UNIVERSITY OF PASSAU

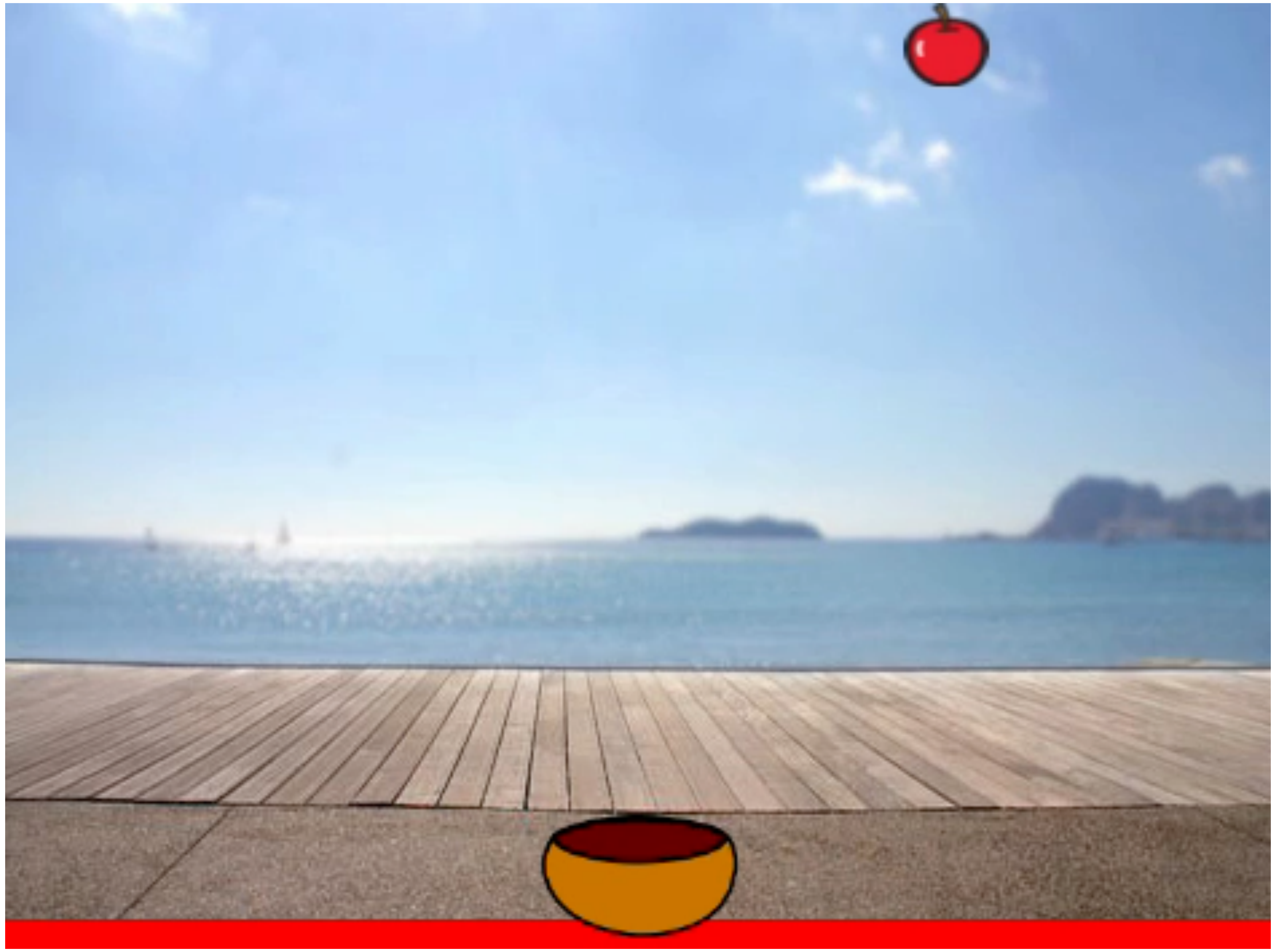
Faculty of Computer Science and Mathematics

Neatest

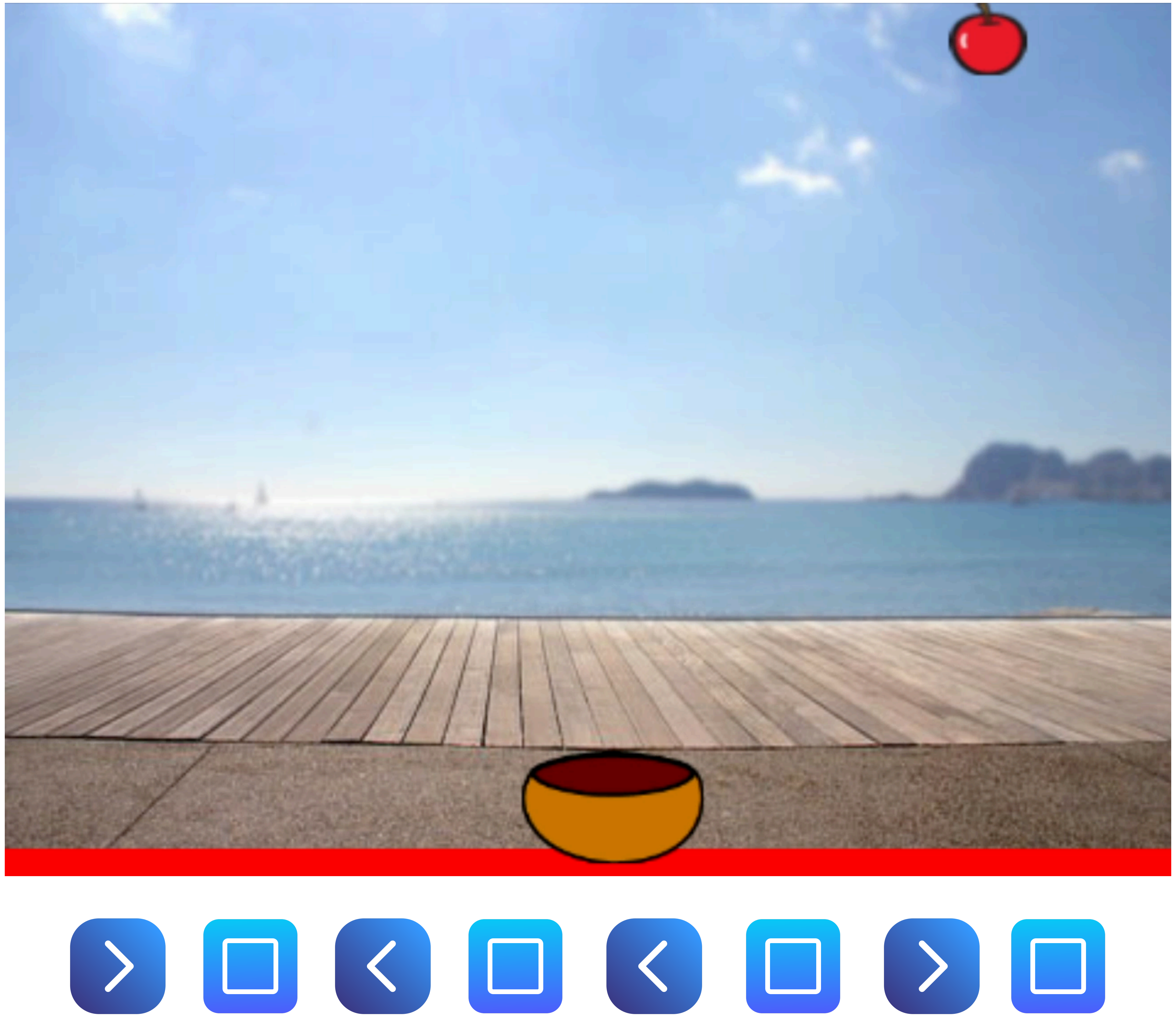
Neuroevolution-based Generation of Adaptive Tests

