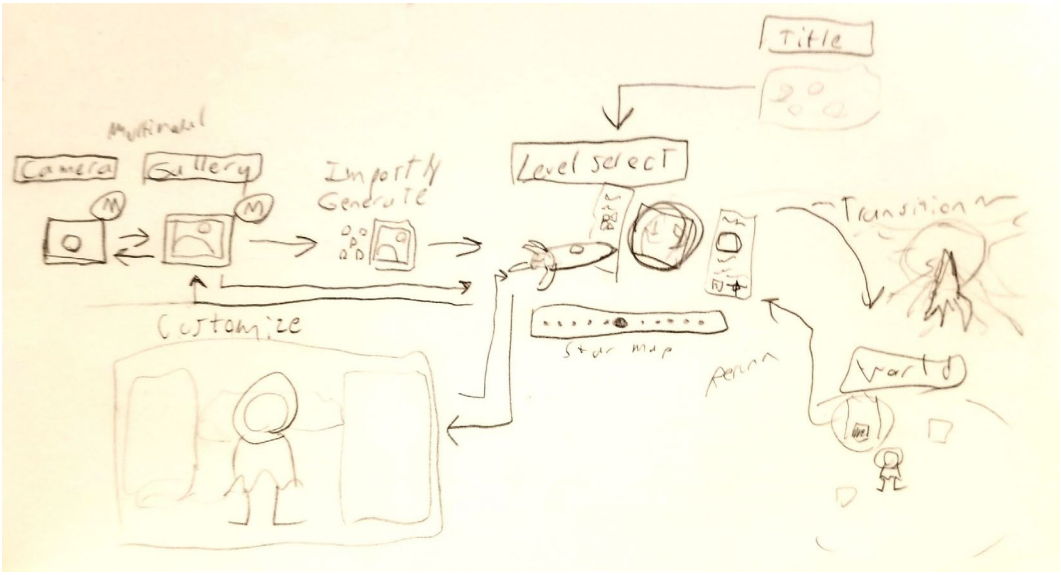

Figure 3: interface map sketch of prototype

HUB SPACE will have options to flip between generated worlds (LEVEL SELECT), CUSTOMIZE the player character, select between OPTIONS, or CAPTURE a multimodal image and generate a world from it.

LEVEL SELECT

I envision the level select as a space rendered in 3D with the primary colour of the world reflected on its planet. Probably will keep details low for now in the 3D space. There will be a preview on the right with a small scaled version of the original image, as well as a preview of the features of the planet based on the image's contextual data, perhaps? A discard option as well.

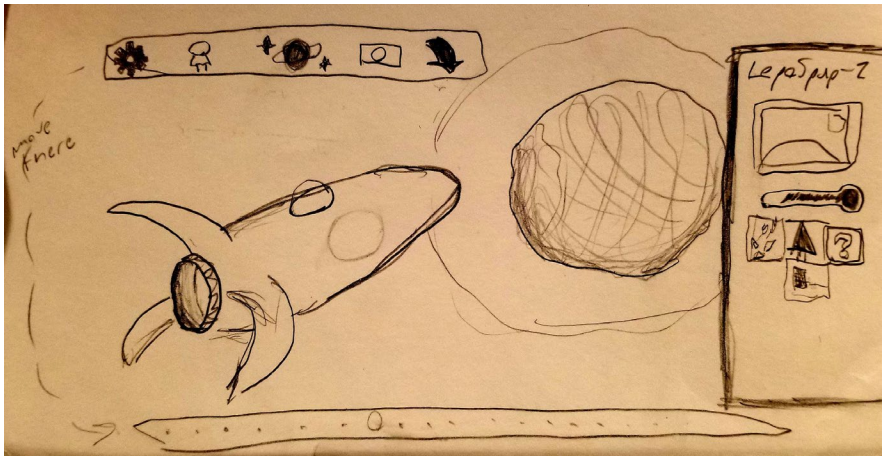

Figure 4: level select screen sketch

CUSTOMIZE

A pop up window with the player character rendered in 2D, showing off highlighted areas affected by MODIFIERS, which are selectable on the right side through the EQUIP option.

Each level will have a collectible modifier. The player can equip up to four modifiers before entering a level. Modifiers can affect things such as primary attack, health, access through lock mechanisms (should only be one type of lock for any given level, if at all), speed buffs, and primary attack buffs. Probably a discard option as well. I have this little almond-headed fellow drawn for the character with a bicycle horn like gun. Something fun with clearly replaceable parts.

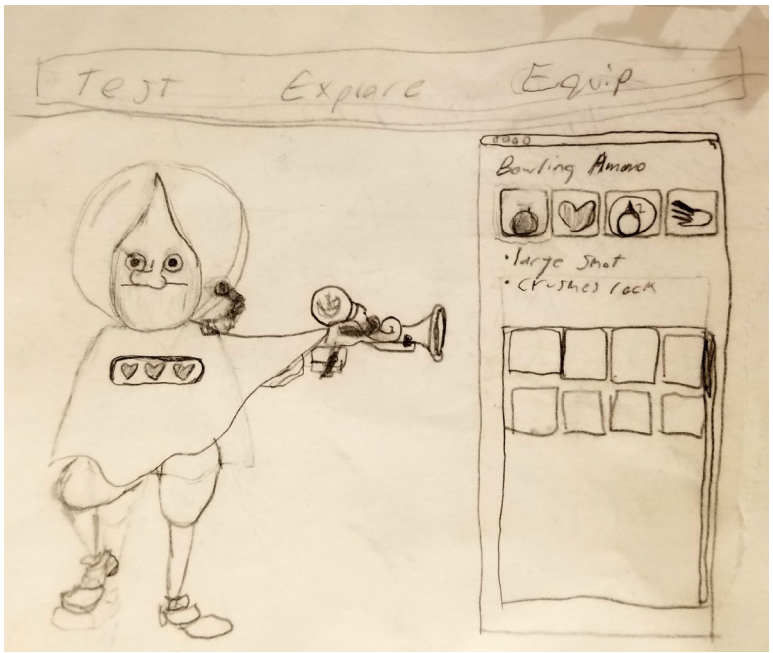

Figure 5: equip screen sketch

CAPTURE

I'm hoping for the most part I can just import my prototype and reuse its features here. Definitely will need to sketch out some new aesthetic, which I imagine will be inspired by whatever work I do on the other screens. Worst case scenario, I keep the camera wholly separate from the game and improve the old prototype. Would love an animation for generating worlds… perhaps something staticy that brings a player out of the gallery and into the level select, and the world grows into place. Perhaps the gallery can have a similar aesthetic to the camera for the time being.

WORLD

Presumably there will be a transition from the 3D level select to the 2D world.
I've been thinking about the level generation process and, for the sake of keeping things looking natural instead of highly gridlike, I think I will have the generation follow a technique similar to Vlambeer's model in *Nuclear Throne*, where blocks of rooms are premade and forced to settle out from a starting area by randomly moving in a cardinal direction, one step at a time until there is no longer a collision. I could experiment with limiting the movement of each new block after its first vertical or horizontal move (if left, then only left, if up, then only up; select between up and left randomly), which would reduce the time for levels to generate substantially over average by preventing redundant moves. The blocks would represent vacant space, and the initial block can house a starting space for the player; walls can be added after each space next to the vacant regions, and this method guarantees that each block will be adjacent to another block after finally settling.

Afterwards, content: I need to add fun details, clutter, and such. I could scan through the generated space for regions that qualify for a particular chosen type of block, then insert them into the space accordingly. I'm still not entirely sure how to evaluate these choices, but I imagine with some experimentation I should be able to find a happy medium. I have considered using the A* method to evaluate distance from particular spots to ensure that certain elements are distant enough from each other or the start… maybe not necessary, I imagine there's a simpler way. Since I am generating rooms of the levels in blocks, it shouldn't be hard to have generated features, enemies, etc, be relative to a room block's center, perhaps.

