Ayan Gangopadhyay
M.Sc. BDA 2019-2021

Climbing Celeste

For my Summer Project

Introduction

Celeste

A brief description

- Celeste is a side-scrolling platformer game developed by two friends Maddy Thorson and Noel Berry
- Received massive critical acclaim and was a contender for the game of the year 2018
- Players almost always describe the game as being “fun to play” and has a massive (and very much active) modding and speed-running community
- The physics of the game is extremely well defined and follows consistent rules and which can be interacted with in only a relatively small number of individual actions.
- Level design is non-linear having many branching paths which one can take to reach the end of a level
- Each level broken into multiple transition screens

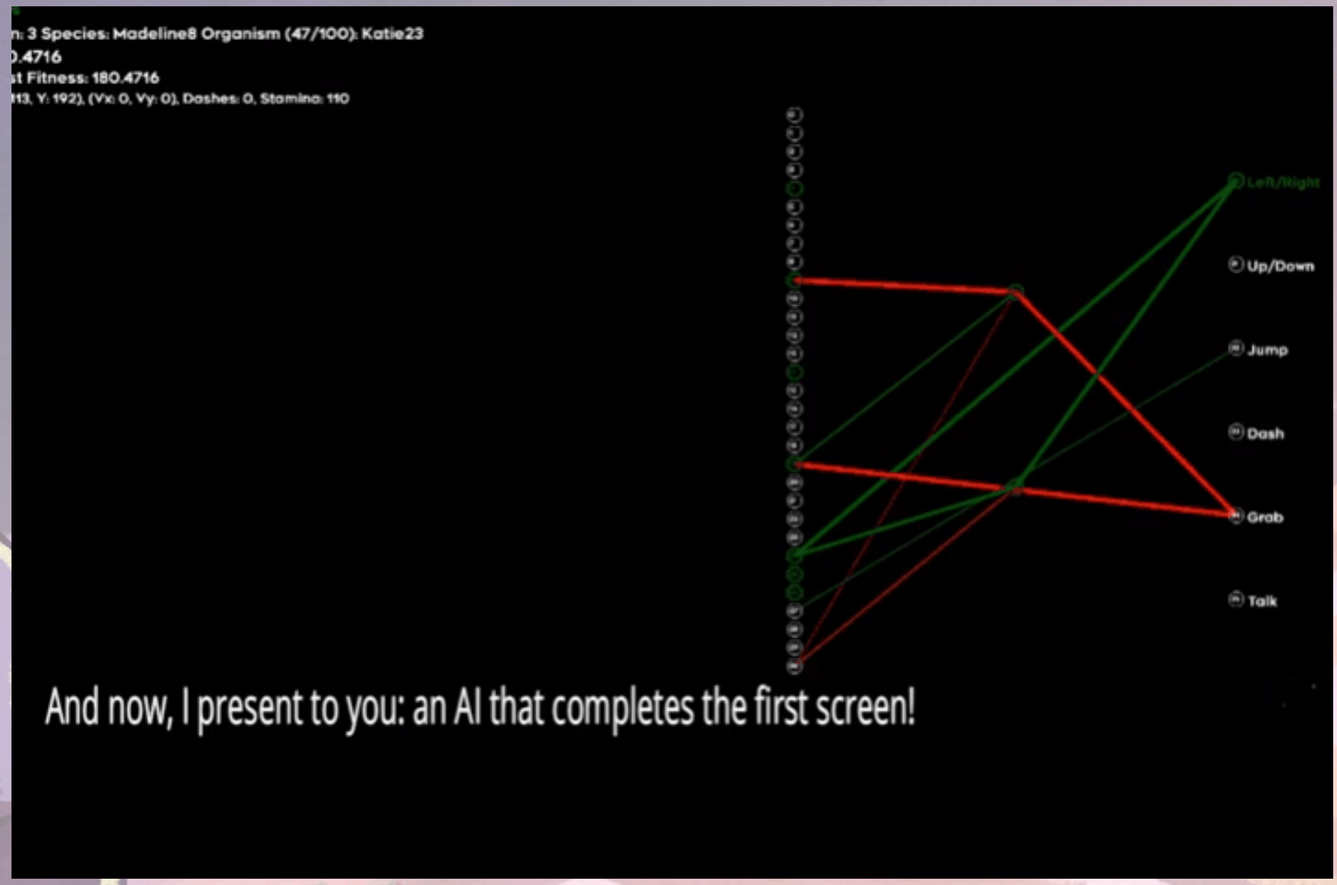
The Player Character Standing at the beginning of a screen of the first level.

