SICP

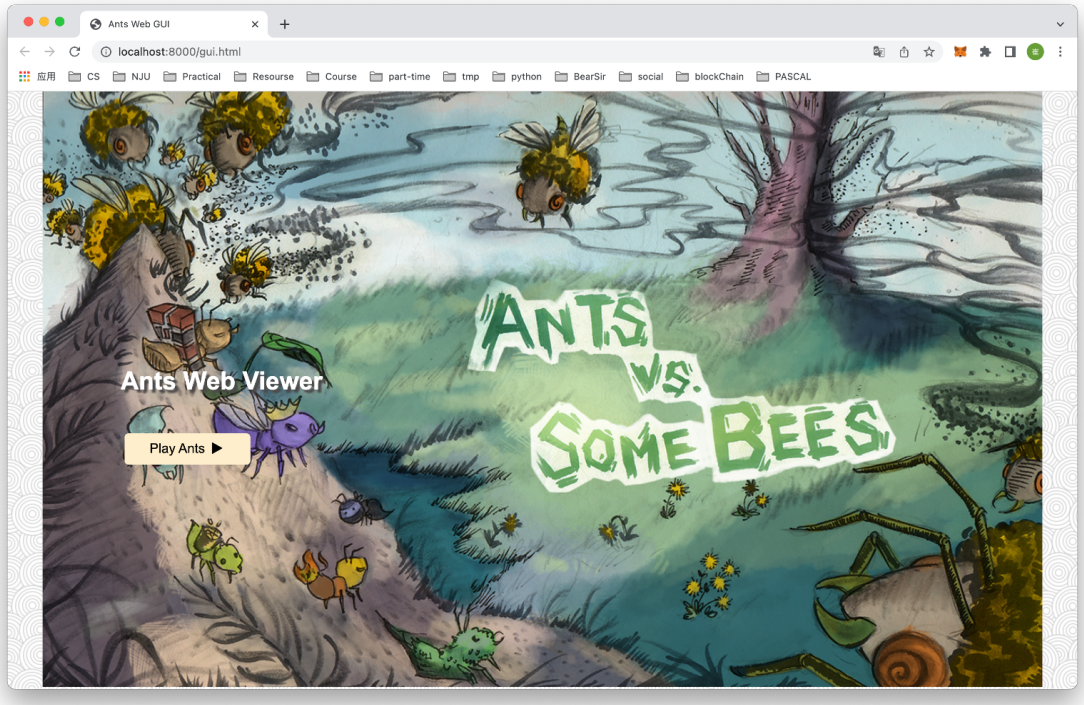
God's Programming Book

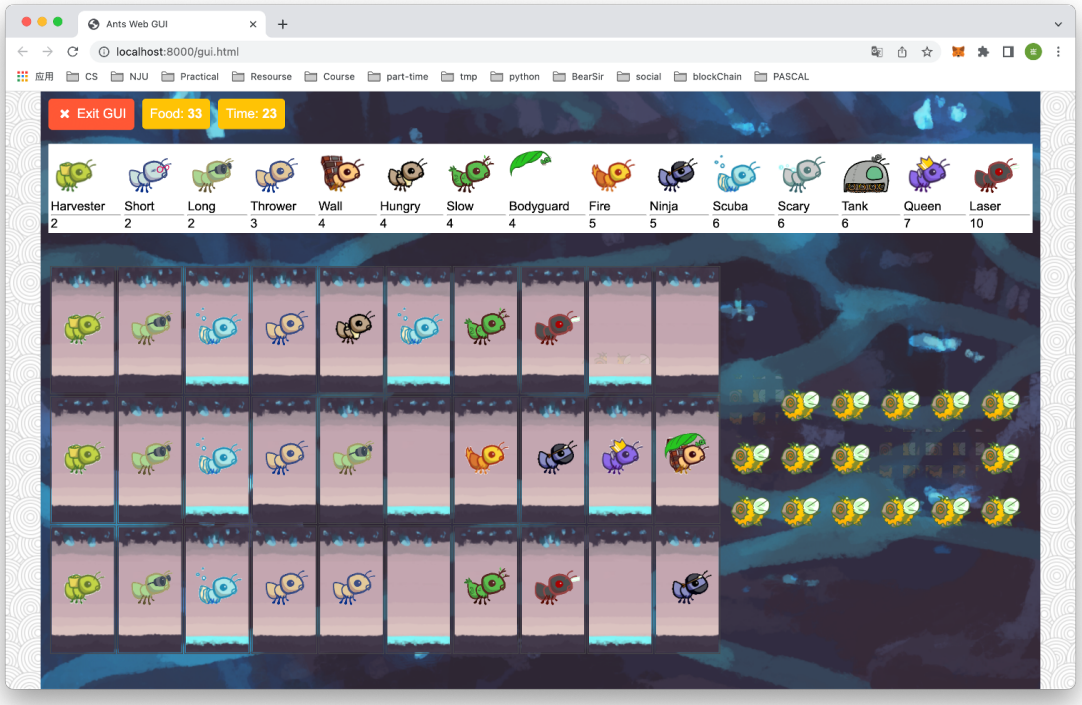
Project-03 Ants

Project Adapted from cs61a of UC Berkeley

Goal

What will you have after the project?





A Tower Defense Game Called *Ants Vs. SomeBees*

Materials

What have you got before the project?

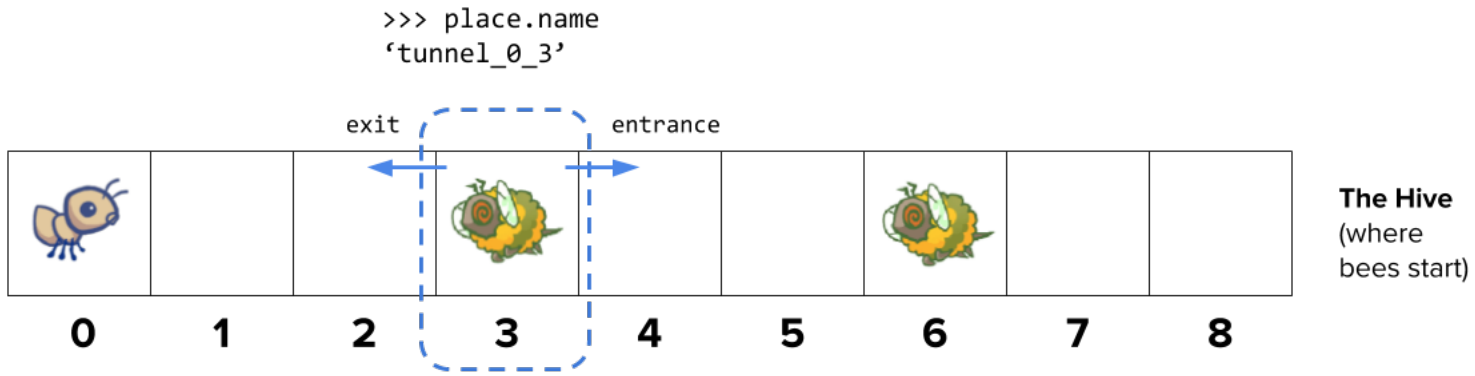
- Skeleton code of the project and an autograder ok
