PETS (working title)

High Concept

PETS (working title) is a first-person, 3D Tamagotchi-style *mobile* game where you get a random creature. You have to take care of this creature by exploring a cute tiny procedural island and foraging for food and resources while it follows you around. The creature requires daily care, as it will feel things like hunger or loneliness if you take too long to pick up the game and play. Caring for the creature will let it grow up and develop abilities, which can be used around the world to get new resources or open new areas on the map.

Must-haves and must-not-haves:

- Asynchronous gameplay - the world keeps going while you are not playing - your creature gets hungry, your house needs maintenance, etc.
- Creatures must be generated based on a seed (like a “DNA”). Every player will have a unique creature and this will be achieved with procedural generation.
- The player will start with one creature, but they can have up to 3 creatures. This might be a paid feature in the future (pay $ to unlock up to 2 more creature slots!). Game is completely functional and fun with only one creature, however.
- Upgradeable creatures and home base.
    - Creatures will have a “skill tree” which the player will be able to slowly unlock to upgrade them and evolve them.
- Procedural world - Maybe your island has deserts and mountains, maybe it has plains and jungles, it all depends on the seed generated.
- The world is not modifiable like in minecraft. The world, albeit procedural, will be generated with a variety of prefab pieces of the environment, which for the most part will be unchangeable (you won’t be able to dig, mine, but you will be able to change small pre-set parts of the world, like fixing a bridge, or destroying a boulder blocking a path). This is to add focus to the pet part of the game.
- The creature does not die. It will be debuffed and weakened if you don’t feed it, but it will never perish. You will always be able to bring it back to health.
    - Instead of dying, the creature will just grow more slowly or not grow at all. So a creature would take double the time to mature if they are not well taken care of, and if they are constantly starving, it would not mature at all until the player remedies that.
- Creatures will drop resources when they are pet, resources that can be used to maybe craft items or sold into gold that can be used to upgrade their base.
    - This system needs to be carefully programmed to follow the operant conditioning curves (variable ratio intervals).

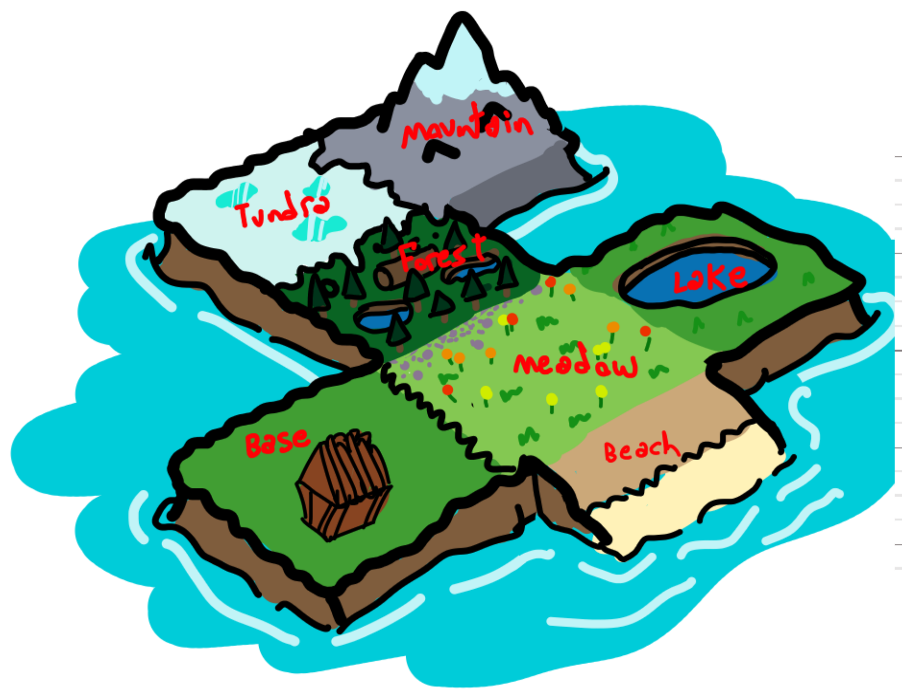
Moodboard

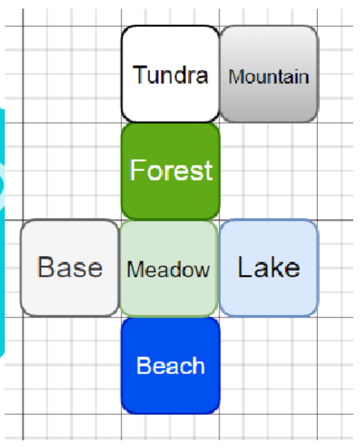
https://docs.google.com/presentation/d/1MaLqKPpOteCf0LrzAF75DnZeQ0lMzKUHnsvft5RG4oc/present#slide=id.p
If no music plays, hover your mouse on the slide until the slide controls on the bottom left appear, then try to change the slide (even though there is just one)

Basics

- The initial release is on Android, but if the game does well it will be released on iOS as well. Though iOS should not be the focus currently.
- While gameplay ideally will focus on the player having 1 creature, we will let them have up to 3 total creatures, so the system should be able to handle multiple creatures
    - I'm assuming the solution is to have each creature's data be kept on a class or scriptable object.

World Generation





You play on an island composed of different "chunks", each containing a random biome based on your game's seed. During gameplay, only one chunk is loaded at a time, so the play area is always constrained to the chunk you are in. The player's base is located in a special biome, the "base biome", containing the player's house, farm, etc.
Every island will **always** have a meadow biome which is connected directly to the base biome, and a beach biome connected directly to the meadow. This is so we can have a tutorial on the

meadow, which will always be the first biome the player visits, and they are allowed to go to the beach, too. From there, after the tutorial is done on the meadow and on the beach, the player can proceed to the rest of the island.

This is a concept of a meadow chunk (with some added concept of different creatures). For more information about chunks, see the next section (Chunks and biomes).