Literature Review?

Past Work

Adam Noworolski's approach and results

- Adam Noworoloski is a student at UC Berkeley who tried to solve the same game using a genetic algorithm based approach called NEAT learning
- This did not work very well and could not complete even the tutorial level
- Learning from pixels was infeasible due to the substantial compute required to process image as well as control agent
- NEAT learning uses extremely local information, basically detecting only the surrounding of the player characters and trying to find the best action to take
- Here we see the final outcome of the bot that he created the neural network controlling the bot is seen on the right side
- A little insight into NEAT

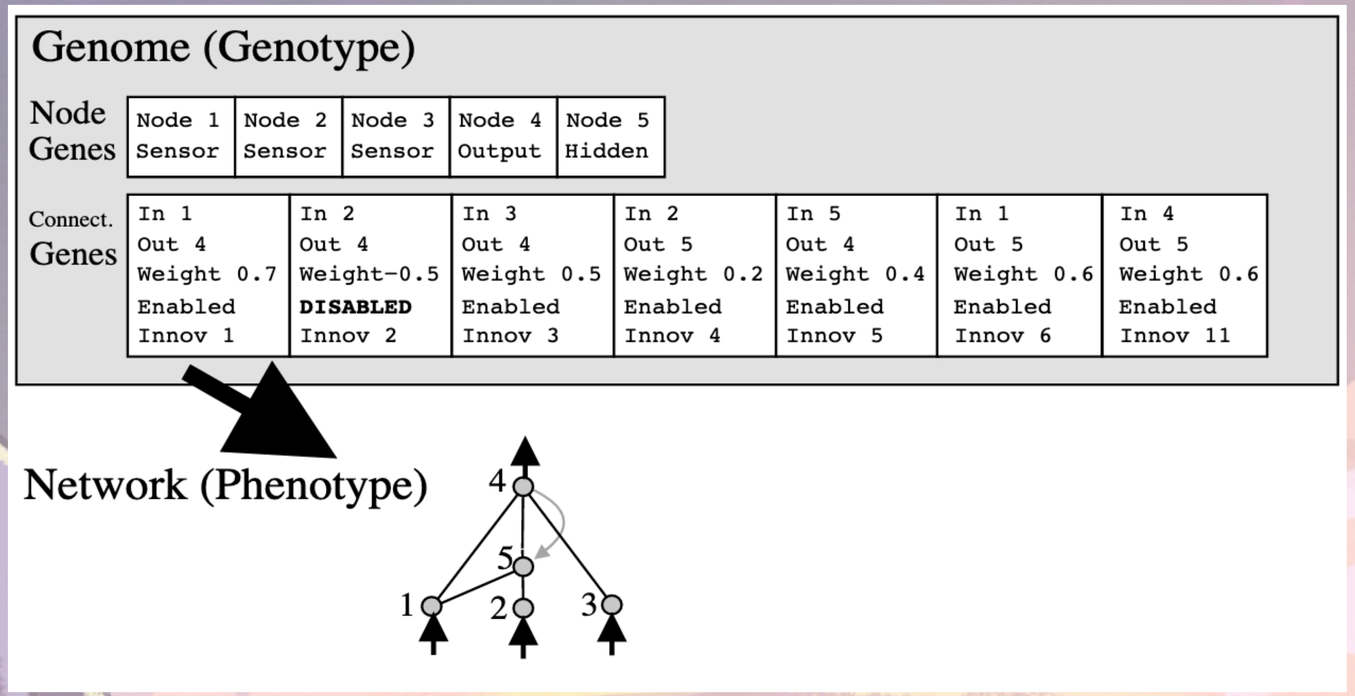


The final outcome of Adam's work.