I absolutely need to establish maximum and minimum values, and the range of possible options so that I may cover all possibilities A debug menu and window will help through the process to display world stats and the product of the generation. Ultimately, I think this level generation process is most important as a starting point from a research standpoint, and will pursue it first, right after I get the GPS working as a seed up to a particular accuracy. I wager that I can incorporate the API collected data as detailing factors after I get the process associated with them working first.

Anthrome and Biome seem clear candidates for aesthetic. Weather for some form of temperature effect… Ice, Lava, Hot Sand, Snow, plenty of possibilities. Thinking far, it might make sense for a native of an area to get starting boons that are related to the space and make exploring local territory easier. Time and light seem obvious connections. I might ignore dates if I already have weather and temperature and biomes; I need to have a clear use for the dataset before I implement it, naturally.The amount of content seems overwhelming at this time. We shall see how it goes. Will certainly need placeholder content.

Figure 6: The first room

I managed to do some preliminary setup work today and have the beginnings of a Procedural Generation algorithm! It doesn't actually generate anything yet beyond the start space, but some sample rooms have been coded out for later placement. The screenshot above is from a bugfix-free less build… I'll take care of the work tomorrow, probably! I'm using images made quick in Aseprite to test out the visual layout of the vacant zones.

2020-01-14: [Proto2] ProcGen

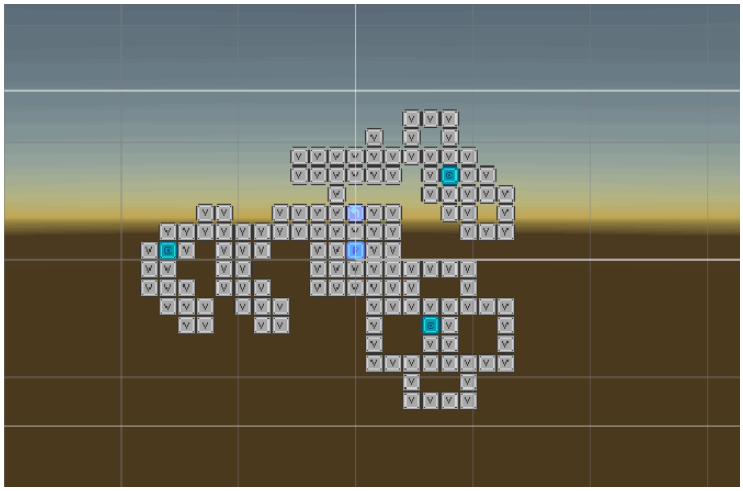

Figure 7: Procedurally generated floor map

First pass on the ProcGen algorithm is looking good! The process currently works by taking a shape designed within an 8 by 8 grid, placing it on top of the center of the room, and forcing it to move in a cardinal direction until the whole shape is no longer positioned over an existing element of the world. Then, it is combined with the existing elements and the process repeats for as many shapes are desired.

Definitely noticing that basic shapes tend to look more natural than very deliberate designs. Might also be worth investigating making room pools for certain sets of metadata in the future.There's definitely some charm in particularly odd tile shapes when used sparingly.Although after adding the solid generation script and some additional rooms they're beginning to look a bit more like creatures and generally more exciting as navigable spaces.

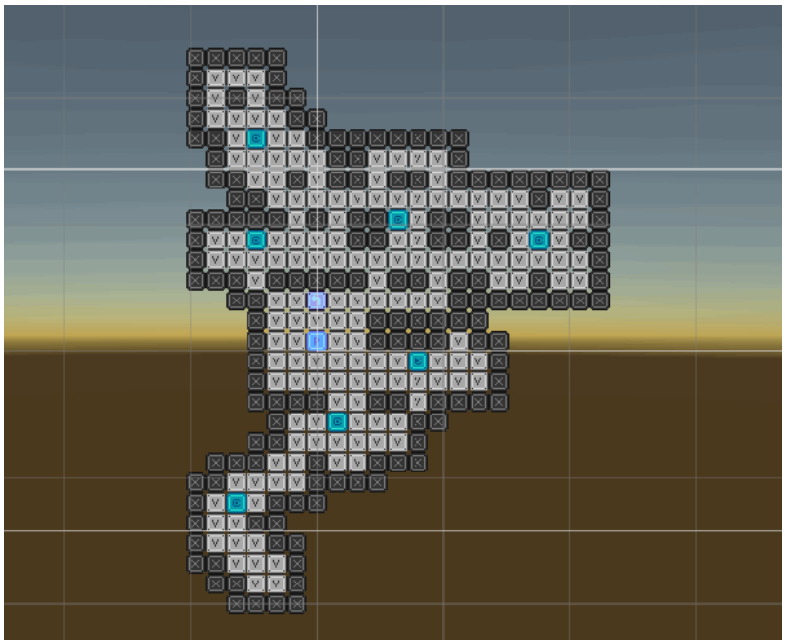

Figure 8: Procedurally generated floor map (with walls)

Figure 9: Procedurally generated floor plan (with Larger Scale)

2020-01-17: [Proto2] Global Positioning System Seeding

When I was first experimenting with the proof-of-concept prototype, one of the things I immediately noticed was that the reported GPS value tended to jump between ranges of at least 10 meters while I was inside an urban building, even when near a window with the device motionless. From what I have read, GPS has limited accuracy in urban areas, valleys and other regions with large vertical surfaces as they tend to reflect incoming and outgoing signals on their way to satellite receivers. It's clearly important to keep in mind potential

sources of inaccuracy. Imagine if I was in the mines! Even storms can affect signals; they are after all essentially thick masses of particles.

I figure I can deal with this issue in a heuristic manner that works with my intentions of level generation: to reduce the number of digits taken into consideration from the GPS report to approximately 100 meters, which would require players to travel such a distance to obtain a different level seed. However, GPS does not work in metric units but rather degrees of latitude and longitude. Each degree can be broken down into minutes and seconds, although many APIs report these fractions as decimals. The distance of a degree of latitude ranges between 110.567km to 111.699km due to the earth's ellipsoid shape (not perfectly spherical); this makes for an average of approximately 111km, which makes 1m equal to approximately 9 millionths of a degree of latitude. Therefore, if I want an accuracy of approximately 100m, I need to focus on reducing the accuracy to 0.0009 of a degree. For convenience sake, I will use 0.001 of a degree (111m accuracy).

Longitude is more of a challenge to enumerate, since the distance shrinks from its widest at the equator (111.321km) to zero at the poles. I'll need to give this more thought, but for now I'm using the same ratio value as latitude, which should be fine as far as addressing my general concerns goes. I'm not exactly expecting a large percentage of players in the antarctic.

In any case, I hooked up the GPS latitude and longitude values to UnityEngine.Random.InitValue() as a single number seed (I multiplied latitude by several tens as to leave the longitude digits untouched when added together) and it seems to be working fine for pseudorandom number generation out-of-the-box; it always generates the same level. Looking good!

2020-01-20: [Proto2] RuleTiles

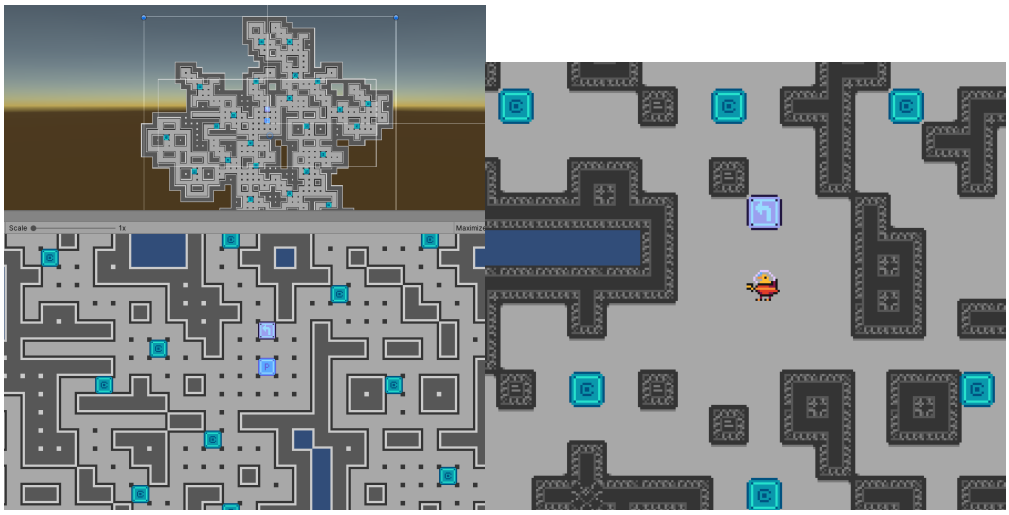

Figure 10: RuleTiles in action

Using the 2D Extras package, I added RuleTiles to the generator to replace the debug icons I initially made. This will allow me to decorate the spaces in relation to environment aspects down the line. I also drew a quick rubber duck to represent the player for the time being. Nothing else to report today, but I'm excited to doodle more with Aseprite.