Connectivity Rules

Please refer to this visual guide that allows you to easily see the connectivity rules.
Biomes can only connect to certain other biomes. Other than the connectivity rules, there are a few hard rules to be followed:

- Biomes must not repeat.
- Every island will have a “base” biome, which is a special biome that generates the house on which the player is based.
- The meadow biome will always be generated adjacent to the base biome, in any of the four cardinal directions.
- The beach biome will always be generated adjacent to the meadow biome, in a random direction (that isn’t the base biome, obviously)
    - No other chunk can be generated across from the beach biome.
    - To make this possible, as soon as the beach is generated, all possible spaces across from the beach (on the opposite side of the meadow) should be flagged so they can’t have chunks generated there. This is to prevent having biomes on the other side of the beach.

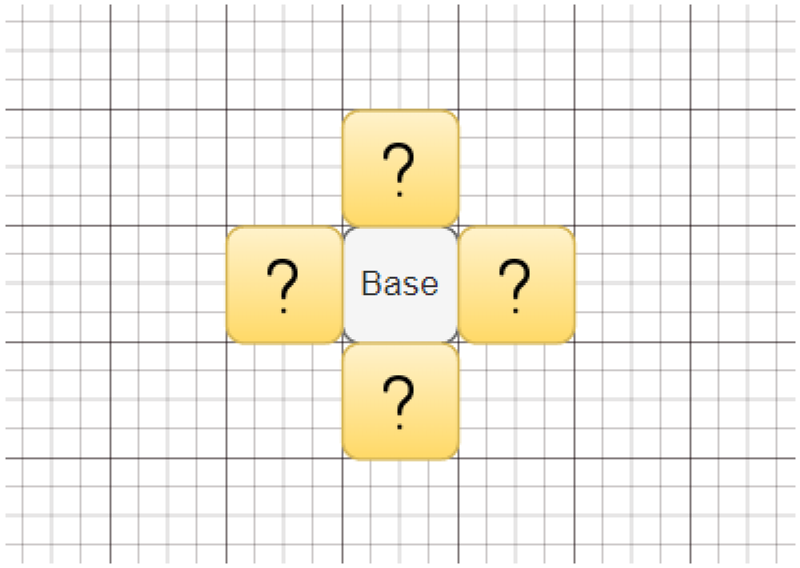
Meadow:
Connects to **Base**, **Desert**, **Lake**, **Swamp**, **Forest**, **Tundra,** and **Beach**
Forest:
Connects to **Meadow**, **Tundra**, **Jungle**, **Swamp**, and **Lake**

Jungle:
Connects to **Lake**, **Swamp**, and **Forest**
Desert:
Connects to **Meadow**, **Lake**, and **Cave**
Swamp:
Connects to **Meadow**, **Forest**, **Cave**, **Lake**, and **Jungle**
Lake:
Connects to **Forest**, **Desert**, **Jungle**, **Meadow**, and **Swamp**
Tundra:
Connects to **Forest**, **Cave**, **Meadow**, and **Mountain**
Cave:
Connects to **Tundra**, **Desert**, **Swamp**, and **Mountain**
Mountain:
Connects to **Cave**, **Desert**, and **Tundra**

Island Iterations

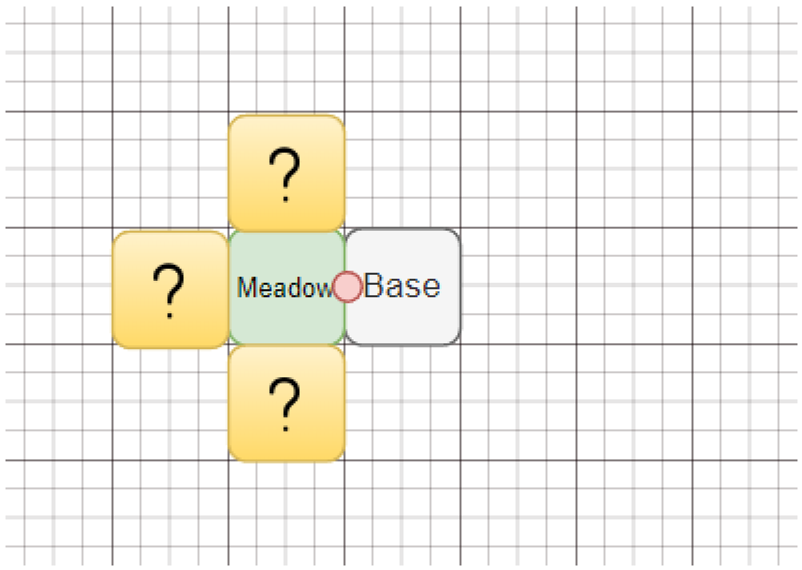
The following steps are instructions on the iterations needed to be done in order to generate an island with biomes.

1. Place the base biome chunk in the middle of a grid. Add the spaces adjacent to it to the pool of potential spaces to choose from.

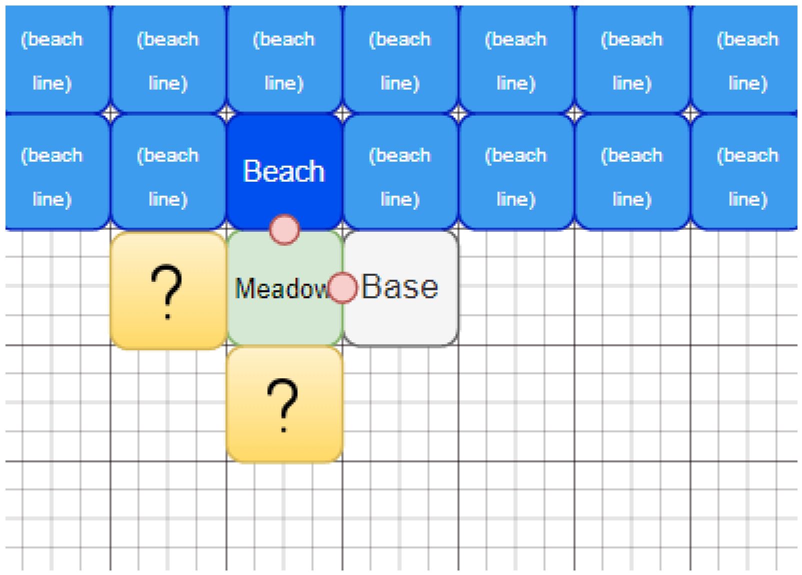


2. Add the meadow to a random space from the four potential spaces you had. Remove the remaining potential spaces around the base (so there's no chance an additional chunk can generate around the base - the base should be a dead end that only connects

to the meadow). Connect the meadow to the base so there's a passage.

