Evolution Help Slides
A Quick Reminder

Fill out the **Final Project Declaration Form** by Saturday midnight!

- This form is *binding*
- Starting Sunday, you will have to email the HTAs to switch your FP.
Part 1: Flappy Bird
Game Specs

- Just like Flappy Bird, but the pipes are always scrolling at a constant rate
  - No platform scrolling!!!!!!
- Birds move completely independent of the pipes
  - Similar to Doodle — they fall due to gravity and jumping resets the velocity to some constant
- Birds die when they hit a pipe
- Constant horizontal distance between pipes
- Vertical gap in pipes randomly generated—try to avoid them going too close to the bottom/top of the game window for playability purposes
Suggestions

● Use good OOP to set yourself up well for Part 2
  ○ Ex: you could code this pretty easily without a Bird class, but it will make Part 2 a lot harder if you don’t have one

● Have a way to access your nearest Pipe
  ○ Technically you only need to check for collision with the Pipe nearest to your Bird
  ○ This will make it easier to calculate inputs your Birds to decide whether or not to jump in Part 2
Part 2: Decision Making and Learning
What is Machine Learning?

- Machine learning is used to solve one of two tasks
  
  - Classification
    - The learning of a function that classifies objects or scenarios based on historical data
    - Should we sell given these market conditions?
    - Does this person have a life-threatening disease given these symptoms?
    - Is this vehicle a car, truck, bike given wheel count, size, horsepower?

  - Decision making
    - Given information about the current environment and a history of past actions and their results, what should we do to maximize some reward?
      - Jump or not jump?
      - Turn left or right? Speed up or slow down?

- Evolution would fall under decision making.
What is a Neural Network?

- A neural network is a model of a brain
- Receive inputs → perform computation → return output → learn
- Nodes (circles)
  - A node contains a value. Nodes store data used for later calculations.
- Weights (arrows)
  - Value between -1 and 1
  - Multiplied by node values and stored in other nodes
  - Held in a 2D array called the weights array
  - Changes during the learning process between generations.
Input Layer Values

- The values of input nodes are descriptors of an agent’s environment.
- More inputs allow agents to use more information when making a decision, and may eventually lead to faster convergence.
- Students should experiment with different input value selection and compare results.
Hidden Layer Values

- Values in hidden layer are arbitrary and carry no apparent meaning. The hidden layer stores an intermediate value in computation.
- The size of the hidden layer can be changed, which increases or decreases the complexity of the Neural Network.
- You should experiment with different amounts of hidden layer nodes and compare results.
Output Layer Values

- Value of the output node(s) determines the action that the agent will take
- Each output node typically corresponds to an action
  - For example, a Neural Network learning how to play DoodleJump could have one output value between 0 and 1:
    - 0.5 or less would mean move left, higher than 0.5 would mean move right
- What might the output node(s) be for FlappyBird?
Thinking (Forward Propagation)

1. Receive input values
2. Multiply input nodes with the first weights matrix ($W$) to compute hidden layer
3. Multiply hidden layer nodes with the 2nd weights matrix ($W'$) to compute output layer
4. Make a decision based on value of output node(s)

Note: if you want, you can think of this as a 2D dot product:

$h = x \cdot W$
$p = h \cdot W'$
Using your Neural Network

- The NeuralNetwork is how your Birds think!
- Each Bird should have its own NeuralNetwork, and use it to decide whether or not to jump
- This can be broken up into 3 steps
  - SEE: Calculate input node values
  - THINK: Use forward propagation in the Neural Network to receive output node(s)
  - DO: Based on value of output node(s), perform an action
Learning

- With classification problems, we use backwards propagation to learn, which nudges weights to change output based on error
  - This is complicated
- Instead with decision making problems, we use neuroevolution to change the weights
Neuroevolution and Implementation
Start

- Weights are all **randomized** to begin
- Some birds will constantly jump (**blue**)
- Some birds will never jump (**green**)
- Others will jump sometimes (**red**)

![Evolution Simulation](image)
Evolution Process

● Selection
  ○ After all birds die, the fittest birds (birds that survive the longest) will pass on their weights to the new generation

● Mutation
  ○ Every weight has a small chance of being changed to a random value, or nudged in either direction, providing more diversity in offspring
Population

- In order to implement these properties of neuroevolution, you’ll need to have a way to keep track of all your Birds
- You will have to keep track of the fitness of each Bird
- The population should be no more than 50
  - Challenge yourself — smaller populations will learn slower, so if your design quickly becomes optimized, decrease the population!
Design Questions
Food for Thought

- In code, how will you encapsulate the game and the neural net?
- How will you keep track of each bird’s fitness? How will you track a population’s performance as a whole?
Evolution is a super new project!

- Our TAs might not be as familiar with this project as others, but they have been working on this all semester, and are fully dedicated to helping you.
- This is a semi-independent project—explore on your own, but our TAs will always be here for advice.