2020-01-21: [Proto2] Colour Replacement and Player Movement

Today I accomplished a few things! We now have camera movement, player movement, tiles with colliders, and the beginnings of a recolouring system. I need to dig up my recolour generator code from some of my past GameMaker work, which has some excellent logic for how to deal with generating ready-made colour palettes with shifting hues, values and saturations.

In any case, here's the colour replacement in action! It utilizes several copies of a UnityAsset Store extension called Color Replacement Image Effect controlled by a colour manager I quickly scripted[3]. Looking forward to hooking this up to the generator.

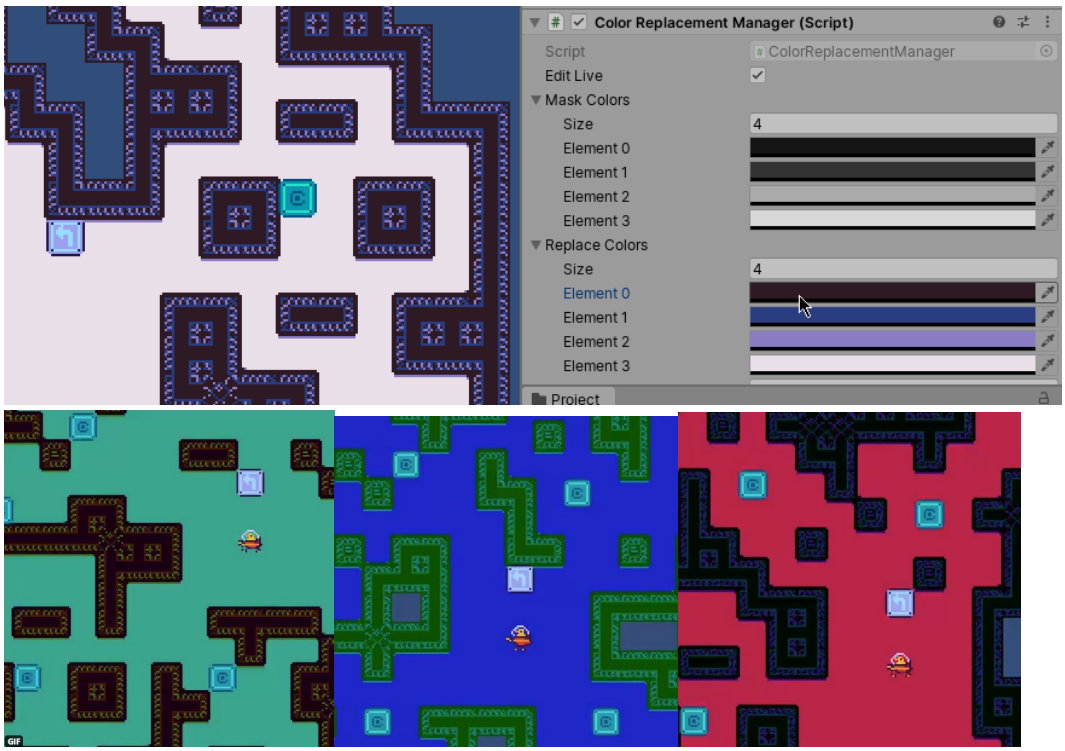

Figure 11: Colour Replacement Manager in action

2020-01-23: [Proto2] Colour Generation

[3] I have since replaced this with another asset: see 2020-02-13.

As promised, generated colour palettes! The code is based on an old project, used here to generate the look of the levels. This will be modifiable further down the line and just serves to show that each level can have a unique scheme.Once the data is passed in we can have more ‘representative’ presentations.

2020-01-25: [Proto2] Data Persistence and Saving / Loading Mechanisms

I forgot to document this earlier, but I’ve reduced the ‘bloat’ from my previous prototype by removing unneeded assets such as the Enviro asset I used to simulate the weather. Hoping to represent this again later, but in a more efficient manner. I am returning to this prototype because I will need to redesign it with the whole application in mind and have the data produced and loaded by it converted into levels for the game. This involves passing data between scenes and some ability to load information from a save file.

Figure 12: Test Mockup of Main Menu

Today’s project was to make the first steps towards having a ‘main menu’, load and save this data, and package it from the camera app into a format that can be used by the generator. At the time of writing, I have scripts persistently holding the data between rooms and the framework for saving and loading prepared. Now I just need to actually encapsulate the data and have it accessed by the generator.

2020-01-27: [Proto2] Context Data from Camera to Generator

I’m continuing the process of porting data from the camera to the game today. Noticing a few things from a developer’s standpoint. Naturally, not all the data needs to be used, so a developer would be wise to have seperate classes for holding the data initially gathered

from the photo and for use actually in-game. I will certainly need to revisit the aesthetic of the camera and the main menu UI, but for the time being my focus is to just get all the parts working. (I did!)

2020-01-29: [Proto2] ArcGIS Implementation

Having now created a working pipeline, I decided today to return to a prioritized element of data collection: ArcGIS tables. I am specifically interested in these for their arrays of locative information, including anthromes and biomes. Today, I coded out a function that is capable of reading the ASCII format of an ArcGIS file sourced from the Laboratory for Anthropogenic Landscape Ecology website here: http://ecotope.org/anthromes/v2/data/[4]. Now I just need to test it and integrate it within my multimodal camera.

2020-02-01: [Proto2] UI Mockup

Figure 13: Sketching UI elements

This weekend I spent some time exploring what could act as the 'gameplay' element of the project. The idea came to me of arriving to the levels and having to forage for the necessary materials to then leave it (e.g. fuel). With that in mind, I prepared some UI assets to model these values I wanted to track: ammunition, health, oxygen (a timer) and fuel.

2020-02-03: [Proto2] Anthromes and Biomes

[4] This and other assets used are listed in Section 8.3

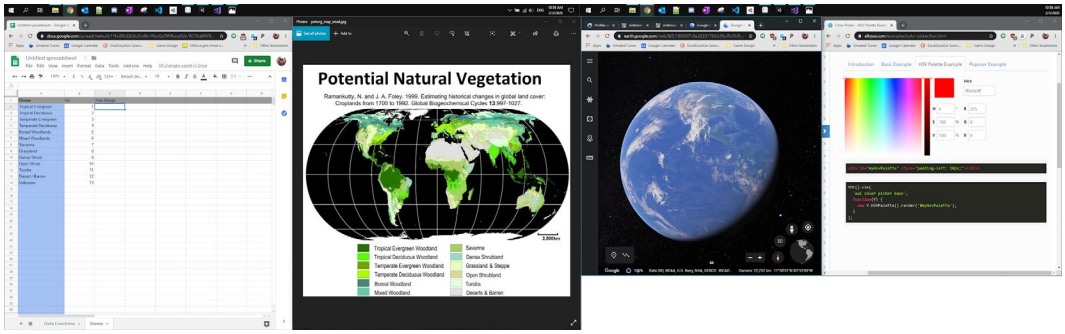

Figure 14: Desktop layout for Biome exploration

The above is a screenshot of my current desktop spread; I'm looking to explore the colour schemes that would best reflect different biomes of the world today based on the mapping provided by the dataset for biomes included in my project. I tried to get a general colour hue range of the foliage in the world's many biomes, although it seems evident that there isn't a large amount of variety… largely tone ranges between orange and green. Of course, this hardly is representative of the colours of flowers or land masses like the Peruvian Rainbow Hills.

