Sprint 5 -- 10 day duration (Ends 12th of May, 2017) 60 hours of work.

Product Backlog Tasks

1. There are a large number of resource types that players can engage with, all of which have a unique place in the game's economy and all of which have an important part to play.
2. Maps have features or terrain that permit the construction of some societies but not others.
3. As a player of any skill level, I can make interesting choices in how I develop my region.

Basic Design

Story
The story of the game remains largely the same from the previous four sprints. No more elaboration has been given to it because it's not a priority.

Mechanics
In addition to the mechanics developed in sprint 4, players will now have additional societies and more complex chains of ascent and descent. Societies and resources will be divided into four tiers of production: raw materials, light industrial products, heavy industrial products, and knowledge products. Each of these resources has one and only one society capable of producing them, and each tier requires more complex societies in order to produce. Societies of a higher tier require productions from the lower tier, while societies of a lower tier benefit from some of the products of the higher tier. Certain societies have certain terrain requirements, especially the societies that produce raw materials, so that not every society can be placed on every node.

Aesthetics
This remains a question beyond the scope of my abilities. I'm going to maintain the simple geometric shapes and monotone colors that I've been using up to this point.

Technology
Still Unity deployed to PCs.

Risk List

1. I don't know how many resources at every tier of production I should make.
   a) I'll need to do some preliminary design work before I even begin coding, to see how various resource types might interact with each-other. The precise bag of resources I end up using will likely be informed by other needs, and is likely to be something I'll work on throughout the development cycle. I'll focus on designing something that has some possibly interesting properties and explore its consequences rather than try to solve this problem all at once.
2. Adding additional resources, and additional societies to produce them, completely destroys the tentative balance of costs I've relied on up to this point.
   a) It was bound to happen at some point, wasn't it? Given that it's not clear that the original cost paradigm was good, I don't know how big of a loss that was. For now, I'll have to relinquish the sort of mathematically precise balance I've been looking for and try to stake out the big-picture. My design for this new paradigm ought to be looser, with the admission that this will lead to balance issues but that their resolution won't be a critical task until later in development. Broad strokes to begin with; I'll deal with the specifics later.
3. I don't know if my resource tiers should be a pyramid (where raw materials are the most

diverse resource tier and diversity goes down as you go up) a column, an inverted pyramid, or some other paradigm.
    a) There's something to be said about the pyramid. The game already implements that sort of pyramidal structure in resource quantity, where large numbers of simpler societies are necessary to support a small number of complex ones. During my preliminary design period, I'll try and build a resource paradigm for each of these three models and see which one makes the most sense. My suspicion is that the normal pyramid's what I'm looking for, but we'll see how the early designs work.
4. Adding too many resources could balloon complexity to such a degree that the game becomes unplayable.
    a) There are a number of ways I can address this problem. One is to make sure that every society produces a single resource, which will make it a lot easier to figure out how to produce a resource and where a given resource came from. Another is to enforce a sort of pattern on the needs of societies of higher complexity, so that all of the needs of societies on the same production tier follow the same logic. And establishing a more consistent logic between tiers will likely make it easier to reason about the tiers hollistically.
    b) Making sure to choose good names and fictional qualities for all of the resources and societies may also help players reason about the problem. For instance, having a light-industrial society called Textile Village that requires Food (green) and Cotton (white) and produces Clothing (blue) is a lot easier to reason about than a blue-producing village absorbing green and white.
5. Adding too many resources could ruin the UX and make the game unusable.
    a) This will have the greatest effect on the highway permission UI. I can mitigate this by organizing resources by their tier and maintaining their color-coding. But that sort of division will only get me so far. It's very important, then, to keep the number of resources low, though how low they'll need to stay is something I'll test thoroughly with people who aren't me.
6. Enforcing terrain restrictions on societies might severely limit the number of options players have and thus reduce player expressivity.
    a) I honestly have no idea how true this is. Limitations, after all, can breed creativity as much as hamper it. In my mind, terrain-based limitations primarily affect the production of raw resources. You need flatland to farm, forests to harvest lumber, mountains to gather metal, things like that. But higher-level societies might not have such restrictions. Terrain, then, informs what sorts of production you have available to you, but doesn't directly inform where higher-tier societies are placed. That's still left to the player to decide.
7. It's not clear what effects terrain types are going to have on the structure of play.
    a) I'll start with terrain just affecting the placement of raw material producers and see how that feels. I might consider adding restrictions to complex society placement in certain places (cities cannot be built in forests, for example) or permit some limited terrain changes (turing forests into fields by chopping them down, fields into forests by regrowing the trees). But all of that will be secondary. Anchoring the raw production is the first step.
8. Adding more resources makes ResourceDepot's storage paradigm more ridiculous.
    a) I can address that fairly simply by removing the per-resource capacity and setting a maximum capacity common across all resources. That might cause issues for particularly high-throughput ResourceDepots, but that's a problem to identify and address when it arises, not beforehand. That goes for this risk, as well. I'll keep an eye on it, but until it breaks something I ought not to fix it.
9. I already don't have enough things to devote certain resources to, nor do I have sufficient niche protection as it stands. Adding more resources is only going to cause more productions