NeuroEvolution of Augmenting Topologies*

A genetic algorithm to learn neural network architectures

- Basic idea is to use a genetic algorithm to learn a neural network to handle a specific task
- Relatively old paper described in *Evolving Neural Networks throughAugmenting Topologies* by Stanley and Miikkulainen (2002)
- Direct representation of a neural network
- Weights updated using new mutation



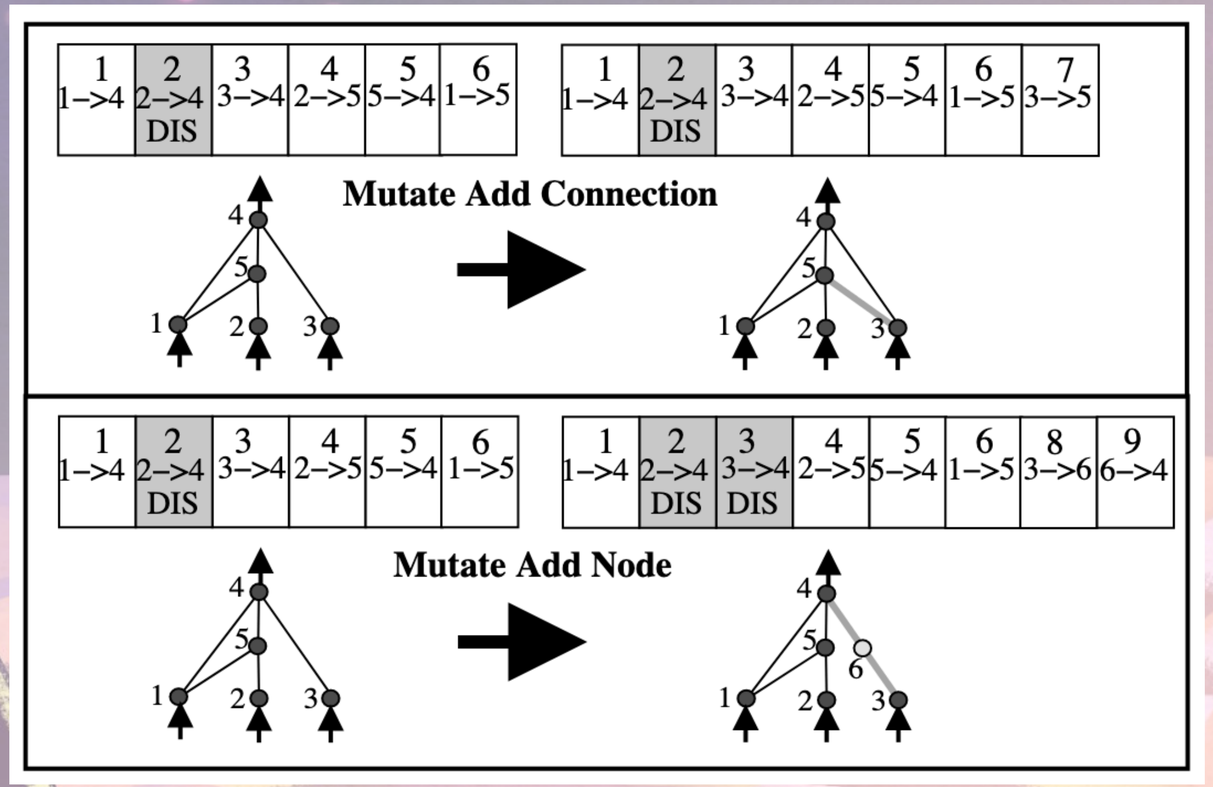
Direct representation of a small neural network used in NEAT.
(Taken from the original paper)