I feel that this limited range might cause levels to have a repetitiveness to their aesthetic, so I'm imagining that it may be better to focus on representing these biomes in object form rather than colour… I like how the existing random generated colour schemes create very distinct looking spaces and I'm worried that applying such schemes would diminish the variety aspect I want to achieve. I'll make some sketches later.

Biome	no.	Natural Hue Range (summer)		Fall Max	Winter Min	Height	Remarks
Tropical Evergreen	1	63	126			Tall	Skinnier Vegetation
Tropical Deciduous	2	63	107			Tall	
Temperate Evergreen	3	77	126	49		Tall	
Temperate Deciduous	4	77	107	15	54	Tall	
Boreal Woodlands	5	54	107		54	Tall (Grassy)	Hybrid of tall & low
Mixed Woodlands	6	77	107		54	Tall (Grassy)	
Savanna	7	54	107			Low (Grassy)	
Grassland	8	54	107			Low (Grassy)	
Dense Shrub	9	54	107			Low (Shrubby)	
Open Shrub	10	54	107			Low (Shrubby)	
Tundra	11	24	107		54	Flat	Snow, Dark Patches
Desert / Barren	12	24	54			Flat	
Unknown (Water)	13	145	250			Flat	Water, Ocean

Anthrome	no.	Buildings	Domesticated Features	Remarks
Dense Settlement	1	High to Mid Rise	All	
Village	2	Mid to Low Rise	Half	
Cropland	3	Low Rise (Sparse)	Half	
Rangeland	4	Low Rise (Sparse)	Low	
Seminatural	5	One to None	Minimal	
Wildland	6	None	None	
Unknown (Water)	7	None	None	Probably water

Figure 15: Table describing colour ranges for various biomes, and building heights and quantity of domesticated features for various anthromes.

I also made a tile today with vague 'urban' stylistics. I feel that it has an interesting balance between looking like an outdoor cobblestone appearance and a fancy indoor carpet

appearance. It also reminded me that these levels don't necessarily have to all be 'outdoors', especially in the case of more densely populated areas.

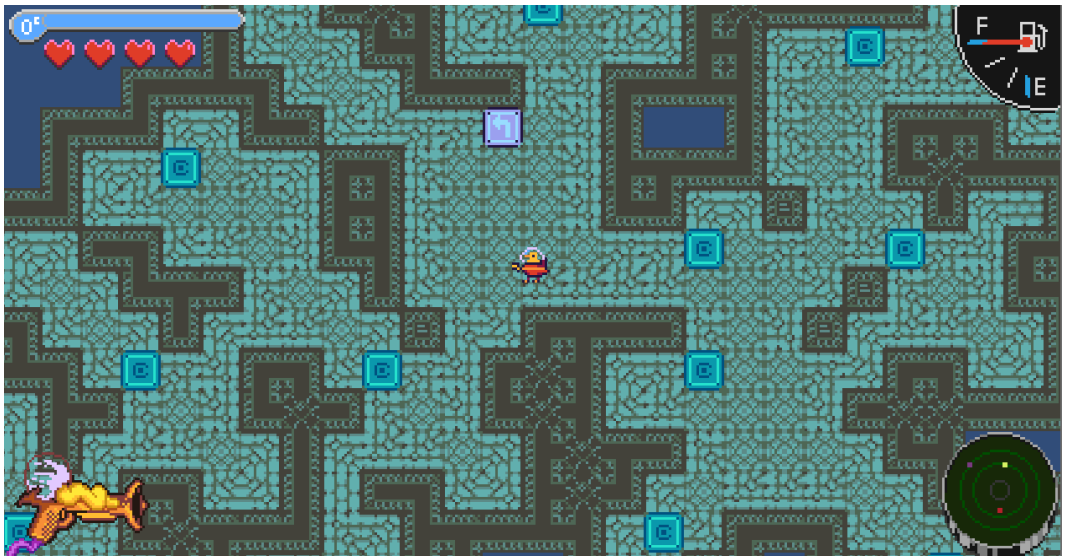

Figure 16: Urban level layout with new tileset

2020-02-10: [Proto2] Trees and Grass

Figure 17: Palm tree and tall grass

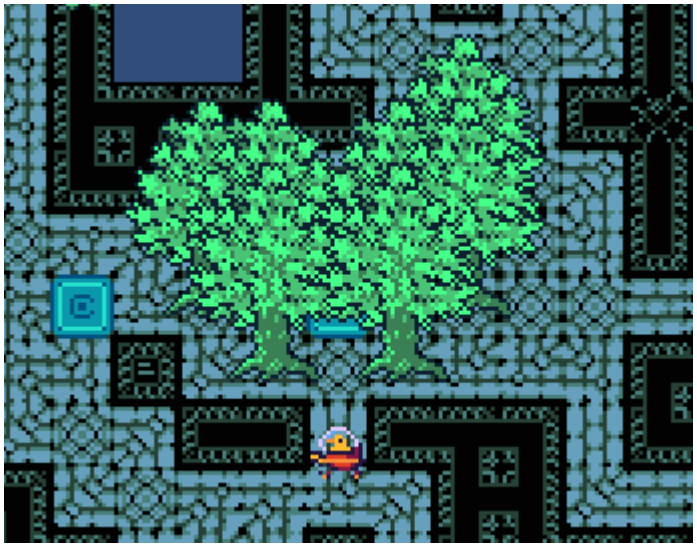

Figure 18: Pine / Oak trees

Sharing some additional new drawings. An idea: pink elements as 'fruit' / healing item from plants. Removeable, spawns on only some of the plants, hits the ground when a tree is struck.

2020-02-13: [Proto2] Back-End Adjustments and Generation Work

Yesterday I realized that the recolouring extension that I currently use only works for colours from the camera's perspective, so I had to go back and find a new method to replace it. I ended up selecting the Color Palette Swapper Shader by Nost Games from the Unity Asset Store. It uses shaders with embedded Texture objects representing the one-for-one original and swapped palettes, which was very easy to manipulate by means of creating and injecting new palettes to the 'swapped' palette texture.

On the generation side, I've done some work to create scripts that automatically generate pseudo values and LevelData objects so that I can test vast data fields without having to create data objects manually. One thought that has crossed my mind is instead of adjusting the hue uniformly for all plants, I could have them vary by a slight margin. I am currently distinguishing their 'seasonal' appearance based on the current temperature, although I imagine some sort of pseudo-'season' feature would be more effective.

2020-02-24: [Proto2] Artistic Ideas for Context

I've been working largely on graphical content for the project. It is slow paced, but I've noticed that when elements are rushed they really don't have the same *feeling* that I'm trying to accomplish with the project. But the project is coming along. I have biome level generation working, and now I'm in the process of applying rules to which elements are generated, under what conditions certain colours are more prominent, and the like. I'm not

sure how many of these considerations will make it to product but every time I make an application, even small, they really do feel impactful. And as I work, more ideas are coming to mind. Here's a few:

- I am currently using a colourization script that generates a random palette for the level's floor and wall tiles.
    - Some of the colour palettes that are being produced remind me of nighttime colourations, sunrises and sunsets, and other 'moods' that are evoked by time, weather, seasonal conditions. I am a little torn between maintaining the randomness in the colouration and potentially having time, weather, etc. inform the colour palette chosen from defined ranges associated with them.
    - Sometimes the random palettes can be really interesting, if a bit unrealistic. Other times, the palettes can look very unappealing. This may have to do with the 'inversion' of colours randomly applied by the algorithm, but I feel that I need to determine exactly where the colourations go wrong and look into fixing it to be 'nice' more often. Perhaps I'll reduce the likelihood of the inversion effect to have it as a 'surprise'.
    - I just added a script to handle background colours. I have a nice effect with averaging the colours of the tiles that has significantly improved my impression of the colour schemes. Still more experiments and tests to do...
- The foliage selection…
    - Some regions of the real world have very unique varieties of flora and fauna that characterize the area. It would probably take a deal of research to make accurate representations of different regions, but as an expansion concept it's a very appealing touch that could incorporate GPS or altitude. For now I'm going to settle with a vaguely representative batch and go from there.
    - A lot of the trees seem a bit too large, so I'll have to experiment with toning down the size.
- Water regions…
    - I currently don't have a process of distinguishing these because of a lack of data from the original dataset… maybe there's an ArcGIS table out there that distinguishes different bodies of water, but it's a low priority when I have no expectation for players to go swimming with the game.

