Kanban and Reflections

# Academic Reflection

### Prototype 1

For my first prototype I produced a simple, turn-based game which uses only one button and an analog stick as controls. My time management in particular improved greatly during this prototype.

Although my time estimation for tasks improved over the duration of the two prototypes, it still isn't perfect. Interestingly, I noticed a pattern emerging during the early and late stages of the first prototype, my time estimations for how long tasks would take was reasonably accurate (although less so with the first prototype), often within 10 minutes of the real time. However, both projects suffered poorer accuracy during the middle period. Reflecting on the specific tasks, and how I felt when working on this middle-period, I found myself feeling somewhat directionless at times. Literature supports the idea that uncertainty or changes in the direction of a project will negatively impact the accuracy of time estimations[1] (Popli and Chauhan, 2014). I responded to this phenomenon by making more notes about how exactly to achieve individual tasks, and trying to keep better track of exactly how far through the project I was.

The main thing I could have done better is planning my sprint durations. This would have been easier if I had known how many sprints were expected from the start, and would have allowed me to do more even length sprints; my 1-day sprints were particularly unhelpful. I also found that non-stop pressure sprints made me quite stressed about the project, and always on my mind, particularly due to trying to squeeze in multiple sprints per week. Most agile development schemes use sprint lengths of at least a week[2] (Shiohama et al., 2012), a duration that also varies depending on the individual project. From this, and my experience, it is reasonable to say that overly short sprints are detrimental to productivity and wellbeing.

Overall I was quite pleased with the prototype I produced during this sprint. It wasn't particularly challenging but it gave me a useful introduction to planning and scoping projects and tracking my time and productivity.

### Prototype 2

For my second prototype I created an isometric exploration game, where the player gradually explores a procedurally generated map and waters plants. During this prototype I made use of several procedural generation techniques and honed my understanding of architectural considerations for complex systems.

In previous projects, I often found myself working late at night, and in very unevenly distributed blocks. My commitment to a project might vary depending on how much direction I have for the task at hand. I noticed this tendency reappearing with certain tasks during development: larger, less well-defined tasks became harder to motivate myself towards and I felt more aimless as a result. However, following the agile methodology for these prototypes lead me to minimise this effect: one key feature of this methodology is breaking the overall product down into individual, achievable modules, a practice called Feature Driven Development[3] (Kumar and Bhatia, 2012). Following this technique made me more productive, especially early on where I took on a complex system for procedurally constructing a chunk-grid-based world. By breaking this down into discrete parts (a basic framework, support for custom chunk patterns, support for flood-fill analysis, plants, details) I was able to maintain an idea of the whole without being overwhelmed by it, focusing on only the current part being developed. Conversely, this encouraged me to consider the architecture of the procedural generation system, and how I could build it such that new features could be added to it later.

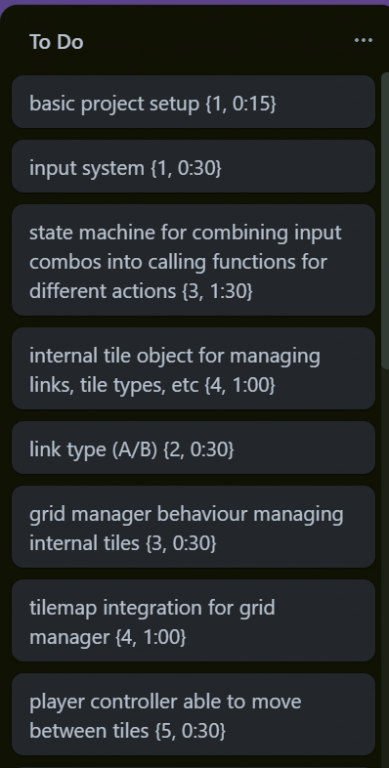
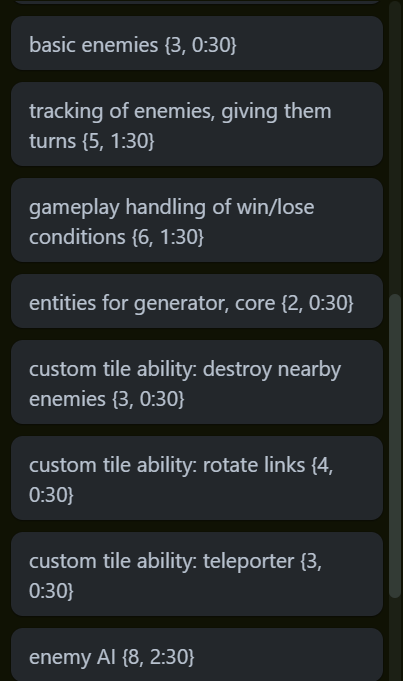
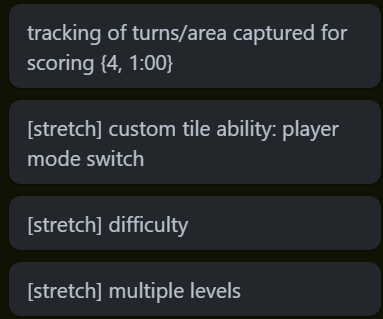
The second prototype pushed me to consider carefully when to use appropriate programming paradigms. Reading the book *Game Programming Patterns*[4] (Nystrom, 2014) not only expanded the range of techniques I understand, but also made me reconsider the benefits of some that I was already aware of. For instance, Nystrom particularly questions over-use of the singleton paradigm: this lead me to minimise my use of this technique in my second prototype. I originally considered using singletons for each of the MapGenerator, the WorldController, PlayerController, and GameController, but after more consideration I eliminated the need for singletons on all but the GameController. This now acts primarily as a bridge to allow different controllers to interact without keeping numerous global references hanging around. I also found use for the flyweight pattern when designing the Map Generator: instead of keeping a list of every possible pattern for every tile, each tile keeps a list of indices, which index into a list of possible patterns, massively reducing data duplication during generation. I made limited use of inheritance for creating different types of hold-able and interact-able objects, being careful to balance minimisation of code reuse with unnecessary inheritance hierarchies.

