Sprint 6

10 day duration (Ends 2nd of June, 2017) ~60 hours of work

Product Backlog Tasks

1. As a player, I am motivated to expand my transportation network and develop my region.
2. As a player, I have access to a variety of maps that offer me different challenges and opportunities.
3. As a designer, I have a collection of intuitive and robust tools that allow me to design maps efficiently without concerning myself with the technical details of the game's implementation.
4. The game's UI displays all the information a player needs to know in a primarily visual manner.

Basic Design

Story

Players are some sort of governing figure attempting to develop a largely empty region by establishing an efficient transportation network that allows societies to grow and complexify.

Mechanics

The current game is an implementation of prototype 3.2, plus the addition of some scoring component or victory condition that has yet to be determined.

Aesthetics

The aesthetic style of the game remains simple geometric shapes and flat shades, as going beyond this is beyond the scope of my abilities.

Technology

Unity deployed to PCs, and possibly to browsers, as well, depending on performance issues.

Risk List

1. My design isn't based on rigorous and repetitive playtesting, and it seems unlikely I'll be able to playtest it properly.
    a) While there is an obvious solution to this problem (namely, to playtest the bloody game) I just don't think that's a feasible goal at this time, for external and internal reasons. For this reason, I think I need to start wrapping this project up, focusing less on developing and iterating upon new mechanics than wrapping the game up in a reasonable package and preparing it for presentation. It's about time that I move onto the next stage of my life, anyways.
2. There isn't a clear mapping between resource names and resource colors, and players might not understand which resource is which.
    a) I need to figure out how to insert the color of a given resource into the descriptive text in SocietyDisplay so that players can build a map between blob color and resource names. This will also allow players to tell, at a glance, what a society needs, wants, and produces.
3. Players have little structure to guide their play, and no clear reason to participate besides the intrinsic enjoyment of the game's mechanics.
    a) The simplest solution would be to score players on the various levels of complexity they manage to achieve. There are a variety of ways in which I could do this. One way would be to give players a certain amount of time to generate a configuration with the highest score. Another would be to present some score that players must achieve and give them as much time as they need to achieve it. I could also judge players by the total number and value of exports they produce, but that seems a much more difficult and complex task.
4. Any designer not intimately familiar with the codebase will most likely have a frustrating time creating and saving maps.
    a) There are a host of issues with the interconnection between MapNodes, MapEdges, and MapGraph that I need to resolve. I should probably devote time to creating an editor window to manage all of the complexities that emerge from my design. At the very least, I need to make sure that nodes and edges destroy and copy gracefully with no complications. I might also consider automatically adding a MapGraph to a scene that does not have one, and automatically handling parenting relations upon subscription.
5. Map designers have few tools by which they can design maps, and this might make it a lot harder to reason about the process of map design.
    a) In addition to resolving the problems described above, giving designers the ability to make prefabs containing arbitrary collections of MapEdges and MapNodes ought to do much to improve the map design process. Doing that in a sensible way will require changing the way in which MapNodes and MapEdges reason about their parent graphs, will require an expensive Find operation, or else will require me to compromise the relatively decoupled architecture I've established.
    b) Designers will also need to be able to save maps. I'm not entirely sure how I'd end up accomplishing that. Using ScriptableObject and generating an asset out of any completed map might be a good solution, though I don't know how that will integrate with the points of coupling between Assets.Map and other modules. I could try using scenes, though that would lead to an unpleasant duplication of factory and core components that would make the codebase less maintainable.
    c) I'll need to take a good look at my options and decide which one suits my needs the best. I should especially try to use features already existing in Unity rather than rolling my own system, and shouldn't be afraid to adopt a less robust solution in exchange for a quick and sufficient resolution.