- A detailed handout covering everything about the project
- My version of solution

https://github.com/JacyCui/sicp-proj03.git

Core Concepts

Core Concepts

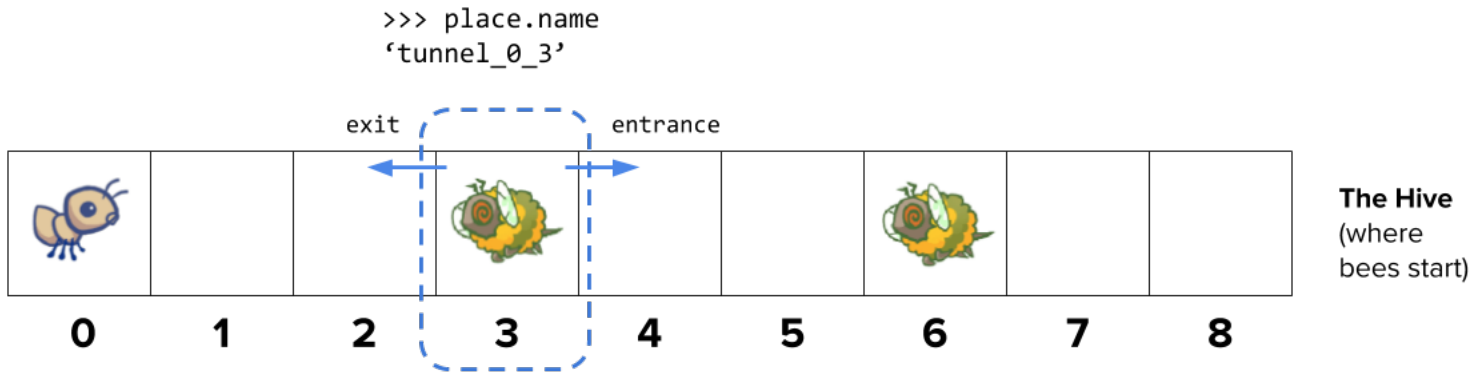
- **The Colony**.
  - This is where the game takes place. The colony consists of several *places* that are chained together to form a tunnel where bees can travel through. The colony has some quantity of food that can be expended to deploy ant troops.