*Much of the information presented here is from an excellent article on NEAT by Hunter Heidenreich (https://bit.ly/3wFiDgz)

NeuroEvolution of Augmenting Topologies

A genetic algorithm to learn neural network architectures

- Mutation
- If new connection is added between two nodes, a random weight is added
- If a new node is added, it is placed between two nodes that are already connected. The previous connection is disabled The previous start node is linked to the new node with the weight of the old connection and the new node is linked to the previous end node with a weight of 1. This was found to help mitigate issues with new structural additions.

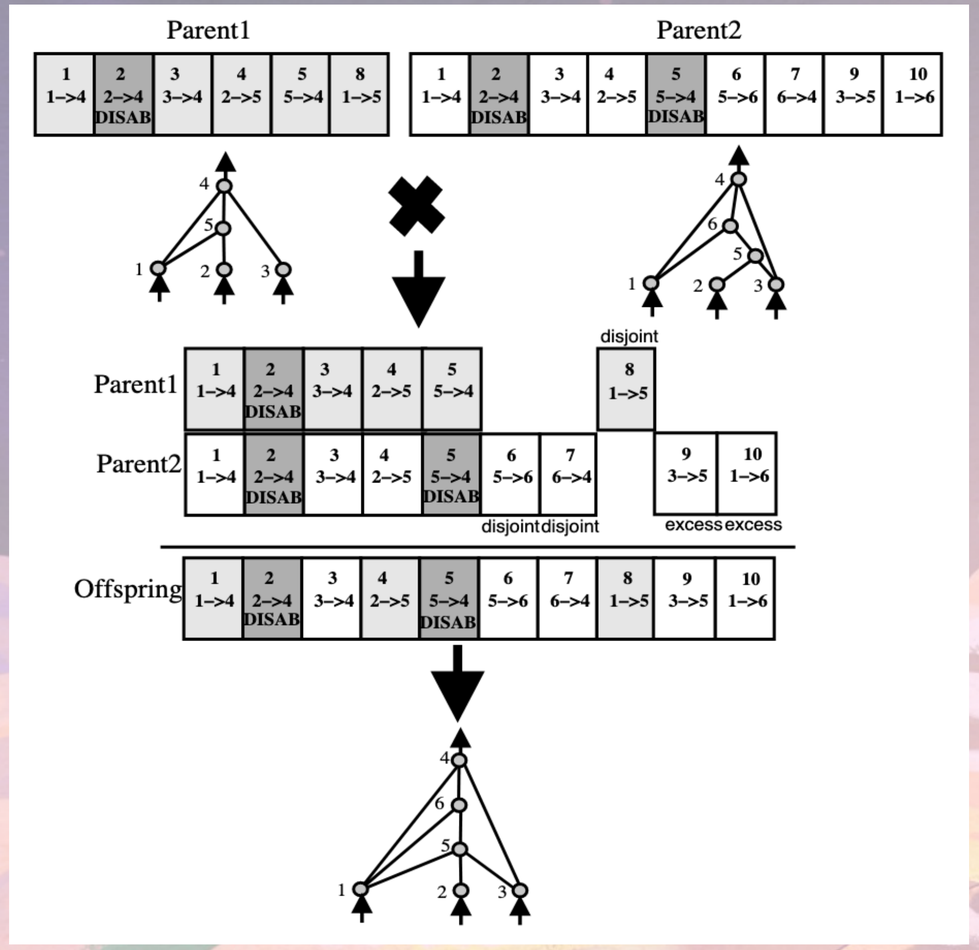


Schematic of mutation as shown in the original paper.