that do nothing (like blue) or that overlap too heavily (like yellow and white).
    a) Terrain might be a good vehicle for resource differentiation. For instance, players might be able to use some heavy industrial goods to chop forests down and turn them into fields, and might be able to use knowledge goods to rehabilitate fields into forests. That gives players a lot of expressivity and a lot of options, not to mention an incentive to reach more sophisticated societies in order to optimize their networks further. In the short-term, I can also just create resource dumps that aggregate consumed resources into some sort of score. That addresses problems with player motivation, as well, though in a lazy and heavy-handed way.
10. It might be difficult to display the terrain of a MapNode in a clear and appealing manner without major changes to the display model of the game.
    a) The clearest way I can think of to display the terrain of a MapNode is to draw some sort of colored polygon around it. Ideally, the surface area of the map would be partitioned into sections defined by the positions of MapNodes, but figuring out how to do that dynamically might be quite challenging, and doing it manually quite frustrating. For starters I'll probably just draw a large square behind the MapNode and ensure sensible separation between them so they don't overlap. More advanced builds ought to have more advanced ways of drawing the regions.
11. Adding more societies and more conditions is going to cause UIControl and SimulationControl to God Object into unmanageable proportions.
    a) Before I start adding features, I'll need to separate SimulationControl and UIControl into its constituent pieces. The separation will probably occur to the interface, implementation, and testing all at once, for both sides. That way, individual components of UIControl only couple to the individual components of SimulationControl that they need. I'll probably make that separation the first programming priority after I've worked through some of the preliminary design problems.
12. I already lack good names for my resources. Adding more exacerbates that problem.
    a) Dividing resources into tiers helps that cause a bit. Probably names will be part of that preliminary design problem, as the fiction for resource transferral is critical to make it feel intuitive. I don't think there's any way of mitigating this problem other than to address it during the preliminary resource exchange design.
13. Having more resources will make delicate balancing acts a lot harder to achieve, and it will become more difficult to ensure precise ratios of societies for stable configurations.
    a) The easiest solution would be to abandon these delicate balancing acts at all, to make it so that production and societies do not get by on thin margins given a certain map configuration. The point of the game being expressive is to permit nondiscrete degrees of success, to make optimization a worthwhile goal, so in that case early design precision isn't quite as important. I'll need to set aside delicate balances for other reasons anyways (just the difficulty of reasoning about more resources makes the task infeasible in the short-run) so perhaps worrying about balance isn't important at this stage in development.
14. There is still no obvious motivation for advancing societies, and adding more difficulty to the process might make that lack of motivation a bigger problem.
    a) The short-term solution to this problem is to quantify the development and production of a region. I could keep track of score, giving players points based on the societies they've managed to create, with more complex societies providing more points. I could also create resource sinks that can measure a configuration's surplus production and calculate points based on that, as well. Neither of those solutions would be particularly hard to implement, and it would bring the game closer to having formal objectives, and it incentivizes the sort of optimization I'd like to see players perform.
15. It's not clear how wants are going to play into a multi-resource paradigm. Do societies of the same tier share the same wants, or do they have different wants?

a) Having lower-tier societies share the same wants makes resources from higher-tier societies more valuable because they're more versatile. Given that terrain compositions will now place restrictions on raw resource production, there's a good chance that higher-level productions will become important for addressing resource shortages. That might cause me to change the way in which satisfied wants affect production, but it's definitely a consideration I should look into.
b) Having shared wants also promotes a more pyramidal model for resource production, with a small number of highly versatile productions at the top and a larger number of more specialized productions towards the bottom. That makes other considerations easier.