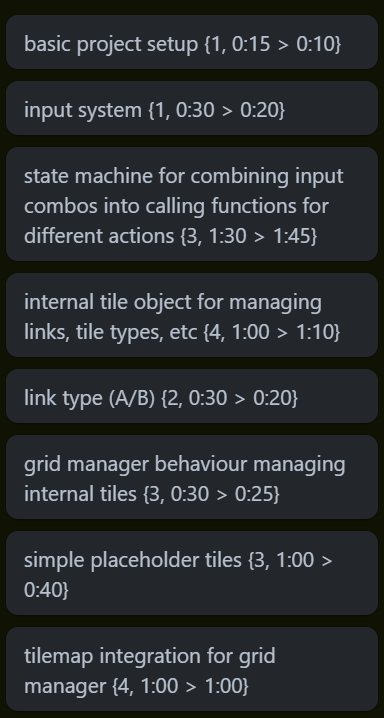
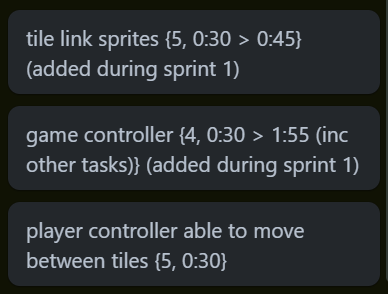
One particular issue I encountered when building the system for the player picking up objects during the second prototype was preventing duplication of hold-able objects. When a chunk loads, it needs to know which detail elements (such as plants, watering cans, etc) to load with it. This raises the question of how to handle the player picking up and moving an object out of its home chunk. The key problem here was one of interrelatedness of scripts: the World Controller doesn't know about the held state of an object, and the held object doesn't know about the chunk map inside the World Controller (the result of the practice of decoupling, a key best practice in programming in general). My initial solution lead to objects being duplicated if a chunk reloaded while the player was holding an item, however with some consideration I produced a solution which avoided this by keeping a list of references to hold-able objects which had been instantiated into the world, to prevent duplication. This avoids multiple problems; it fixes the initial bug, avoids having the HoldableObject script talking directly to the WorldController resulting in coupling, and avoids the brute-force solution of having the chunk position of objects be checked every frame.

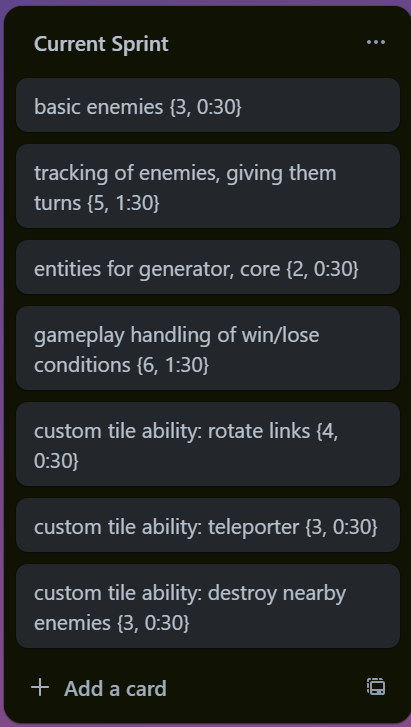
Overall, I actually enjoyed working within the agile framework for these projects. I feel it made me more efficient, and less stressed, and as a result I'm proud of the two prototypes I produced. Particularly for the second prototype, I'm very pleased with the multiple procedural generation techniques I employed: wave function collapse, flood-fill, and a simplified version of the void-and-cluster algorithm proposed by Ulichney[5] (Ulichney, 1993).