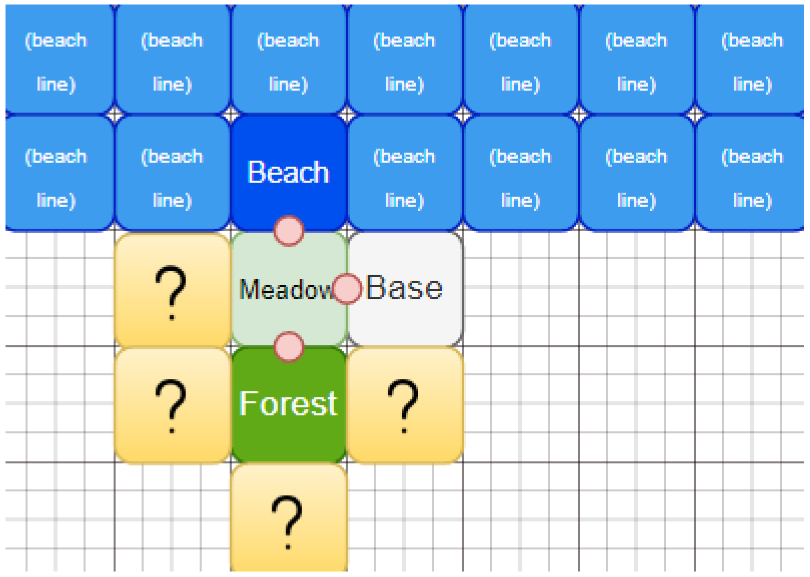


3. Add a beach biome to a random possible space. Add a "beach line" to the possible spaces across from the meadow - if the meadow is south of the beach, add a beach line to the north of the meadow (like in the image below). Connect the beach to the meadow.

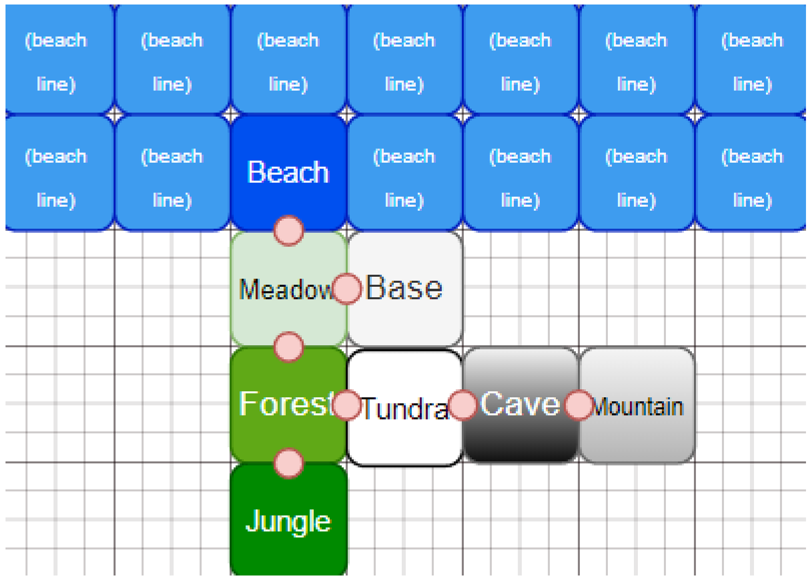


4. From now on, generate a random, valid (as per the connectivity rules) chunk from the pool of possible spaces. Connect that chunk to the chunk it was generated from, and

add the possible spaces around it to the pool of possible spaces.