Sprint Backlog

To Do

Critical (0 hours of work remaining)

1.

Important (7 hours of work remaining)

1. Add a ScoreTracker that measures the total development of a player's configuration and generates a score based on how complex it is.
   a) Estimated 1 hour to complete.
2. Add a ResourceSink that consumes resources and makes some measure of the number and types of resources consumed.
   a) Estimated 1 hour to complete.
3. Build a BlobAlignmentStrategy that keeps blobs of the same type in a cohesive group without causing unintelligible patterns of movement.
   a) Estimated 2 hours to complete.

Desirable (9 1 /2 hours of work remaining)

1. Rework ResourceDepot so that it does not store resources by type and has an upper limit to its contents independent of the number of resource types.
   a) Estimated 30 minutes to complete.
2. Figure out how to sensibly partition the display area of the map into regions defined by the placement of MapNodes.
   a) Estimated 8 hours to complete.
3. Design a UI that indicates on the world map whether a Society has its needs satisfied and how close it is to descending if its needs are unsatisfied.
   a) Estimated 1 hour to complete.

In Progress

Critical

1.

Important

1.

Desirable

1.

Completed

Critical

1. Sketch out a few different resource need, want, and production paradigms whose resource types are bottom-heavy, top-heavy, and evenly distributed.
   a) Estimated 5 hours to complete.
   b) Took 2 1/2 hours to complete.
2. Replace the implementation of UIControl with the Chain of Responsibility pattern on each type of UISummary and separate its current code into a collection of more decoupled modules.
   a) Estimated 2 hours to complete.
   b) Took 2 hours to complete.
3. Separate the interface, tests, and implementation of SimulationControl into some number of smaller, constituent modules of behavior.
   a) Estimated 3 hours to complete.
   b) Took 2 1/2 hours to complete.
4. Reorganize, augment, and repair all of the UIControl unit tests so that each constituent module is tested independently.
   a) Estimated 2 hours to complete.
   b) Took 3 1/2 hours to complete.
5. Augment and repair all of the old SimulationControl tests so that each Control module is unit tested thoroughly.
   a) Estimated 5 hours to complete.
   b) Took 2 1/2 to 3 hours to complete.
6. Establish the notion of terrain types in MapNodes that can be modified at runtime, and establish some rudimentary way of displaying these terrain types to the player.
   a) Estimated 1 hour to complete.
   b) Took ~ 40 minutes to complete.
7. Add ConstructionZoneProjects, ConstructionZoneControl commands, ConstructionPanel buttons, and PrefabBuilder methods to create societies of any tier 1 complexity.
   a) Estimated 1 hour to complete.
   b) Took ~35 minutes to complete.
8. Modify the Society complexity, ascent and descent systems to handle the sorts of many-to-many transitions proposed by Prototype 3.1, as well as terrain restrictions.
   a) Estimated 3 hours to complete.
   b) Took ~4 hours to complete.
9. Update the SocietySummaryDisplay so that it sensibly displays the multiple ascent and descent options available to societies in Prototype 3.1.
   a) Estimated 2 hours to complete.
   b) Took 1 1/2 hours to complete.
10. Test and debug the new Highway UI to make sure it gives proper permissions to all of the new ResourceTypes.
    a) Estimated 30 minutes to complete.
    b) Took ~ 10 minutes to complete.
11. Debug Society ascent and descent, in the UI and the mechanical logic, as well as BlobSite capacity changes based on available ascent transitions.
    a) Estimated 1 hour to complete.
    b) Took ~ 1 1/2 hours to complete.
12. Modify the program so that it possesses and properly manages all of the resource types as specified by Prototype 3.1.
    a) Estimated 2 hours to complete.
    b) Took ~ 2 hours to complete, but also overlapped with other tasks. It's not clear how

much time was devoted specifically to this task.