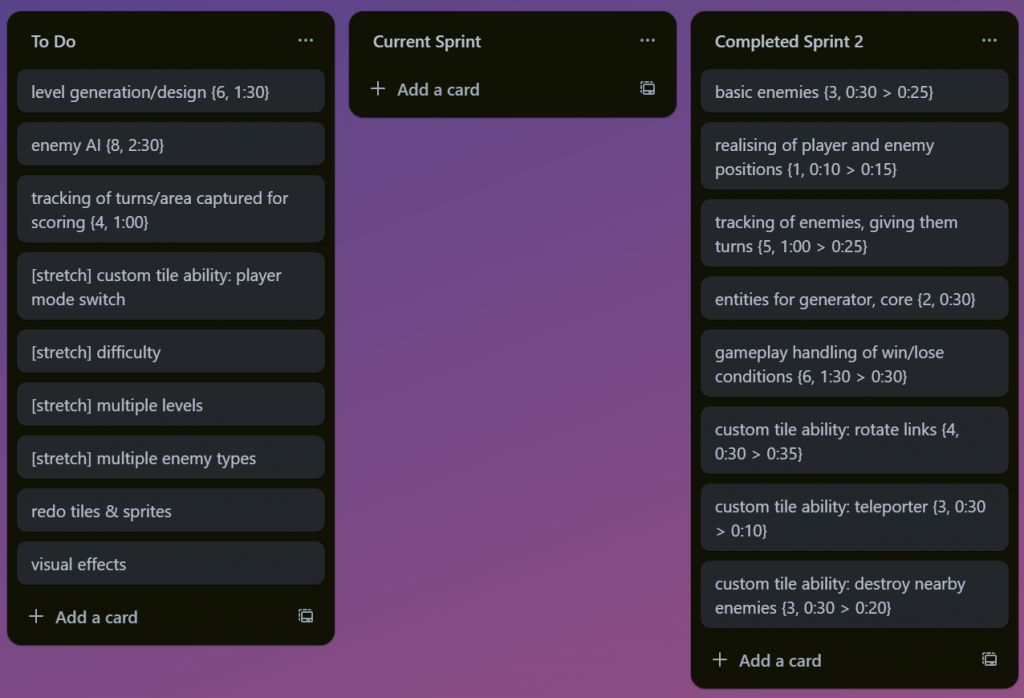
# Kanban Board Progression

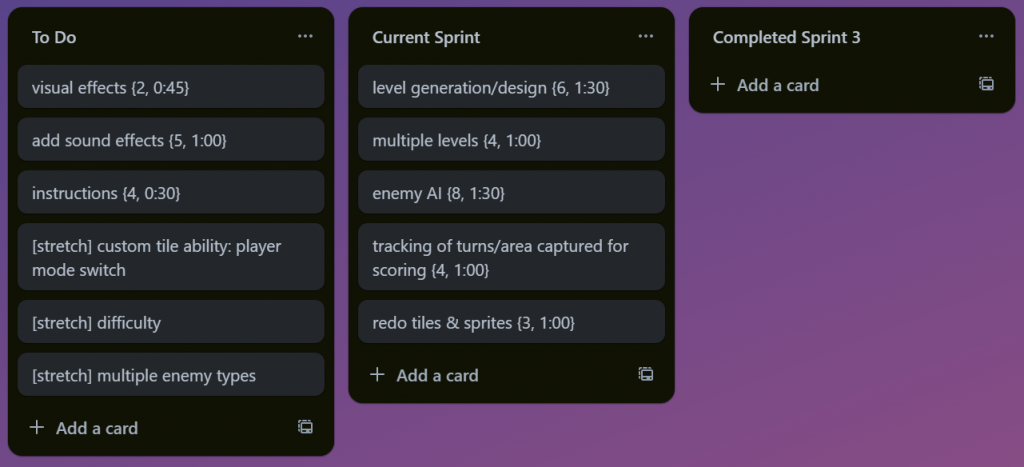
## Prototype 1

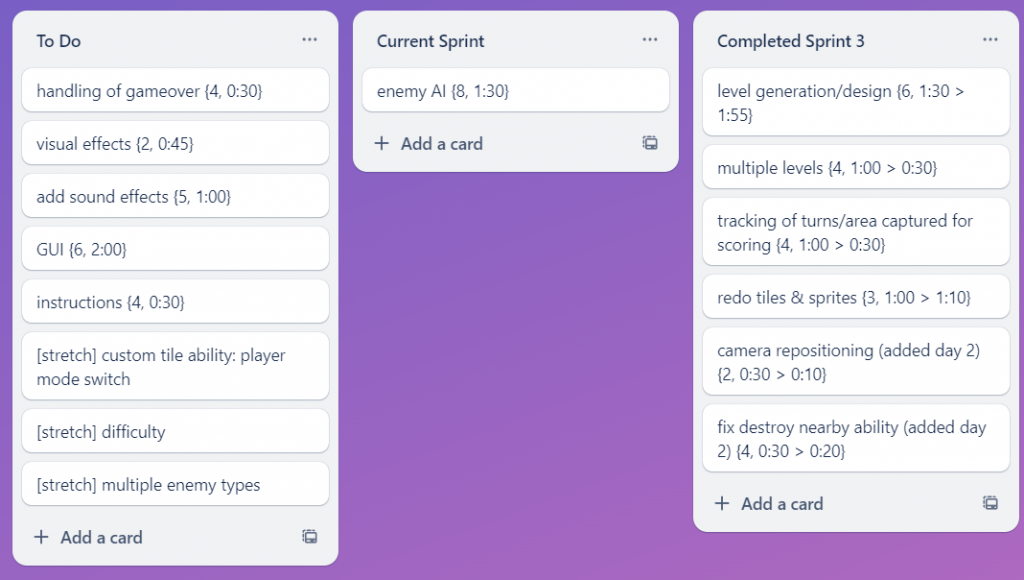
Start of sprint 1:   

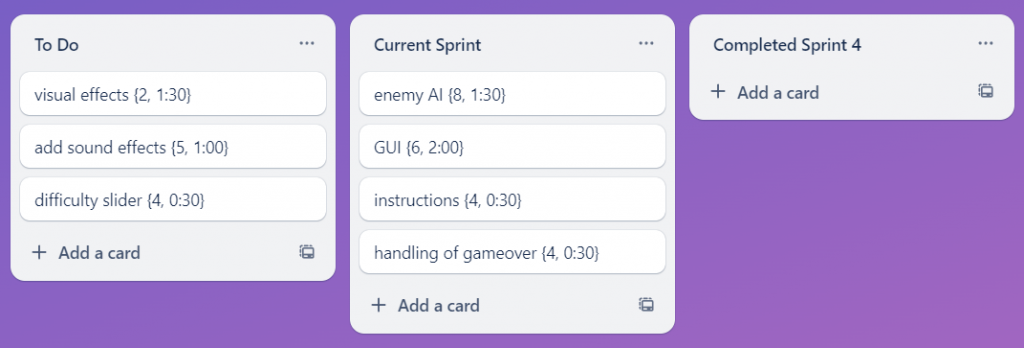
End of sprint 1:   


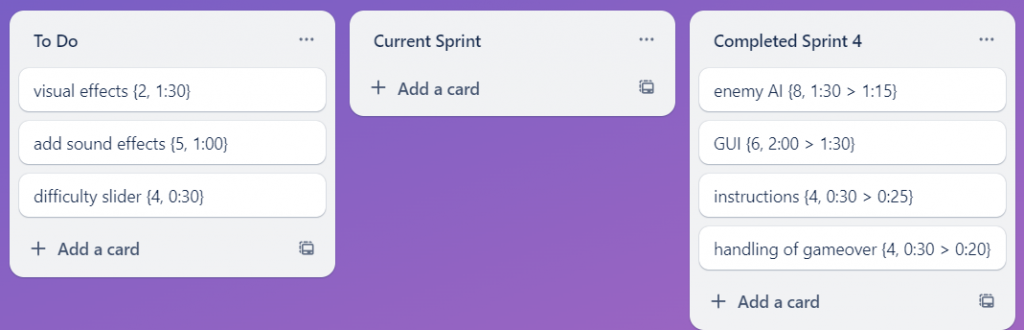
Start of sprint 2: 

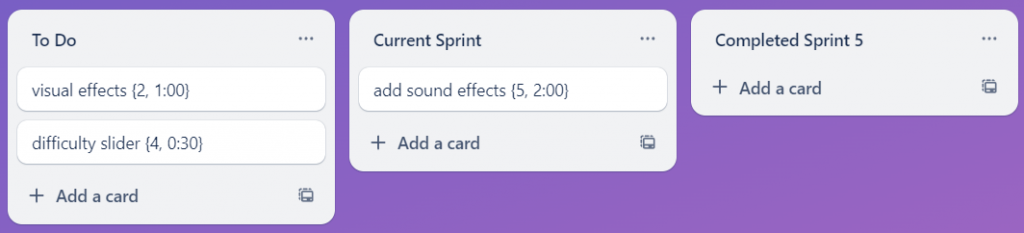
End of sprint 2: 

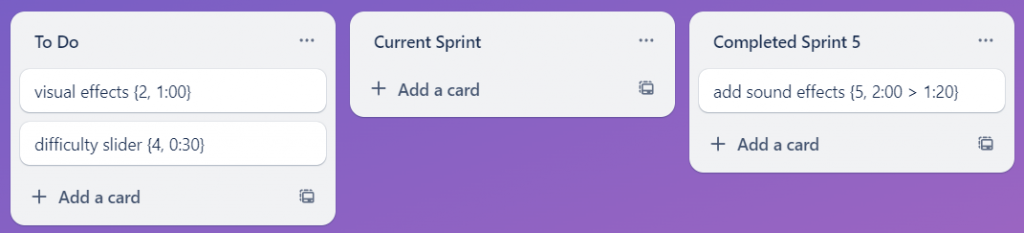
Start of sprint 3: 