2020-03-02: [Proto2] Advisory Meeting Notes, Mobile, and the Weather

- Toronto is a city, so having urban anthromes for the sake of the final presentation is important. However, representing areas such as High Park could be problematic because of the resolution of the source ArcGIS table. I need to make sure that I note the resolution of these maps (& cite their sources) in order to justify why these

spaces are misrepresented, while pointing to future developments to improve the resolution of these lookup tables.

- On a related note, there seems to be a delay caused by the ArcGIS lookup algorithm… the time might impact whether the data is reported accurately on mobile. I will need to test and bugfix accordingly, and ideally find a more optimized solution of getting anthrome and biome data.
- Otherwise, mobile has been tested and it works fantastic. If I use a similar UI technique between what is used in the generation room and the menu/camera rooms, I should be able to have something that works on many resolutions very effectively.

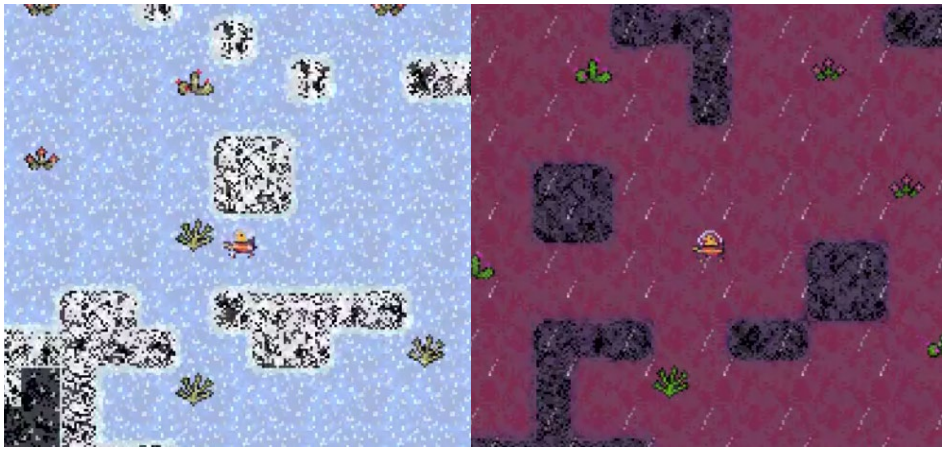

Figure 19: Snow and rain weather conditions

- Today, I implemented the weather. I used graphics that I had previously drawn up and pulled together fog, clouds, rain, snow, and sand. Will probably round them off tonight or tomorrow. Aiming to do anthromes this week as well.
- The plan for anthromes is to have anthrome-specific rooms generate at a frequency that is dependent on their density in ratio to the number of biome rooms generated (as far as the algorithm goes)
- My advisors noted my rationale for level design is worth noting. For the desert, I used broad open spaces for particular rooms to evoke the expansiveness, with the occasional niches where shrubs imaginarially are more abundant due to shade and moisture levels. For the grasslands, I tried to imagine rolling spaces with occasional lighter patches.
- On the fence as far as whether I should include a “random level” button for the floor demo. Could be fun, but could also detract from the photo-specific spaces.
- Regarding the exhibition, I will probably endeavour to make a web tool that works in conjunction with a maps API to pick a geoposition, then creates a injected photo accordingly. This way, I can demonstrate the tool to people interested in picking out their own locations for levels to generate.

- I think having the finance table will be an interesting mix of fun and controversy with poorer locations having less ‘loot coins’ than wealthier locations, and what that could possibly mean to the future of mobile experiences (reflecting on proportional wages to local minimum wage models for example)
- Still need to work on colouration. Things are still moving nicely but there is still lots to do….

2020-03-04: [Proto2] Bug Fixing the Tile Colour Scripts

Today’s task was to review the colour script for the tiles and see what I could do to improve its appearance in-game. At first, I thought I may have forgotten to include colour value spacing in the algorithm (as there was a related snippet of code I left out when I first created the colour generation algorithm in GML code), but it turns out that I made an error!

The algorithm has methods for creating hue, saturation, and value elements for individual colours. ‘Hues’ can loop around the colour spectrum, ‘values’ determine the brightness of a colour and must be spaced out enough to be distinguishable, and ‘saturation’ can be more or less any number because of how the value is handled. With this in mind, my algorithm creates an initial displacement from zero for each colour element, then offsets the values of the elements by a consistent amount for each colour in the palette. This results in a series of colours moving from dark to light, one hue to another, with a scaling saturation value. Further, the algorithm can reverse the change in the elements from being incremental to decremental, thus allowing for inverted colour palettes as well.

The bug was causing the colour elements to be miscalculated, resulting in some colours that were very close together or identical. Having now corrected this, I am significantly happier with the available colours for the tiles. In fact, I am considering how I might use the inverted value palettes to represent ‘night’ and the regular value palettes to represent ‘day’, and aligning the background colour with this as well.

I intend to test out a package called CoordinateSharp to see if it can accurately give me sunrise and sunset times so that I can determine this for many different locations in the world.

2020-03-06: [Proto2] ArcGIS Headaches

Today and yesterday, I was working to reduce the wait time involved in the ArcGIS table search and resolve an issue where my mobile version would report unusual or unknown

values for a particular location's biome or anthrome. Turns out I had a bit of an oversight in terms of my expectations of the data!

Figure 20: Black and white rasterized versions of ASCII ArcGIS Tables for Biomes (left) and Anthromes (right)

Figure 21: Example of discrepancies between ASCII ArcGis Tables for Biomes (left) and Anthromes (right)

The graphics in *figure 54* show raster versions of the ASCII tables I use to get the geospatial data for biomes and anthromes. Black regions have values, while white regions do not (lakes, oceans, ice, etc). I wrote a script to export these images and indicate where on the map the GPS position I collect is being located in the table. It turns out that some locations reported in white in one table were reported as black in another; since Toronto is right next to Lake Ontario, this means there is a likely chance that data collected near the water will report as white instead of black due to the resolution of these maps (both approx. a 4000 x 2000 grid). I later found out by reading a journal documenting research involved in the creation of the original data that "Pixels missing from the original dataset, constrained by a relatively restricted land mask (mostly missing some island and coastal areas), were filled using a nearest neighbour algorithm" (Ellis et al. 2010), which cleared up why there was a discrepancy in the data (see *figure 55*).

Since I will be presenting this prototype in mobile format and desktop format at a Toronto exhibition, it was important to me that I have terrestrial reportings for these regions. My solution was to 'blur' the report: when a white region is detected, the script checks the surrounding values for a black region and reports it instead. This way, any coastal areas will match the neighbouring biome and anthrome. This also gave me the means of identifying 'coastal' regions in the contextual metadata whenever this method was triggered as a bit of a bonus.

2020-03-07: [Proto2] Improving PhotoStego Methods

Today's goal was to integrate new data from CoordinateSharp's Celestial reporting as well as some unused data reported from the weather API. I ended up doing a large amount of refactoring in the process, as the old code had many sections with repeated string values, especially for the encoding / decoding of steganography. Essentially, the code now references dictionary entries to determine the data type, encoding label, and so forth required in the steganography process; fill in an entry, update the photoData object, and the bulk of the work is done. Handy!

The celestial data that can be retrieved through CoordinateSharp has some interesting data points, including eclipse times, sunrise, sunset, whether the moon and sun are above the horizon, and even zodiac signs. The documentation wisely notes that not all sunrise and sunset values are available[5]. For example, in the Antarctic there are sometimes 24-hour days in the summer where the sun does not set. To keep things simple before exhibition, I've decided to just go with provided boolean checks to determine whether the sun and moon are above the horizon, as I do not wish to confirm how my algorithm and PCG code handles NULL values and related complications at this time. Maybe if I feel inspired later! In any case, I now have access to a larger breadth of data than before and the work that I have done should simplify the process of working on the UI and the Injector, as the data is much easier to reference now than before.