6. Players have no way of knowing how well they're doing, and this might create a lack of feedback.
    a) Adding score-tracking mechanisms, in one form or another, should help resolve that problem. Adding tiered score (say, bronze, silver, and gold medals for various score achievements) might help give players an idea of the range of difficulties within a given map. I can't say for sure without playtesting the thing, and this may very well be a fake problem, one that has little bearing on reality.
7. I might be injecting my own biases about what I want to work on into the risk list, assigning myself tasks that may or may not be relevant to the play of the game.
    a) Avoiding this would require a level of introspection and self-awareness I'm not sure I possess. I might be able to mitigate it somewhat by addressing problems I know with certainty already exist. For instance, I am quite familiar with the problems surrounding the map designing system. The problems I've identified with it are likely to be far more legitimate than any hypothetical player issues. Beyond that, any things that I've personally struggled with while trying to play the game have a higher chance of being legitimate problems that hypotheticals I've never tested and never thought about.
    b) Scoring is, perhaps, a component of the game that deserves the most skepticism. While I'll likely need some sort of victory condition (or at least some measure of progress) for completeness sake, it likely makes less sense to engage with detailed decision-making in that arena. That would suggest that I should make only a rudimentary scoring system, focusing more concerted efforts on other tasks.
8. There are no ways of selecting between maps, which will make it impossible for players to engage with multiple challenges.
    a) I'll need to create some sort of menu screen for completeness anyways. Within that screen can be a place from which players can choose between multiple maps. How exactly I'll implement that screen depends on how I manage to save map configurations.
9. There is no way of saving and loading a session in progress, which will make it difficult and frustrating for players to engage with complex topologies requiring a lot of thought and effort.
    a) Depending on how I implement the scoring system, I may or may not need a system for saving and loading. But I do think it's a good idea to code in persistence to my game, to expand my options and to demonstrate one of the fundamental tasks any game must perform. Actually managing that persistence may be somewhat challenging, though I suspect I can manage it through a pair of classes, GameSaver and GameLoader, that take the current set of highways, societies, resource depots, and whatnot, turn them into some serializable format, and then use the various factories to reconstruct everything upon loading the game.
    b) Doing that will require specifying MapNodes across runtime environments, which will be a problem given the current way in which I specify IDs. I'll need to change the paradigm under which MapNodes are assigned IDs so that every node has the same ID everytime it is loaded. How I will do that remains up for debate.
10. MapEdges are intolerant of the movement of MapNodes, which makes tweaking the position of nodes a frustrating experience.
    a) MapEdges need to listen to the movement of the MapNodes they are attached to and refresh their endpoints accordingly.
11. Components of MapGraph do not function well with prefabs, which makes it difficult and tedious for designers to store and make use of neighborhood patterns.
    a) This is a considerable issue, and resolving it will require making the Map module more robust to external manipulation. I think the mitigations of risk 5 cover this one fairly well.
12. It might be very difficult to save the state of a particular match, given the coupling of objects to their factories (and other, similar entanglements).

a) The mitigations of risk 9 cover this risk, as well.

13. Terrain might not be very well described or displayed, leading to an ugly map that players don't want to spend time looking at.
    a) I don't think this is a priority right now. I have more fundamental problems, like being able to design, save, and load maps, or being able to save and load sessions. Appearance can come as a polish task later on in the development cycle, possibly as a desirable task during this sprint but most likely as a task in the next one.
14. Performance issues might interfere with player experience, especially if the framerate drops below 30. These performance issues might be an even greater concern if I deploy to browers.
    a) Performance, too, is fairly low on the priority list. I'll consider it at a higher priority to displaying terrain, but will address it only after the numerous more important tasks have been resolved.
    b) Performance will start with profiling, figuring out what methods are consuming the most time. It will also likely involve object pooling on at least ResourceBlobs, for which object instantiation is likely a large and unnecessary overhead. I'd imagine that most of the problems emerge from ResourceBlob and possibly BlobDistributor. They might also emerge from unseen inefficiencies in using Tick methods instead of performing tasks in Update (for which Unity might provide considerable optimizations). All of these considerations ought to wait until performance becomes a priority, however.

Sprint Backlog

To Do

Critical (0 hours of work remaining)

1.

Important (0 hours of work remaining)

1.

Desirable (6 hours of work remaining)

1. Profile the game and identify performance bottlenecks that are hurting the game's framerate.
    a) Estimated 1 hour to complete.

In Progress

Critical

1.

Important

1. Devise a system for partitioning the map into sections based on the positioning of MapNodes in a sensible manner that makes the map look like a more cohesive object.
   a) Estimated 5 hours to complete.
   b) Spent 6 hours without completing it.

Desirable

1.

Completed

Critical

1. Change MapEdges so that they reset their position whenever one of their endpoints is moved.
    a) Estimated 30 minutes to complete.
    b) Took ~40 minutes to complete.
2. Modify MapNodes and MapEdges so that they gracefully copy and destroy themselves without disrupting the state of MapGraph.
    a) Estimated 2 hours to complete.
    b) Took 4 hours to complete.
3. Change the Map module so that it properly handles prefabs containing MapNodes and MapEdges.
    a) Estimated 1 hour to complete.
    b) Took 5 1/2 hours to complete.
4. Devise some sensible way to save and load map configurations in a portable manner.
    a) Estimated 3 hours to complete.
    b) Took 4 hours to complete.
5. Change society creation at design-time so that it permits the creation of appropriate societies on the terrain types that accept them, and forbids their creation otherwise.
    a) Estimated 1 hour to complete.
    b) Took 1 hour to complete.
6. Add a custom editor to Society that makes it easier to change the Society's current complexity to some relevant value.
    a) Estimated 1 hour to complete.
7. Develop a title screen that allows players to choose which map they would like to play on and gives basic statistics about that map.
    a) Estimated 1 hour to complete.
    b) Took 2 1/2 hours to complete, though the scope of the task expanded somewhat.
8. Change HighwayManagerDisplay so that it properly records the upkeep of the selected HighwayManager and highlights the highways it is currently managing.
    a) Estimated 3 hours to complete.
    b) Took ~3 hours to complete.