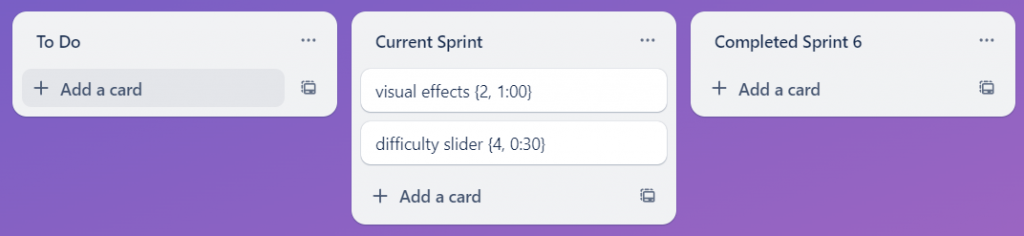
End of sprint 3: 

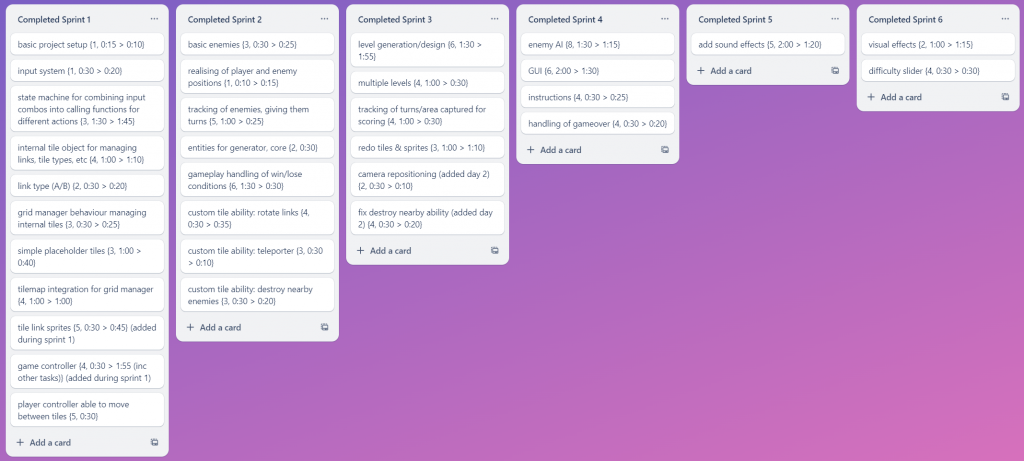
Start of sprint 4: 

End of sprint 4: 

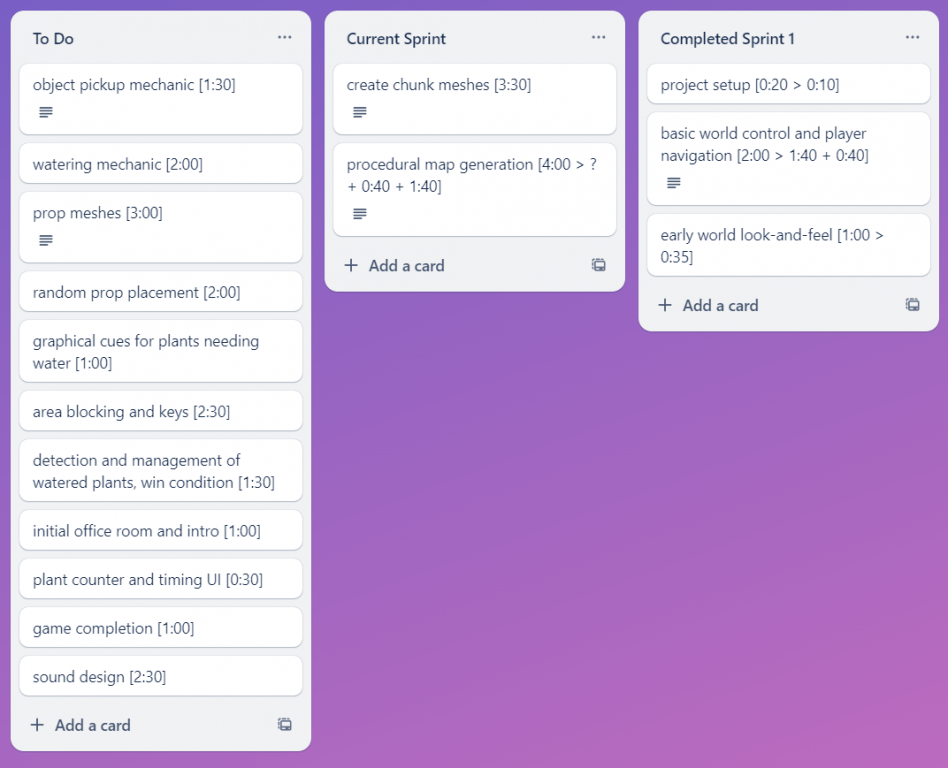
Start of sprint 5: 

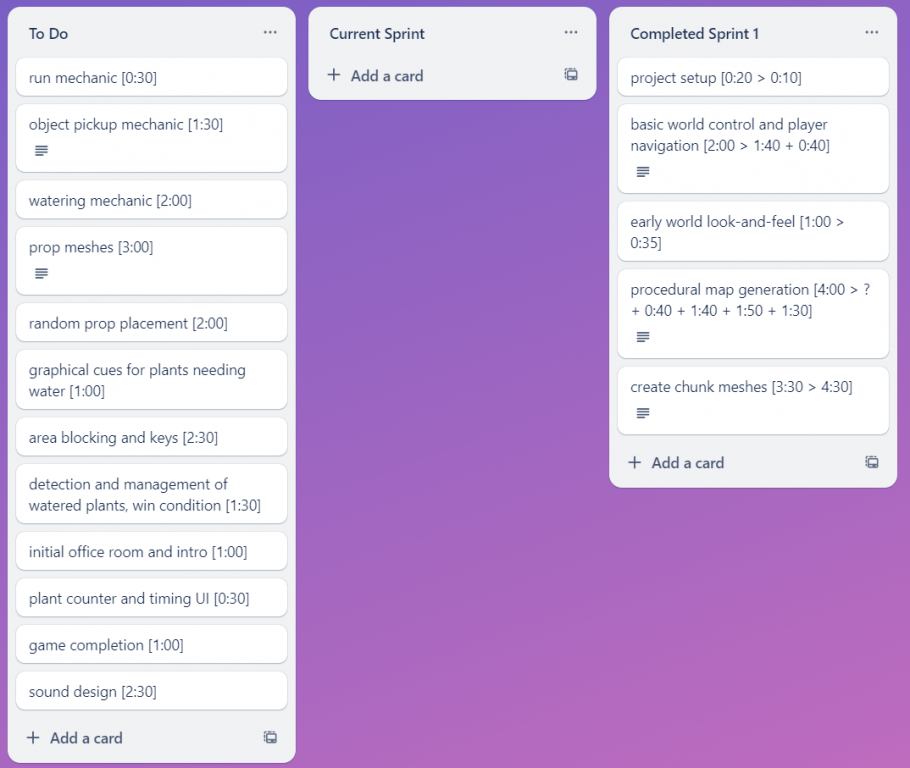
End of sprint 5: 