Patric Feldmeier, 23.11.2023

Testing Games

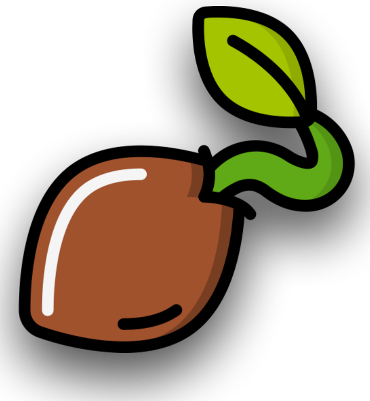


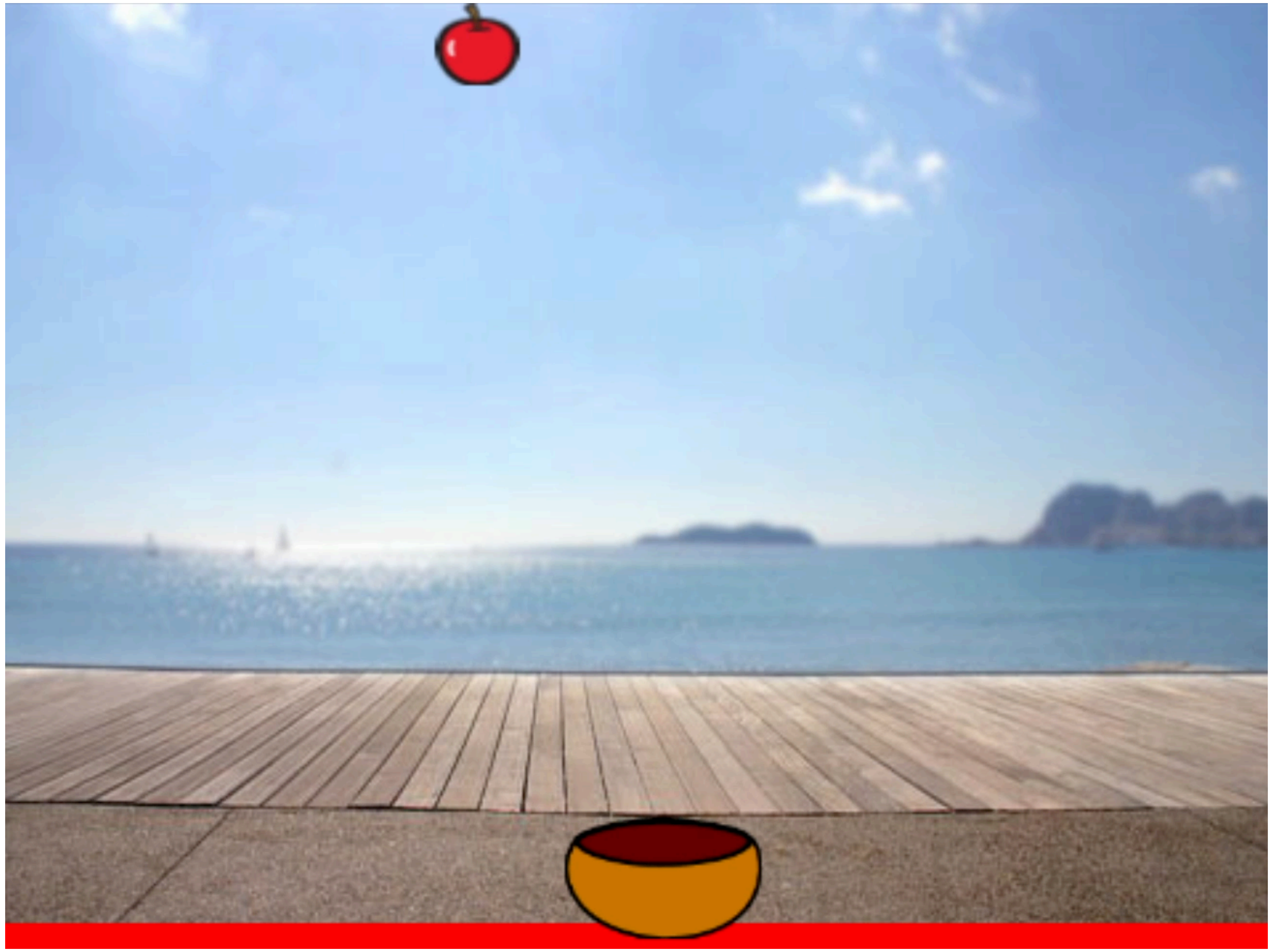
Testing Games



Challenges of Game Testing

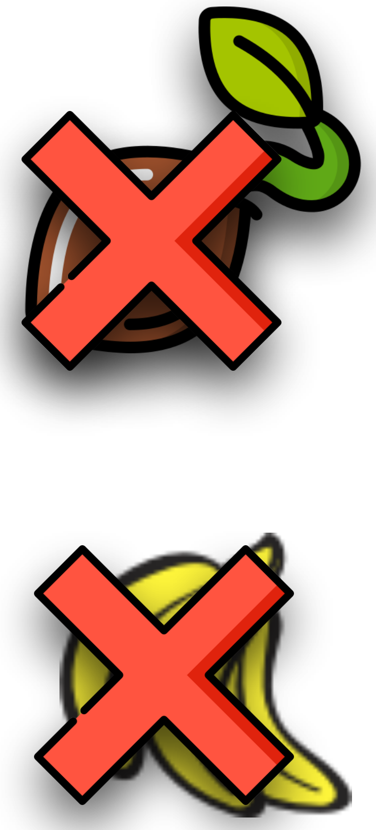
Randomisation

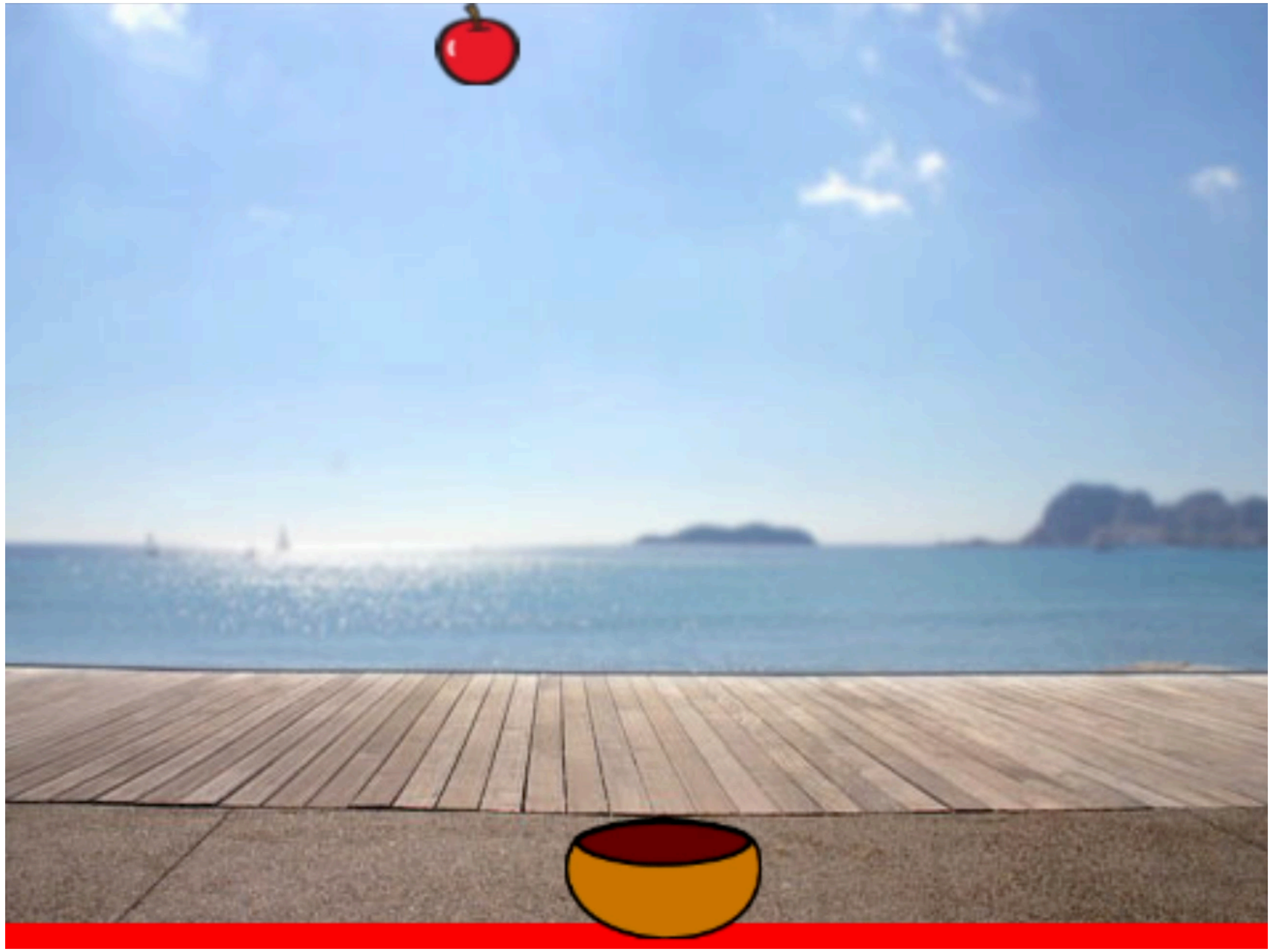




Challenges of Game Testing

Randomisation



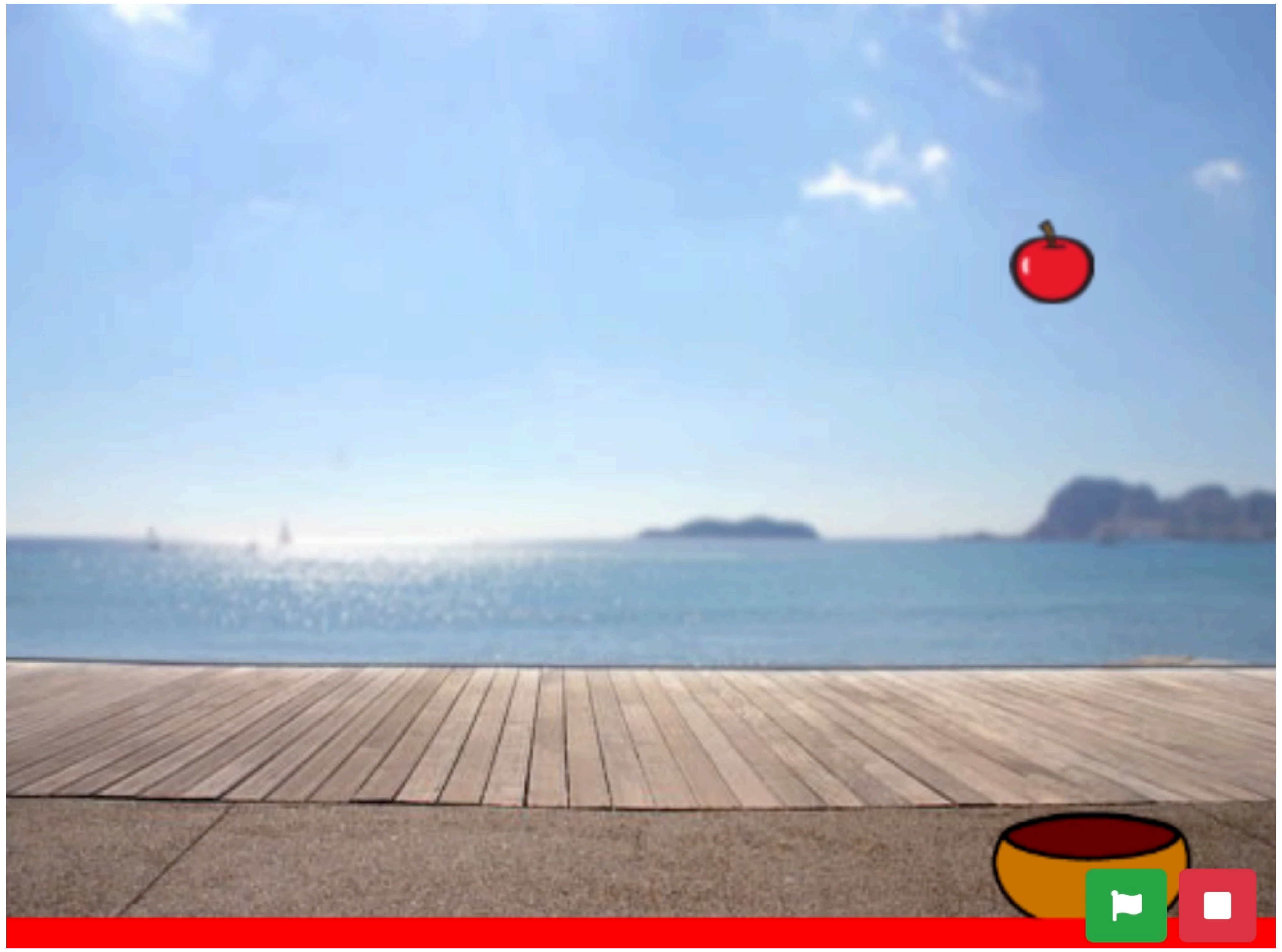


Challenges of Game Testing

Randomisation

Challenging program statements

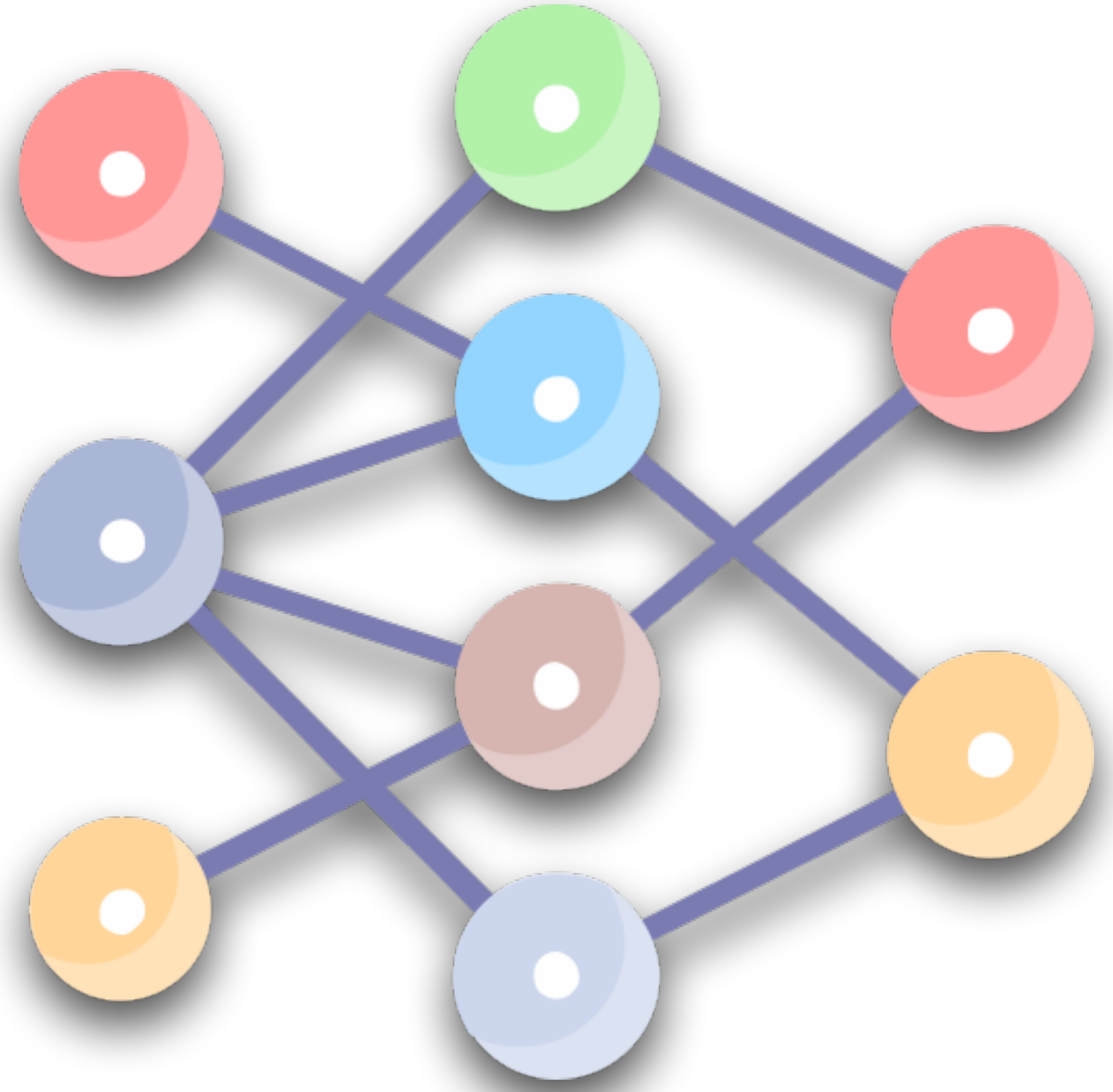
Neatest



Learn to play

Validate behaviour

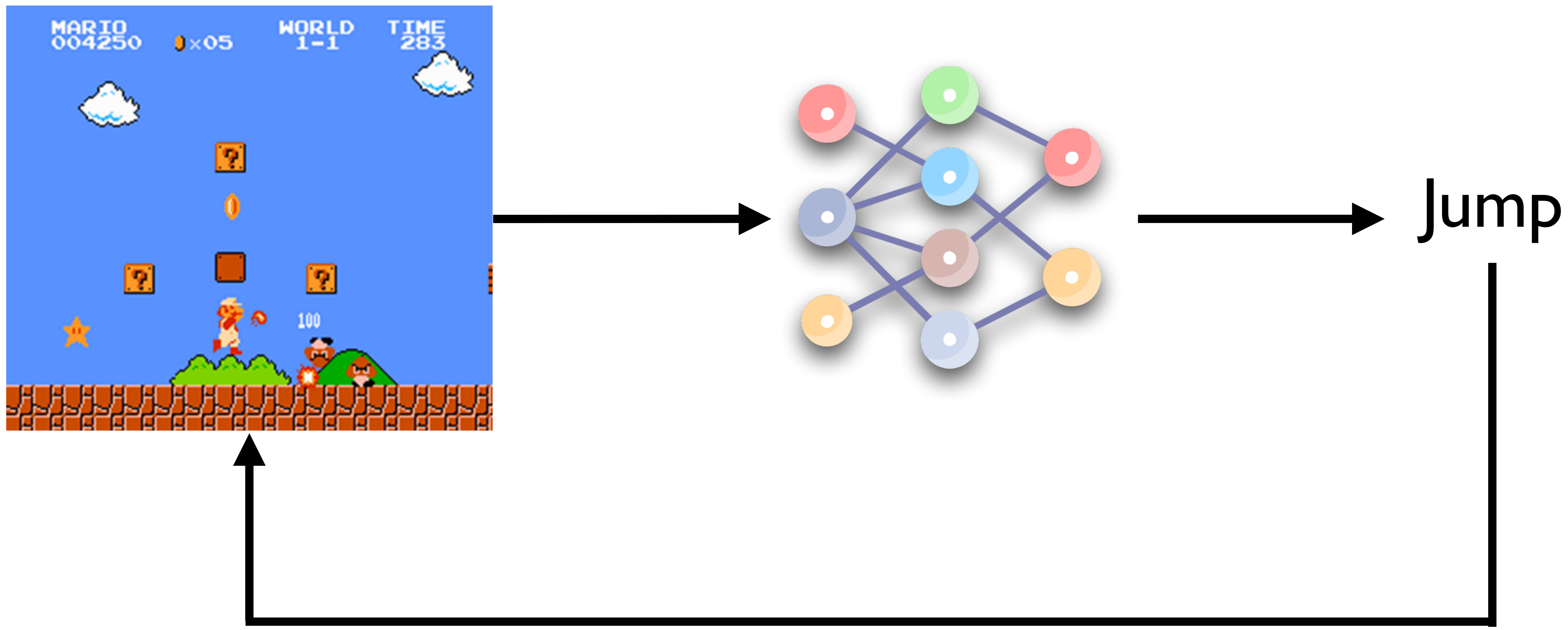
Dynamic Test Suites



Playing Super Mario Using Neural Networks

- Player (Mario) has to travel to the right as far as possible
- Game Over if the player touches an enemy or falls into an hole
- Mario can be navigated to the left/right using the left/right arrow keys and can jump using the space bar

Playing Super Mario Using Neural Networks

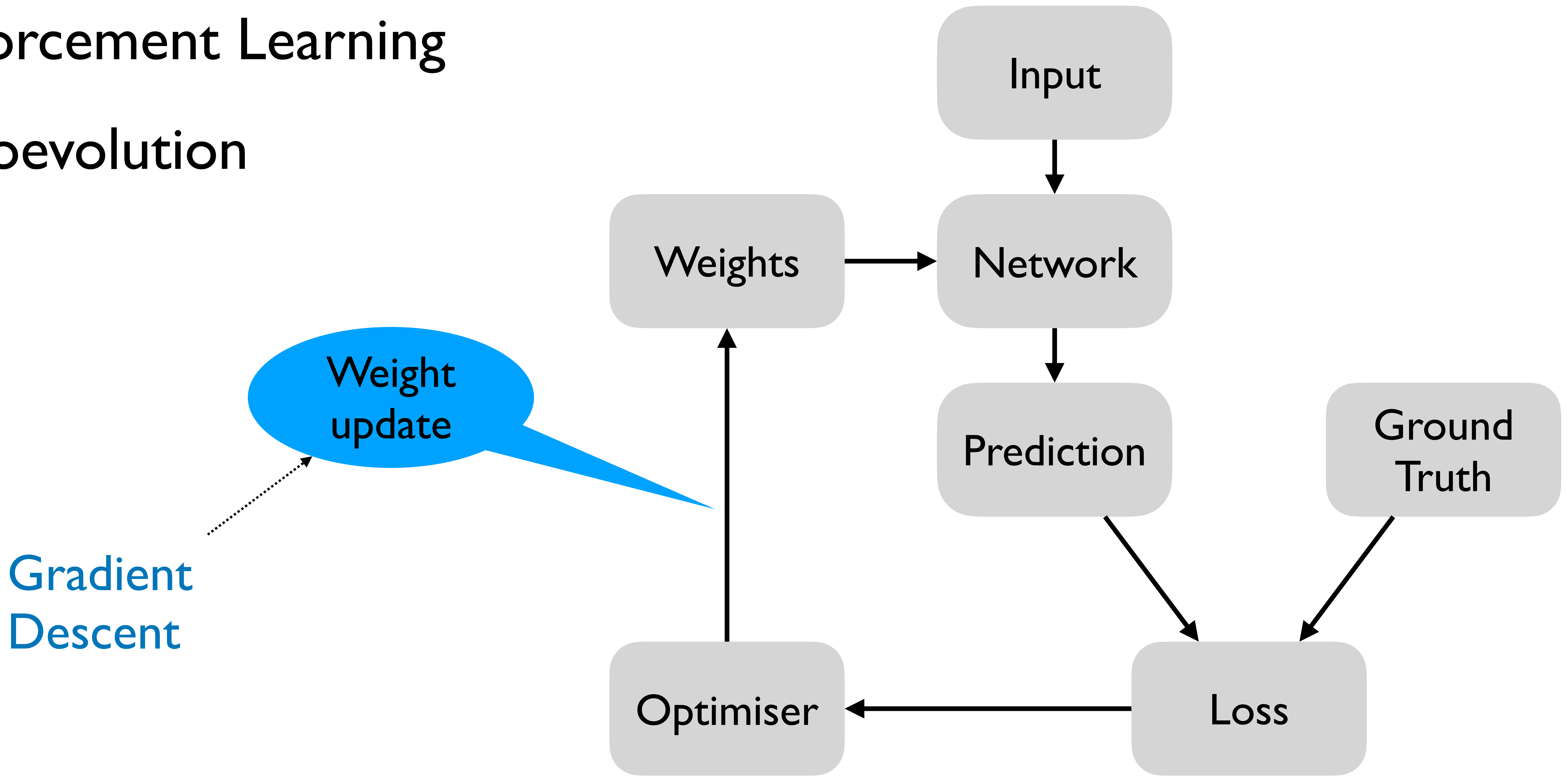


Supervised Optimisation

Problem: requires manually labelled data

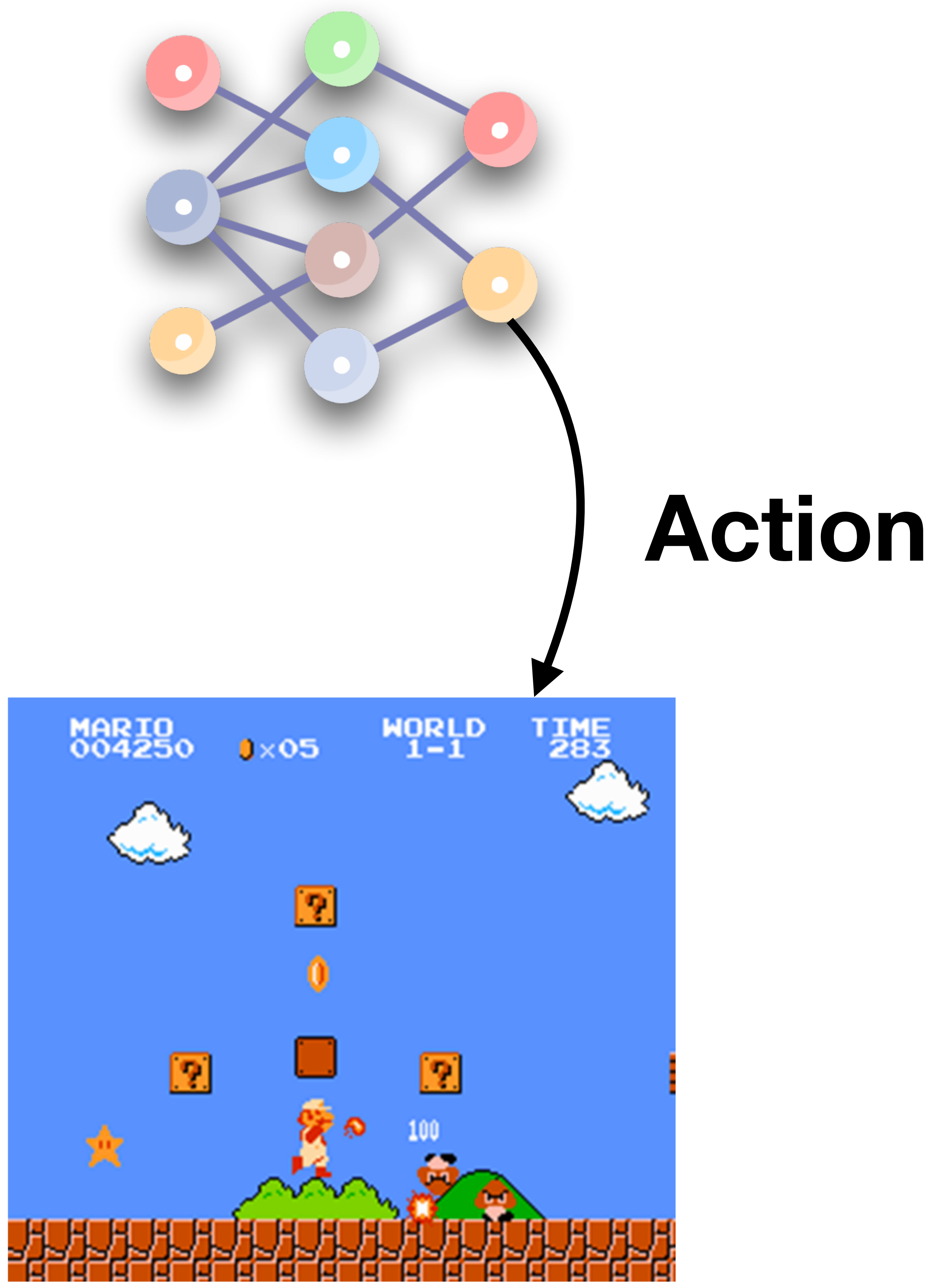
➡ Reinforcement Learning

➡ Neuroevolution



Reinforcement Learning

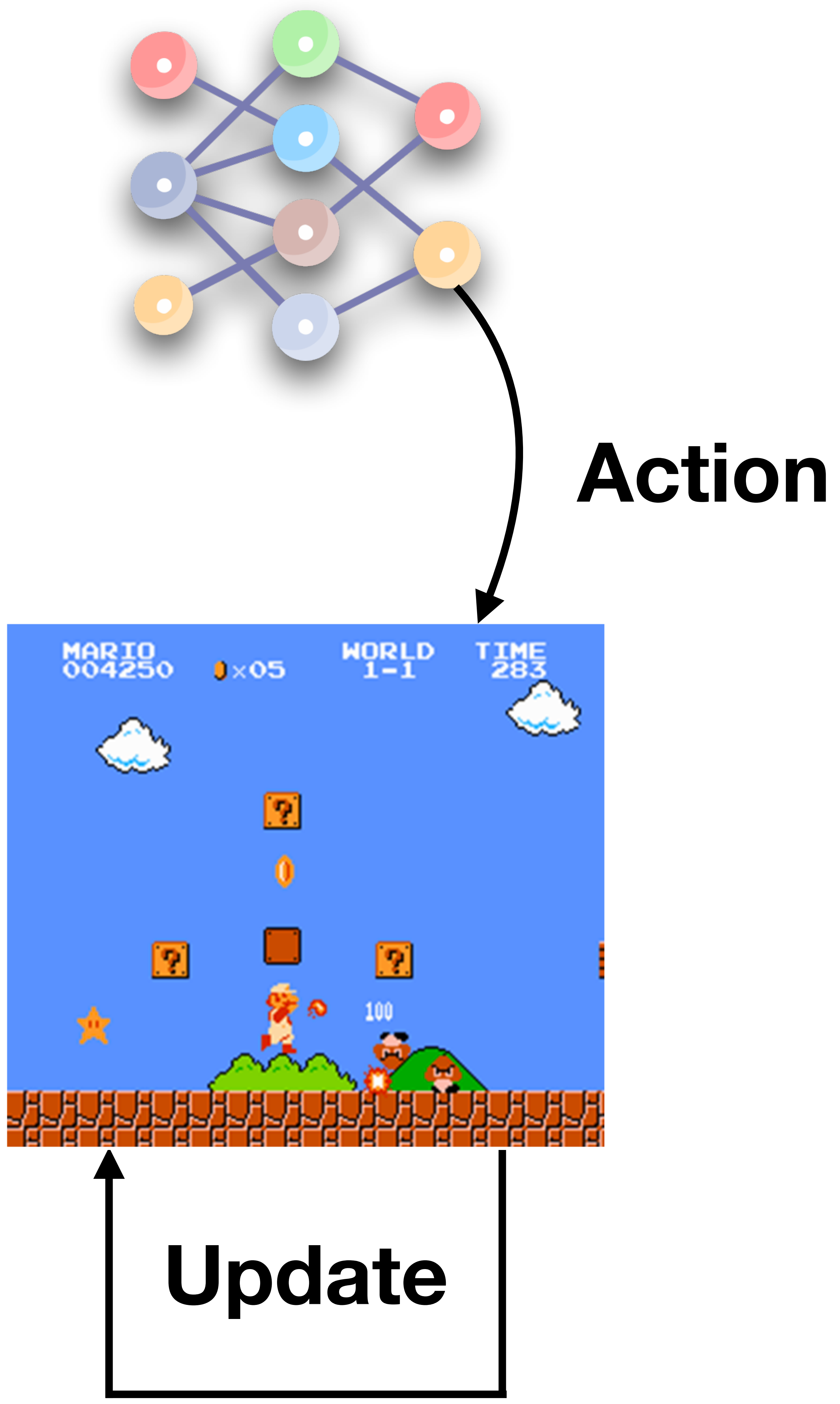
- An agent is placed into an environment in which, it can perform certain actions.



Reinforcement Learning

- An agent is placed into an environment in which, it can perform certain actions.

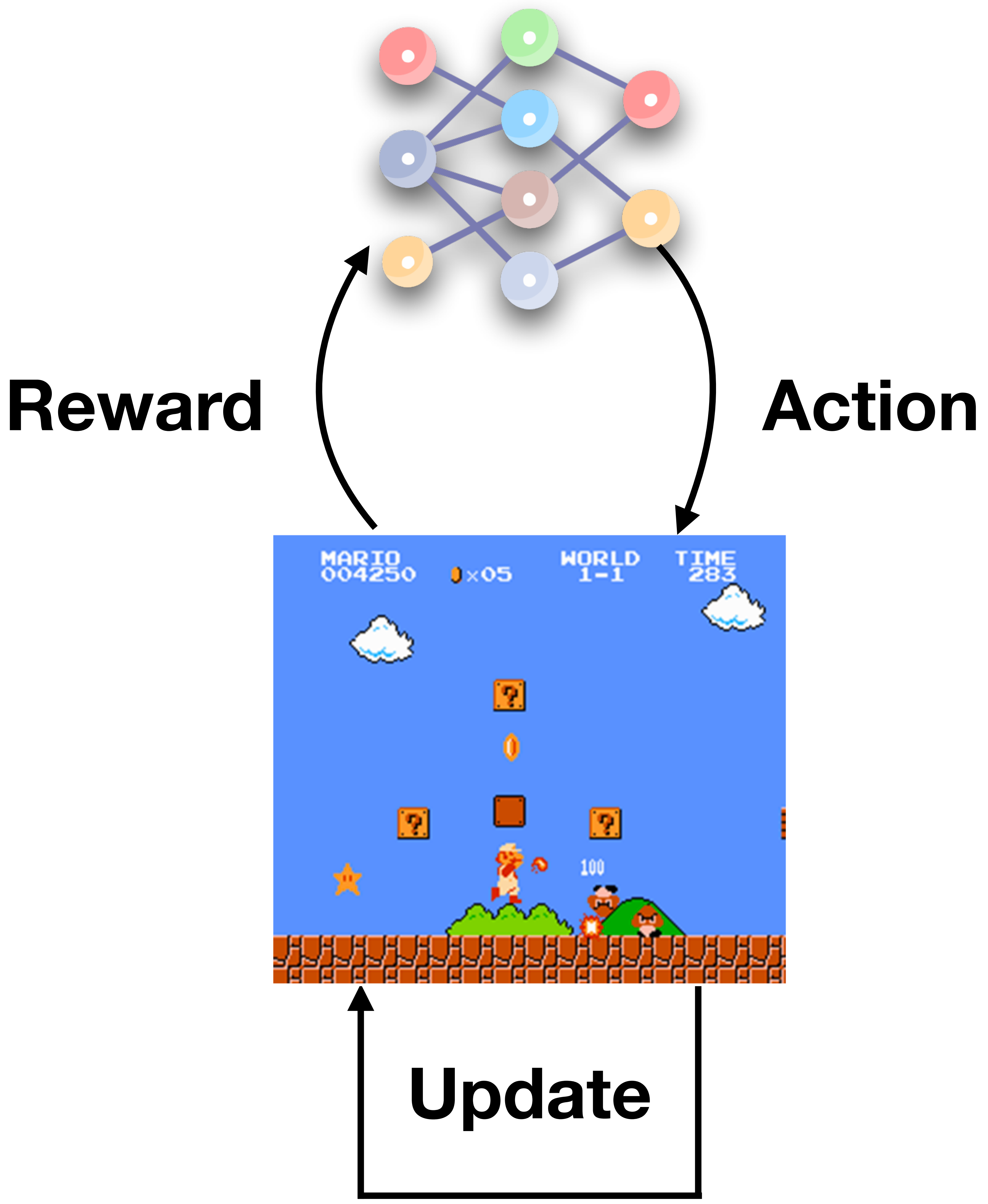
- Based on the selected action, the environment is updated.



Reinforcement Learning

- An agent is placed into an environment in which, it can perform certain actions.
- Based on the selected action, the environment is updated.
- The agent is assigned a reward (fitness value) based on the reached state after applying one or several actions.

➡ Has to represent the intended goal as closely as possible!