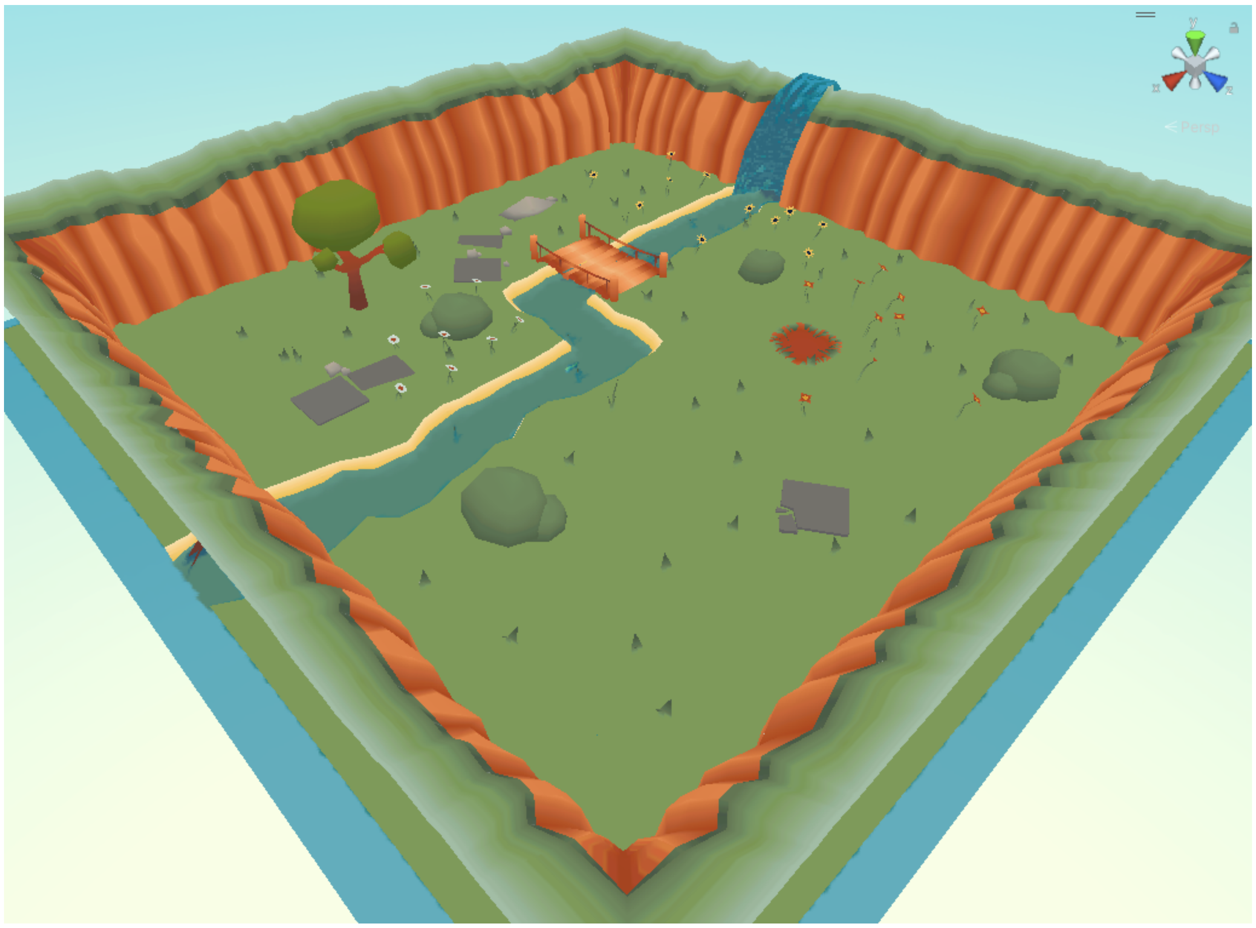


5. Do this until there are no more possible chunks to be placed (due to exhausting possibilities because there are no more biomes that connect to the currently available biomes) or until you have no more than 8 total chunks, including the base, meadow, and beach. There are 11 biomes in total, so every player will be missing a few biomes, making their world unique, still keeping their world large - 8 biomes should be big enough for the player to explore and play around with the game mechanics, while still giving a reason for players to pay to generate a new world if they desire.



This video shows the system working step by step:
https://www.youtube.com/watch?v=CstQ5OsiOvI

Chunks & Biomes

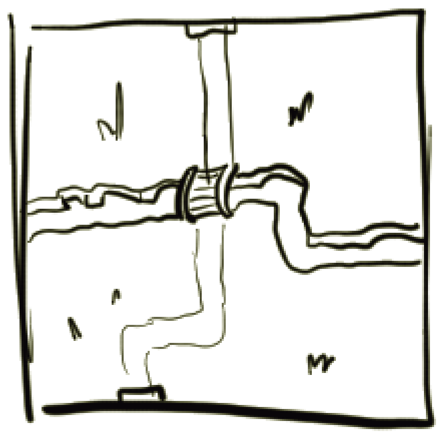

Aerial view of a first version of a meadow chunk.

Chunks will have different elements and layouts depending on the biome. Chunks of the same biome will always generate with similar layout and features, but will still be randomly generated.

Biomes have resources that can be harvested, and eventually regrow or reappear, such as trees, minerals, sources of mud, bones, etc. So a player that has a jungle biome and a forest biome will be able to collect lots of wood before waiting for it to regrow, etc.

Each biome will have certain essential features that set it apart from other biomes.

Meadow



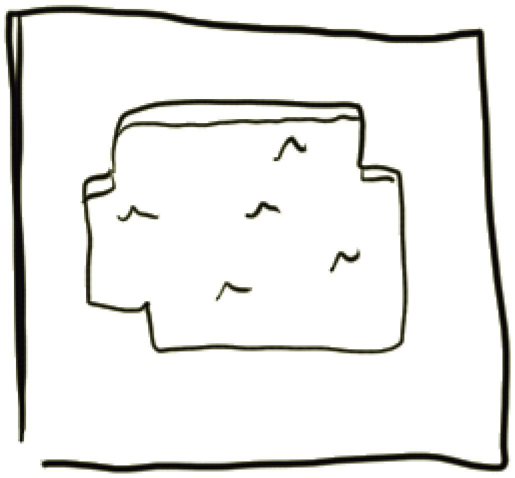
Has a clear, straight path connecting to all passages. Also has a river, containing a broken bridge, which will need to be repaired in order to proceed to the rest of the island, and will act as the tutorial of the game (for this reason, during world generation, meadows are always the first biome the player visits when leaving their base biome).
Contains lots of flowers, potentially some old stone ruins, some mud, a few trees, and occasionally small bones. Since all players will have this biome, it should have a little bit of everything, to make it impossible to get players soft-locked in case they need a resource to progress.

| Wood | Stone | Mud | Calcium |

Beach

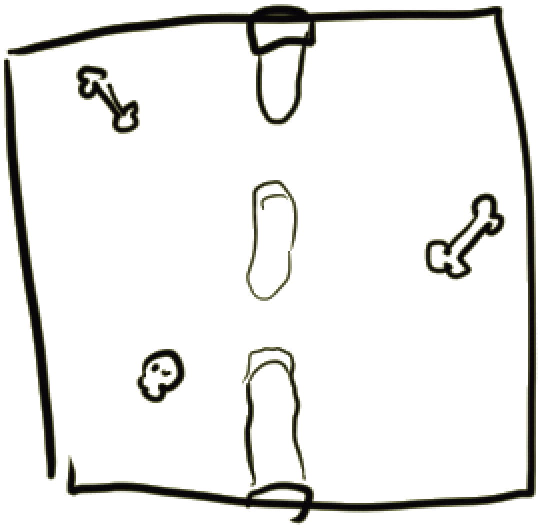
A biome that is half land, half water. The land half will always “face” the exit where the meadow inevitably is.
Contains seashells, and driftwood.
| Calcium | Wood |