NeuroEvolution of Augmenting Topologies

A genetic algorithm to learn neural network architectures

- Competing Conventions
- It is expected that randomly crossing over different specimens will not give desired results
- We track history using innovation numbers
- A new node or connection when created is assigned innovation numbers ensuring that when crossing over different specimens these are not combined in a way that important connections are deleted

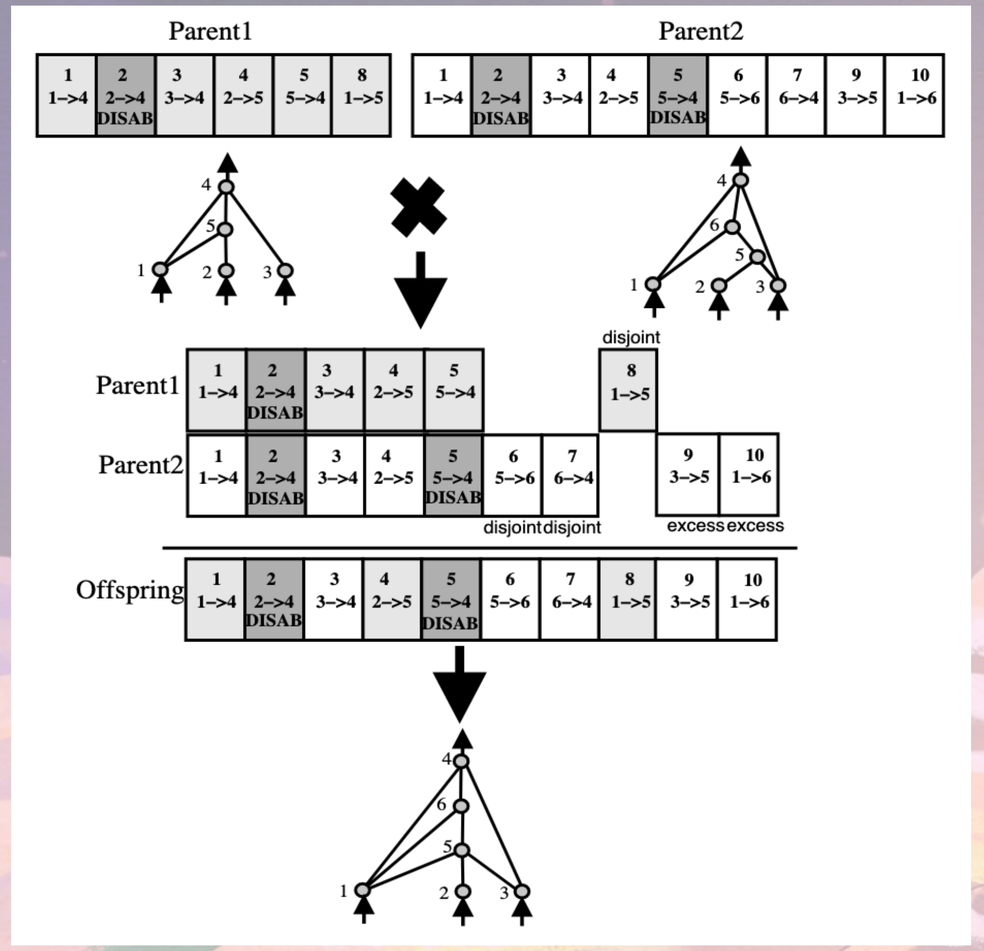


Schematic showing the idea of how innovation numbers are used.
(Taken from the original paper)

Problems

And my approach to them

- Learning from pixel data infeasible due to limited compute availability
- Initial idea was to try to write an agent for the entire game
- Tried simplification of the process by designing simple stages and handling them first
- Windows environment, extremely unfamiliar
- C# and .NET framework also unknown
- Current effort primarily going into creation of environment



Schematic showing the idea of how innovation numbers are used.
(Taken from the original paper)