Fitness Function

Measure travel distance

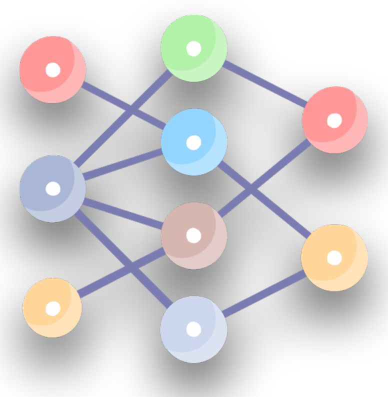
Fitness Function

Measure travel distance + level progress

Neuroevolution

Neuro evolution

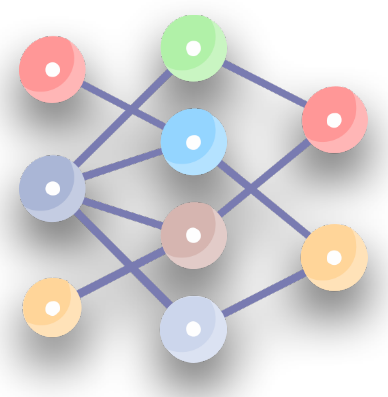
Neuro evolution

Neural networks 

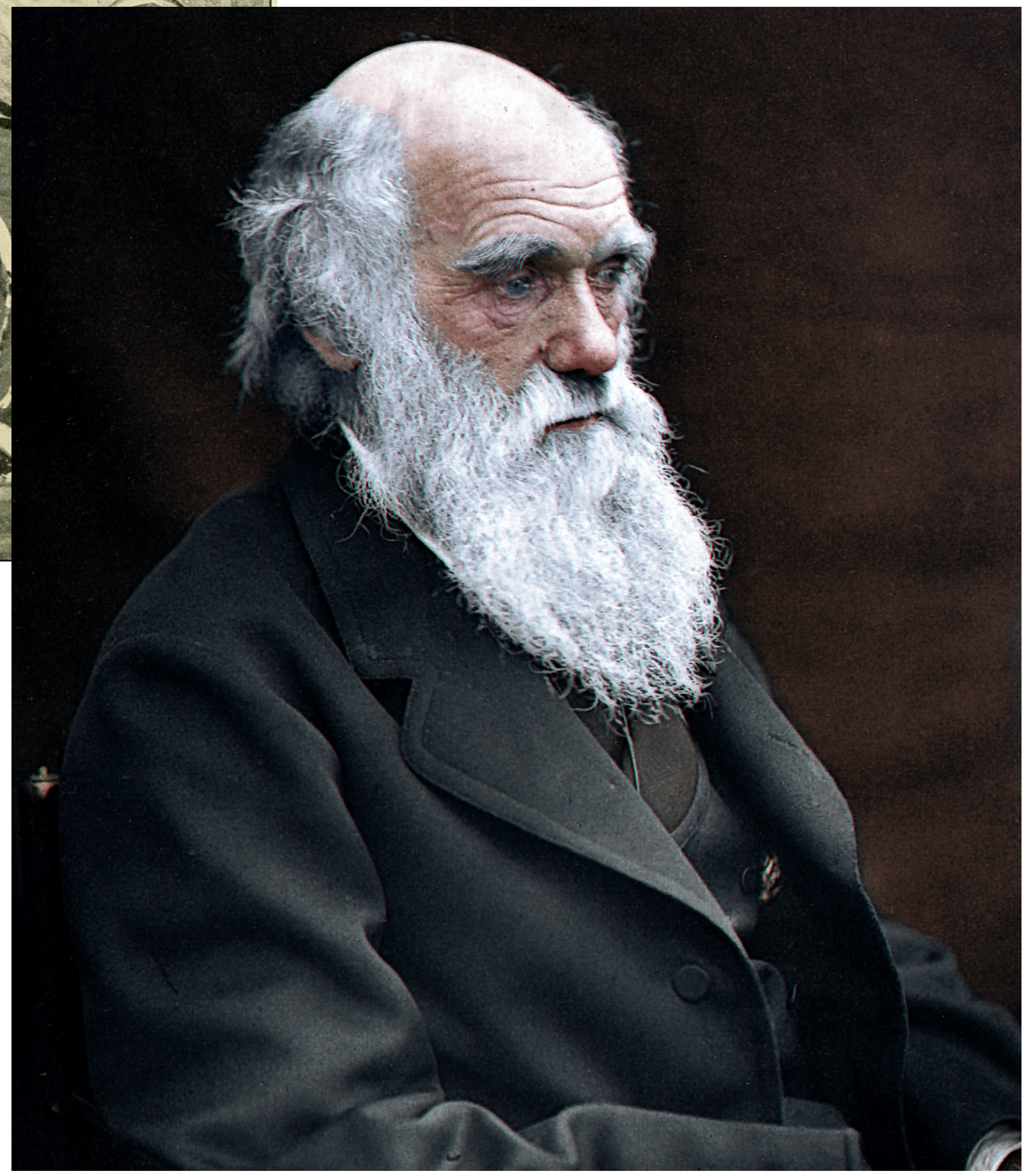
- Mimic a human brain to solve complex tasks
- Must be optimised for each task individually

Neuro evolution

Neural networks



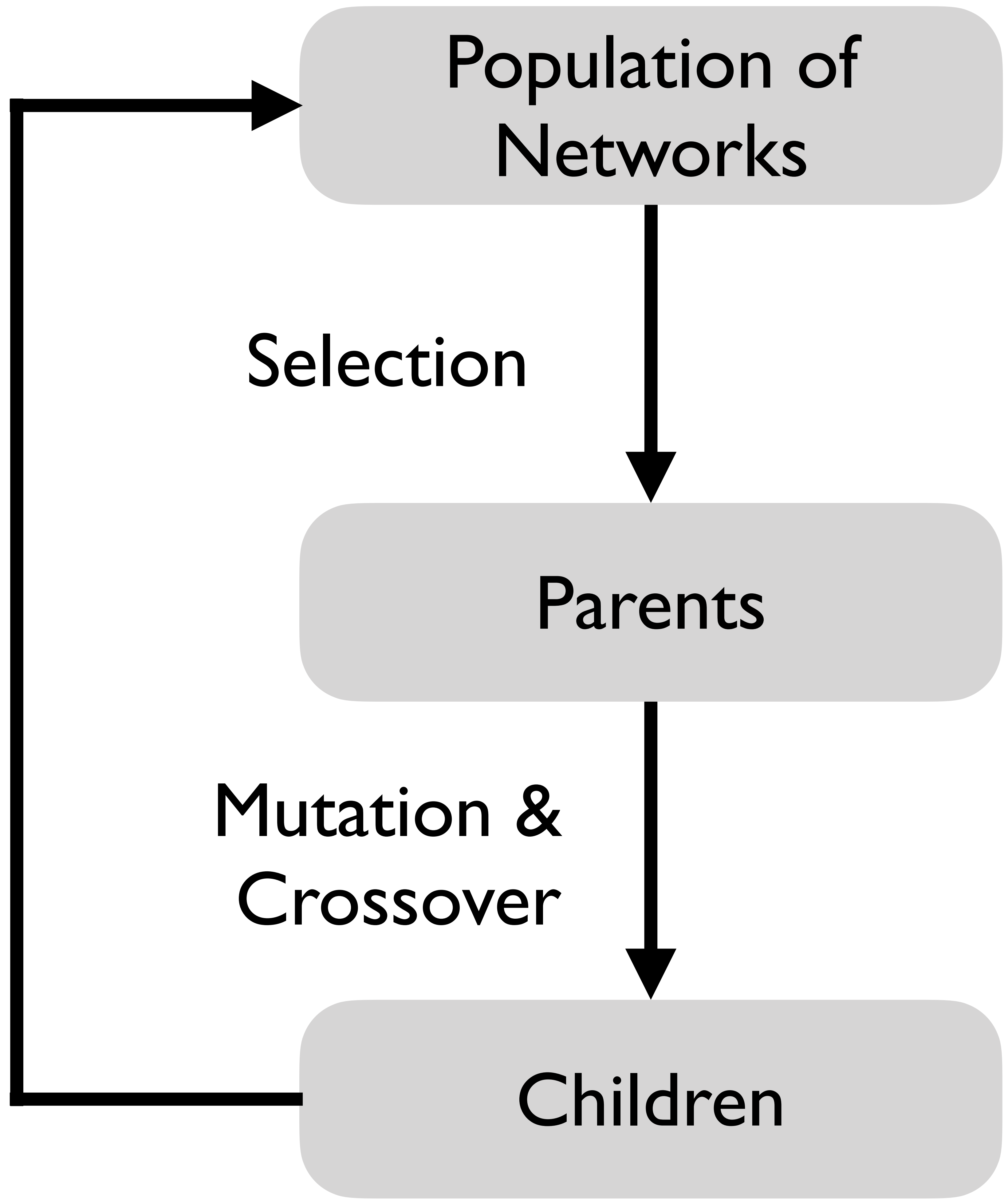
- Mimic a human brain to solve complex tasks
- Must be optimised for each task individually



Neuro evolution

Neural networks

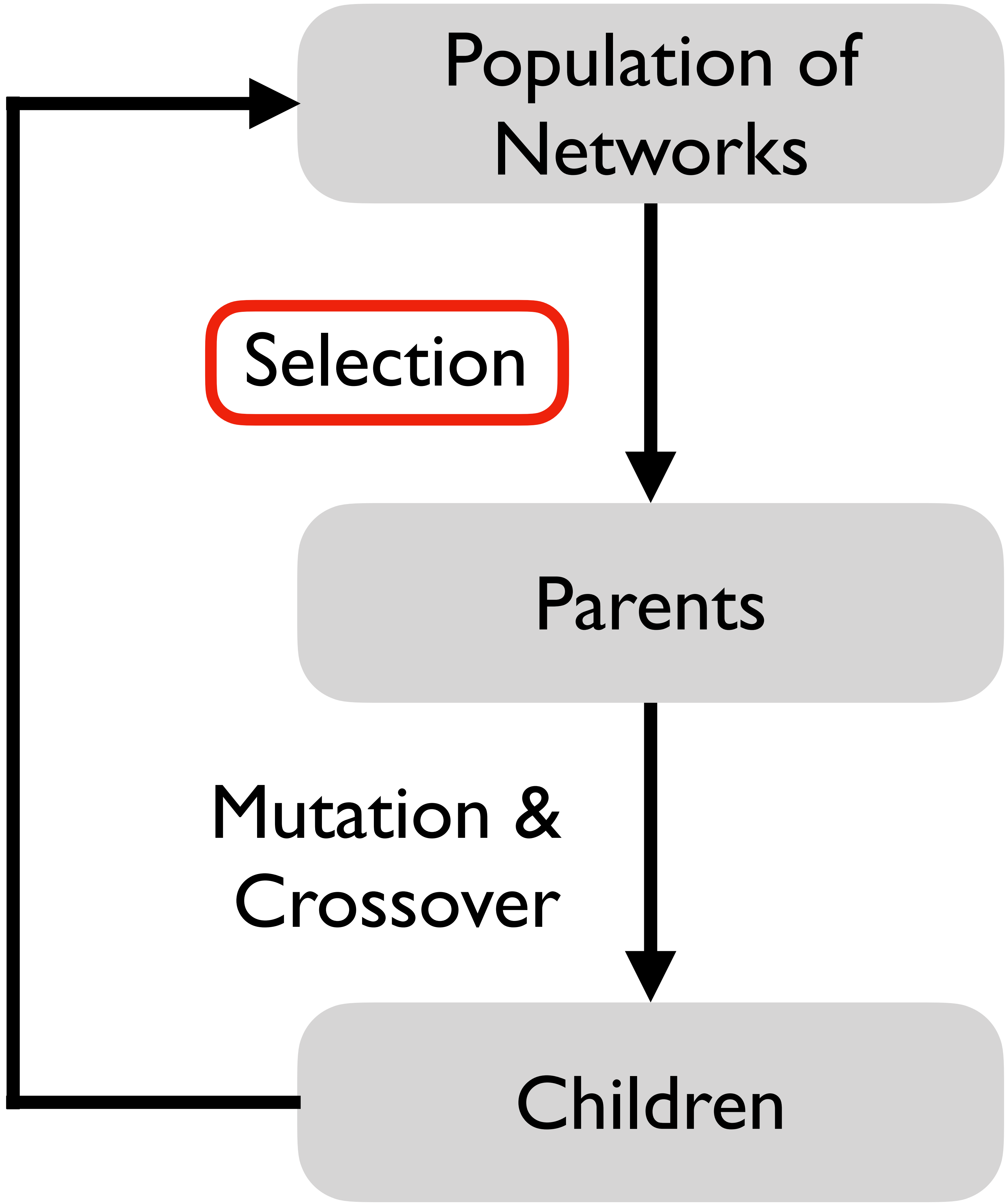
- Mimic a human brain to solve complex tasks
- Must be optimised for each task individually



Neuro evolution

Neural networks

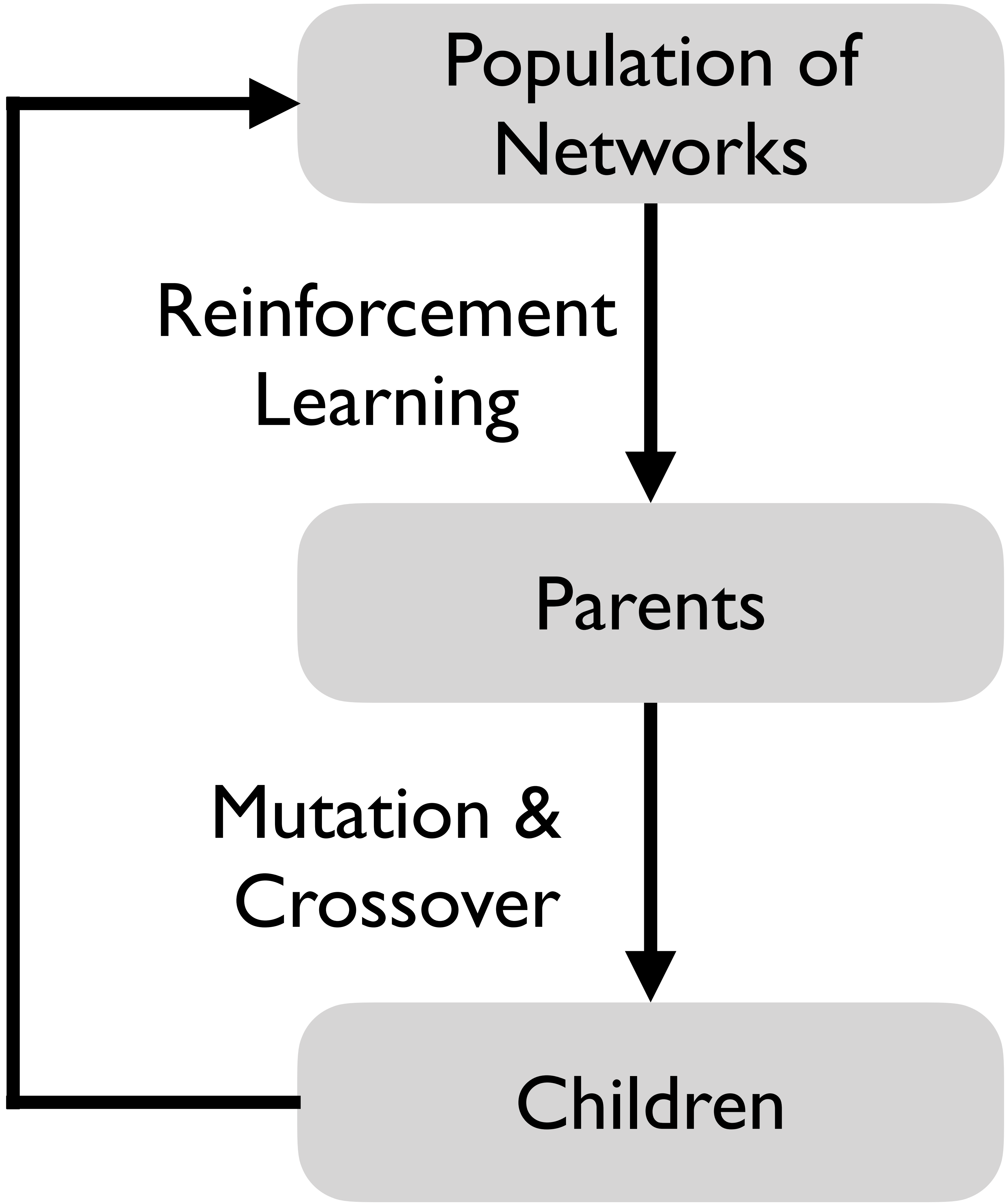
- Mimic a human brain to solve complex tasks
- Must be optimised for each task individually



Neuro evolution

Neural networks

- Mimic a human brain to solve complex tasks
- Must be optimised for each task individually



Neat

Evolving Neural Networks through Augmenting Topologies

Kenneth O. Stanley and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
{kstanley, risto}@cs.utexas.edu

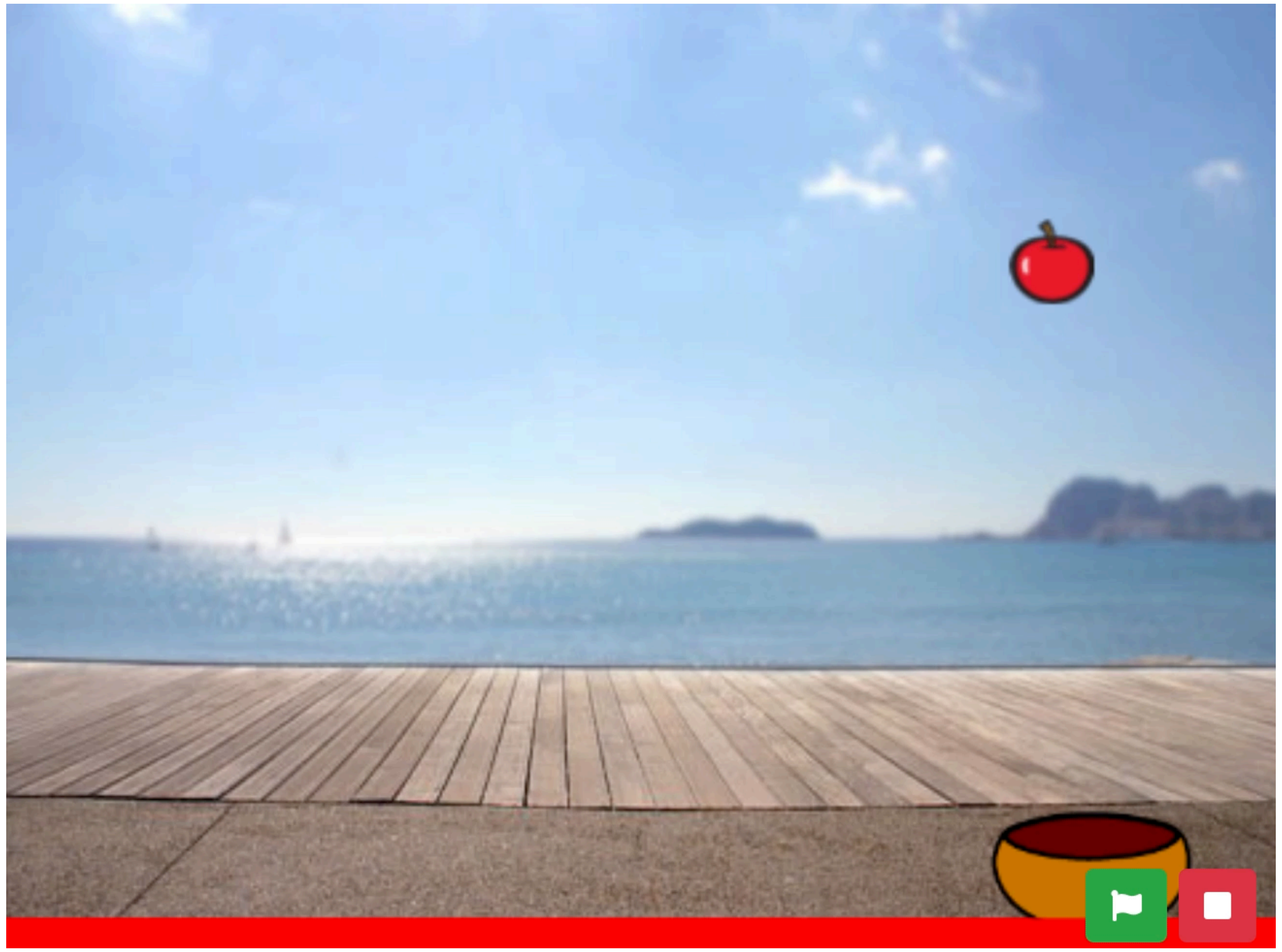
Technical Report TR-AI-01-290

June 28, 2001

Abstract

An important question in neuroevolution is how to gain an advantage from evolving neural network topologies along with weights. We present a method, NeuroEvolution of Augmenting Topologies (NEAT) that outperforms the best fixed-topology method on a challenging benchmark reinforcement learning task. We claim that the increased efficiency is due to (1) employing a principled method of crossover of different topologies, (2) protecting structural innovation using speciation, and (3) incrementally growing from minimal structure. We test this claim through a series of ablation studies that demonstrate that each component is necessary to the system as a whole and to each other. What results is significantly faster learning. NEAT is also an important contribution to GAs because it shows how it is possible for evolution to both optimize *and complexify* solutions simultaneously, offering the possibility of evolving increasingly complex solutions over generations, and strengthening the analogy with biological evolution.