Lake



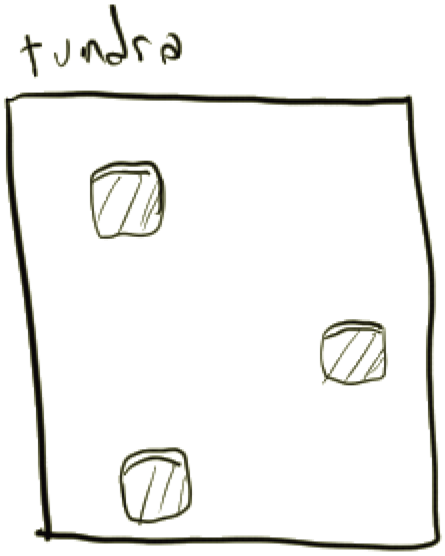
This biome has a major pond as its centerpiece.
Contains a lot of mud and some trees.
| Mud ++ | Wood+ |

Desert



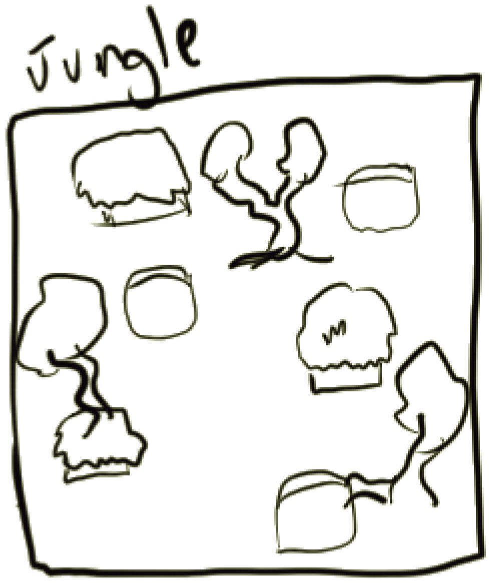
The desert biome has a faded dusty path going along its entrances.
Has tons of bones and a little bit of coal and stone
| Calcium +++ | Coal | Stone |

This biome has some small puddles of water that slow down the player as they move.
Has some bones, tons of mud, and some trees.
| Calcium+ | Mud +++ | Wood+ |

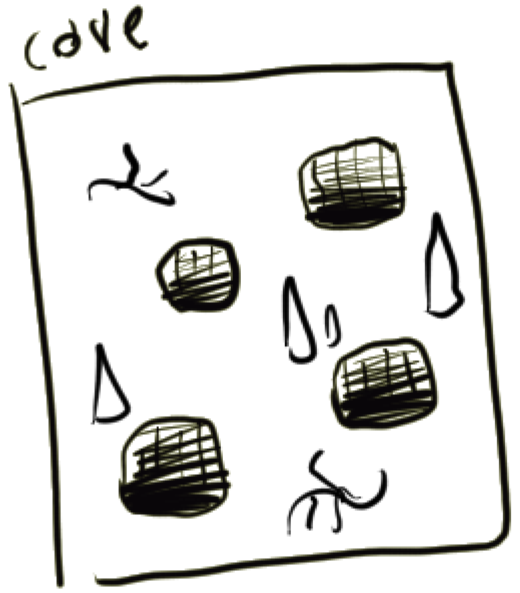


The tundra biome has frozen puddles spread throughout, making the player slide when they walk on top of them.
Has a lot of ice and some stone.
| Ice +++ | Stone+ |

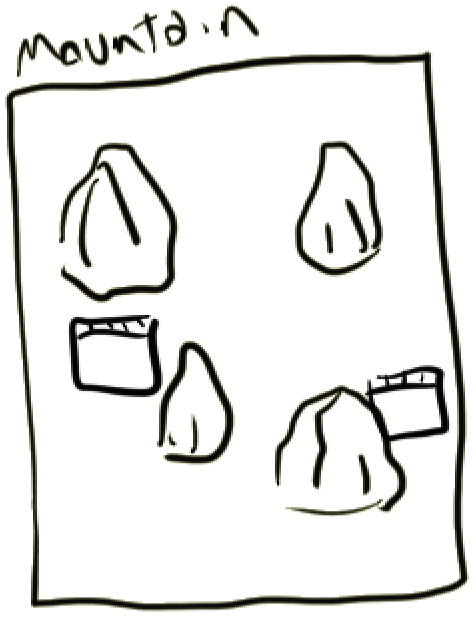
The forest has trees and small impassable rock formations sprinkled throughout.
It has tons of wood and some stone around.
| Wood +++ | Stone+ |



The jungle has trees and impassable rock formations, as well as puddles that slow the player down.
Has a lot of wood and some mud.
| Wood ++ | Mud+ |



The cave has holes, cracks, and stalagmites spread around its floor.
Has tons of coal and some stone.
| Coal +++ | Stone+ |



The mountain has huge impassable rock formations and some holes.
Has tons of stone and some coal and a little bit of ice.
| Stone +++ | Coal+ | Ice

Milestones

1.0 Foundational Milestones

This milestone will barely have any gameplay. Its objective is to lay the systemic foundations so game mechanics can function.

1.1 Basic System & Debugging: This milestone lays the basic operational foundations of the project - a repository, the programming architecture to be used, and a debug menu.