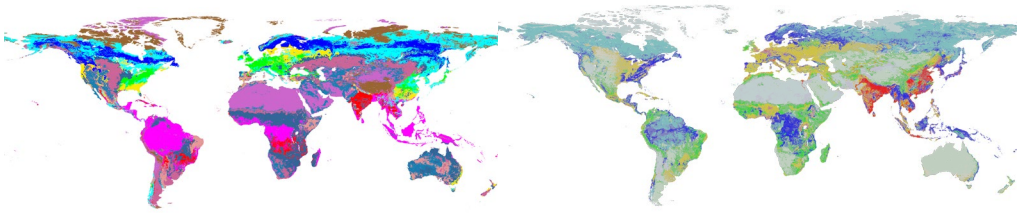

Figure 22: Rasterized versions of ASCII ArcGIS Tables for Biomes (left) and Anthromes (right), with regions differentiated in color (created from the ASCII-format dataset by Ellis et al. 2010)

Later in the day, I ended up writing a method to dump the biome and anthrome data in the form of an image instead of using a text file. I knew Unity read from textures much faster than text (at least my implementation, anyways), and it was the simplest fix I could think of without getting into deep levels of computer science. The only real caveat was making sure that the textures are precisely loaded and *not compressed*. The default

[5] https://coordinatesharp.com/DeveloperGuide

maximum size for textures was not accommodating for the exported raster graphics, so I had to adjust it to the maximum for use.

Oh, before it slips my mind: the order in which Random values are called is important for PCG; changing the order will change when decisions are made. If we use a lineup for a gumball machine as an example, and every person in the lineup has a ticket number with their order in the line, then their numbers will not match if someone cuts ahead of them, and the gumball that each person after the cutter receives will be different than they would otherwise have received. I mention this because I need to remind myself to be careful about the order in time in which I call the UnityEngine.Random.value so that elements added in the future do not modify the manner in which levels are generated.

2020-03-11: [Proto2] UI Work

Figure 23: User Interface Sketches

I drew these UI mockups for the Main Menu and the Camera application. With the deadline for the thesis paper submission coming up, I felt that it was important to have these resolved soon, as the UI elements for these portions of the project remain very undeveloped and unpresentable.

I didn’t manage to get the elements integrated in time for inclusion in the paper, but I did finish drawing the elements in vector format along with the rest of the graphics I wish to complete my thesis with.

Here’s some updates on the game visuals, though! You may notice a strange purple portal near the starting area: this is the world exit! I have also added a few more varieties of deciduous trees to better reflect variations in plant life in the wild.