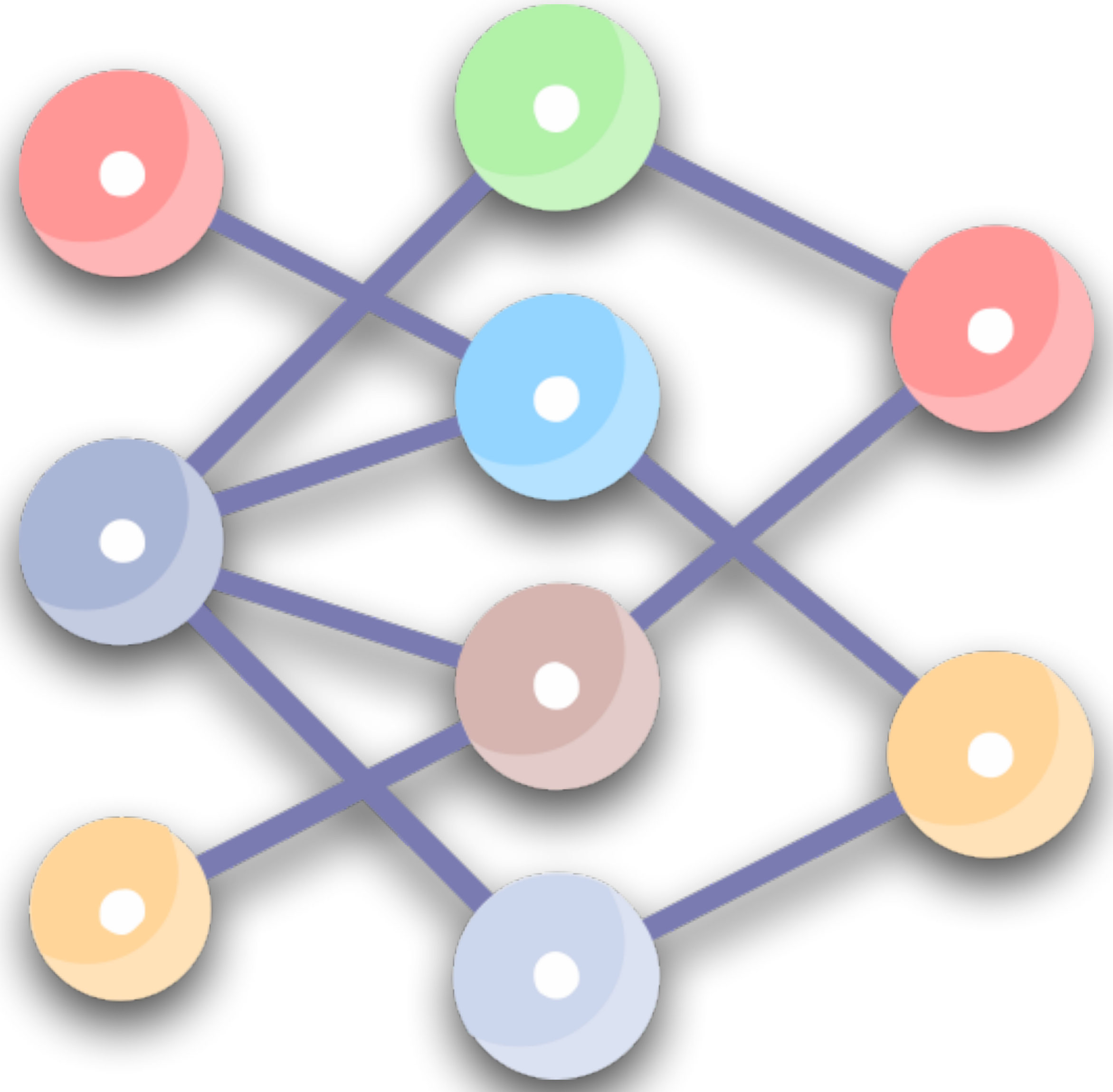
Neatest



Learn to play

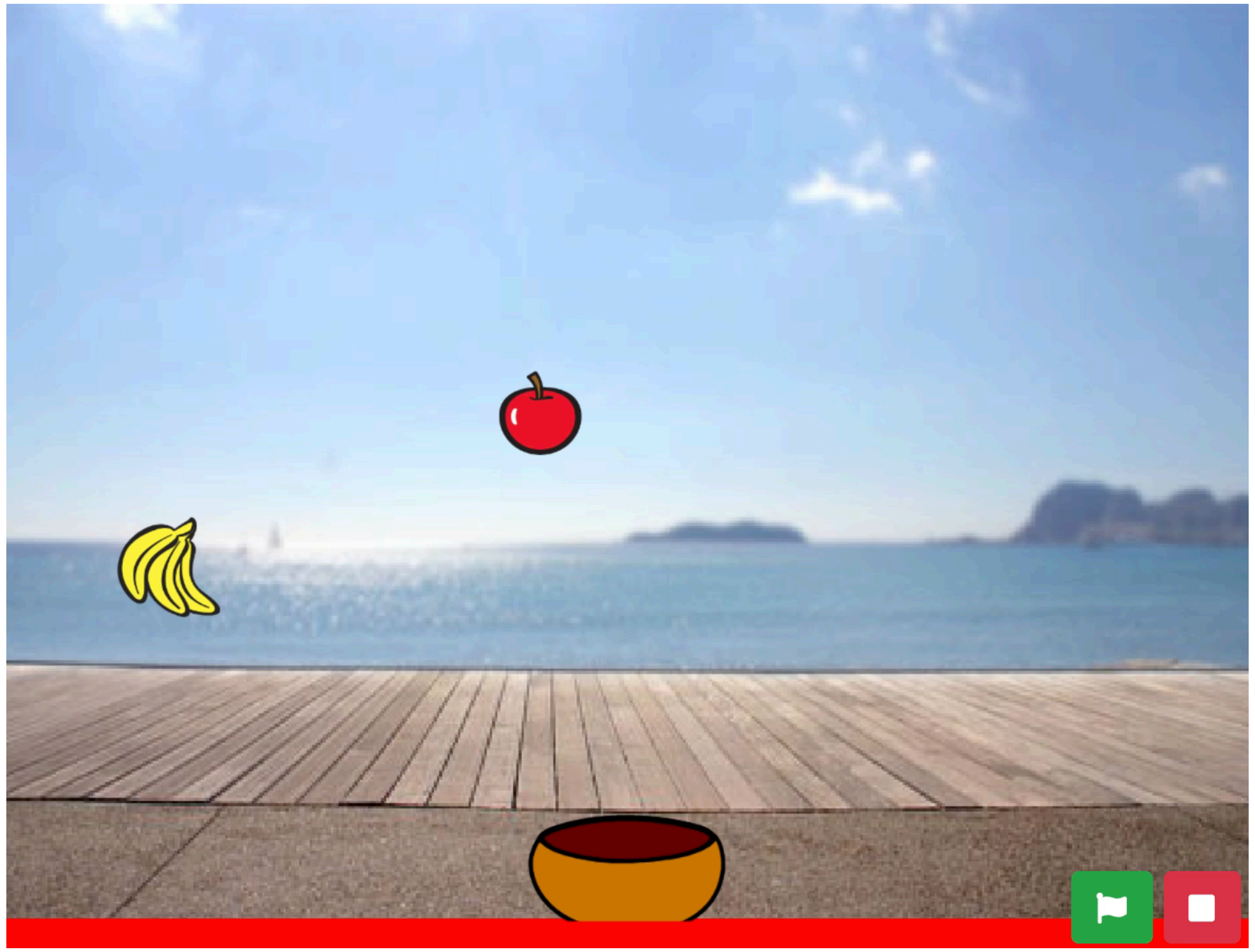
Validate behaviour

Dynamic Test Suites



Generating Dynamic Test Suites

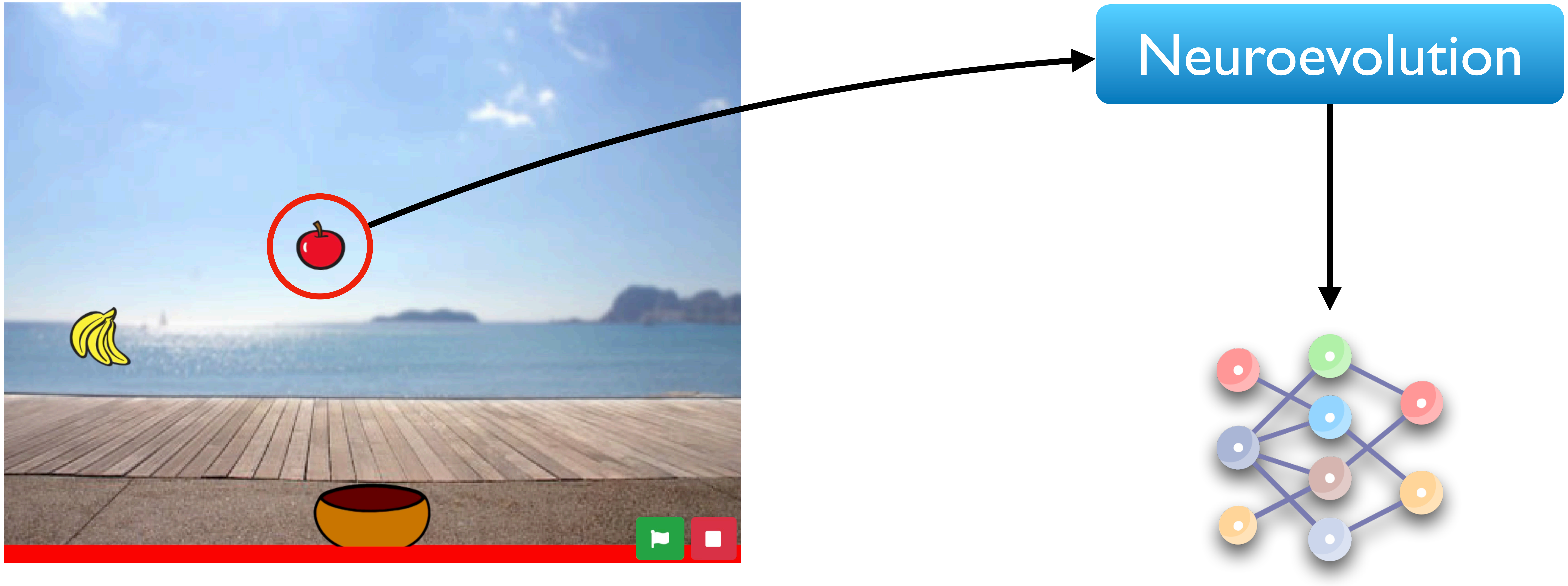
1. Select a target statement



Generating Dynamic Test Suites

2. Optimise networks to cover the selected statement using Neuroevolution

➡ Fitness = distance to target statement

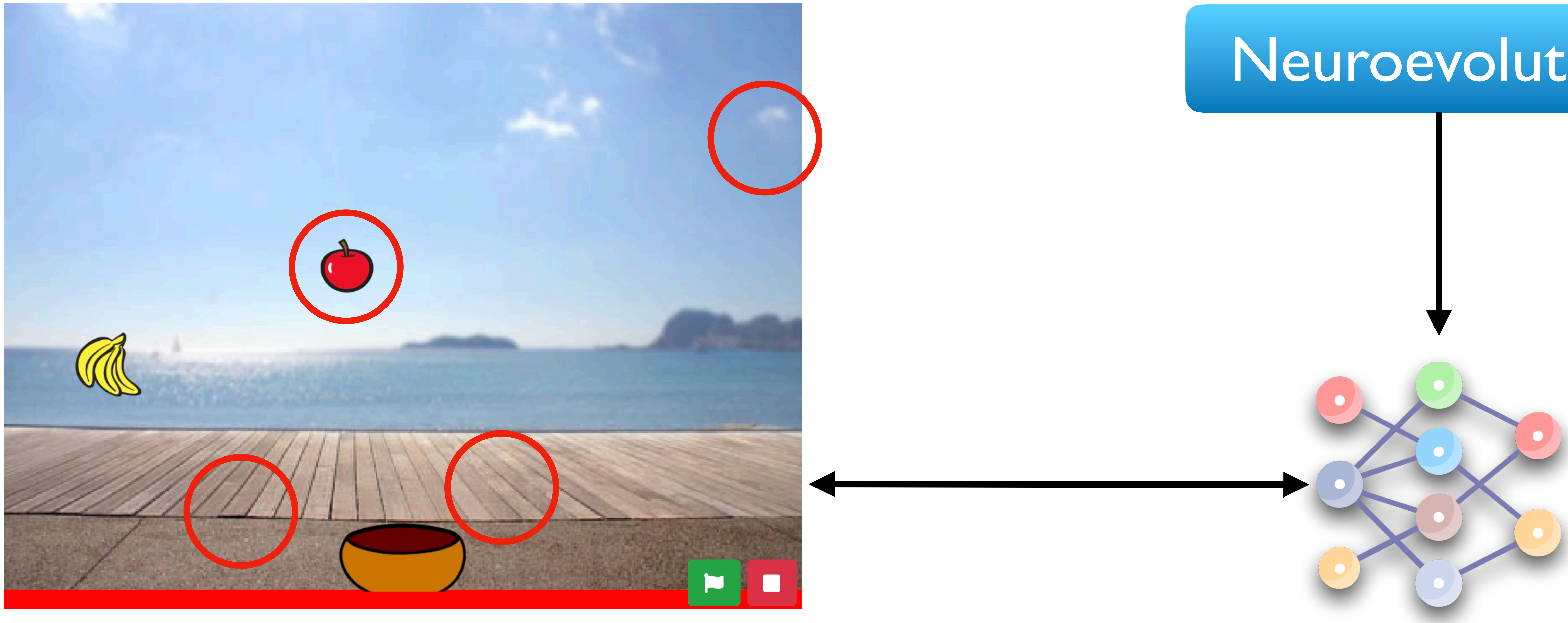


Generating Dynamic Test Suites

3. Validate and improve the robustness of networks

➡ Cover 

multiple times using different seeds

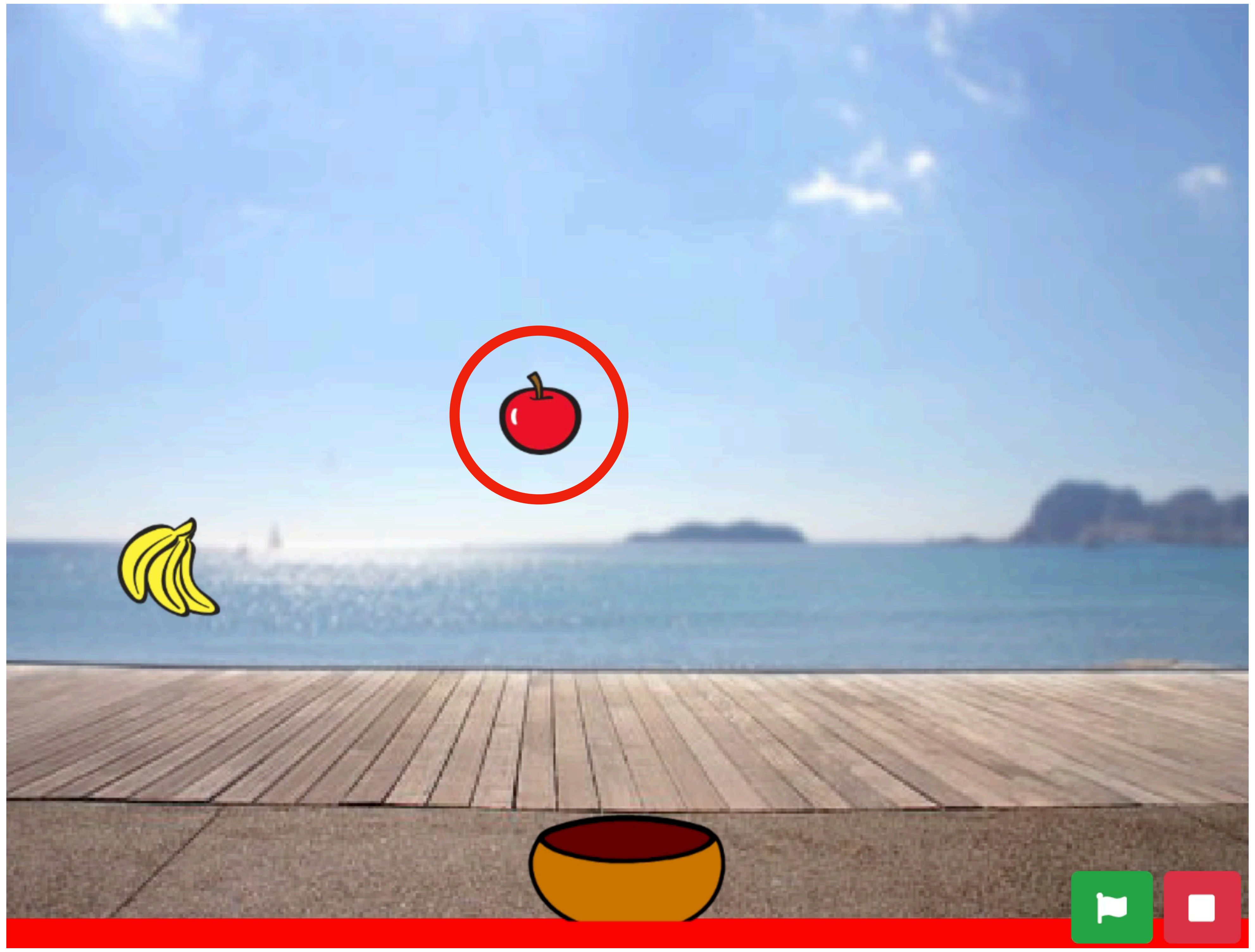


Generating Dynamic Test Suites

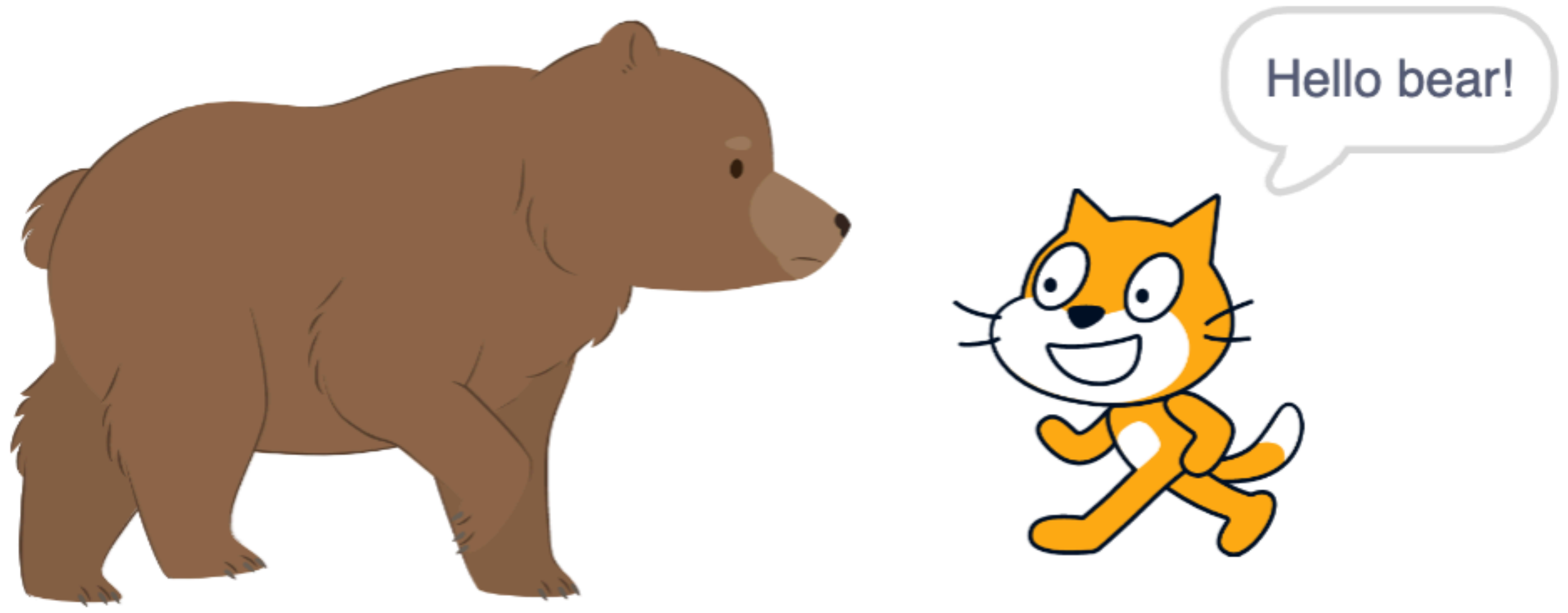
1. Select a target statement

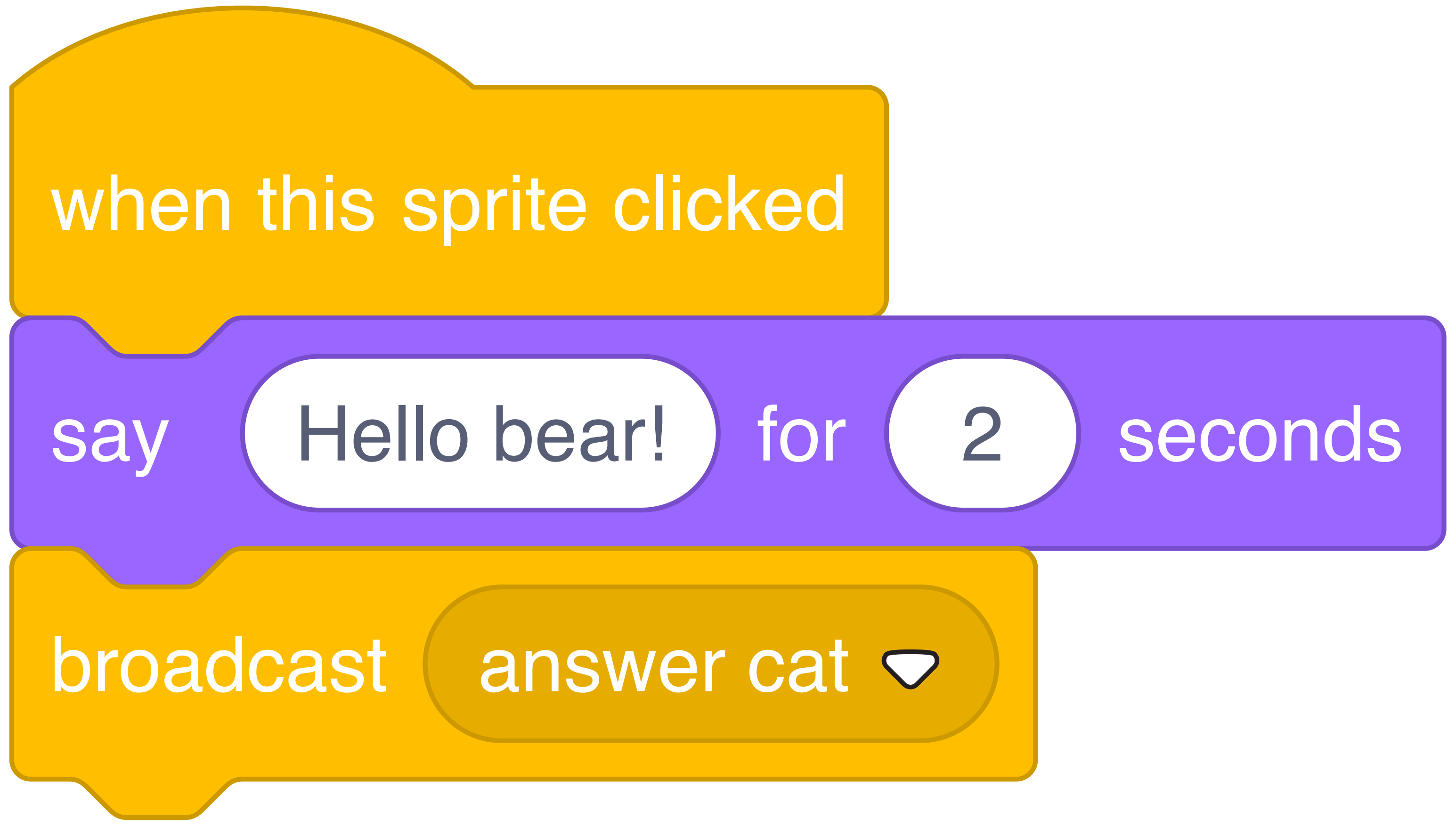
How to select a target?