- GitHub must be set up so other people on the team can push changes and assets.
  - GitHub must be set up properly, with a proper gitignore.
- The project must use scriptable object architecture - if you are unfamiliar with it, I ask you to please watch the video linked.
  - There is a free unity asset on the asset store that has the entire system ready, so we don't need to implement it from scratch in our project, we just need to adopt this architecture.
- There must be a debug system inside the game. It should contain three screens:
  - **Cheat menu:** A menu that contains buttons for cheating.
    - For now, this can be empty.
  - **Debug log:** A menu that has everything that you print using Debug.Log or Print().
  - **Debug menu:** A menu that will contain general useful information about the game, like your position, the seed, etc
  - Items in orange on this list are things that should be added to this debug menu.
  - You open the debug menu by tapping an invisible button in the top right of the screen.

1.2 Basic Saving and Loading: This milestone ensures a foundational implementation of a save system.

- The game must have an **untamperable** save and load system
  - Save system must use good encryption and not things that are easy to tamper like playerprefs.
- The game will need to use Unity's Authentication so we can avoid users simply deleting their data and redownloading the game to "reroll" their creature - they must stay with the creature they get.
- The save system must use Unity's cloud saving as well, so users can play the game from multiple devices.
- Save file should be able to contain different types of data.
  - Creatures: the player must be able to save the state, seed, level, and other stats of their creature or creatures (so this should be dynamic and be able to store N amount of creatures)

        - The creatures “DNA” (seed) which will be used to know the creature’s features.
        - The creatures level, hunger level, age, and other stats
        - Creature skill tree progression - how far have you progressed through your creatures tech tree
    - Procedural world progression status - if you have fixed the bridge to cross the river, the game should save this information
        - The island is procedural, so what if your island does not have a river with a bridge, but it has a valley with a boulder blocking it? How could we differentiate while also dynamically being able to save progression? We should discuss this in order to decide.
    - Other values such as your home base’s level, etc.
- Cheat menu should have a force save, force load button

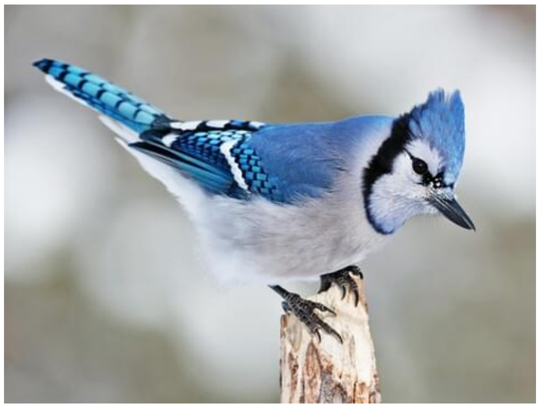
1.3 Basic Asynchronous gameplay: This milestone lays the foundation for time-based events in the game.

- Game must have an **untamperable** clock that works like animal crossing. Wherever the player is, if it’s 8pm for them, it is 8pm in the game. Players should not be able to change the game time by changing their phone’s clock.
    - I do not think the game will have an internal clock, however. The time will only affect the sun/moon cycle, lightning, and perhaps when the creature sleeps
- Upon opening the game, the game must check the amount of real-world time that has passed since the game was last opened so that in the future we can apply any changes necessary:
    - Example: If the creature takes 5 hours to get hungry, and you fed them 7 hours ago and opened the game, they should be very hungry when you open the game
    - Every few hours or so your house will spawn something “dirty” that the player can clean up. Dirt on the floor, or a cobweb on the ceiling, etc. We need to know how long it has been since you opened the game last in order to spawn an appropriate amount of dirt in the house.
    - A reminder that these things like creature hunger or house dirt should not be implemented right now - these are just examples. But we should have a system that debug prints “It has been X hours since you last opened the game” when you start the game.
- Cheat menu should have a button to force set time of day, force fast forward X hours

1.4 creature data: Ths milestone will ensure that we have proper data containers so we can add creature generation and functionality in the future.

- The creature must be procedurally generated. It could be anything from a pink chicken with a toucan’s beak to a green fish with butterfly wings. What defines the looks and behaviors of a creature will be stored in the data container stated here.
- This data can be kept in the form you find most efficient and functional. It could be a scriptable object, a custom class, or whatever method you prefer.