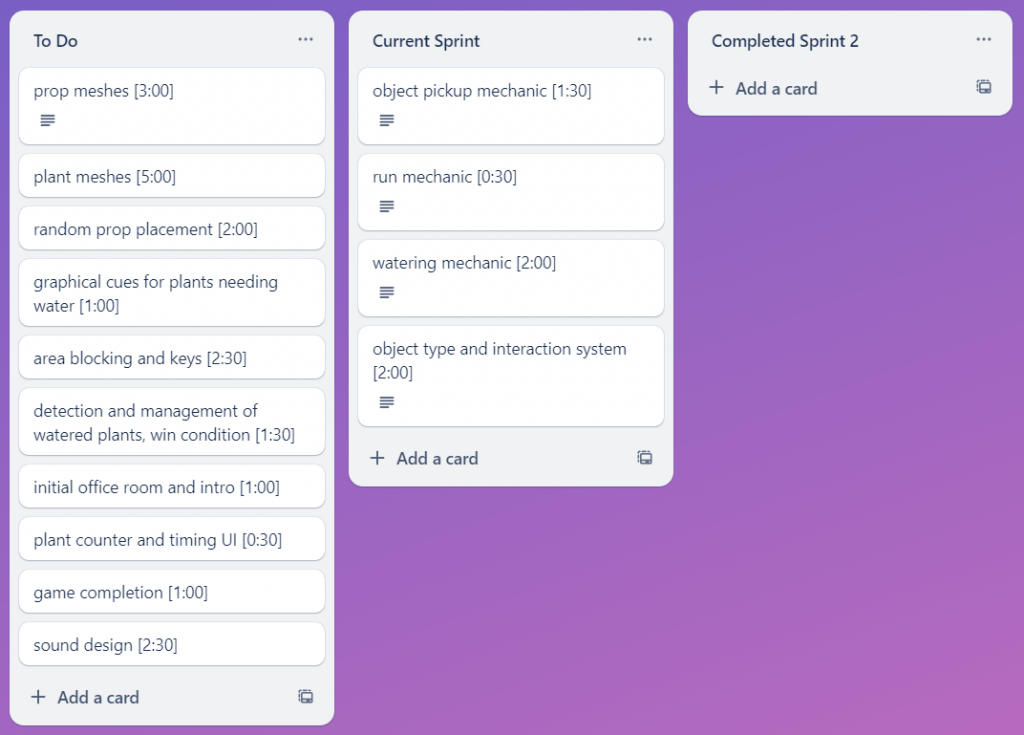
Start of sprint 6: 

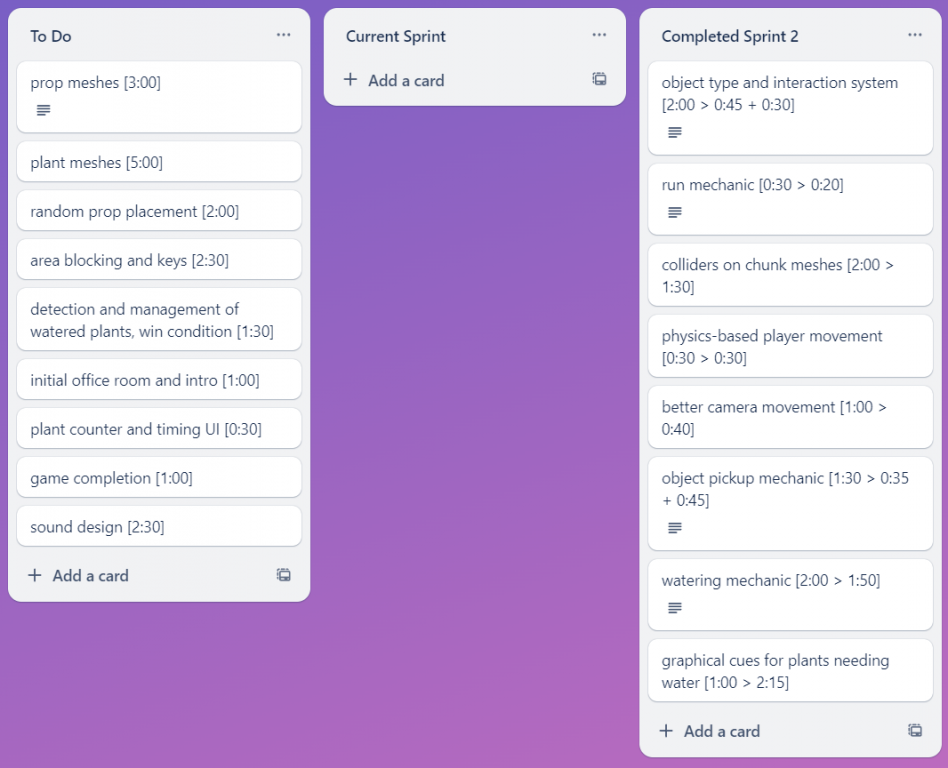
End of sprint 6 (entire board): 