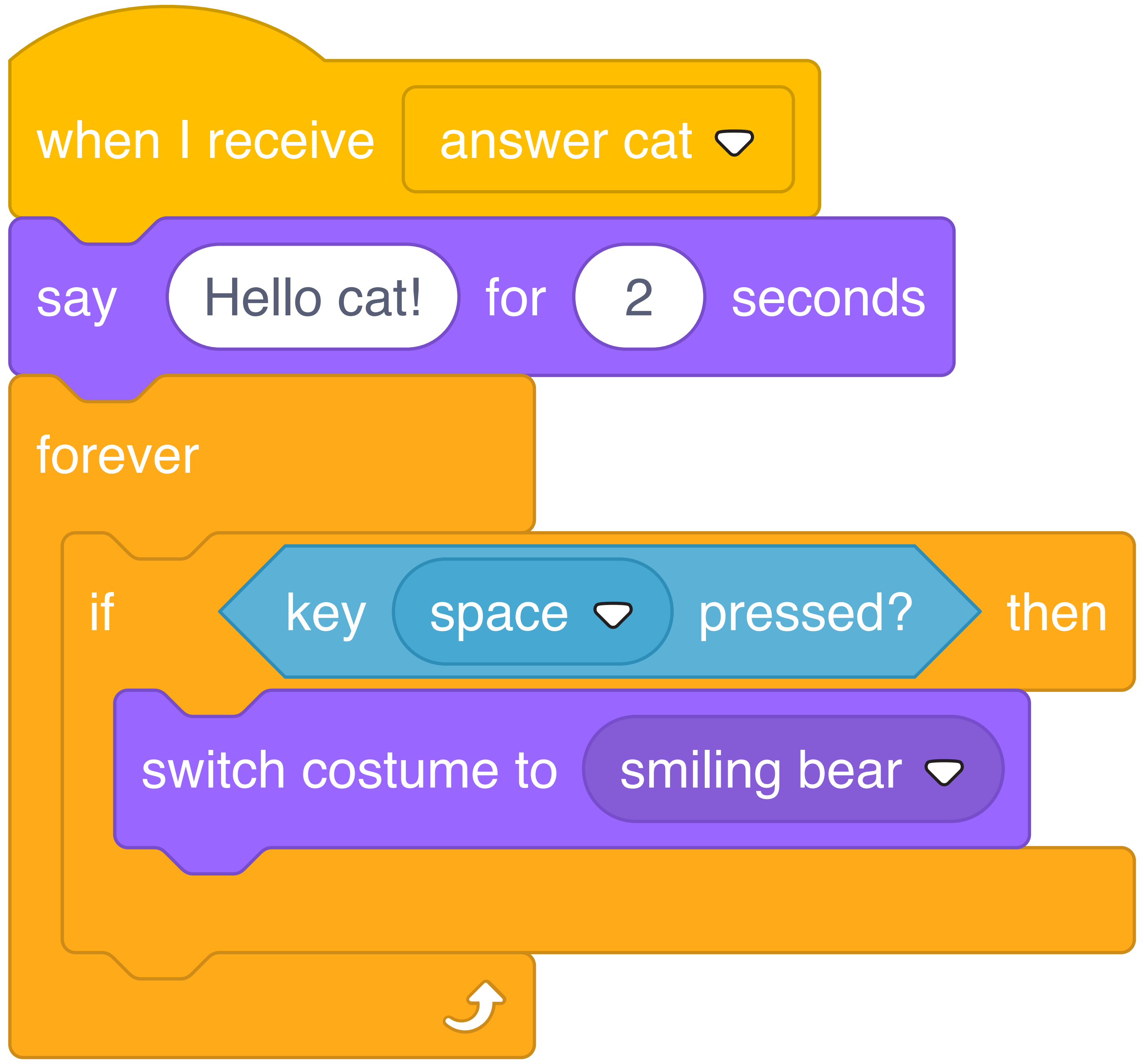
Explore the CDG



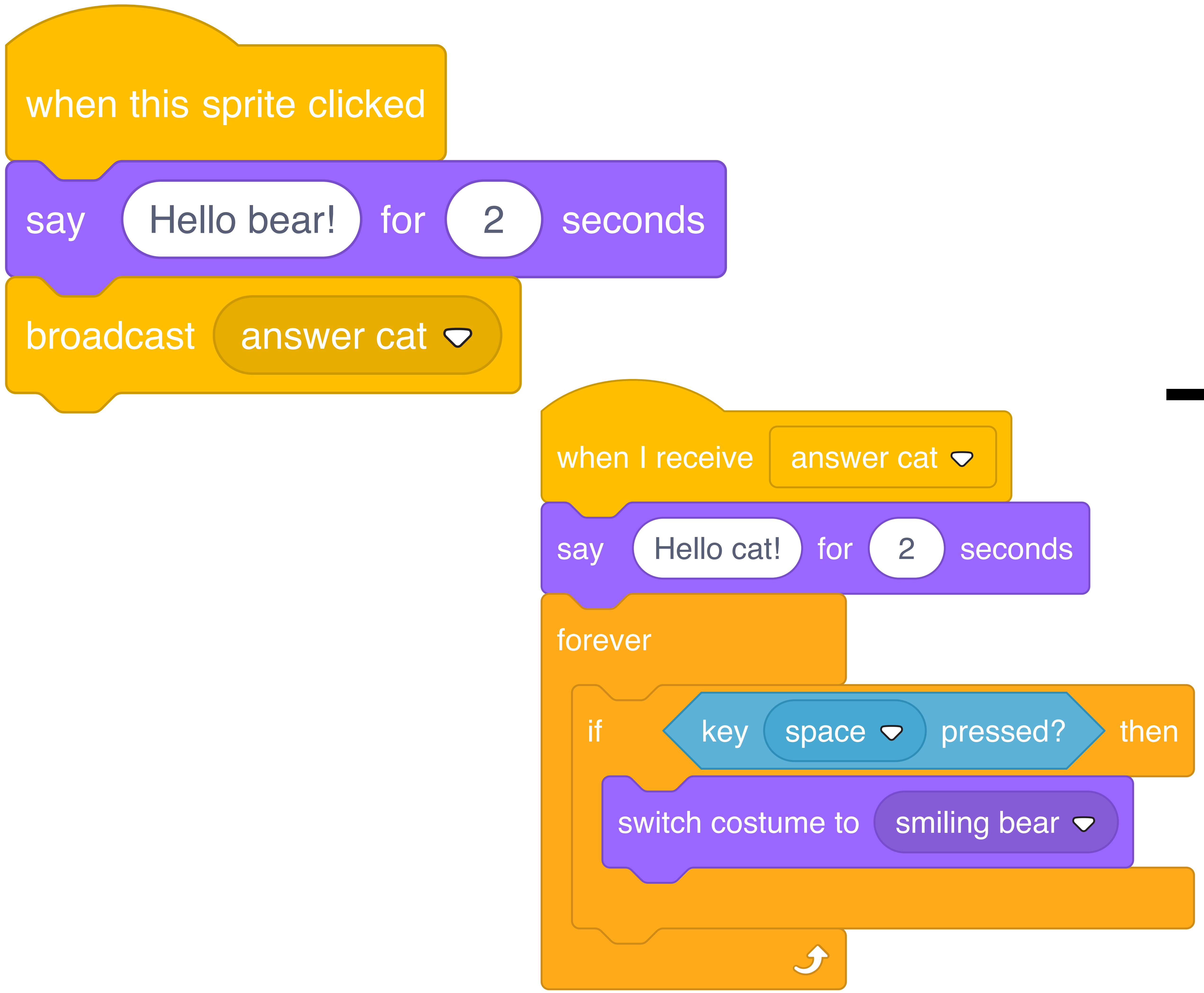
Explore the CDG

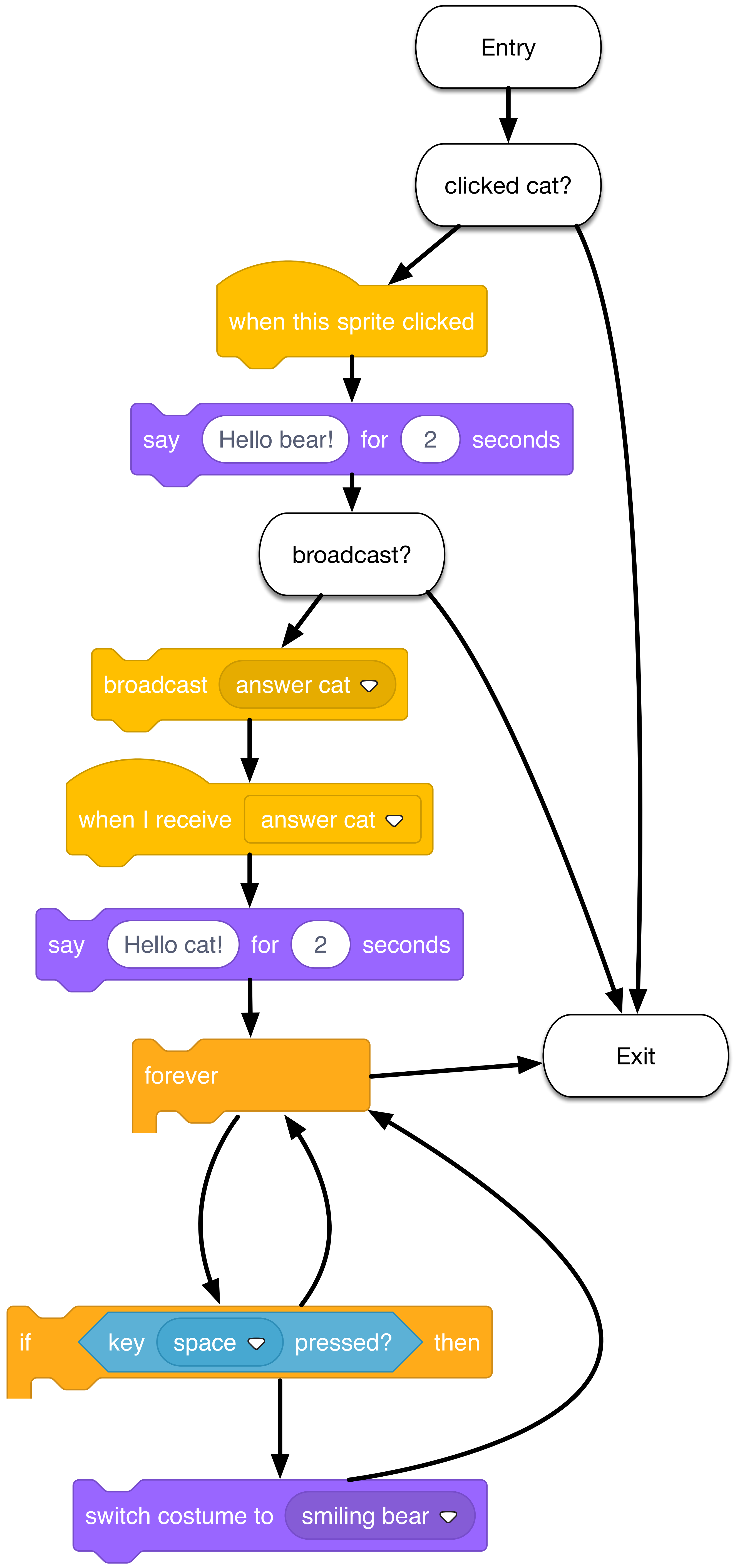




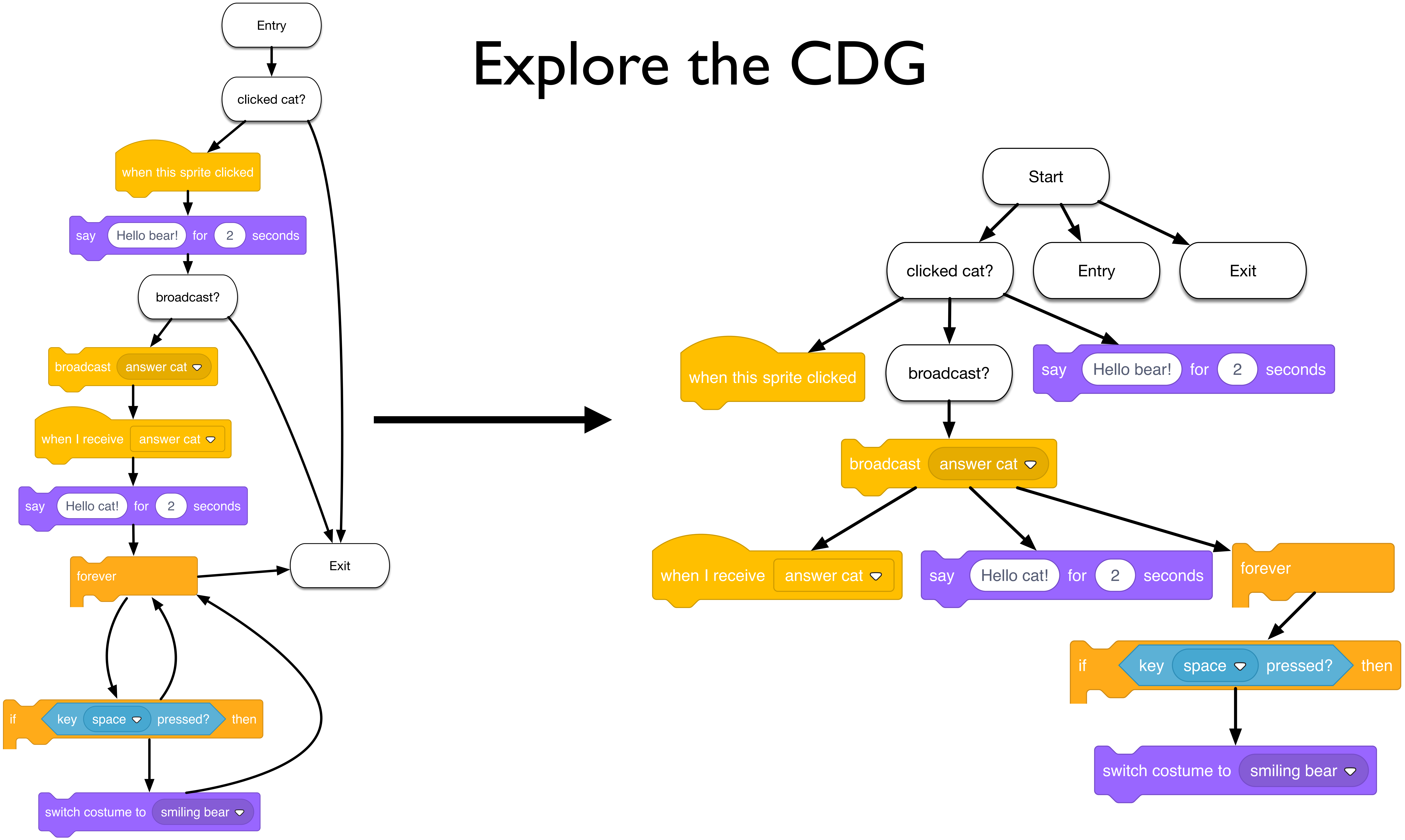


Explore the CDG

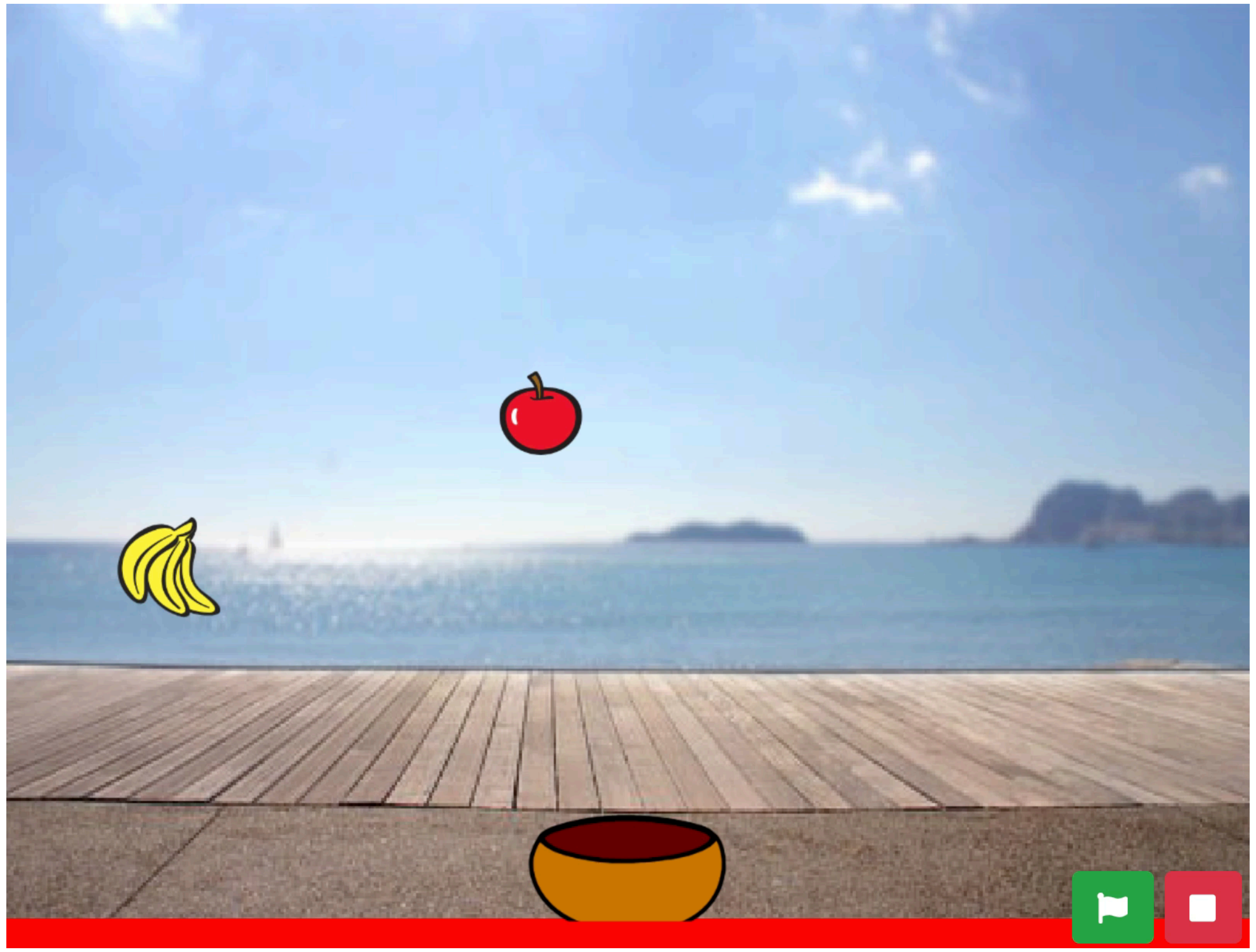


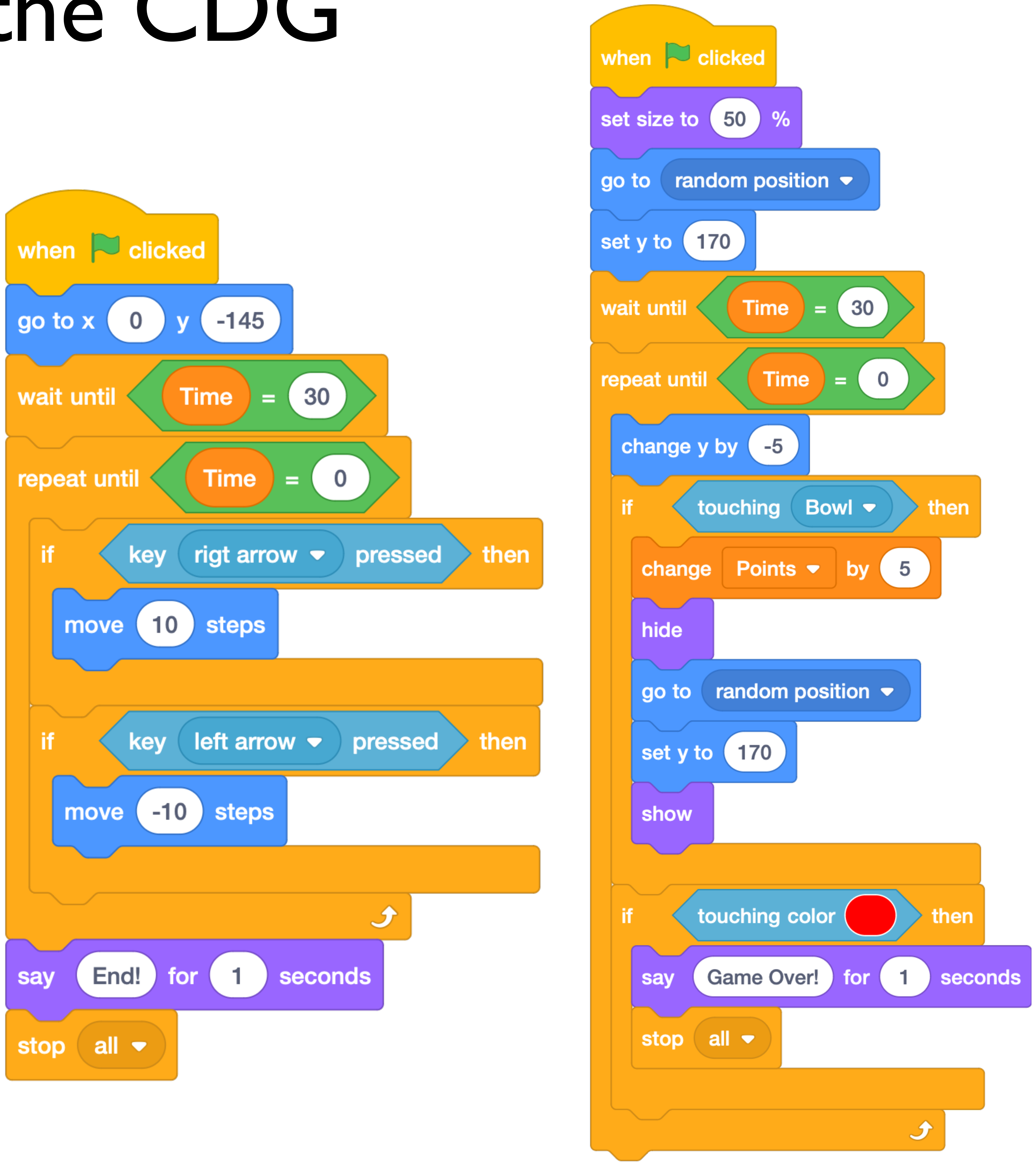


Explore the CDG



Explore the CDG



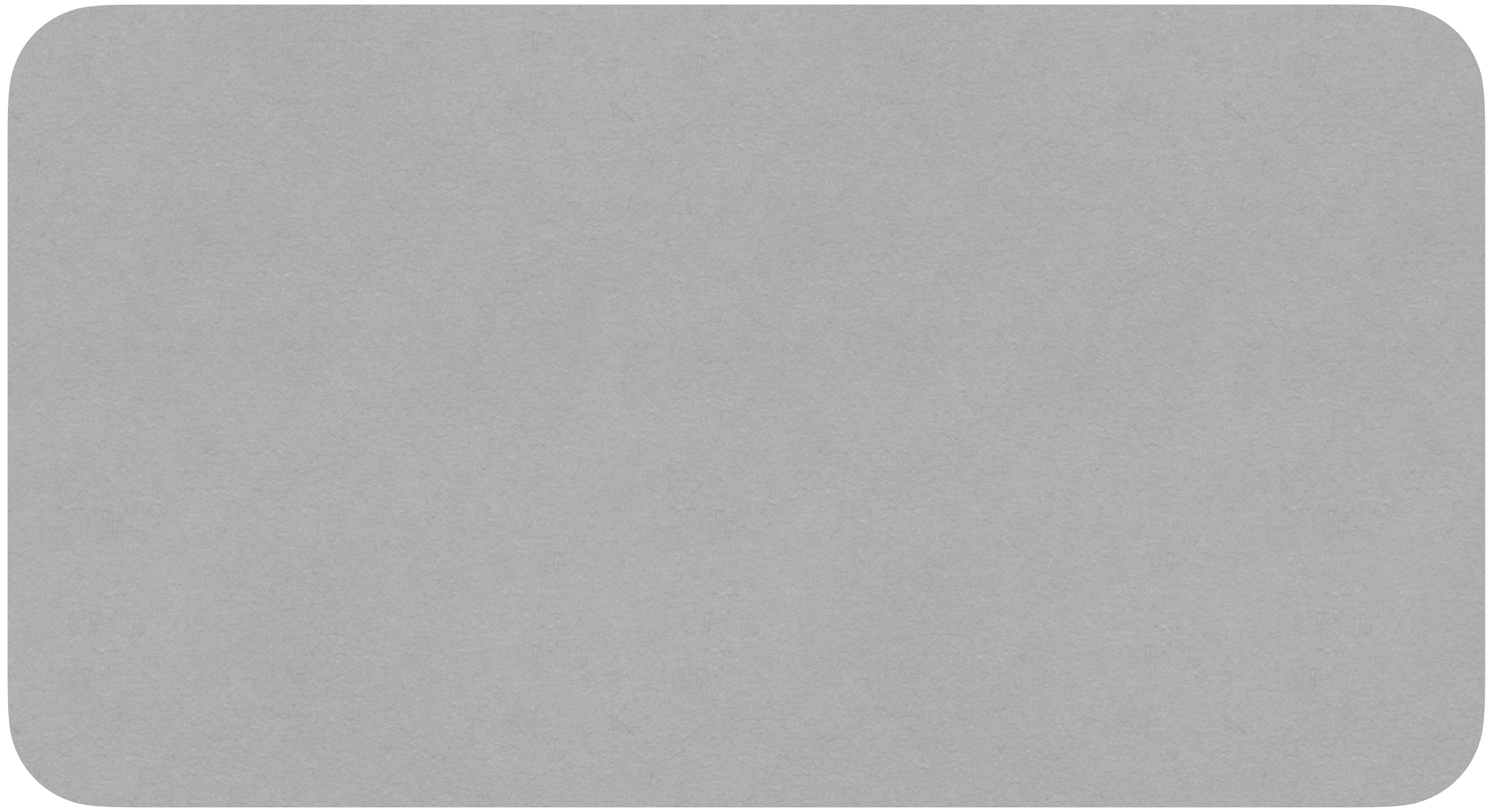


Explore the CDG

Explore the CDG

- Select direct children of covered control nodes

Dynamic Test Suite



Explore the CDG

Dynamic Test Suite

- Select direct children of covered control nodes

Explore the CDG

Dynamic Test Suite

- Select direct children of covered control nodes

Explore the CDG

Dynamic Test Suite

- Select direct children of covered control nodes

Explore the CDG

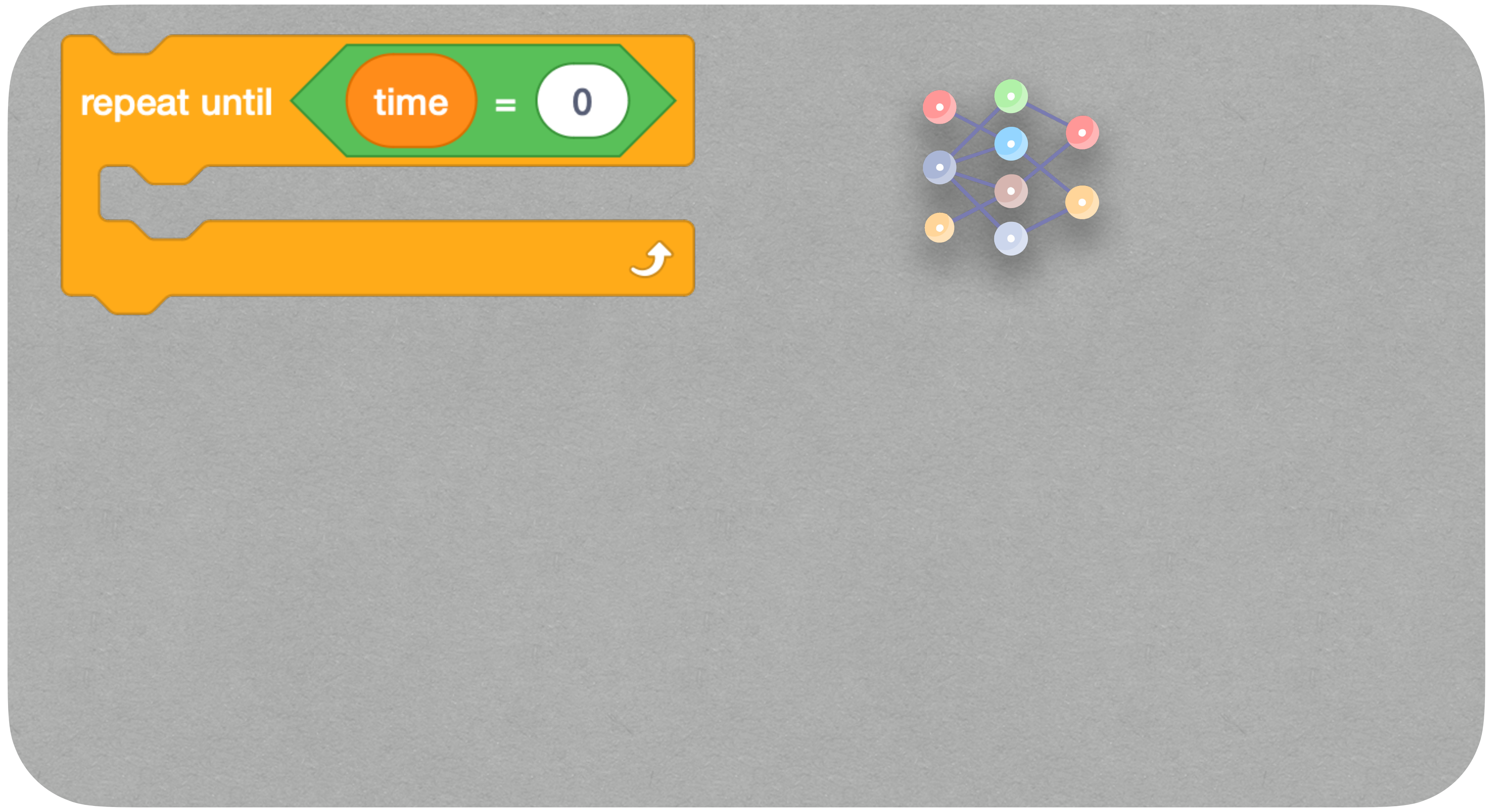
- Select direct children of covered control nodes

Explore the CDG

- Select direct children of covered control nodes