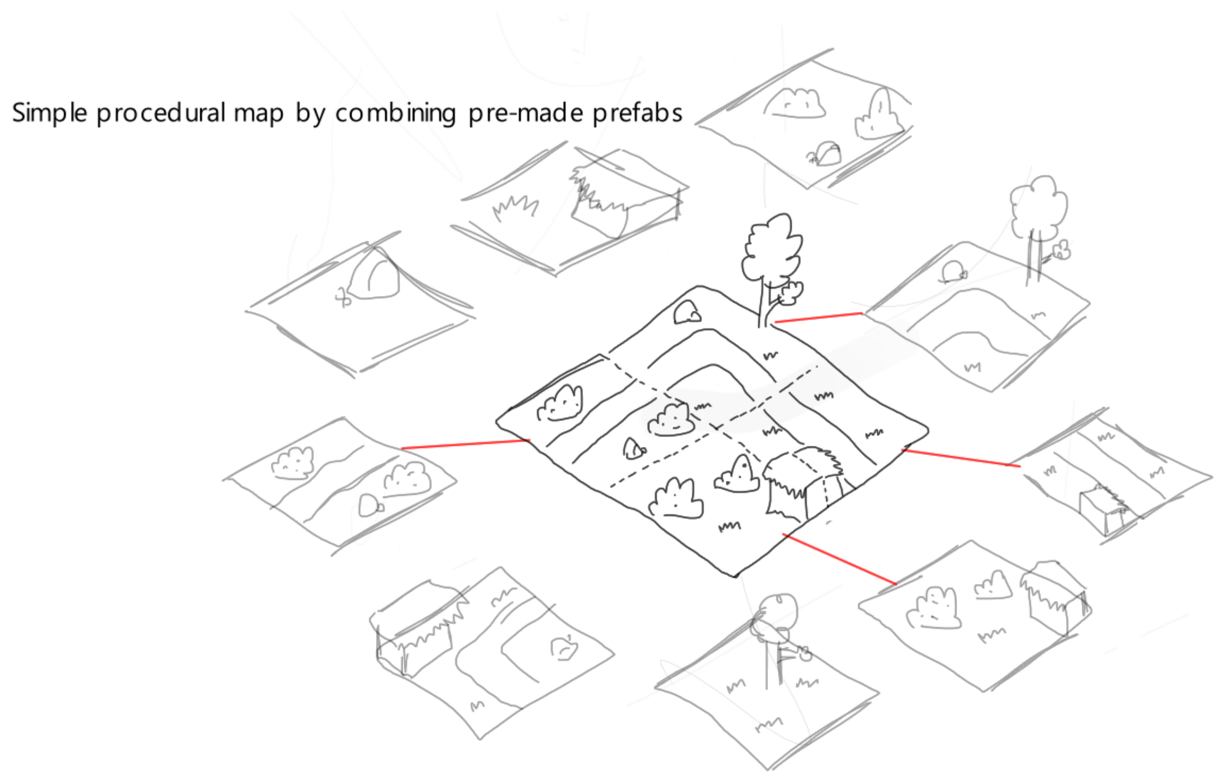
    - What is important is that whatever data container has these attributes below, we must be able to have **multiple** of them - so players could eventually have creature A, creature B, creature C, etc on the same save file.
- There are certain parameters that the creature will have. This task is not to create the full creature but to create the backend data container and its attributes
    - **DNA (the "seed")**
        - The DNA is the unique string of numbers that will define different parameters of the creature, such as what it looks like, its colors, how long it takes for it to reach adulthood, etc.
    - **General attributes**
        - Age: What is the age of this creature measured in days? [int, from 0 to unlimited]
        - Stage: What life stage is this creature on? (egg, newborn, infant, adolescent, adult) [enum]
        - Stage thresholds - at what age does this creature advance stages? Creatures will be unique so some creatures might reach adulthood at 30 days, some at 20, some at 50, etc. Creatures start as an egg, that will be hatched in a matter of minutes of gameplay, when the user completes the tutorial (walk around, get X item, bring it to the egg, move the egg to a warmer place, etc).
            - Age to reach infancy [int from 2 to 4]
            - Age to reach teenagehood [int from 5 to 30]
            - Age to reach adulthood [int from 10 to 50]
        - Max Stamina [int 0 - 100]
        - Stamina [float 0.0f - 100.0f]
        - Max Hunger [int 0 - 10]
        - Hunger drain rate [float from 0.0f to 3.0f]
        - Current Hunger [float 0 - max hunger]
        - Joy [float 0.0f - 100.0f]
        - Mood [int from -10 to 10]
    - **Archetype -** The type of animal that the creature is based on. It Will affect things like, how the creature evolves, the creature's "skeleton" (for animation purposes), and some neat features that the creature can have (bird-like can fly, fish-like can swim, primate-like can climb trees, etc). Every archetype will have a special ability that will unlock once the creature reaches adulthood
        - Reptilian-like - Cold blooded - able to withstand heat
        - Amphibian-like - Able to walk on land and water at moderate speeds
        - Bird-like - Able to glide or fly against strong winds
        - Fish-like - Flops slowly on land but has flippers that let it swim against currents in rivers
        - Cat-like - Able to climb walls
        - Dog-like - Able to jump large distances
        - Primate-like - Able to climb trees
        - Rodent-like - Small, able to go through small holes

        - Bat-like - Able to use echolocation to help you to see in caves
    - **Color scheme** - The creature will have different colors - a primary color, a secondary color, a tertiary color and a detail color. If you imagine a blue jay (bird), its base is white (primary), it has a blue coating on its back (secondary), black stripes (tertiary), and some parts of its feathers are light blue (detail).
    - 
    - Note that not all creatures will have all 4 colors, some will have less, even just 1, some will start will less and get more colors as they evolve, etc. The parameter that defines the colors a creature has is called a color scheme, which will be an enum. Please check paletton to understand visually what the below terms mean.
        - **Monochromatic**: the creature will have different colors on the same hue. Think light yellow, **yellow** and **dark yellow**.
        - **Adjacent**: the creature will have 2-3 colors on adjacent hues. Think **olive**, **blue** and **purple**
        - **Triad**: the creature will have 3 colors that form a triangle on the color wheel. Think **orange**, **green** and **blue**.
        - **Tetrad**: the creature will have 4 colors that form an X on the color wheel. Think **pink** and **orange**, **olive** and **green**.
        - **Contrasting**: the creature will have 2 colors that contrast each other on the color wheel. Think **orange** and **blue**.
        - **Colorful**: the creature will have different colors of multiple hues.
    - Aesthetic Features - The creature may or may not have aesthetic features. These will be affected by the secondary, tertiary and detail colors from their color scheme.
        - Skin type:
            - Naked, Short fur, long fur, Scaly, Spiky
        - Major feature type:
            - Stripes, waves, dots, half and half
        - Secondary feature type:
            - Stripes, waves, dots, half and half
        - Head:
            - Antlers, horns
        - Back:
            - Shell, wings
        - Tail:

            - None, Long, short, stubby, corkscrew
    - Traits - Positive qualities that the creature has.
        - I have to think about different positive qualities - more ideas will come as the game is more implemented.
    - Penalties - Negative traits that the creature has. Each creature will start with 2. The player can eventually remove them with a potion or something similar.
        - Hungry fella - Requires more food
        - Sleepyhead - Requires more sleep
        - Poopyhead - Poops more frequently
- For this milestone, we can make it so that when you start the game, you generate a purely random creature with purely random values - and we'll fine tune these in a future milestone.
    - The game should save and load the creature properly
    - The creature stats should show in the custom debug menu!

1.5 Basic world procedurality: This milestone sets the basic parameters in order to create a procedural world in the future.