13. Greenlight (and augment if need be) the HighwayManager unit tests.
    a) Estimated 30 minutes to complete.
    b) Took ~30 minutes to complete.
14. Modify ResourceDepot's cost model, add projects for clearing and planting forests, and generalize the inheritance hierarchy of ConstructionProjects.
    a) Estimated 30 minutes to complete.
    b) Took ~30 minutes to complete.
15. Design an informal, neighborhood-heavy map, then attempt to generate a stable city configuration on it, just to get some initial impressions on the new prototype.
    a) Estimated 1 hour to complete.
    b) Took ~1 hour to complete.
16. Develop some map neighborhoods that can be used to support different types of villages with the right placement.
    a) Estimated 30 minutes to complete.
    b) Took 30 minutes to complete.
17. Develop a Prototype 3.2 that uses triangle villages instead of pentagonal villages while still providing meaningful surplus.
    a) Estimated 30 minutes to complete.
    b) Took ~10 minutes to complete.
18. Fix some of the issues with MapGraph that cause manually deleted MapEdges to break the system beyond all repair, and generally make its editor-facing methods more robust.
    a) Estimated 30 minutes to complete.
    b) Took ~ 15 to 30 minutes to complete.
19. Build, run, and iterate on a map and see if the new triangulated villages make the game flow better and allow for more intelligent player choice.
    a) Estimated 2 hours to complete.
    b) Took 1 1/2 hours to complete.
20. Compose some questions to ask about and experiments to perform with Prototype 3.
    a) No estimate provided.
    b) Took 1 hour to complete.
21. Using the technique suggested by questions 10 and 11, develop a collection of neighborhoods that can generate different pallettes of all the tier 2 resources.
    a) Estimated 1 hour to complete.
    b) Took ~ 1 hour to complete.
22. Construct a map that uses the new neighborhood theory that could likely be used to sustain a city configuration.
    a) Estimated 30 minutes to complete.
    b) Took 30 minutes to complete.
23. Run through the new configuration and see what problems arise with its design.
    a) Estimated 1 hour to complete.
    b) Took 1 hour to complete.

Important

1. Fix and greenlight the ConstructionZone unit tests.
    a) Estimated 1 hour to complete.
    b) Took less than a minute to complete.
2. Fix and greenlight the Core and Society unit tests.
    a) Estimated 30 minutes to complete.
    b) Took 25-30 minutes to complete.
3. Modify HighwayDisplay so that it groups resources by their tier.
    a) Estimated 30 minutes to complete.

b) Took ~10 minutes to complete.

4. Fix some of the performance issues that seem to have arisen around the creation of BlobHighways and HighwayManagers.
   a) Estimated 2 hours to complete.
   b) Took 2 1/2 hours to complete.
5. Put MapGraph under unit tests.
   a) Estimated 3 hours to complete.
   b) Took 5 hours to complete.

Desirable

1.

Abandoned

Critical

1. Bring the current implementation of Prototype 3.1 to working order.
    a) Estimated 2 hour to complete.
    b) Task did not have specific enough actions and boundaries and was considered pointless.
2. Enforce terrain restrictions for Society placement.
    a) Estimated 1 hour to complete.
    b) Got wrapped up into other tasks and was rendered redundant.
3. Determine a policy for placement and extraction permissions on Societies that allows them to ascend on their own resources without becoming resource roads.
    a) Estimated 30 minutes to complete.
    b) Got wrapped up into other tasks and was rendered redundant.
4. Take the neighborhoods designed in experiment 1 and figure out what pushing ResourceDepots further into them would accomplish, if anything.
    a) Estimated 30 minutes to complete.
    b) Got wrapped up in other tasks and was rendered redundant.
5. Build neighborhoods that can produce 1 tier 2 resource normally, and another less efficiently if you push ResourceDepots into its interior.
    a) Estimated 30 minutes to complete.
    b) Got wrapped up in other tasks and was rendered redundant.

Important

1.

Desirable

1.

Review

Total hours invested: Somewhere around 48 1/2, plus 3 for initiating the sprint itself.