Example: AntColony with dimensions (1, 9)

Core Concepts

- **Places**.
  - A place links to another place to form a tunnel. The player can place a single ant into each place. However, there can be many bees in a single place.
- **The Hive**.
  - This is the place where bees originate. Bees exit the beehive to enter the ant colony.



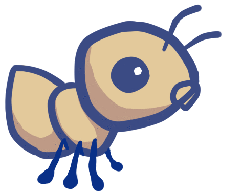
Example: AntColony with dimensions (1, 9)

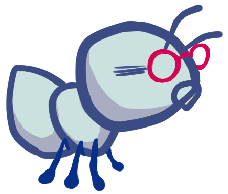
Core Concepts

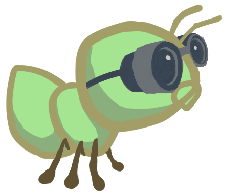
- **Ants**.
  - Ants are the usable troops in the game that the player places into the colony. Each type of ant takes a different action and requires a different amount of food to place. The two most basic ant types are the
    - HarvesterAnt , which adds one food to the colony during each turn, and the
    - ThrowerAnt , which throws a leaf at a bee each turn.
    - You will be implementing many more.

Ants Family


HarvesterAnt


ThrowerAnt

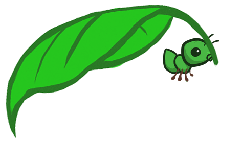

ShortThrower

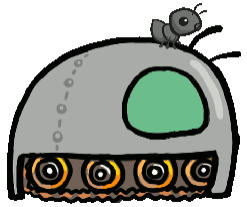

LongThrower


FireAnt


WallAnt


HungryAnt


BodyguardAnt

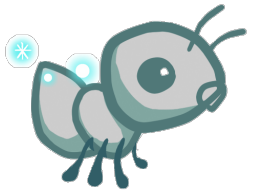

TankAnt


ScubaThrower


QueenAnt


NinjaAnt


SlowThrower


ScaryThrower


LaserAnt

Core Concepts

- **Bees**.
  - Bees are the antagonistic troops in the game that the player must defend the colony from. Each turn, a bee either advances to the next place in the tunnel if no ant is in its way, or it stings the ant in its way. Bees win when at least one bee reaches the end of a tunnel.