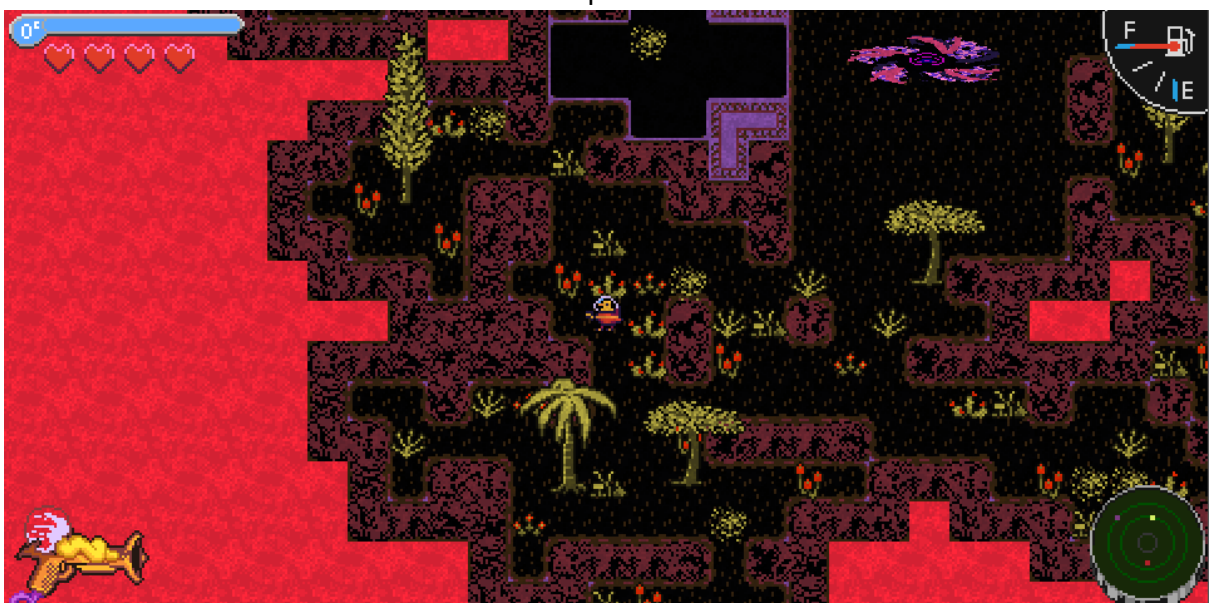

Figure 24: Shrublands biome, cloudy

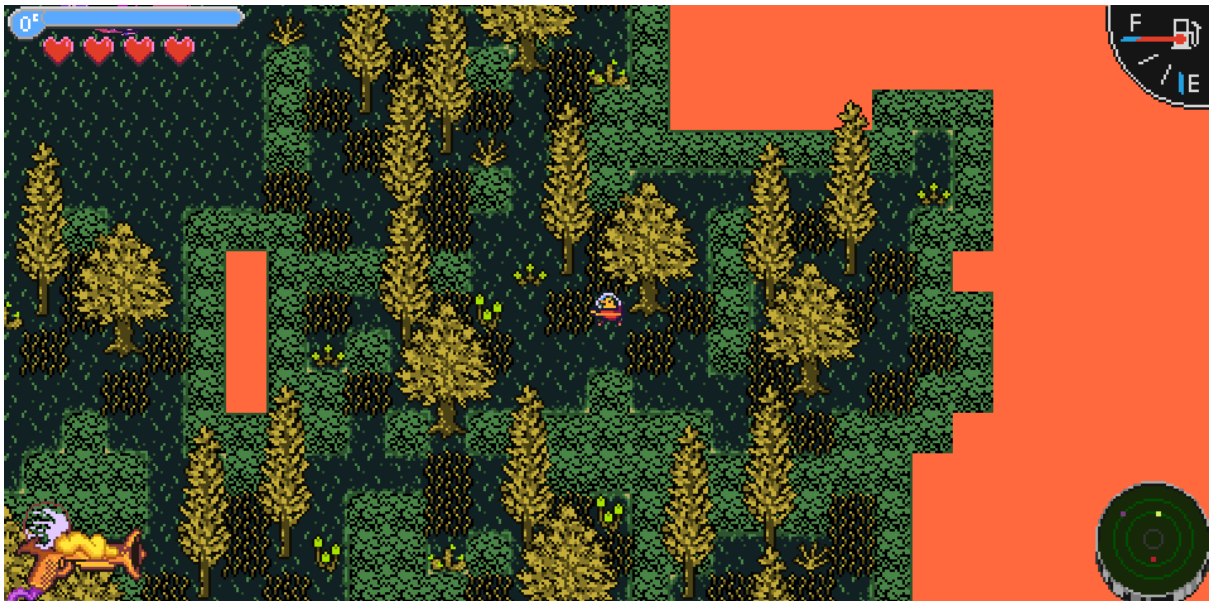

Figure 25: Evergreen biome, clear

2020-03-21: [Proto2] Multimodal Camera UI and Photo Injector

Figure 26: Camera Mode, Light Theme

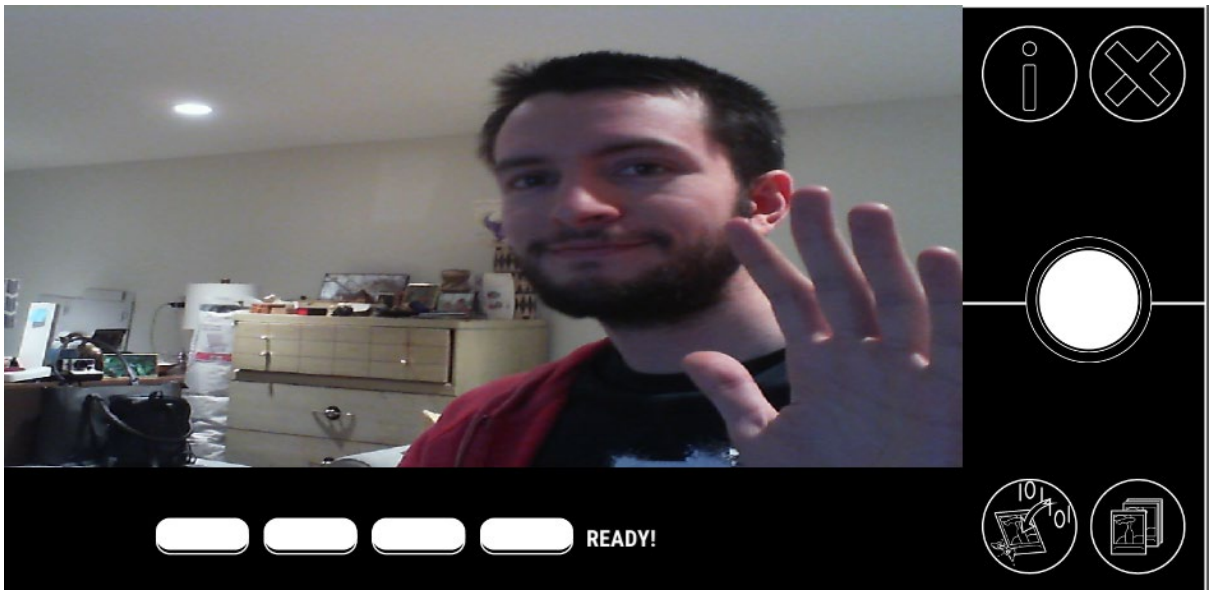

Figure 27: Camera Mode, Dark Theme

As promised, here is the updated graphical UI. I moved back to a two-dimensional display format for these and pulled together two themes… one black, and one colour. I have also added a new ‘indicator’ feature below the camera feed to indicate whether the data is ready for capture; red for waiting and green for ready.

I also wanted to include a photo injection feature that anyone, not just myself, could use to inject the multimodal format into their existing images. This required some significant refactoring of my older code to ensure that images could be sourced from both the webcam and externally, which effectively has made the whole system more robust in the process. I have also included two new assets in the project: TagLib Sharp, a standard C# library which allows for existing metadata tags (including location, time, and date) to be read from the images, as well as Simple File Browser, a freeware Unity extension that allows for file

navigation on a variety of targeted platforms. Essentially, the injection process begins by user navigation to the desired photo of choice, several checks to make sure that the required data (GPS & TIme) are available, the collection of relevant Secondary data, and finally the multimodal saving function that allows the image to be saved to disk with all the collected information. The only caveat to this process is that I currently do not have access to an API that allows me to get historical photo data from the time in which the images were captured, so the weather data that ends up injected into the images is the current weather, rather than the weather at the time the image was captured.

Figure 28: Gallery mode, with details for an image from Reykjavik, Iceland (image courtesy of Mikaela Manley)

In the above screenshots, we can see an image from Australia successfully imported by the Injection method. The process also automatically resizes the image before saving it to PNG format in order to reduce the size of the file and speed up load times. With this new tool, it should be much easier to examine and test real-world locations and make the project more accessible to potential users who already have a collection of photos..

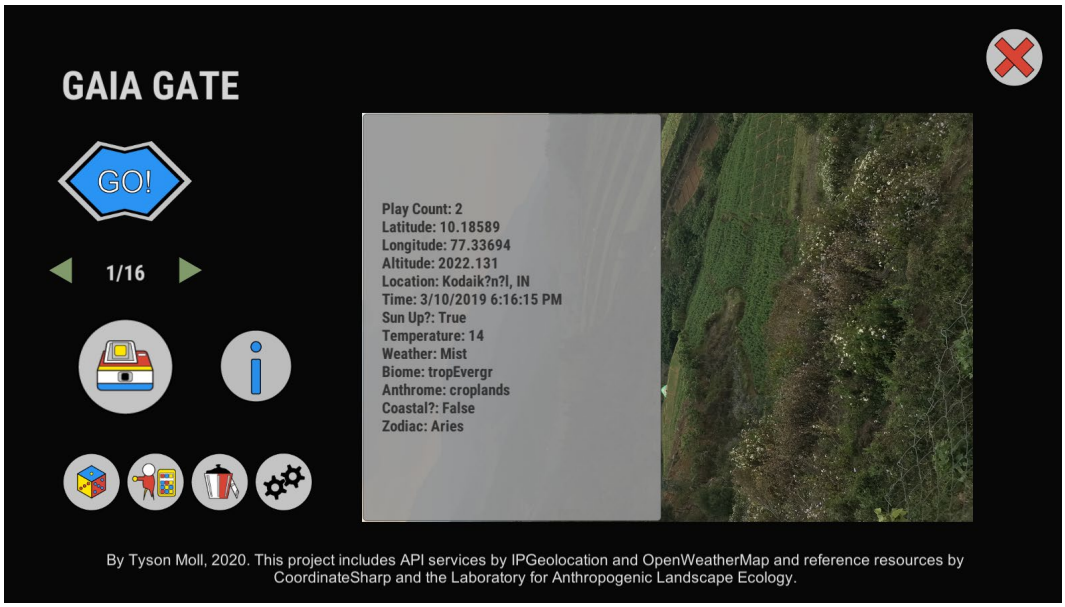

Figure 18: Main Menu of Gaia Gate, with level data created from croplands in Kodaikanal, India (image courtesy of Shikhar Juyal)

2020-03-24: [Proto2] Ease-of-Use Features for Mobile and Desktop

Both the Desktop and Mobile versions of the game have their own respective limiting factors. Today, I looked into investigating means of resolving these issues.

For Desktop users, this currently comes in the form of a lack of means of accessing one's GPS position. Since I can't expect every computer user to have access to a GPS sensor (plus Unity's lack of support for desktop location) I decided to have the computer look up its location by its IP address. I installed a C# library by *ipgeolocation* (https://ipgeolocation.io/) that uses their API as a reference for converting the IP of the computer connecting to the service to latitude and longitude which worked out nicely for what I needed.

For Mobile users, there was a lack of accessible controls for people without an external controller (as pictured earlier). Wanting to simplify the experience, I added the Joystick Pack by Fenerax Studios extension asset for Unity, which provided an almost plug-and-play analog stick for movement.

I also added a pop-up system featuring alerts of any issues that may arise while using the application, some general prompts to guide them into capturing multimodal photos, and the capacity to delete any images or levels created by the application. These features appear to make the application much more accessible and hopefully will inform users how to deal with any errors that may arise during its use.