Important

1. Figure out how to insert sprites representing resource colors inline with the names of those resources, or else attach sprites to the names of resources in some other way.
    a) Estimated 1 hour to complete.

b) Took 4 hours to complete, as the original estimate failed to take into account incorporating the code into the game proper.

2. Build interfaces and unit tests for a new class, NetworkScorer, that analyzes what the player has accomplished and scores them based on some criteria.
   a) Estimated 2 hours to complete.
   b) Took ~2 hours to complete.
3. Greenlight the NetworkScorer unit tests.
   a) Estimated 2 hours to complete.
   b) Took ~45 minutes to 1 hour to complete.
4. Build interfaces and unit tests for a new class, VictoryManager, that checks for victory and defeat conditions, triggering victory or defeat sequences when those conditions are satisfied.
   a) Estimated 30 minutes to complete.
   b) Took 30 minutes to complete.
5. Greenlight the VictoryManager unit tests.
   a) Estimated 1 hour to complete.
   b) Took ~30 minutes to complete.
6. Create and integrate a UI that displays the player's progress towards victory.
   a) Estimated 30 minutes to complete.
7. Build interfaces and unit tests fora new class, SessionSaver, that saves the state of a given session for future play.
   a) Estimated 11 hours to complete (from the estimates of the pre-aggregation tasks)
   b) Took 9 1/2 hours to complete.
8. Design and implement a class, FileSystemLiaison, that reads and writes SerializableSession files to the file system properly and produces a list of available sessions.
   a) Estimated 1 hour to complete.
   b) Took ~1 hour to complete.
9. Implement all unimplemented unit tests.
   a) Estimated 1 hour to complete.
   b) Took ~1 hour to complete.
10. Greenlight all unit tests.
    a) Estimated 1 hour to complete.
    b) Was accidentally wrapped up into the above task.
11. Build runtime UI for saving and loading sessions.
    a) Estimated 2 hours to complete.
    b) Took ~3 1/2 hours to complete.
12. Integrate neighborhoods into SerializableSession.
    a) Estimated 1 hour to complete.
    b) Took ~1 hour to complete.

Desirable

1.

Abandoned

Critical

1. Rearchitect MapGraph so it performs its tasks cleaner and without as much fiddly code.
    a) Estimated 3 hours to complete.
    b) Spent 30 minutes on it.
    c) Ended up being enmeshed in other tasks and was determined to be not a particularly good framing of the problem.

Important

1. Build interfaces and unit tests for a new class, SessionSaver, that saves the state of a given session for future play.
    a) Estimated 3 hours to complete.
    b) Got consolidated, along with the other Session tasks, into a single, larger task.
2. Greenlight the SessionSaver unit tests.
    a) Estimated 2 hours to complete.
    b) Got consolidated, along with the other Session tasks, into a single, larger task.
3. Build interfaces and unit tests for a new class, SessionLoader, that takes the data generated by SessionSaver and uses it to load the appropriate map and repopulate it with all relevant objects.
    a) Estimated 3 hours to complete.
    b) Got consolidated, along with the other Session tasks, into a single, larger task.
4. Greenlight the SessionLoader unit tests.
    a) Estimated 3 hours to complete.
    b) Got consolidated, along with the other Session tasks, into a single, larger task.

Desirable

1.

Review

Total Hours invested: ~55 hours, including the 3 hours it took to initiate the sprint.

Design: I managed to improve the user experience of the game considerably. There is now a sensible mapping between the colors of resources and their names, reiterated in pretty much all of the UIs, that makes it a lot easier to tell at a glance what a society or a construction project needs. I've also vastly improved the UI of HighwayManager, so that it now shows how much upkeep it requires and what highways it manages. I've also added code (currently not particularly well-integrated) to enable victory and defeat conditions and the saving and loading of sessions.

The lack of integration of the various pieces of the program has, for the first time, become a considerable issue. Before I largely did not concern myself with menu flow or game state (whether the game is at its title screen, paused, running, etc) because these weren't elements of the program. But in its current state these issues pose an awkward challenge.