- The seed generated for the creature will double as a seed that is used for the world generation
    - The world will not be as detailed "block by block" as Minecraft so this system does not need to be crazy complex.
    - There must be a cheat button to regenerate the seed.
    - The world will have a few biomes. In bold are biomes present in every world.
        - **Base:** This is a special biome. It is an environment that is always going to be similar, albeit with some simple detail changes. It will be a peaceful area with a small pond, grass, and trees.
        - **Beach:** Starting biome of the fish-like
        - **Meadow**: Starting biome of the bird-like
        - Desert: Starting biome of the amphibian-like
        - Forest: Starting biome of the cat-like
        - Lake: Starting biome of the amphibious-like
        - Tundra: Starting biome of the dog-like
        - Cave: Starting biome of the bat-like
        - Swamp: Starting biome of the rodent-like
        - Jungle: Starting biome of the primate-like
        - Mountain
    -
    - Each chunk will be represented visually in the game by randomizing and matching environment pre-sets and combining them together to form a walkable part of the world. Check the image below.



The combination of preset prefabs will generate a biome.

Reminder that this milestone is not to create the system that will generate the prefab map - but simply creating the parameter for the seed and keeping it somewhere efficient in the project.

2.0 Initial Visual Milestones

This milestone will make it so that the game will have some visual content and some basic functionality.

2.1 Basic creature procedural generation: This milestone ensures that the creature's DNA is randomly generated, and its stats and parameters are properly set

2.2 Basic creature functionality: This milestone ensures that the creature's basic game functionalities work, such as following you around, sleeping, needing to eat

2.3 Basic creature needs and interaction
TBD

3.0

TBD

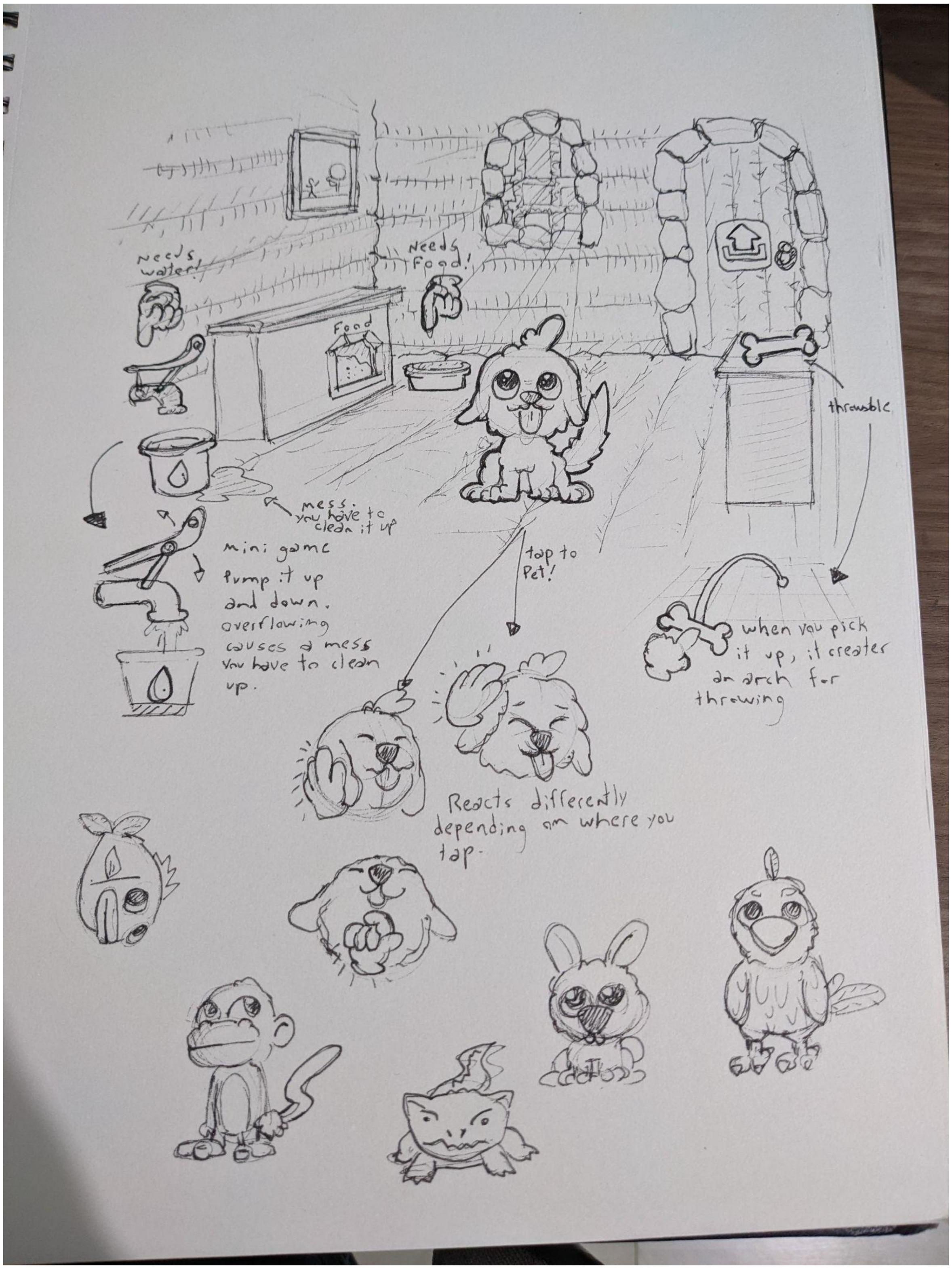
4.0

TBD

5.0

TBD

Gameplay Sketch


Needs water!
Needs Food!
Food
mess. you have to clean it up
mini game
Pump it up and down. overflowing causes a mess you have to clean up.
tap to Pet!
throwable
when you pick it up, it creates an arch for throwing
Reacts differently depending on where you tap.