➡Deep nodes require meaningful gameplay

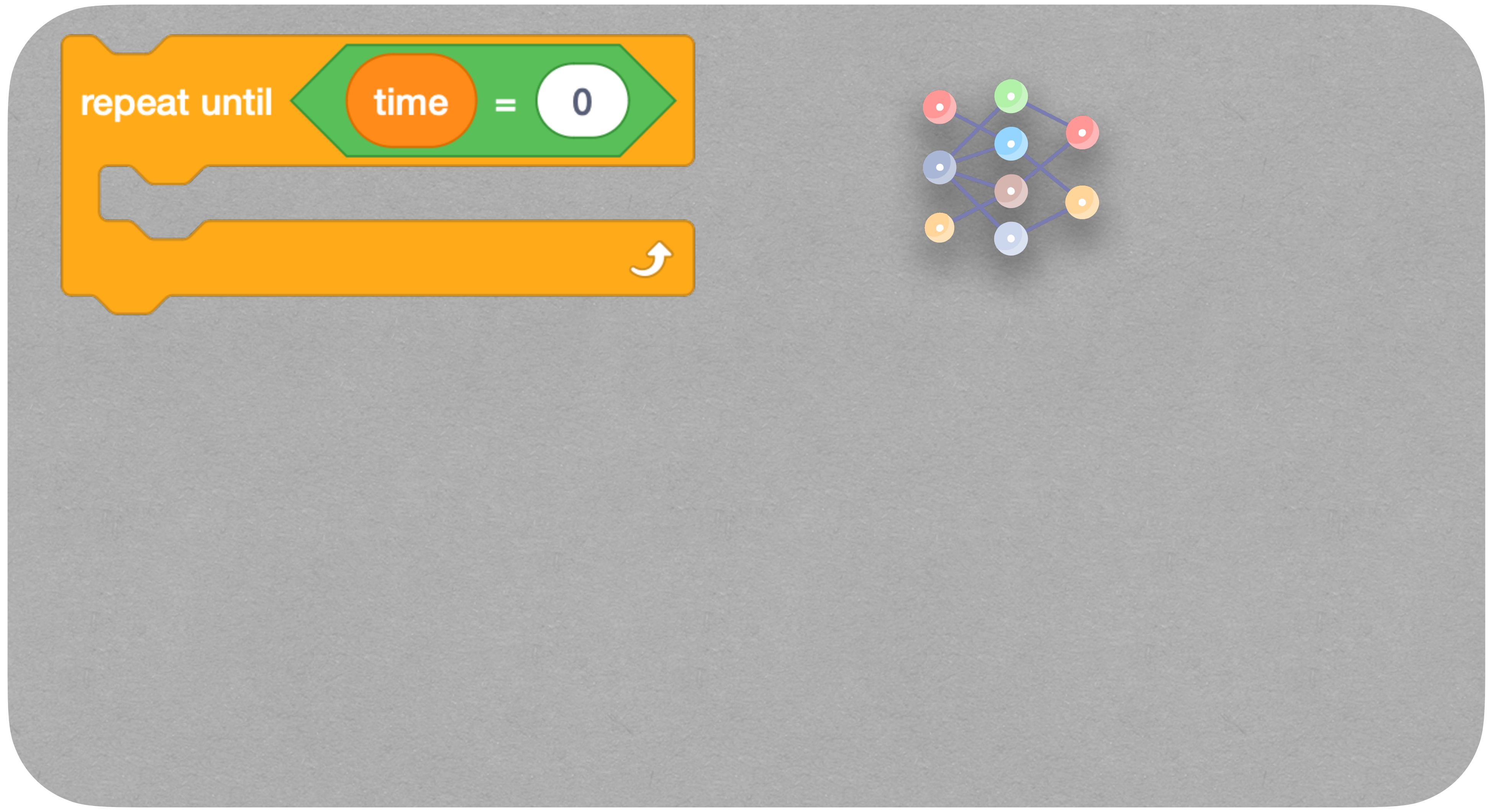
Dynamic Test Suite



Explore the CDG

- Select direct children of covered control nodes
  ➡Deep nodes require meaningful gameplay

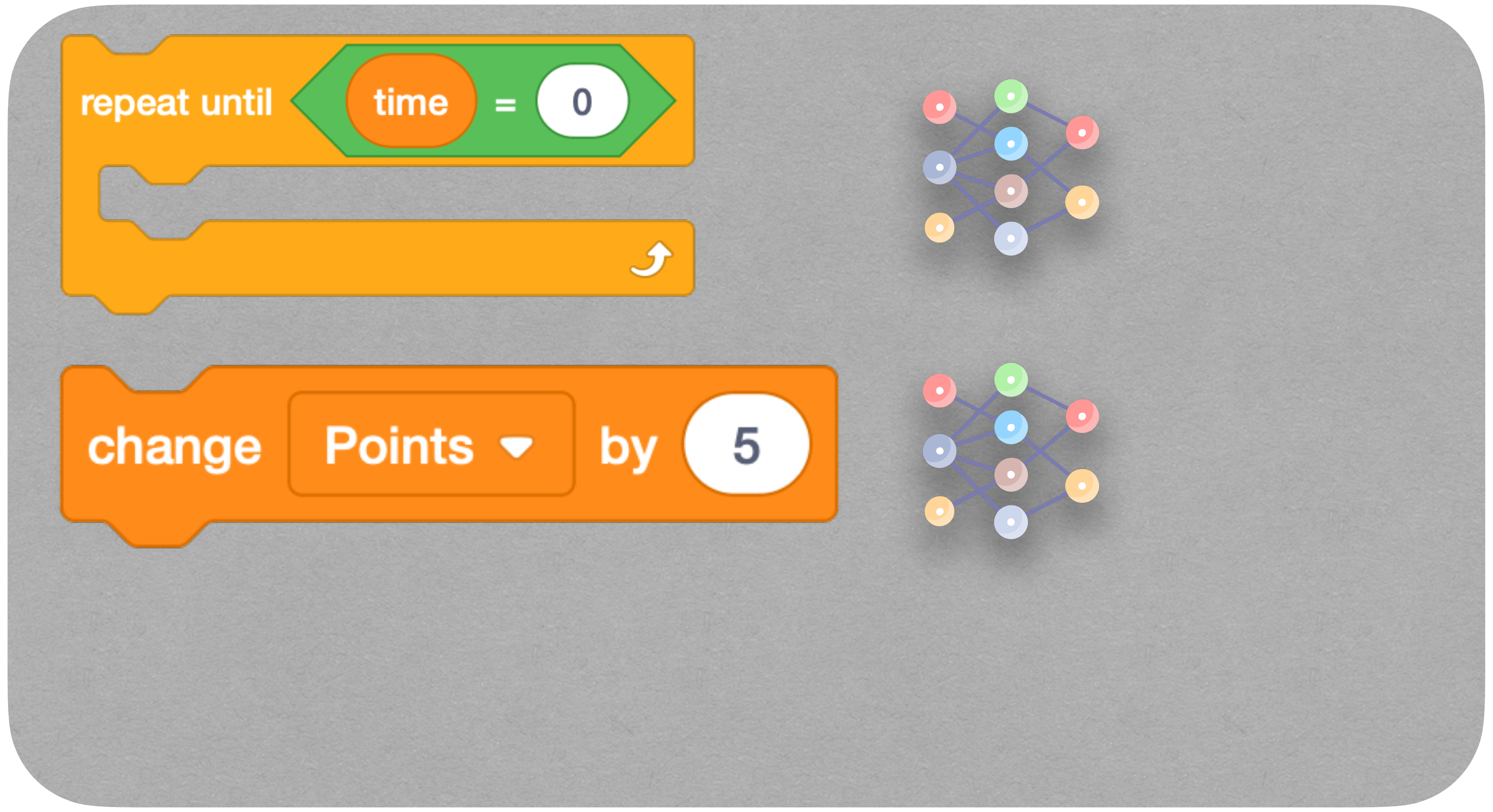
Dynamic Test Suite



Explore the CDG

- Select direct children of covered control nodes
  ➡Deep nodes require meaningful gameplay

Dynamic Test Suite



Explore the CDG

- Select direct children of covered control nodes
  ➡Deep nodes require meaningful gameplay

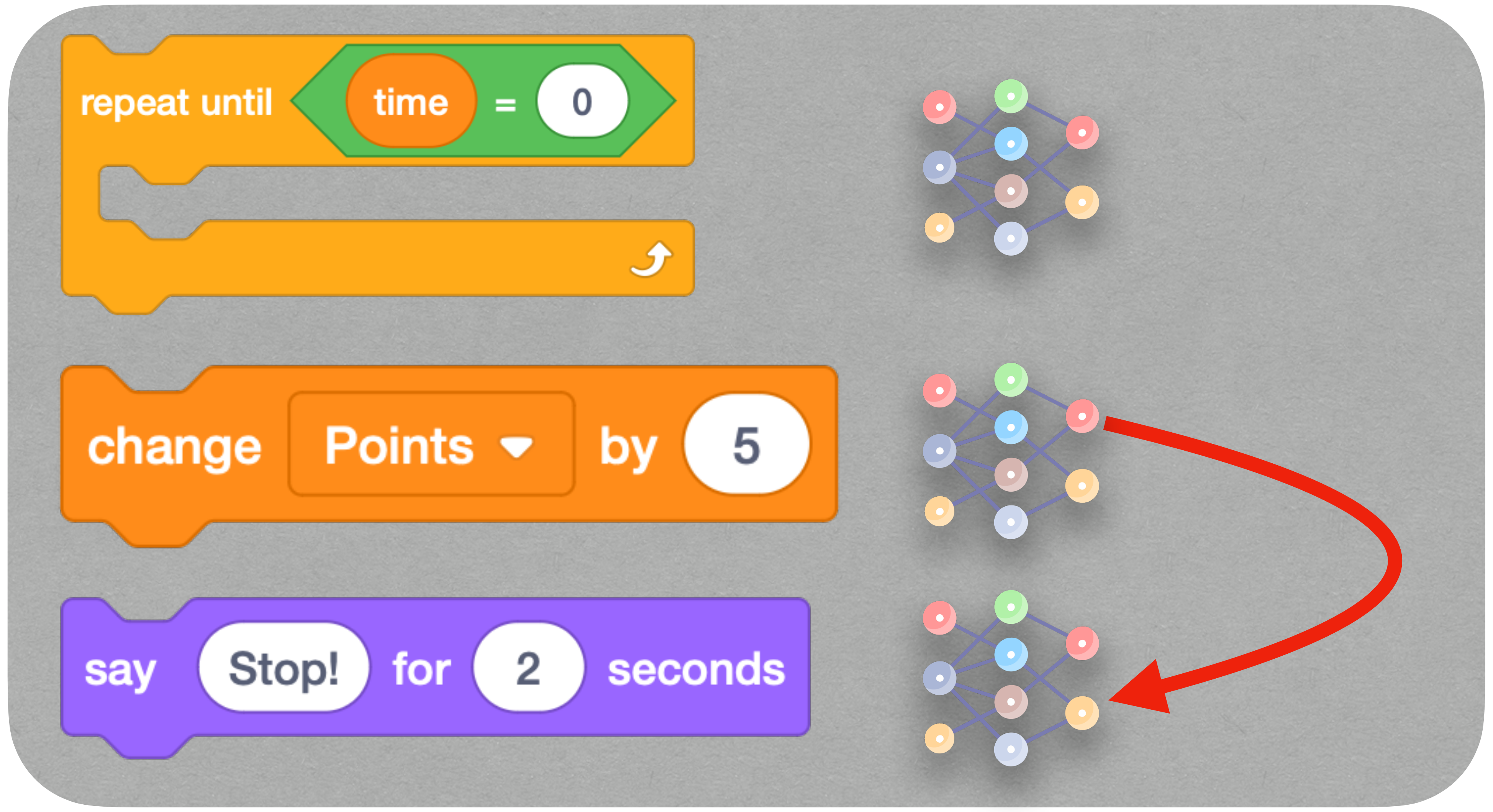
Dynamic Test Suite

Explore the CDG

- Select direct children of covered control nodes
  ➡Deep nodes require meaningful gameplay

  ➡Build upon previously optimised networks

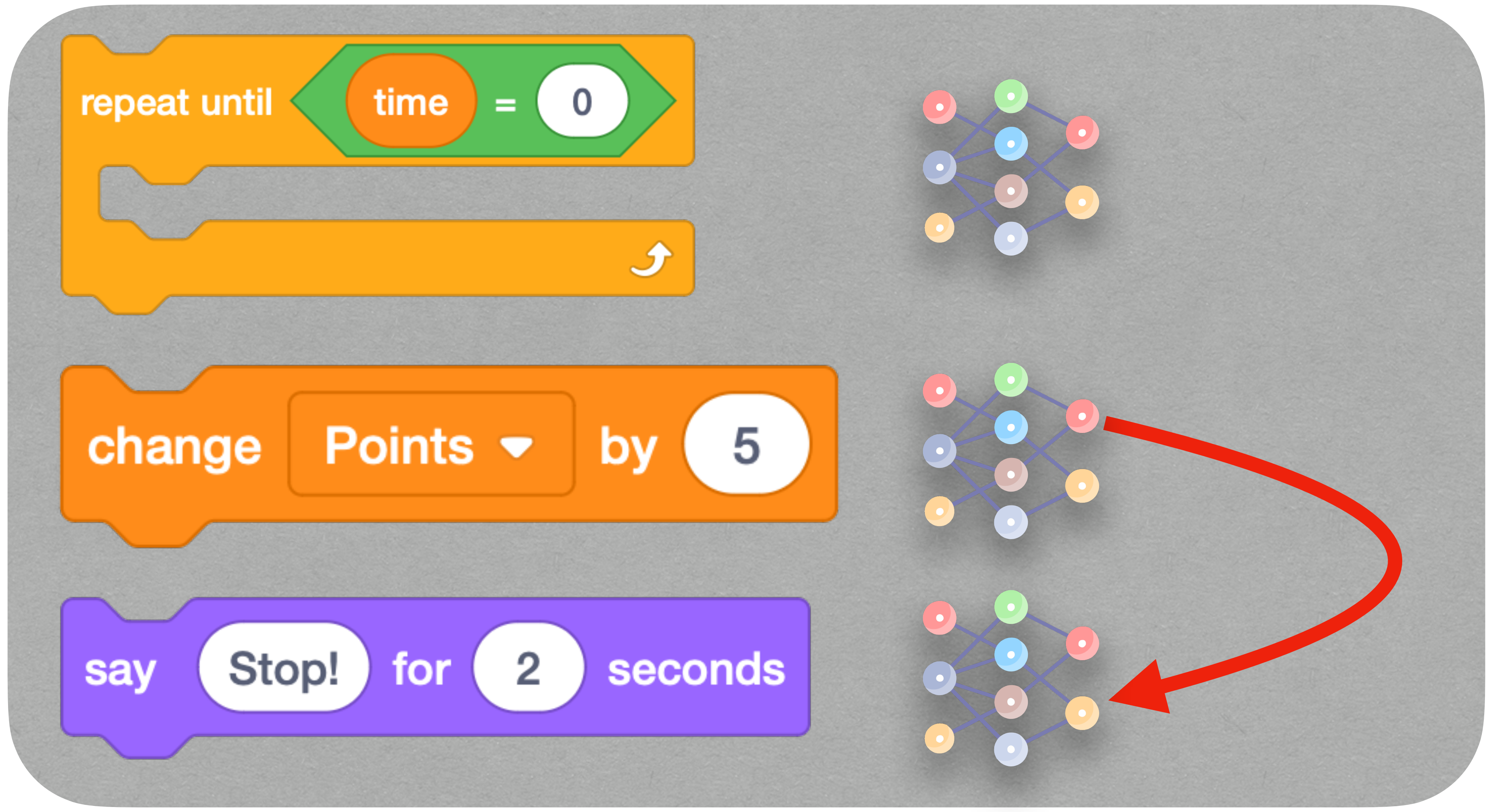
Dynamic Test Suite



Explore the CDG

- Select direct children of covered control nodes
  - ➡Deep nodes require meaningful gameplay
  - ➡Build upon previously optimised networks