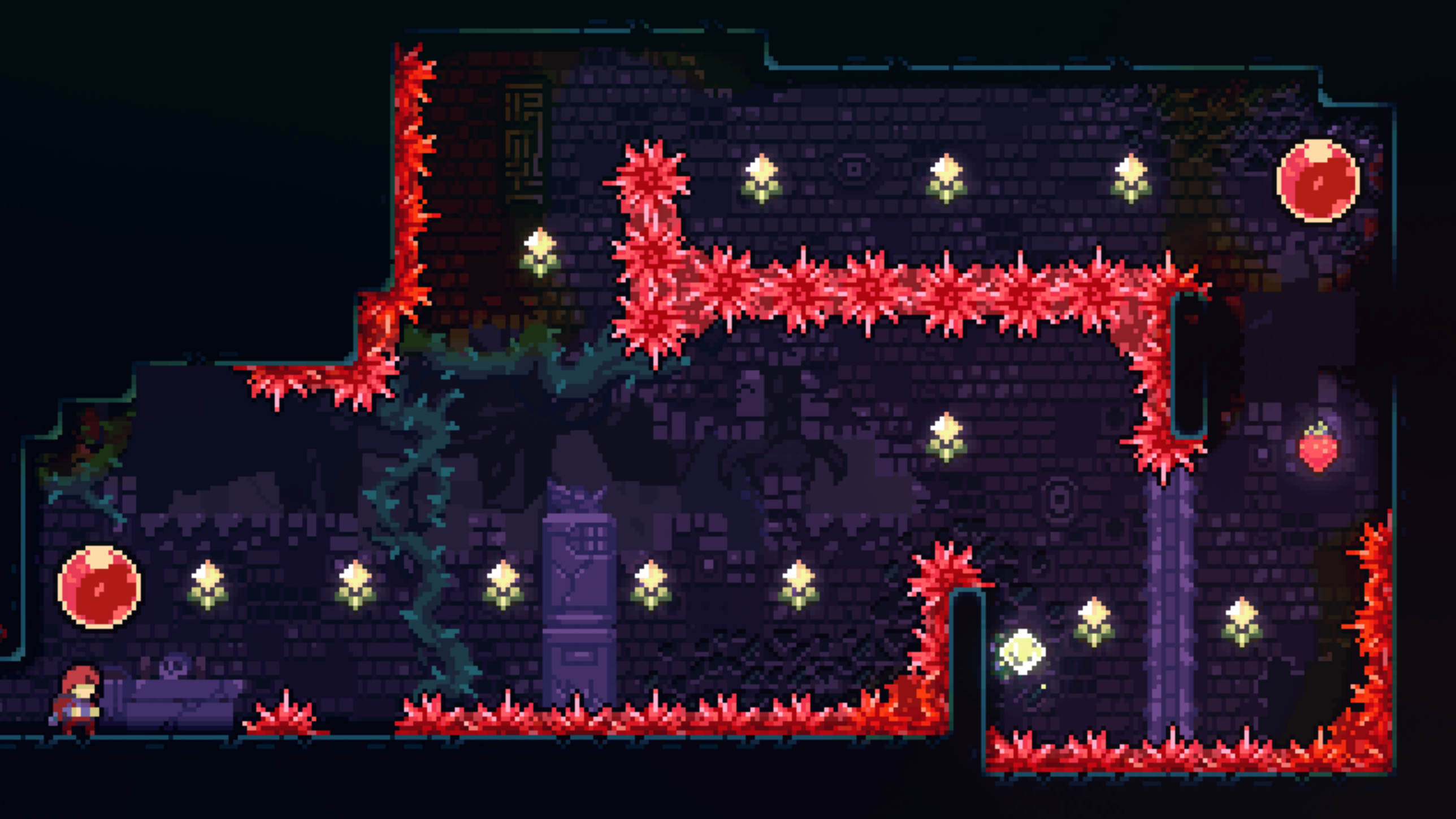
Understanding the Physics of Celeste

Celeste Physics*

Basics

- All of the physics are handled by the classes Solids and Actors
- Solids are, of course, the collide-able level geometry
- Actors are physics objects, such as players, arrows, monsters, treasure chests, etc. Anything that has to move and interact with the level geometry is an Actor
- Solids do not overlap/interact with other solids
- Actors and Solids overlap using collide delegates (which are onCollide events)
- We will see which are Solids and which are Actors in the next gif
- Actors and Solids can only be present in integer locked positions