Design: I managed to design a Prototype 3.1 and 3.2 to mitigate the specified risks. Prototype 3.2 establishes 10-resource paradigm with 10 corresponding societies spread across 4 tiers of production. The raw numbers of the societies' productions and consumptions fit reasonably well within each-other. I also designed a simple notion of terrains that limit the placement of certain societies to certain terrains.

While I struggled to reason about the parameters of the design, I did end up developing a system that gave me some ability to reason about the map design of this game more rigorously. I came to an important conclusion about the nature of neighborhoods and used that to create several neighborhood "prefabs" that I just started using to design maps more intelligently.

I also identified and began to address several problems with the design. It seems highways are too slow and efficiency doesn't scale their speed up effectively when they are long. Building towns from a network containing only liquid village resources (as the network I tended to build did) proves problematic if the town ever decays due to temporary losses of resources, since there are no tier 1 resources to support it. And building towns from empty land was awkward for similar reasons. There are also resource differentiation problems that, I suspect, can only be addressed with additional player verbs or additional differentiation between tier 3 resources.

Unfortunately, I did not perform any playtesting of my design, a recurring issue that's becoming more and more destructive as the design is advancing. Thus my observations are more hypotheses than definitive statements. And I've no doubt there are many problems I'm woefully unaware of.

Development: I managed, over the course of the two weeks, to implement pretty much all of the requirements of Prototype 3.2. Beyond the requirements of the design tasks, I modified the architecture of the control modules (UIControl and SimulationControl), splitting them and their unit tests up into smaller, more modular units of behaviour. While doing that, I stripped out the tests' direct use of implementations and replaced them with mocks.

I finally placed MapGraph under proper unit testing, which revealed several errors with the algorithms I was using. HighwayManager is now considerably more performant; it doesn't freeze up execution upon creation of higways or highway managers. There are, however, general performance issues that either have arisen recently or have been there for quite some time and that I did not notice before. The framerate for the game steadily declines as the transportation network becomes more and more complex. The current low point is around 27 FPS for a graph with several dozen societies producing and distributing resources.

Retrospective

What went well?

- I managed to implement all of the critical tasks arrayed before me, bringing Prototype 3 online and into an iterable state.
- I reinforced the rigor of my program's architecture and the Scrum practice in general by enforcing unit tests on a previously untested module (MapGraph) and by modularizing my

two main Core classes (UIControl and SimulationControl) so that they were more wieldy. I also removed a lot of the direct use of implementations in old SimulationControl tests, which means that the Core classes are more decoupled from external implementations.

- I delayed rearchitecting any of my modules until there was a clear reason to do so. Once that reason appeared, I modified the architecture to address the existing issues and went no further. Examples include UIControl and SimulationControl splitting, and the generalization of the UI (making it so that the UI code didn't address specific ResourceTypes).
- I managed to establish additional ConstructionProjects with considerable ease and no modification of any of the intermediary pieces of the code (no Core classes changed), which demonstrates the extensibility of that section of the codebase.
- I feel like I've designed a fairly reasonable collection of resources and societies that produce them, and defined reasonable metaphors for making the mechanical systems make sense. While there's plenty of balance and differentiation that ought to be done, I think my current resource setup is a solid base upon which to build.
- I quickly identified when tasks were irrelevant and meaningless and brought myself back on track without wasting too much time flailing about.
- I managed to make some progress in how to reason about the design of my map. By reasoning about production alternatives within neighborhoods more rigorously, I was able to generate a better way of thinking about the map as a whole, and actually managed to design a map that was more than a loose affiliation of non-specific neighborhoods. Each region of the map had a number of purposes to which it could be applied, which made it feel more like a map than a random collection of nodes.
- I pretty thoroughly addressed the tasks from the project backlog that the sprint was flagged to work on. There are a variety of resources with at least some differentiation between them. Terrain types now substantially alter where certain societies can and cannot be placed. And players definitely have choices about what they can do with any particular section of the map. While it is false to say that any of these large tasks are resolved, I've at least made considerable progress on them all.
- I addressed and used the mitigations of nearly all of the risks I defined. I can trace pretty much every task I engaged with (with the possible exception of the MapGraph tests) to a particular risk I'd identified and the mitigation for that risk.

