Initial Vision

This document depicts my initial vision of Evolution Simulator.

# High Level Description of Final Product

There is the **environment** and the **creatures**. The environment has a bunch of parameters (wind, food density) and so do the creatures (speed, mass). If the creature finds food, it will survive. If however, after some time has passed, and the creature has not found any food, it will die. Periodically, all alive creatures lay a few offspring. The offsprings’ parameters are inherited from the parent but with a little bit of randomness applied. The user is able to change the environmental parameters to see the effect it has on which creatures survive and reproduce.

Example of environmental parameters:

* Wind direction/speed
* Amount/depth of water
* Color of terrain
* Amount of food
* Food regeneration rate

Examples of creature parameters:

* Mass
* Volume
* Shape
* Speed

# Software Development Process and Methodologies

* Lean
* Highly iterative
* Focus on getting a working product as fast as possible
* Add new features each iteration, as well as increase the quality of existing features
* No TDD
* It’s a prototype that *could* turn into the real thing (but also ok if it gets thrown away)
* Do not focus on documentation, cleanliness, organization, conventions, etc, get something up and running ASAP. Can refactor later.
* Don’t worry about art/sound (can always make it better later)

# Iterations

## Iteration 1

* DONE - Get project set up on git hub
* DONE - Creatures that move around randomly
* DONE - After some time, if they are out of energy they will die
* DONE - Randomly spawn food (when creature collides with food, increase his energy)
* DONE - After some time, surviving creatures lay some off springs, adjust offspring’s parameters randomly

## Iteration 2

* DONE - GUI
  + DONE - Can use GUI to spawn a new creature with specified parameters into the mix
  + Can use GUI to select creatures to delete
  + DONE - Can use GUI change environmental parameters such as food spawn rate

## Iteration 3

* DONE - Make world bigger and camera pannable
* DONE - Make camera zoombable
* DONE - Rename Main to World
* DONE - Creature needs reference to World
* DONE - Fix socket error bug

## Iteration 4

* DONE - Add a “speed up” feature
* DONE - Make sure creatures eat at least one food before replicating
* DONE - Dark creatures tend to live for way too long without food

## Iteration 5

* DONE - Change movement and such to use physics
* DONE - Come up with new creature/world model using physics
* DONE - Food should see if it collides with rigidbody (not area) since creatures are rigidbody now
* DONE - Make it so when you spawn creatures, they spawn in random area of map (not all in same location)
* DONE - Make it so it costs energy to give birth
  + If creature doesn’t have enough energy, can’t do it
* DONE - Increase efficiency: call GD.Load just once on Creature.tscn

## Iteration 6

* Make game full screen (just a project setting)
* DONE - Make a menu (that comes up when you press esc)
* DONE - When creatures collide, sometimes movement in the y direction occurs as a result of the collision; prevent this
  + Ideas
    - On \_integrate forces or whatever
      * Get rid of y component of linear velocity
    - Make creatures not collide with one another
      * Put em all in same layer, make them all not collide with that layer
      * This is what I went with
* DONE - Make camera’s far plane further
* DONE - Make zooming faster when zoomed further out
* DONE - GUI – make sure 1x etc updates when timescale is changed
* DONE - In GUI, add ability to spawn a certain number of food (randomly)
* Also create a food brush
  + Create a mouse handling node
    - All mouse events (down, move, release, and wheel) should go to this node first (in other words, implement the \_input() function)
    - This node should see if there is a MouseHandler object set, if so, forward events to mouse handler
    - If no mouse handler, simply do nothing (so event goes through Godot’s regular way)
    - MouseHandler should do its thang (for example, CreateFood handler can create food in response to mouse events)
* DONE - Also, allow adjustment of bounds via gui