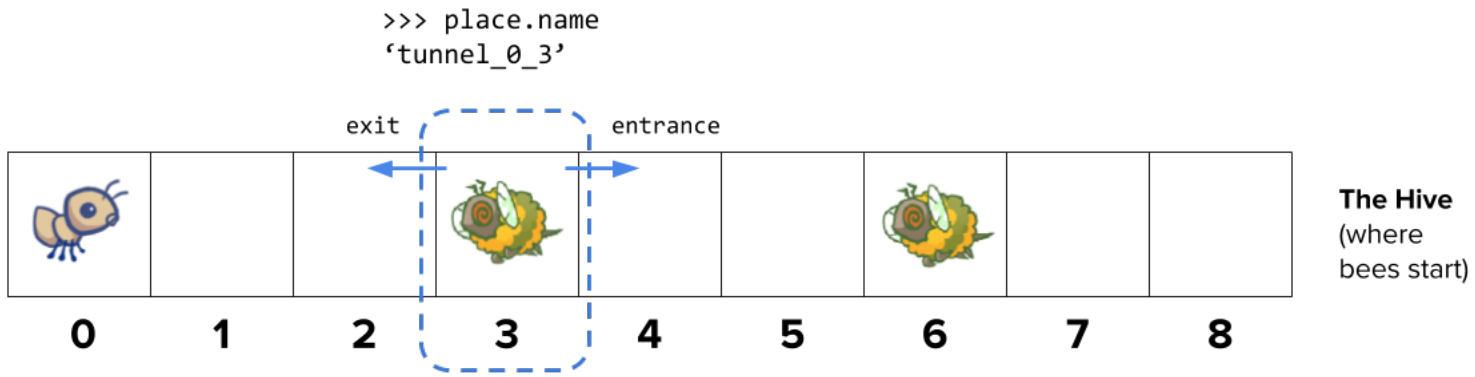
Core Classes

Core Classes

- **GameState** :
  - Represents the colony and some state information about the game, including how much food is available, how much time has elapsed, where the AntHomeBase is, and all the Place s in the game.

Core Classes

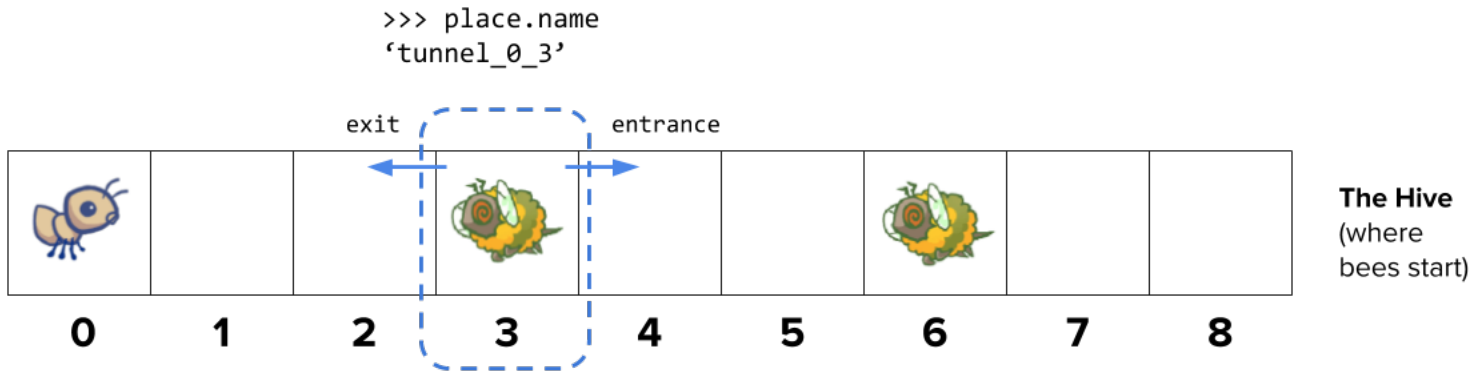
- **Place** :
  - Represents a single place that holds insects. At most one Ant can be in a single place, but there can be many Bee s in a single place. Place objects have an exit to the left and an entrance to the right, which are also places. Bees travel through a tunnel by moving to a Place 's exit .



Example: AntColony with dimensions (1, 9)

Core Classes

- **Hive** :
  - Represents the place where Bee s start out (on the right of the tunnel).
- **AntHomeBase** :
  - Represents the place Ant s are defending (on the left of the tunnel). If Bees get here, they win :(



Example: AntColony with dimensions (1, 9)

Core Classes

- **Insect** :
  - A superclass for Ant and Bee . All insects have health attribute, representing their remaining health, and a place attribute, representing the Place where they are currently located. Each turn, every active Insect in the game performs its action .

Core Classes

- **Ant** :
  - Represents ants. Each Ant subclass has special attributes or a special action that distinguish it from other Ant types. For example, a HarvesterAnt gets food for the colony and a ThrowerAnt attacks Bee s. Each ant type also has a food_cost attribute that indicates how much it costs to deploy one unit of that type of ant.

Core Classes

- **Bee** :
  - Represents bees. Each turn, a bee either moves to the exit of its current Place if the Place is not blocked by an ant, or stings the ant occupying its same Place .

Requirements

What do you need to finished this project?

- Trees
- Mutation
- Iterators
- Objects
- Inheritance

Tips

What you should keep in mind?

- Always figure out what you need to do before writing codes.
- Keep it simple and elegant.
  - Normally no more than 20 lines for each problem.
- Take a challenge to conquer all the optional problems.
- You need to enable proxy on to run the web GUI properly.

Thanks for Listening