What could be improved?

- At one point, early on in the sprint, I set a few placeholder tasks that I intended to replace with more actionable tasks once some prerequisite items had been worked through. This turned out to be a very poor practice and I should not have done it. The very fact that there were prerequisite tasks means that I, at least partially, regressed into a waterfall design pattern. I need to figure out how to define and divide tasks earlier on in the sprint so that I never have a task that reads "there will be more work later, but I don't know what it is." One thing I can try and do is extract the really preliminary design work from the sprint cycle altogether, so that I have a clear idea of what I'm iterating on, what I need to program into being, and what tests ought to be done at hour 0 of the sprint.
- Though I've likely missed a few hours of programming here and there in almost every sprint, this one was particularly bad. There were several days where I relinquished an hour

or more of work and performed other tasks not related to the sprint. It felt a lot like what was happening during the latter half of the fiasco that was Merchant Republics. I think the reason why this happened was because I lacked a comprehensive list of specific, important tasks to address. I was often left not entirely sure what I should be doing, and thus it made it a lot easier for the lesser elements of my psyche to argue for abandonment of work. If I'm to counteract that trend, I'll need to specify my design and development tasks a lot more clearly and precisely, ideally ahead of time, so that at every point in the development cycle I know what I should be doing and why I should be doing that. Worst case scenario is I end up defining a bunch of tasks that become irrelevant as I move forwards and slide them into the Abandoned category, while still holding onto those that are relevant.

- I am now, more and more often, running into barriers in the development process that can only effectively be overcome via thorough playtesting. This lack of external player feedback is making design questions harder and harder to address as the design advances. It's also making it a lot harder to develop tasks and figure out where the design should go from here. I think I need to put my foot down and declare that no more design work may be done until I've playtested my game with at least somebody who isn't me. It made some sense not to playtest during Sprint 1 and Sprint 2, but now it is an absolutely nonsensical prospect.
- Despite the delays I made in implementing them, many of the archiectural changes I made may have been a poor use of my time. I devoted 2 full days--20% of the sprint--to fixing architectural issues. I guess my backlog wasn't jammed full of high-priority tasks, and the changes undoubtedly made things cleaner and easier to worry about, but spending so long on code that will never face the player may be concerning. And there's no guarantee that my changes have even had a positive effect on usability. I won't know for certain whether this was a good use of my time, I think, until later on in the development cycle. But it does worry me, given how huge of a problem re-architecting was on earlier projects.
- A lot of the tasks that remain in the sprint backlog quickly became irrelevant in the grand scheme of things. I think the reason that was is because they weren't part of the core considerations of the sprint. Thus, as time went on, the issues they addressed became less and less relevant to what I was working on. I should endeavour to focus the attentions of any given sprint on a specific collection of features or goals. I suppose that's the whole point of pulling things from the project backlog, but there's likely more specificity that ought to be defined beyond just the higher-level project goals. Either that, or I should have declared the tasks to be irrelevant, though that seems like an equally inappropriate decision.
- The tools I'm using to develop maps are rudimentary and fragile. If I'm to apply myself to the task of designing maps more rigorously and thoroughly, I'm going to need to vastly improve the tools available to me. Whatever sprint I define in the future undoubtedly needs improved MapGraph code, additional editor windows, prefabs, and better copy/paste functionality than the current one has. It would also be enormously useful to be able to quickly change Society complexities, build highways, add stockpiles, and be able to move MapNodes without breaking the MapEdges connected to them. It was important, I think, that I didn't waste time on these features earlier in the development cycle, but now I think I rather need them.
- I completely discounted the use and effects of society wants on the execution of the game. I never found any particular reason to spend higher-tier resources to generate more lower-tier production, nor did it even cross my mind that I should have done that. This suggests that I'm letting parts of the problem slip away from me, either because they are no longer relevant to the execution of the game or because I simply can't store the whole problem in my head at once. I'm not entirely sure what I should do about that, other than attempt to meticulously address each component of my mechanical systems in my risk mitigation, so

that I define meaningful backlog tasks for them and thus incorporate them into the ongoing iteration.