Figure 31: Using Mobile Controls (Tropical Deciduous Biome, Smokey Weather)

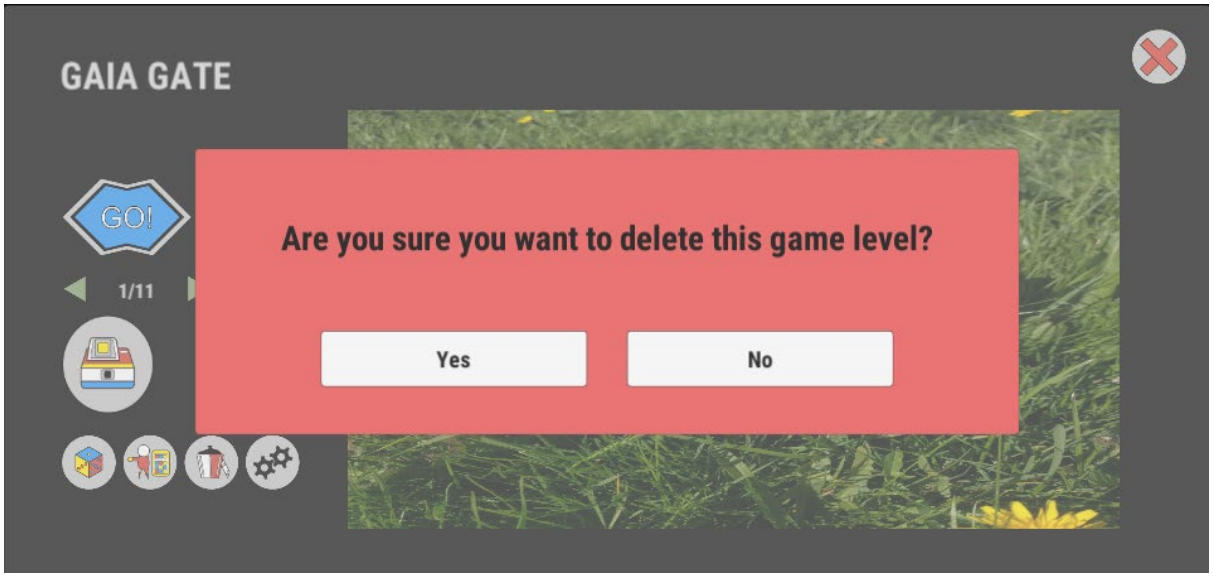

Figure 32: Deletion Dialogue

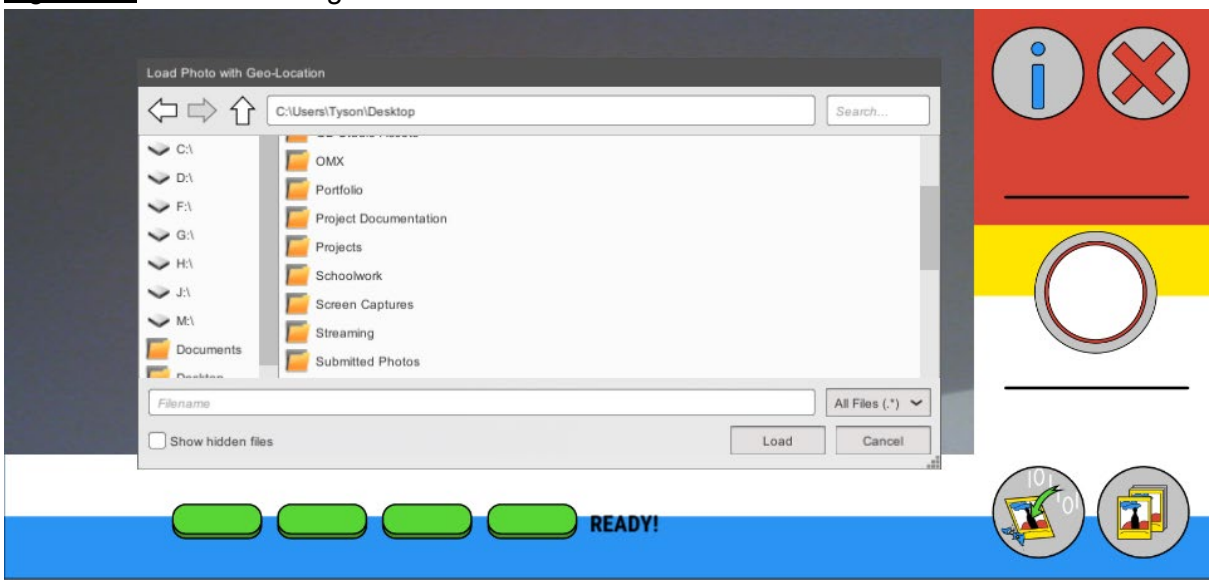

Figure 33: Image Importing

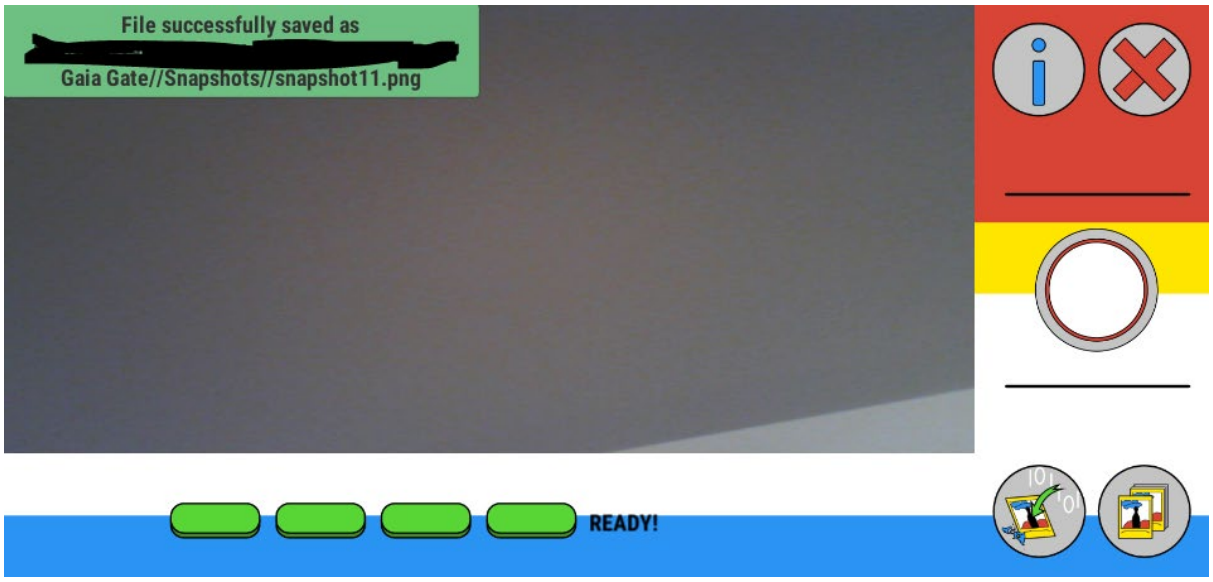

Figure 34: Pop-up Information

2020-03-30: [Proto2] Gameplay Elements and Functional UI

Figure 35: Croplands, Boreal Biome, with destructible posts

This week I focused on developing elements that had an impact on the gameplay of the project. My initial plan was to begin working on anthrome objects, but as I continued to work and integrate these elements I ended up adding features that made the game actually *feel* like a game, if ever so slightly.

Wanting to develop the croplands biome, I initially started by sketching out tiling patterns for soil and posts. Since the posts would likely block a player's movement, I decided that I would make them the game's first breakable element, alongside rocks. I implemented this in the game and improved the player pistol to make them interactive, then decided to create a pick-up 'fuel' element, which led me to 'activate' the UI mockup features that had been thus far still. The oxygen bar now depletes over time, the fuel gauge increases with each pickup, and the radar shows where all fuel items are located. The fuel pickups are randomly placed in a specific manner: first, a handful of the room blocks are elected to have a fuel pickup, then the pickup is randomly placed within one of the designated unoccupied tiles of the room. In a future build, I plan to have this method inform other gameplay element placements to ensure consistency across the procedurally generated space.

2020-04-20: [Proto2] Final Adjustments for Thesis Submission

Before making the final submission, I want to make sure I have accounted for the last of my changes!

I added level layouts for the remaining anthromes as well as some new anthrome objects: crops for the farms, lightposts for the urban areas, and some destructible 'ruins' for various anthrome spaces. At this stage, the anthromes finally feel like they are ready for inclusion! They resemble the human intervention that they represent and do so in a way that looks

good. I also finally added lightning to the thunderstorm weather condition, which makes for some interesting tension in the gameplay (despite ‘damage’ not being implemented at this stage!)

Figures 36-38: Several screenshots of generated levels. From top to bottom: Boreal with seminatural anthrome, desert at night, and Toronto.