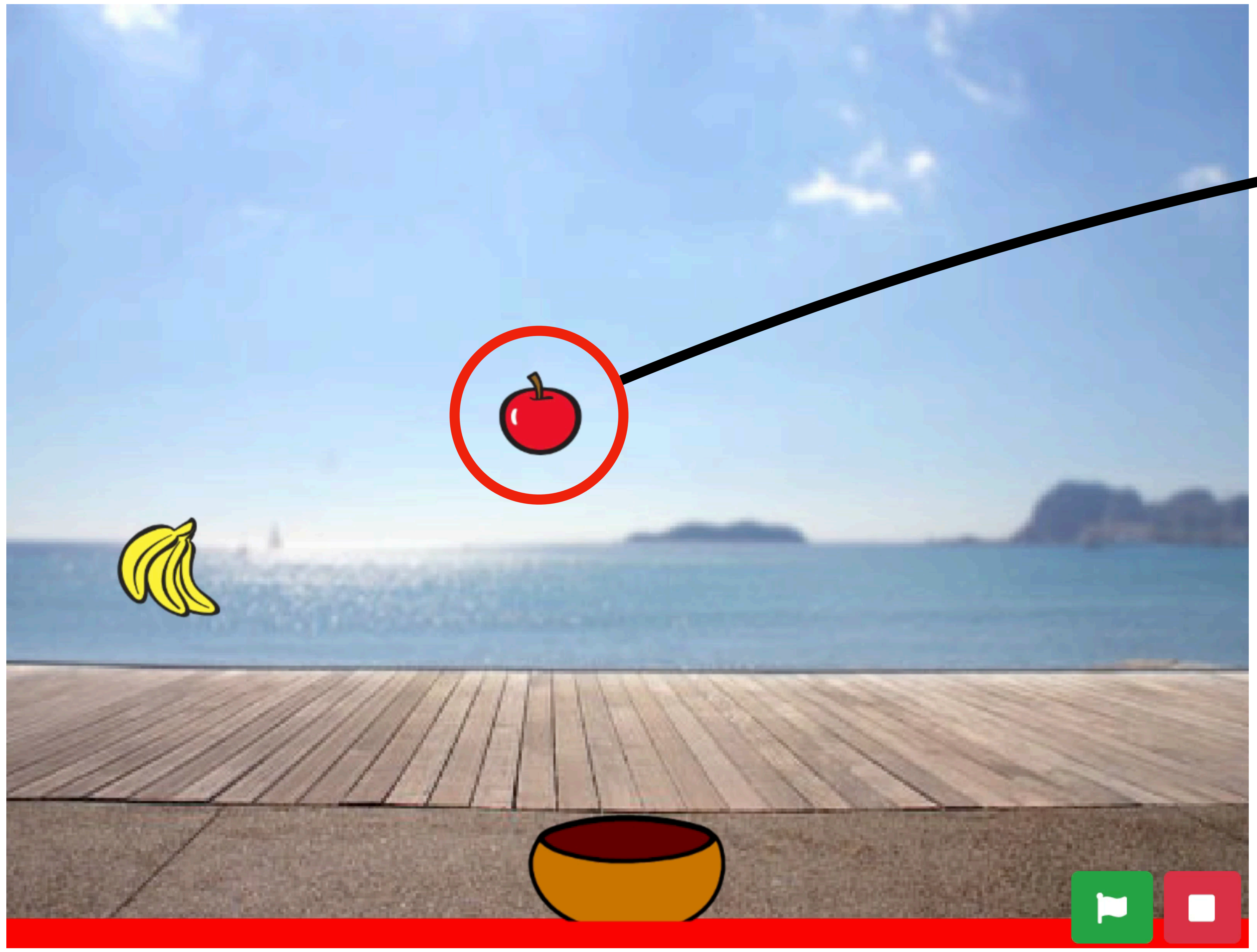
Dynamic Test Suite



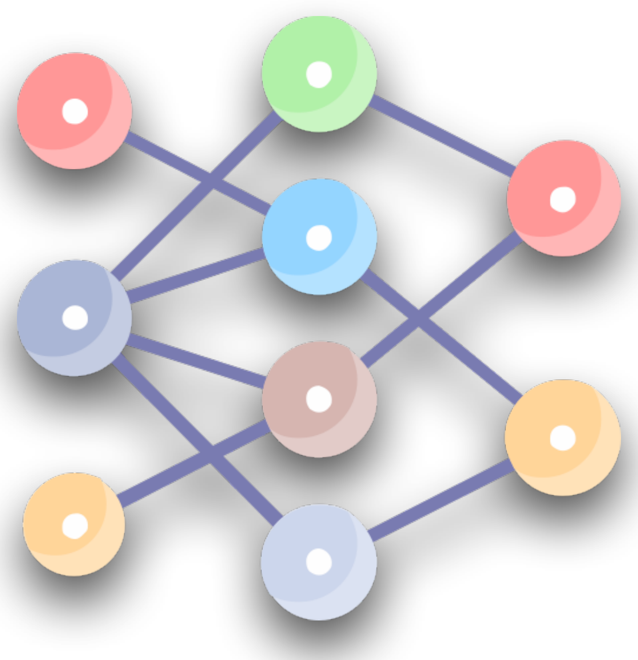
Generating Dynamic Test Suites

2. Optimise networks to cover the selected statement using Neuroevolution

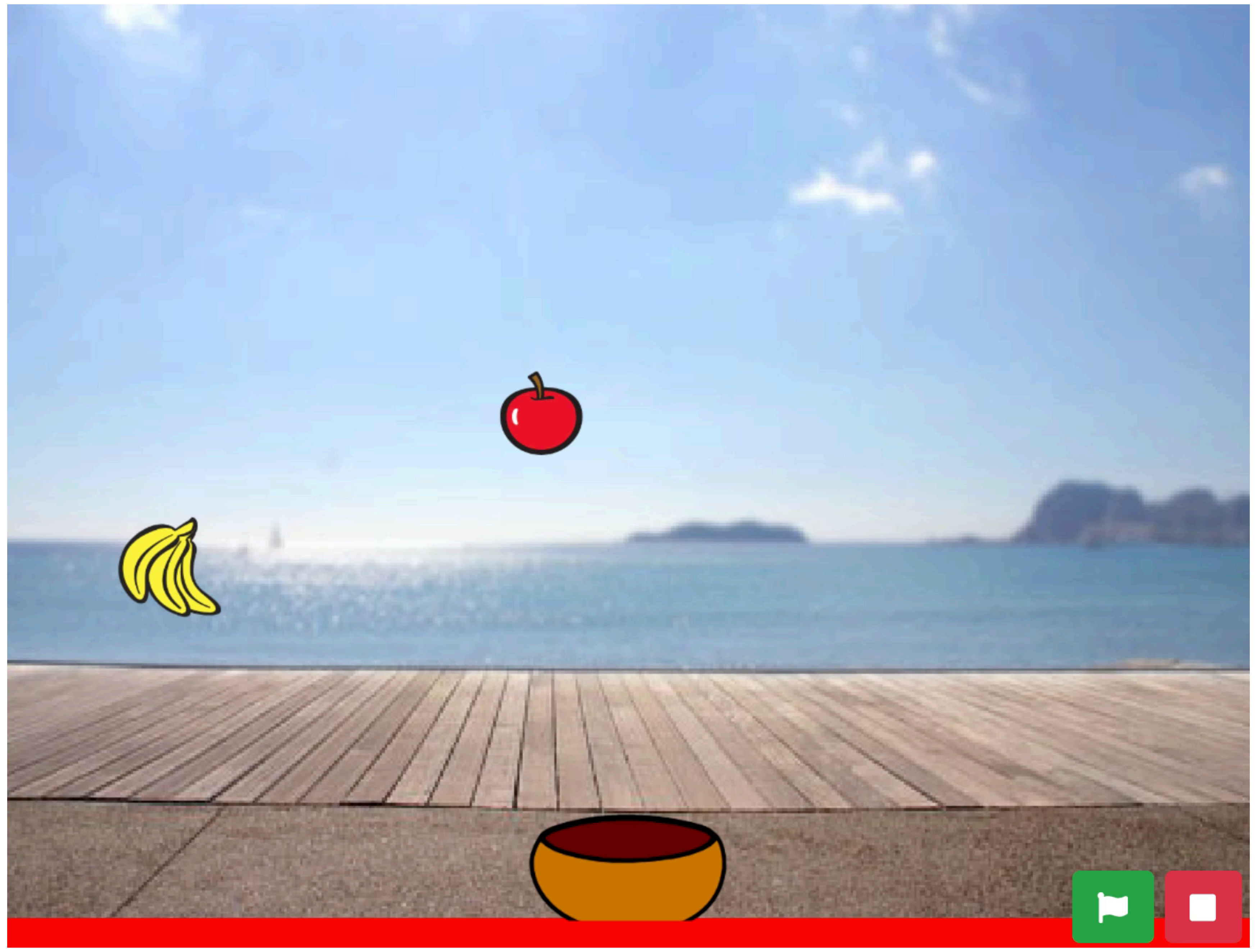
➡ Fitness = distance to target statement