*A lot of the information presented here is from the blog post by Maddy Thorson on Celeste and Towerfall (another game). (https://bit.ly/34nMUUO)

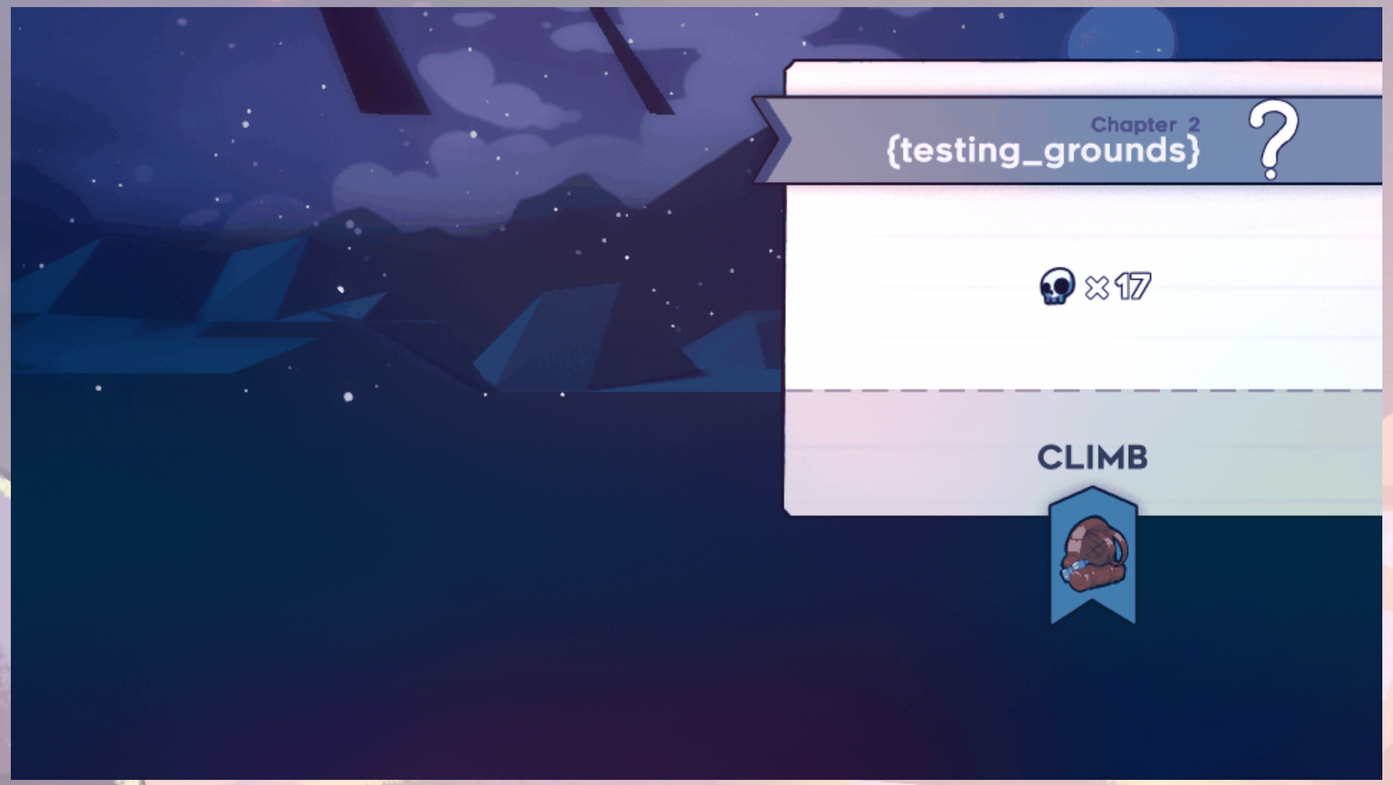


Simplifying the Task

Creating Levels

And how it “Simplifies” our task

- Original game is a lot to take on
- How to make it easier
- Linear levels
- Removal of alternate objectives
- Removal of complex playing techniques
- Task is complicated, define clear goals
- Demonstration of simple levels



Playing through the entire “Simple” level.