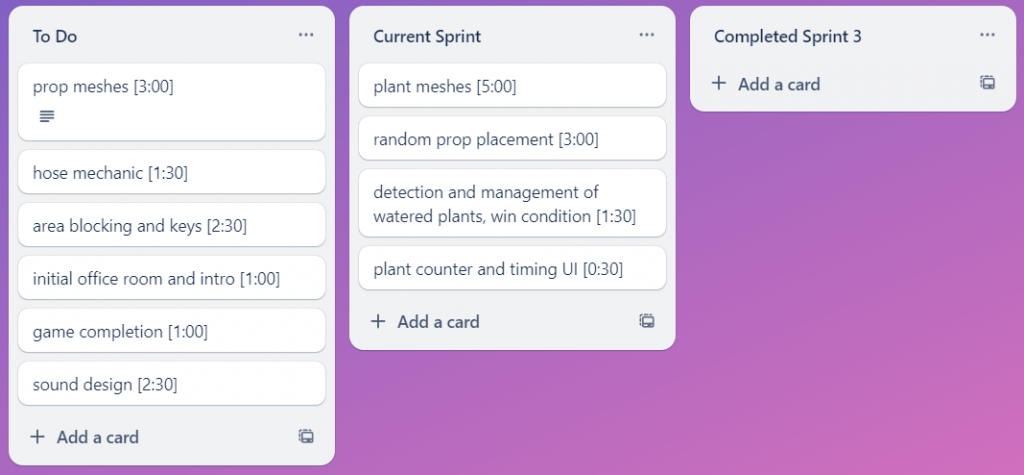
## Prototype 2

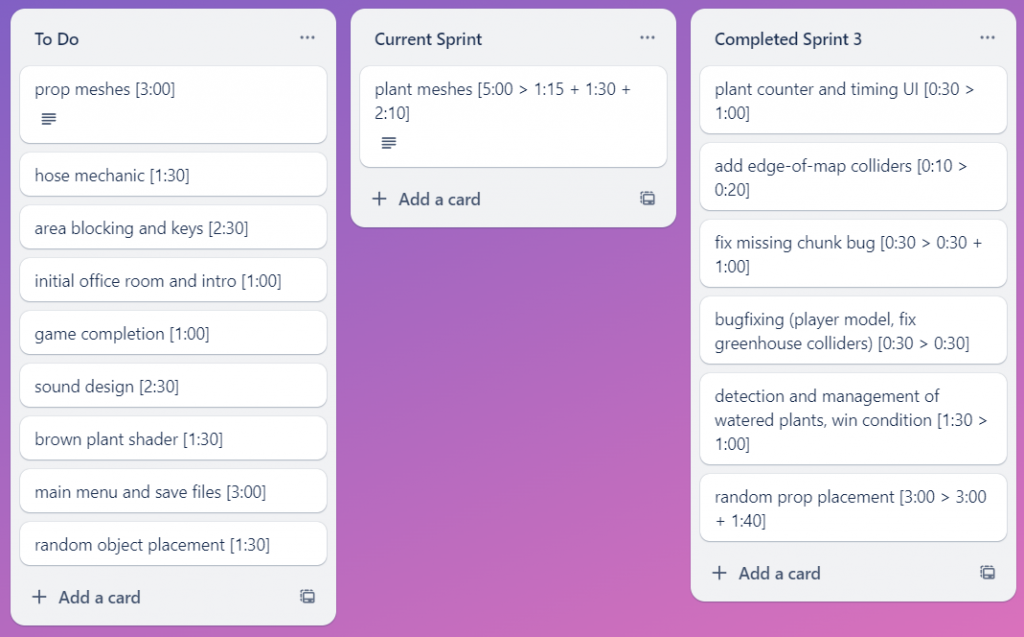
Start of sprint 1: 

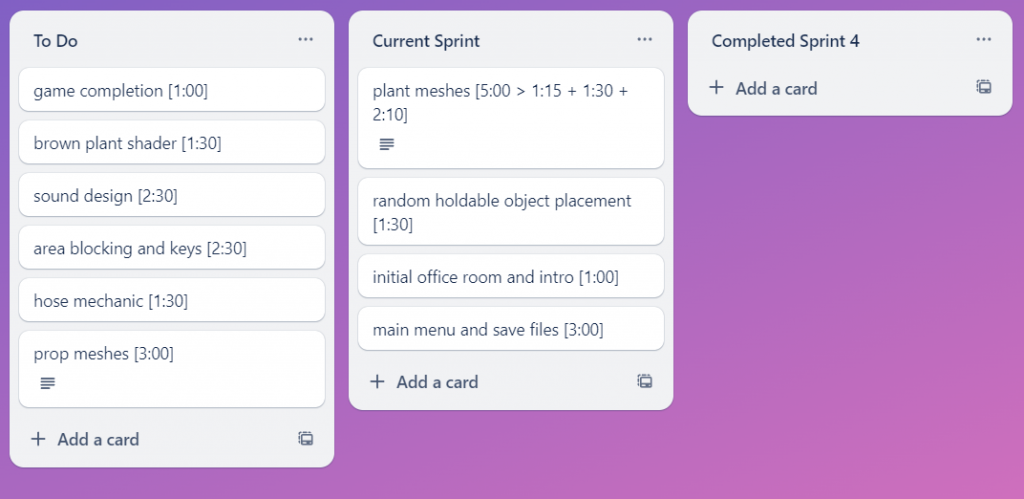
End of sprint 1: 

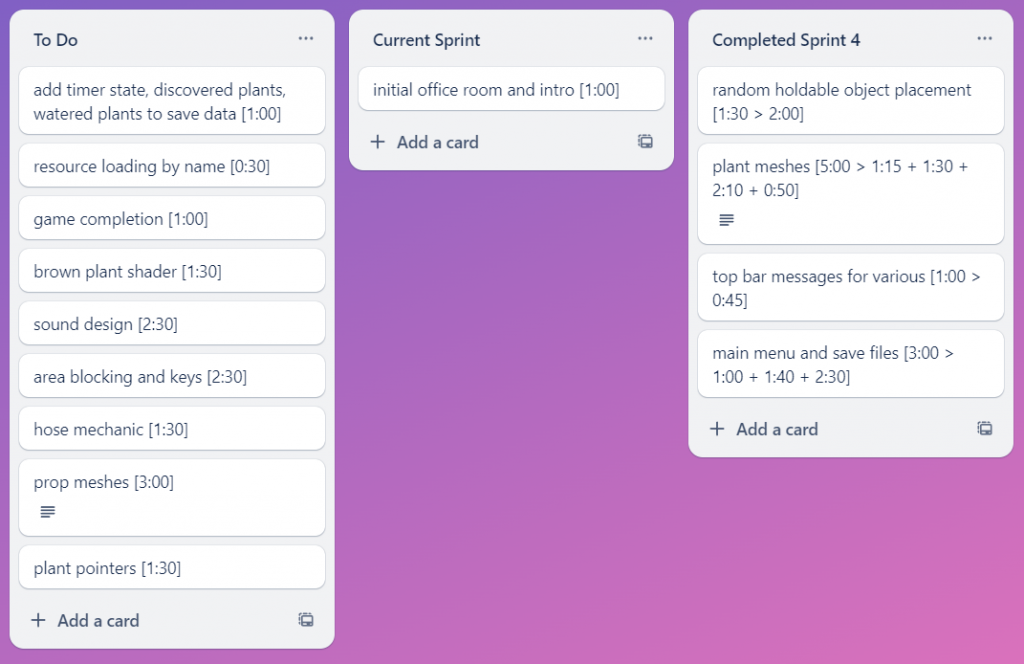
Start of sprint 2: 