As I more or less promised, I avoided engaging in mechanical design changes, given the difficulty of playtesting the game.

Development: I spent a large portion of this sprint working on design-time tools rather than runtime modifications to the game. I managed to substantially improve the robustness and designer experience for developing maps. MapNodes and MapEdges now copy and move intelligently, can be copied into and from prefabs without too much of an issue, and can be serialized into assets. I've also designed ways of doing the same for other elements of the game, both at runtime and design time. Now everything but the position and orientation of ResourceBlobs can be saved to and loaded from a file, which makes it possible to design and save map configurations in a much cleaner way. I've not integrated that full session-saving module into design space, but it shouldn't be much of a task to do so.

Retrospective

What went well?

- Despite what I had felt during the execution of the sprint, I only wasted a relatively small amount of time (~5 hours).
- I managed to work through nearly all of the tasks I had set before myself, including all of the critical and all of the important ones.
- I successfully kept myself from trying to mechanically iterate on the game without some form of playtest. Though it would've been much better to have engaged in playtesting, I at least addressed tasks and improvements that could be done without player feedback rather than continuing to build higher on shaky scaffolds.
- The factory pattern I've been using up to this point paid considerable dividends when I was implementing session saving and loading. Having a single location from which I could pull all of a certain type of object, and a single point of coupling with the runtime environment, made it a lot easier to save session data. Instead of having to grapple with the manual construction and deconstruction of MonoBehaviours, I could simply pull and pass relevant

data into the factory or the object itself, no delicate interconnections required.

- Though I didn't come anywhere near completing it, I think I did a solid job reasoning about the parameters of the map partitioning system. I managed to, through mathematical reasoning, devise a system by which I could solve this fairly complex problem. It left me with a single problem to solve, that of detecting polygons in an undirected graph, which I was able to find a possible solution for without too much difficulty. What I did on the last day of the sprint was one of the most algorithmically intensive things I've done in the project thus far. That I was able to pursue it meaningfully demonstrates to me that I've got solid enough mathematics fundamentals to reason about the world in that manner.
- I managed to, at least once, recover from a severely scatterbrained position and continue working. Such things have traditionally derailed workdays, and I'm sure laziness and lack of motivation claimed several hours during this sprint, as well, but that I was able to catch it at all is a sign of progress.
- I addressed nearly all of the risks I identified, ignoring only the playtesting risk (which I opted to steer around) and the performance risk (which I deprioritized). I've addressed, sometimes completely, everything else I thought might be a problem, even if I didn't ultimately resolve the issue (like with risk 12).

What could be improved?

- There were several tasks whose time for completion I grossly underestimated, one so substantially that I suspect it shall take four or five times as much time to implement as I had originally budgeted. I should look back at those tasks, trying to understand what caused them to take so long, why I thought they would take less time, and see what I can learn from those mistakes to improve my skills at estimation. Being able to intelligently guess the difficulty of a task will, I think, go a long way to improving my time management and determining what things I should or should not attempt.
- My UI currently feels very ad-hoc. The transitions between various screens and states is cobbled together in a largely informal and unorganized way. If I'm ever to expand upon this UI, or build an effective paradigm by which I can design UI in the future, I'll need to figure out a more formal and rigorous way of handling screen transitions, that I can build and iterate them with less repetitive and interstitial code hanging between.
- My solutions for making MapNodes and MapEdges more robustly associate themselves with MapGraph feel hacky and inefficient. I suspect that there are latent bugs living within that system that will emerge at a later date. There are likely better ways by which I could make those connections line up, though it remains to be seen if repairing them is worth my time.
- I poorly divided the session saving and loading mechanisms into four tasks that ultimately got rolled up into a single task. In reality, there were only two tasks to which I should've devoted my time: the unit tests for saving and loading sessions, and the implementation to greenlight them. I still need to get better at dividing tasks that are too large and combining tasks that are too similar. In that case, I should've identified that session saving and loading were two sides of the same coin.
- I think the method by which I am constructing unit tests is inefficient. Having to manually construct all of my mocks takes a considerable amount of my time. There might be ways for me to reduce the setup complexity of my tests as well. I should look into both NUnit and other unit testing utilities more carefully to see if there are any tools by which I might improve my testing process.

- During the sprint, I often justified cutting an hour or so off the programming work cycle by pointing to the small number of tasks left in the sprint backlog. I did this despite the fact that I've always ended sprints with tasks of importance that I ought to perform. I need to stop using that justification to depart from my work, ideally by meditating away that tendency to get distracted and asserting discipline.