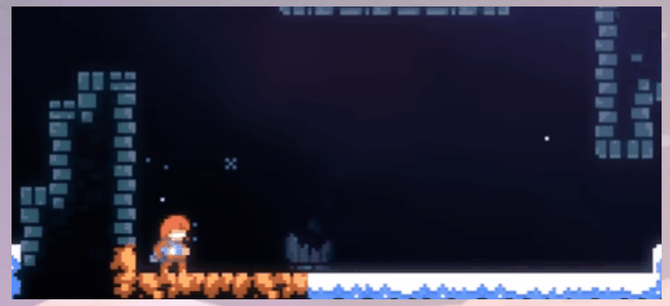
Backend Details

Everest and Code mods

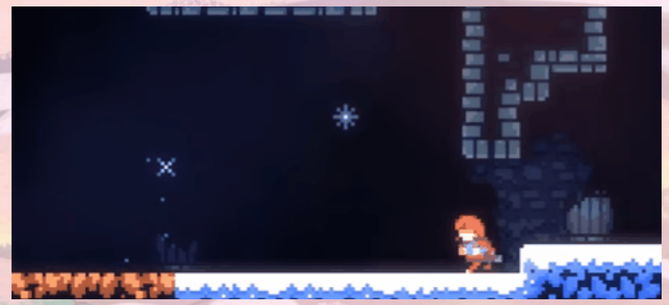
How to actually code an agent and implementation details for Celeste Code Mods

- Adam implemented his bot by implementing it as a mod
- Good amount of C# and .NET framework knowledge needed which I am currently learning
- COM Interop is a technology included in the .NET Framework Common Language Runtime (CLR) that enables Component Object Model (COM) objects to interact with .NET objects, and vice versa
- CelesteTAS is a tool to record and send keypresses to the game, this was forked and used by Adam to allow his agent to interact with the environment, currently trying to learn how this works
- Code mods can be implemented and used to directly interface with the Celeste Game, these can also be used to get a reference for the Player Character and return state information such as Position, Available Dashes, Stamina, Velocity which can be used as states for the Agent
- Main challenge comes from how to define actions for agents, the adjacent gif shows how the relatively simple basic actions can be used to do extremely complicated manoeuvring in the game. This presents a significant modelling challenge which has yet to be solved.

Dash-Cancel



Hyper-Dash



Bounce-Dash

These are some of the variations of “Advanced Movement” in the game. These are born from the simple physics that are present in game.

Present Issues and Progress so Far

Progress Report and Future Work

- Learning .NET Framework and creating an Environment in the form of a mod current priority
- The first iteration of the agent will only learn how to play simple levels
- The challenge of the task is too much to be handled by one person without help and sufficient knowledge, need to have collaborators.
- Currently trying to learn to create code mods by creating documentation and an example module with ColoursofNoise (one of the most active modders for Celeste)
- Little work has been done on the actual modelling front
- Modelling challenges are also significant, the issue of action modelling aside it is difficult to determine if just having local player information is enough to create an efficient agent. Since the game itself contains a lot of state information, we will need to consider carefully what all is relevant to the actual agent. Obviously the minimal local information utilised by the NEAT agent created by Adam was not enough.

In Conclusion

Final Comments

- Need help to model the agent
- Much work required before the environment is usable
- But interesting project
- Possible application of Hierarchical RL
- Long term project, will welcome an opportunity to work with professors here to bring to fulfilment


Thank You