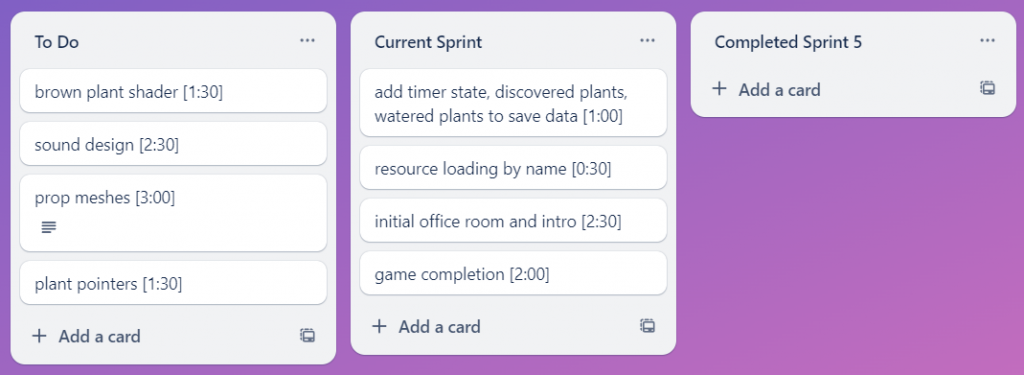
End of sprint 2: 

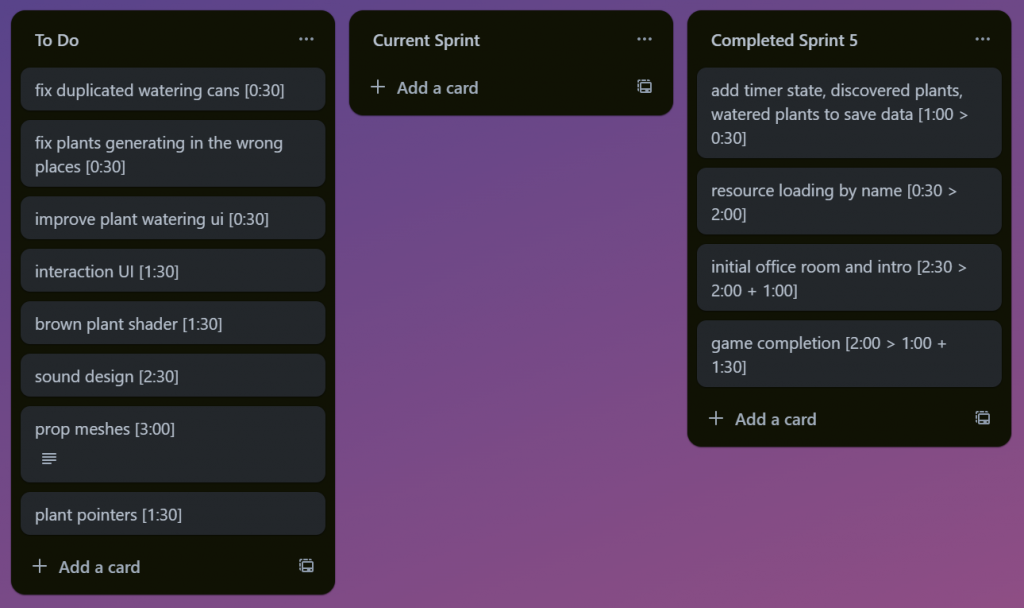
Start of sprint 3: 

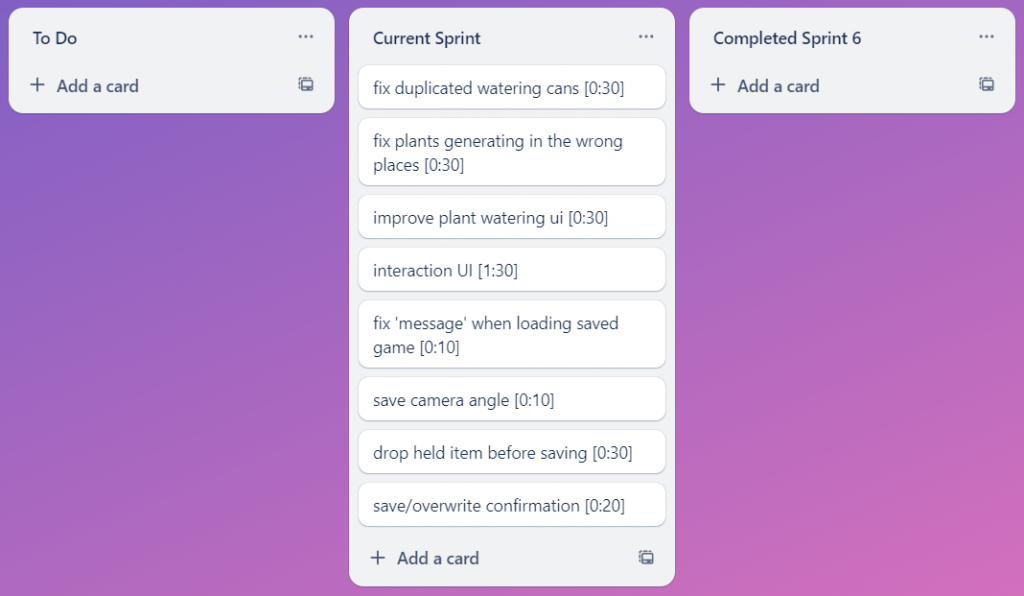
End of sprint 3: 

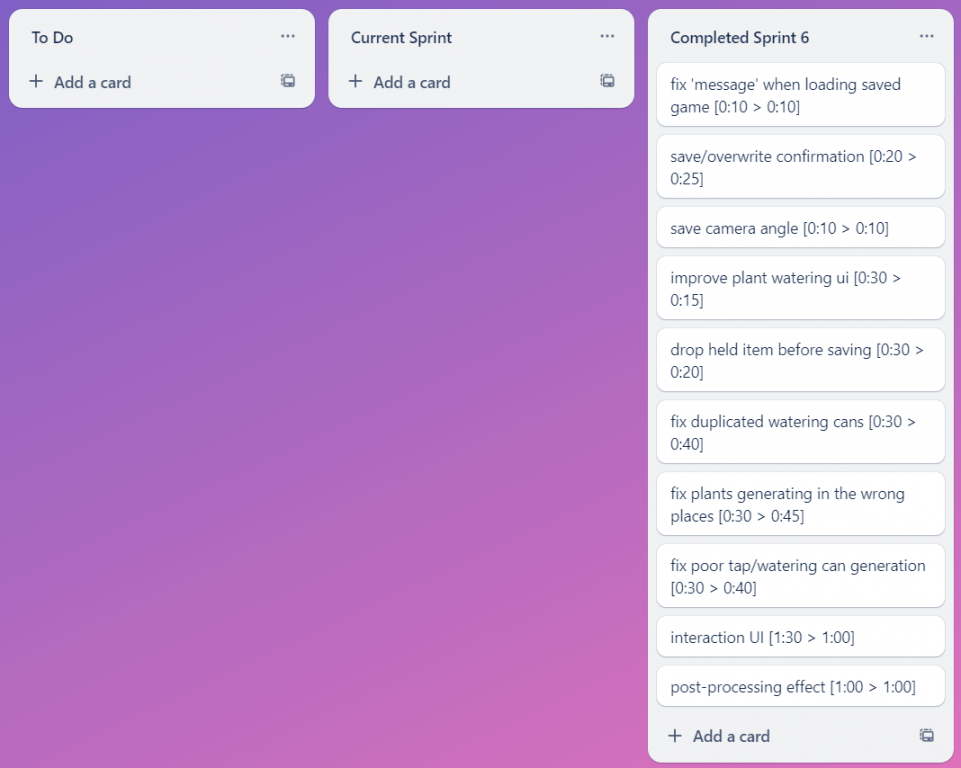
Start of sprint 4: 

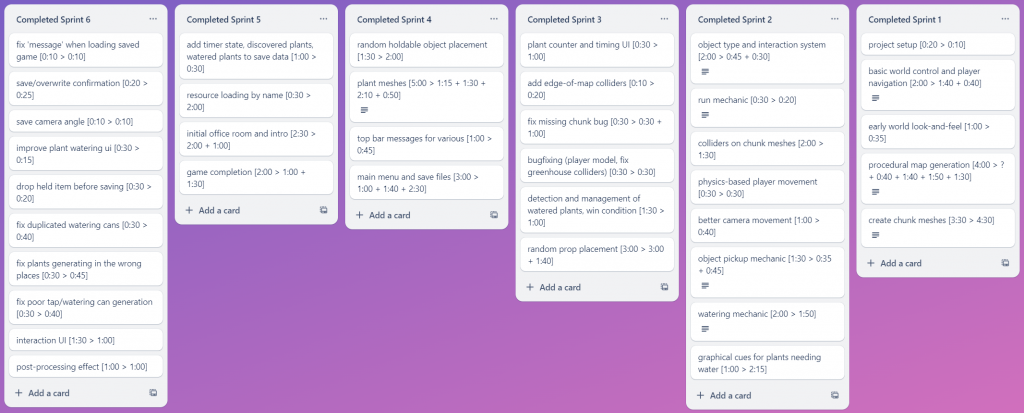
End of sprint 4: 

Start of sprint 5: 

End of sprint 5: 

Start of sprint 6: 

End of sprint 6: 

Entire board: 

# Sprint Time Log

## Prototype 1

I did not keep a proper time log for the first prototype.

## Prototype 2

Sprint 1:  
Tuesday 16/04 - 3:05 (project setup, world control, world look and feel, ~map gen)  
Wednesday 17/04 - 2:20 (more work on world control, ~map gen)  
Thursday 18/04 - 1:40 (~map gen)  
Friday 19/04 - 1:30 (map gen)  
Saturday 20/04 - 4:30 (chunk meshes)

Sprint 2:  
Monday 22/04 - 1:20 (~object types, ~object pickup)  
Tuesday 23/04 - 3:30 (object types, run mechanic, chunk colliders, physics player movement, better camera movement)  
Wednesday 24/04 - 0:45 (object pickup)  
Friday 26/04 - 3:35 (watering mechanic, ~plant UI overlay)  
Saturday 27/04 - 0:30 (plant UI overlay)

Sprint 3:  
Monday 29/04 - 1:50 (plant counter/timing UI, edge of map colliders, ~missing chunk bug)  
Tuesday 30/04 - 2:45 (missing chunk bug, ~plant meshes, bugfixing)  
Wednesday 01/05 - 1:30 (~plant meshes)  
Thursday 02/05 - 2:10 (~plant meshes)  
Friday 03/05 - 4:00 (win condition, ~random prop placement)  
Saturday 04/05 - 1:40 (random prop placement)

Sprint 4:  
Monday 06/05 - 0:0  
Tuesday 07/05 - 3:00 (~save files, random holdable placement)  
Wednesday 08/05 - 0:50 (plant meshes)  
Thursday 09/05 - 0:0  
Friday 10/05 - 2:25 (top bar messages, ~menus)  
Saturday 11/05 - 2:30 (menus, save files)

Sprint 5:  
Monday 13/05 - 0:0  
Tuesday 14/05 - 2:30 (resource loading redo, timer state & plant counts saved)  
Wednesday 15/05 - 2:00 (~office room intro)  
Thursday 16/05 - 2:00 (office room intro, ~game completion)  
Friday 17/05 - 1:30 (game completion)

Sprint 6:  
Saturday 18/05 - 1:00 (fix message, save confirm, save camera angle, plant watering overlay)  
Sunday 19/05 - 2:25 (saving held item, duplicated watering cans, plant generation placement, poor tap generation)  
Monday 20/05 - 2:00 (interaction UI, post processing effect)

1. Popli, R. and Chauhan, N. (2014) 'Cost and effort estimation in agile software development', IEEE Xplore. doi:10.1109/ICROIT.2014.6798284.
2. Shiohama, R. *et al.* (2012) ‘Estimate of the appropriate iteration length in agile development by conducting simulation’, *2012 Agile Conference* [Preprint]. doi:10.1109/agile.2012.16.
3. Kumar, G. and Bhatia, P.K. (2012) 'Impact of agile methodology on software development process', *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, *2*(4), pp.46-50.
4. Nystrom, R. (2014) 'Game programming patterns', United States? Genever Benning.
5. Ulichney, R.A. (1993) *Human Vision, Visual Processing, and Digital Display IV* [Preprint]. doi:10.1117/12.152707.