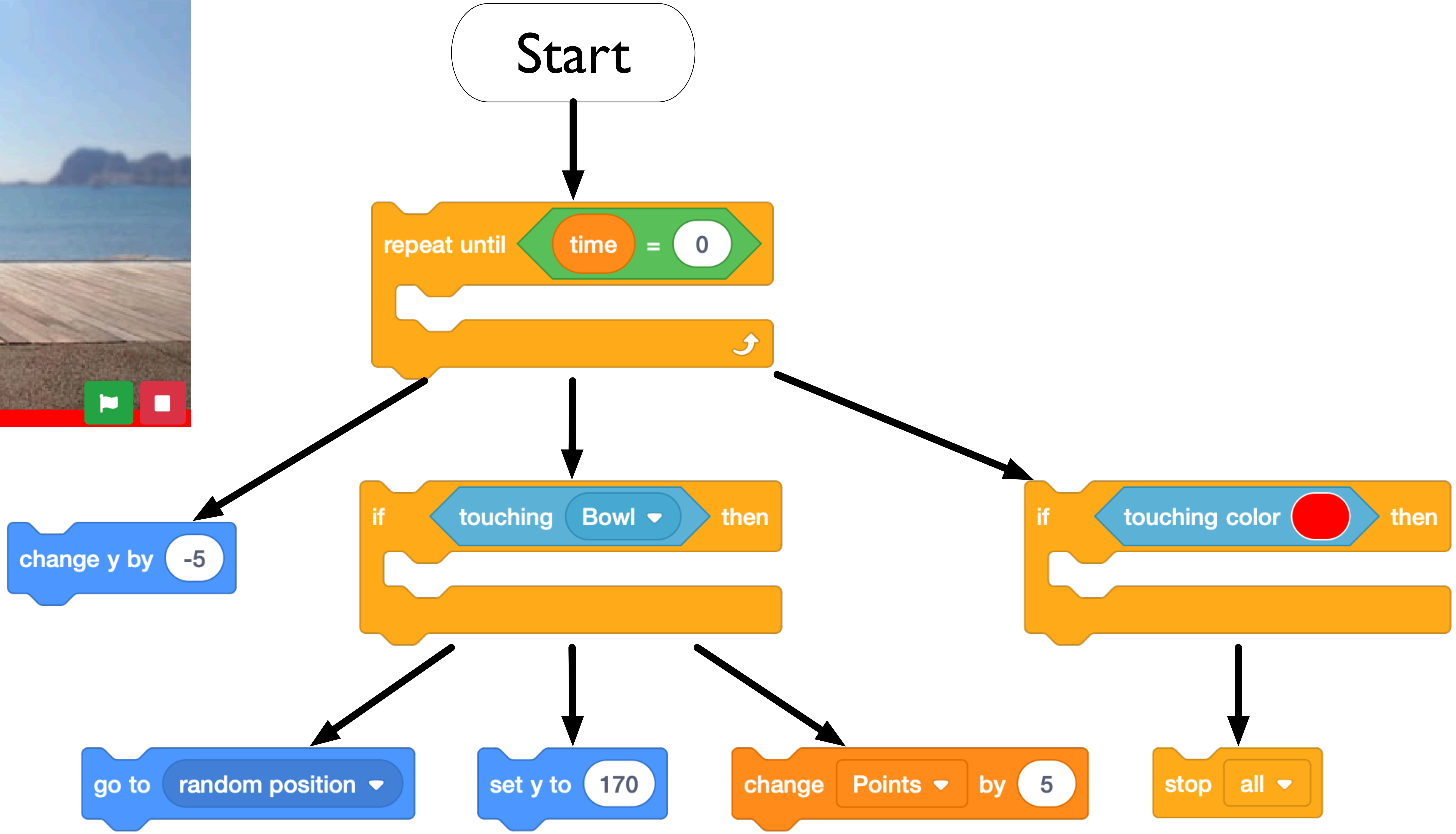


Neuroevolution

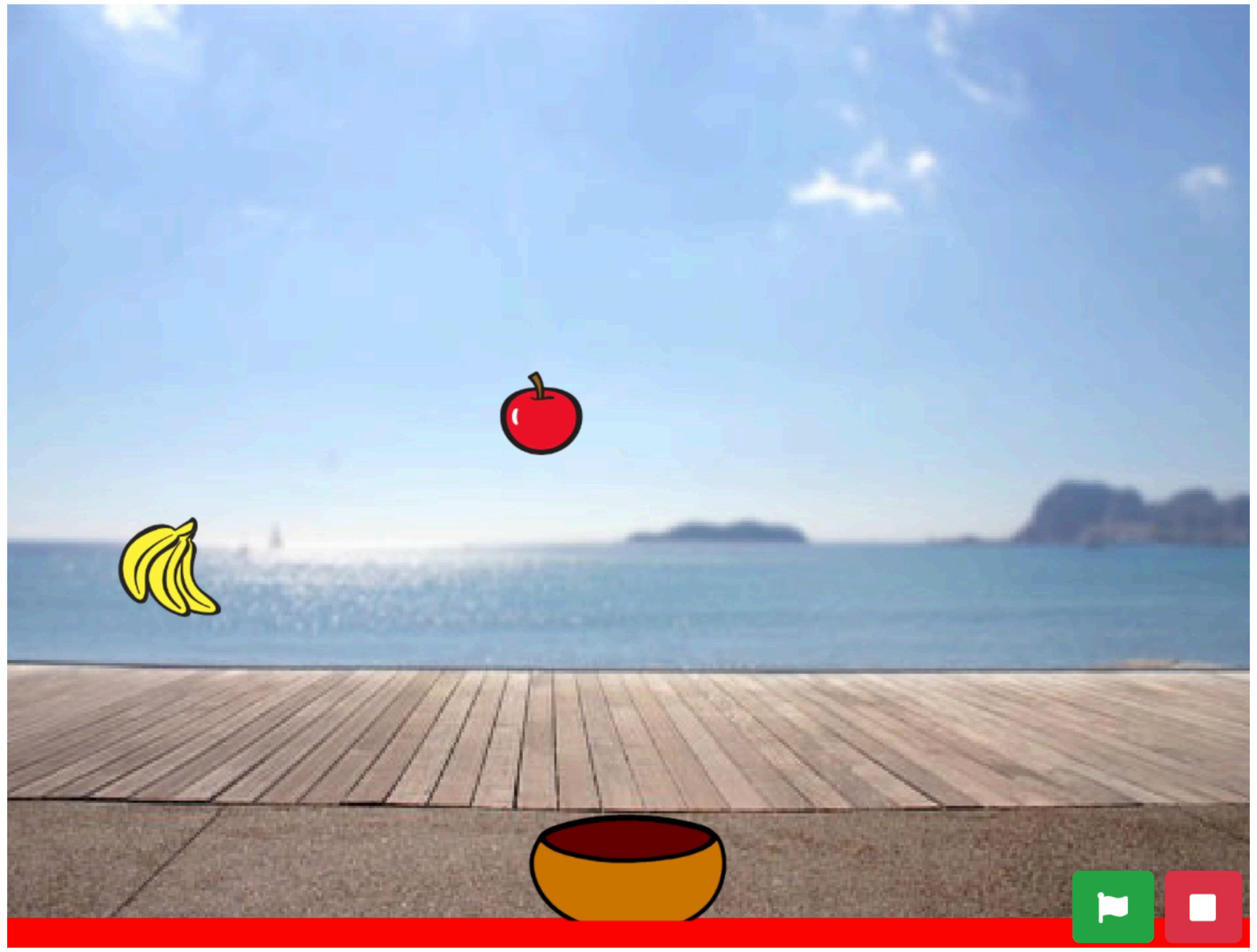


Fitness = Distance to Target





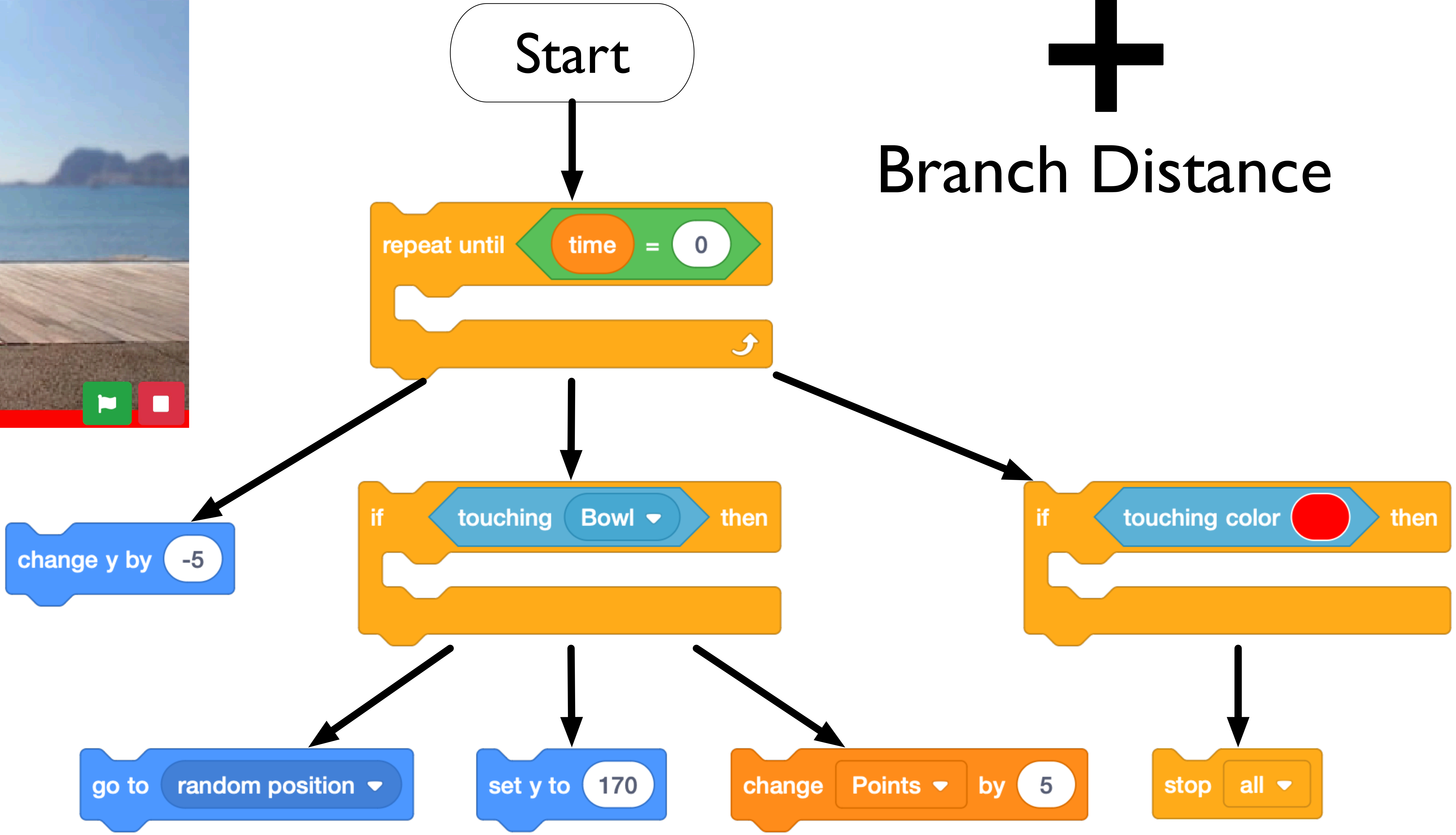
Fitness = Distance to Target



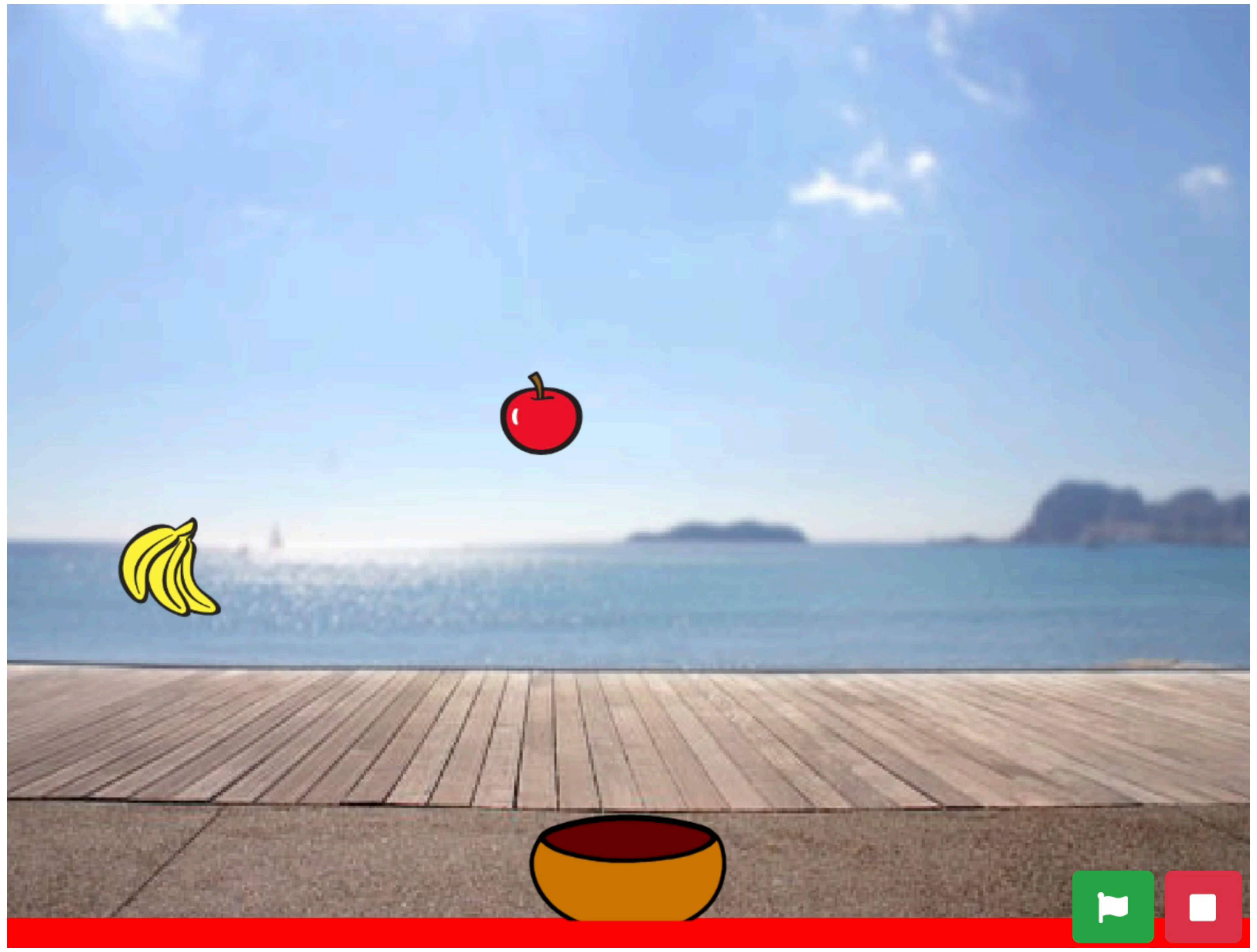
Approach Level

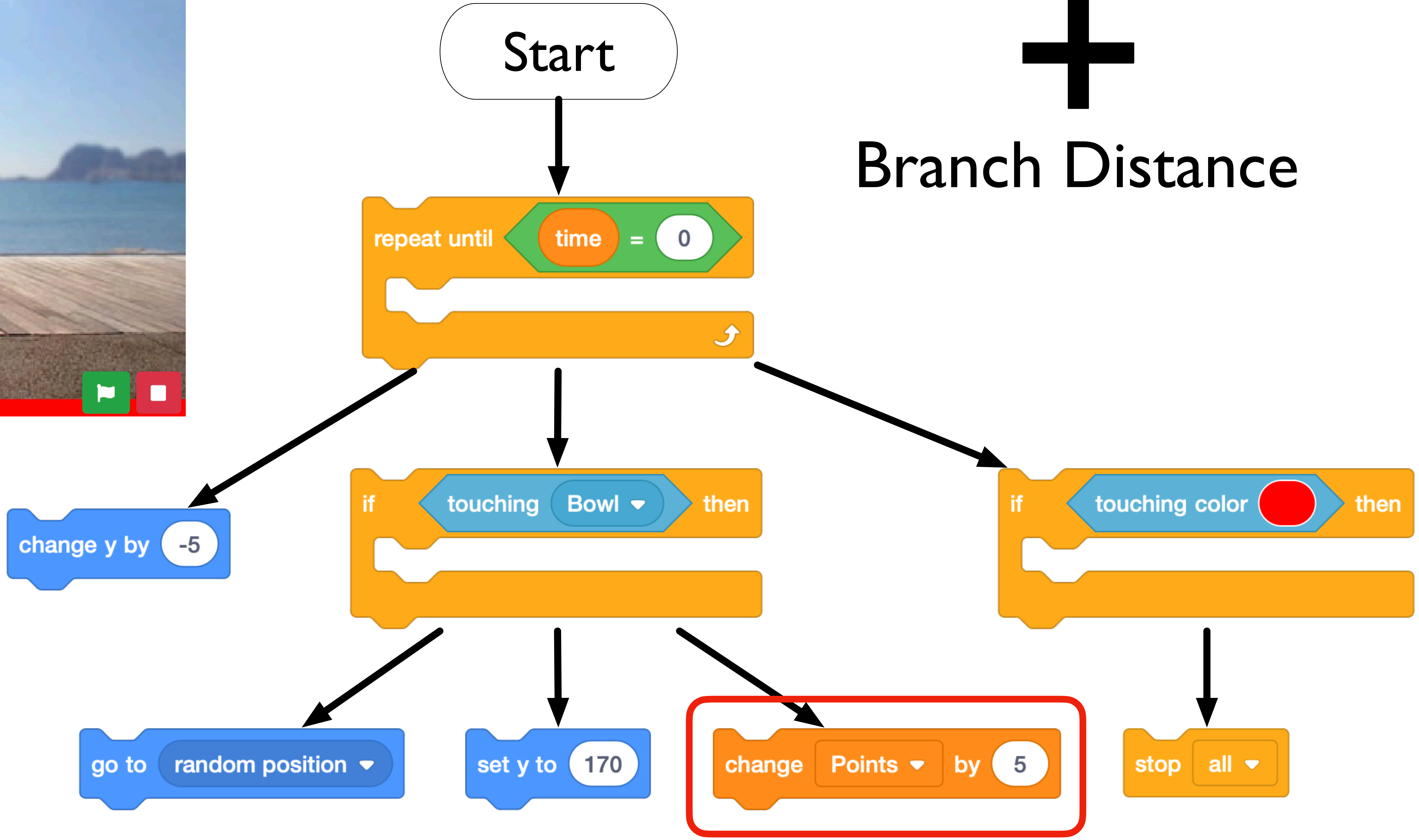
+

Branch Distance



Fitness = Distance to Target



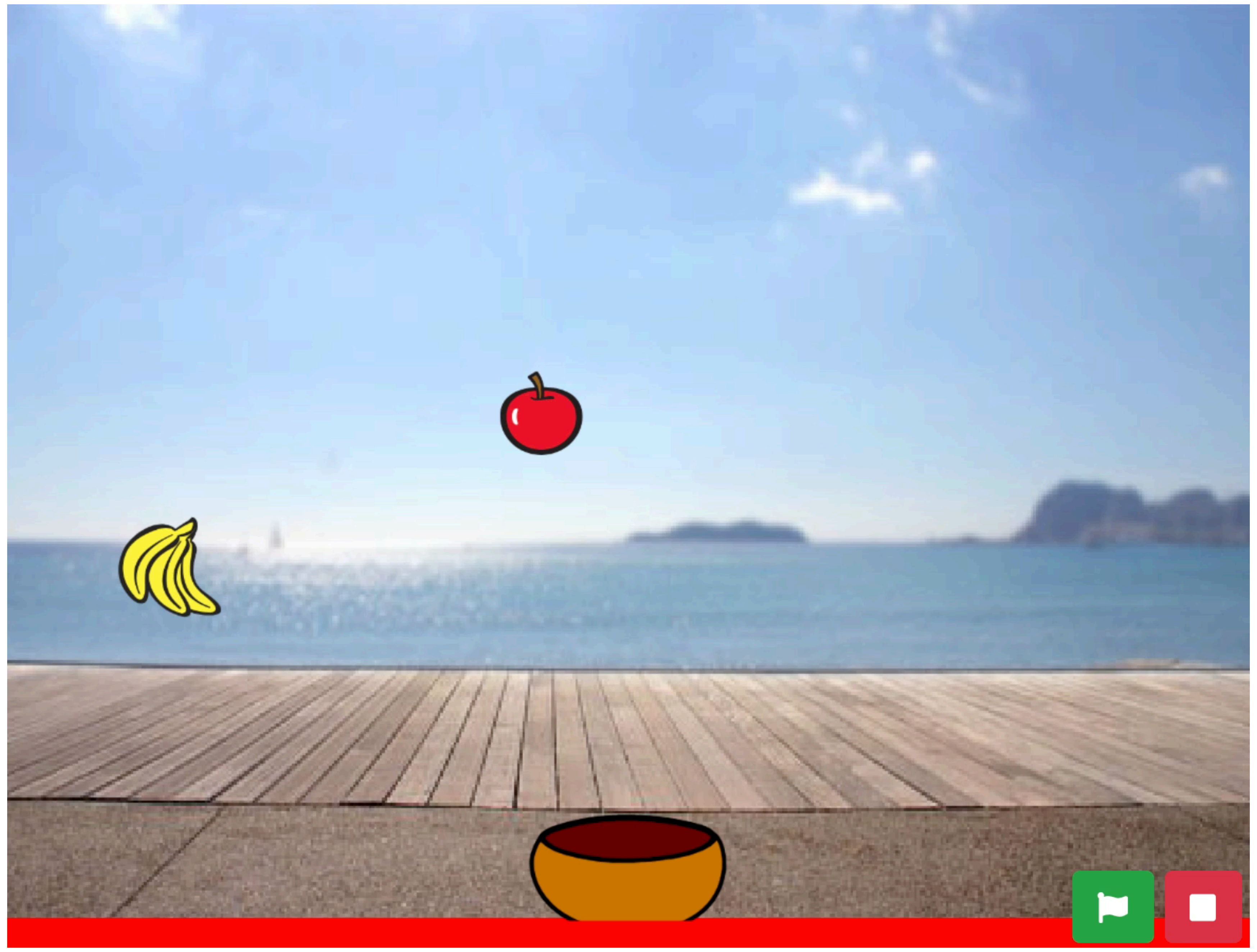


Approach Level

+

Branch Distance

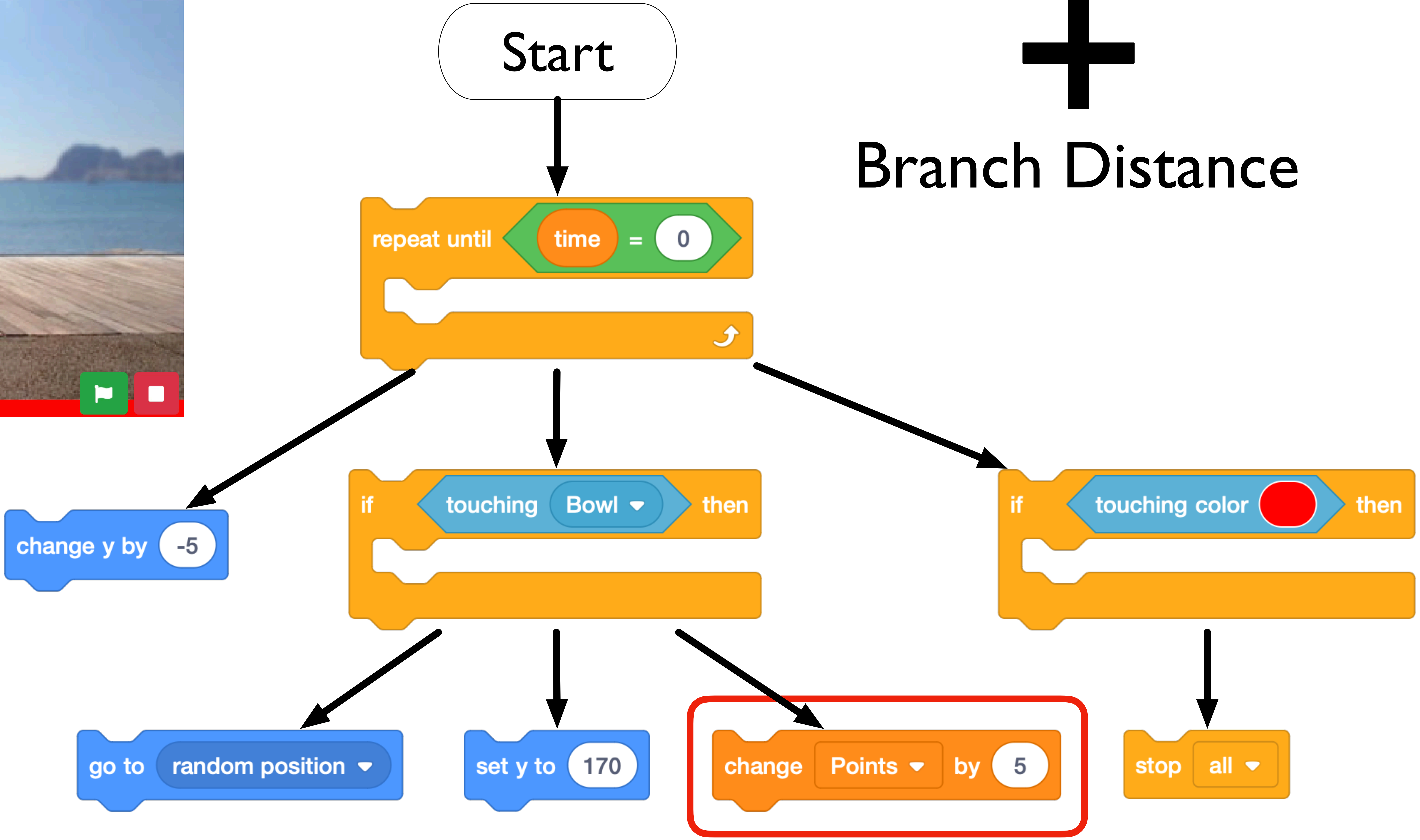
Fitness = Distance to Target



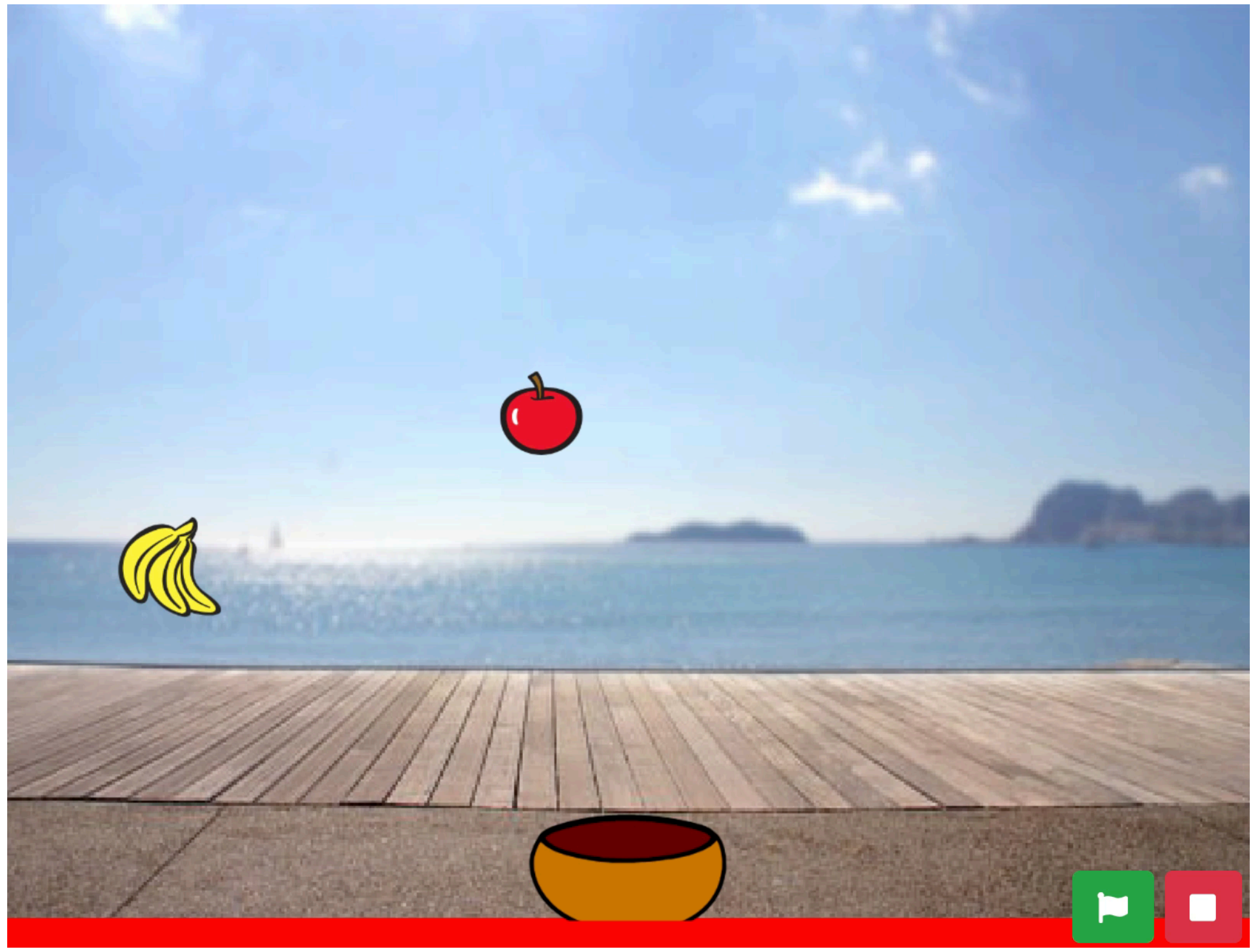
Approach Level = 1

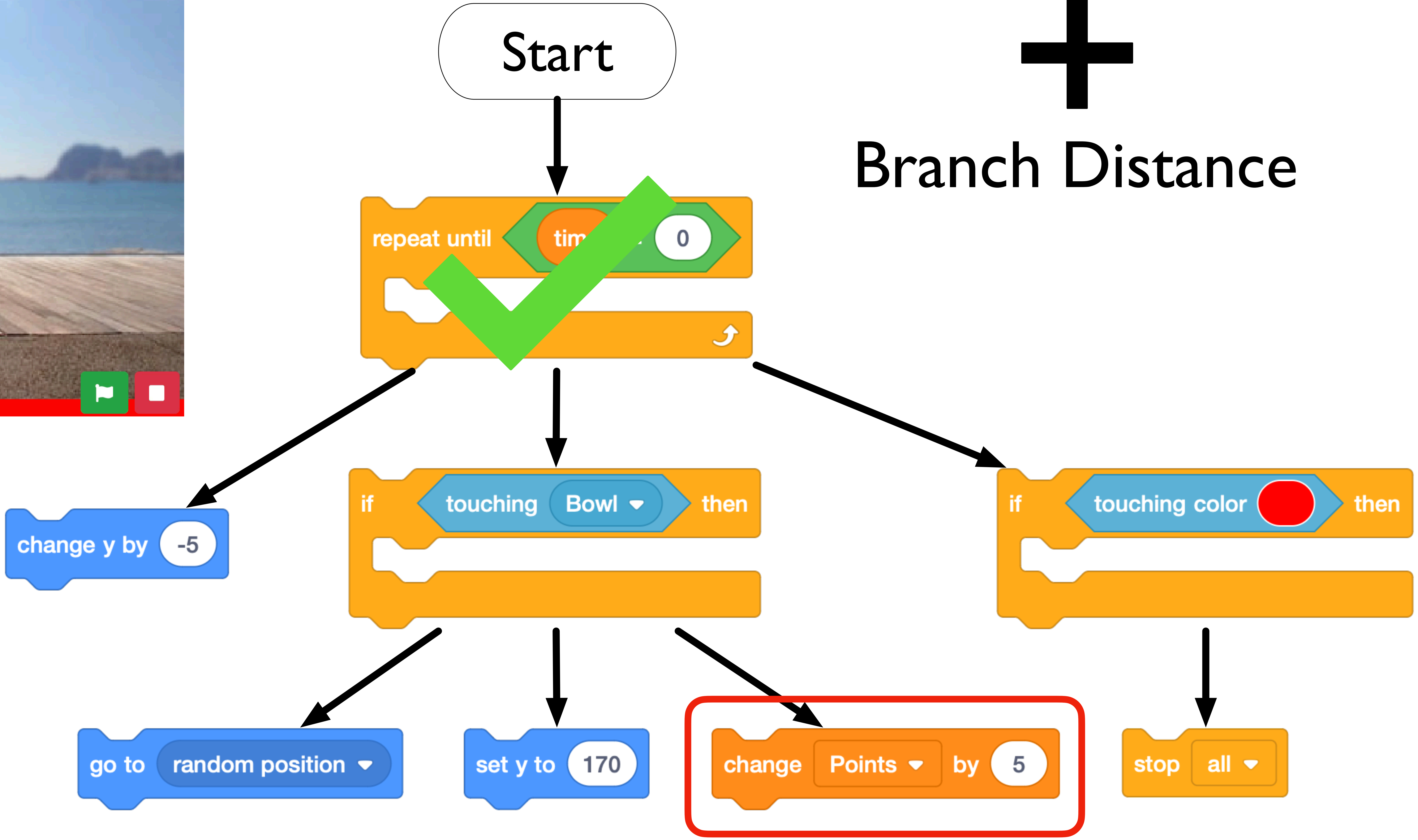
+

Branch Distance



Fitness = Distance to Target



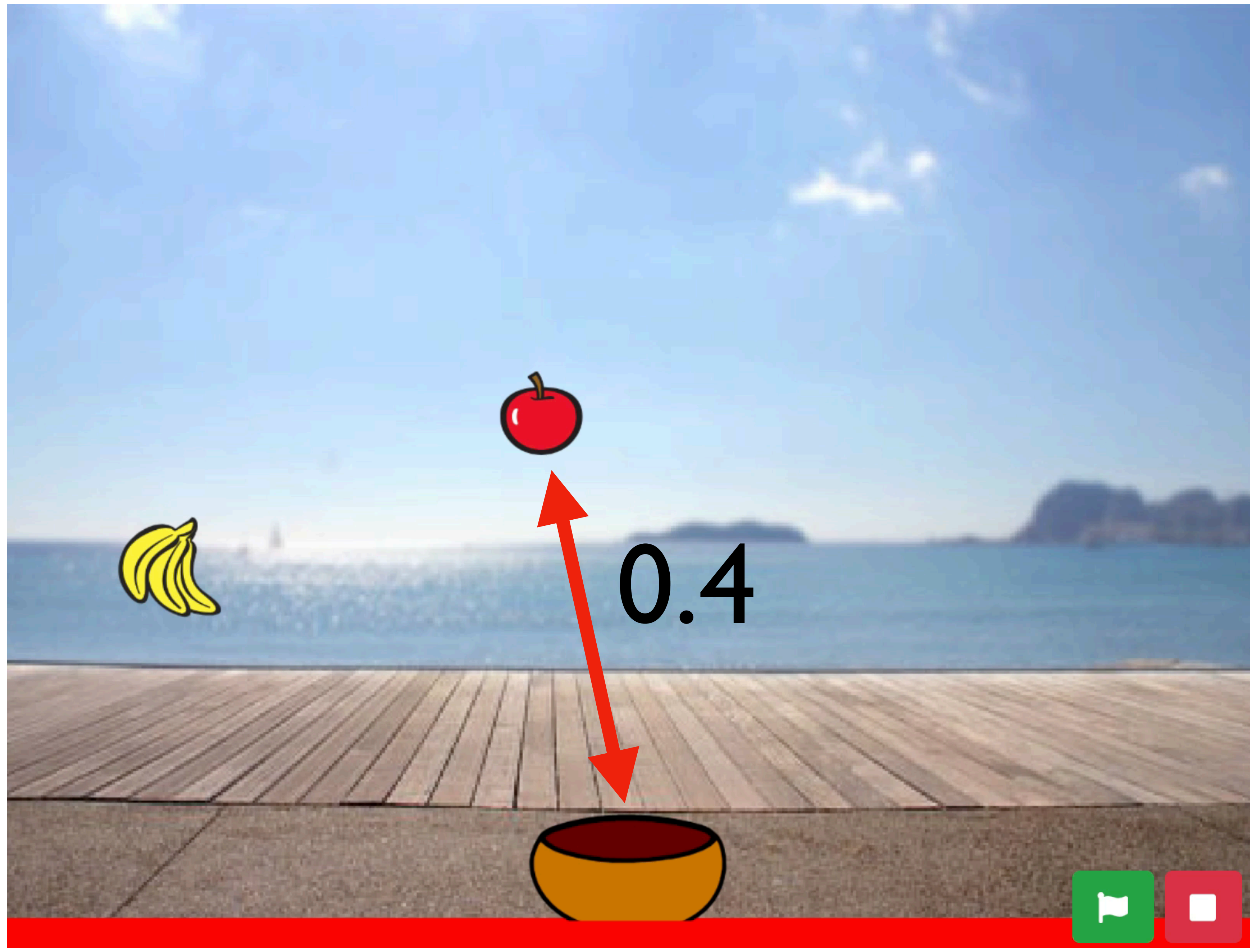


Approach Level = 0

+

Branch Distance

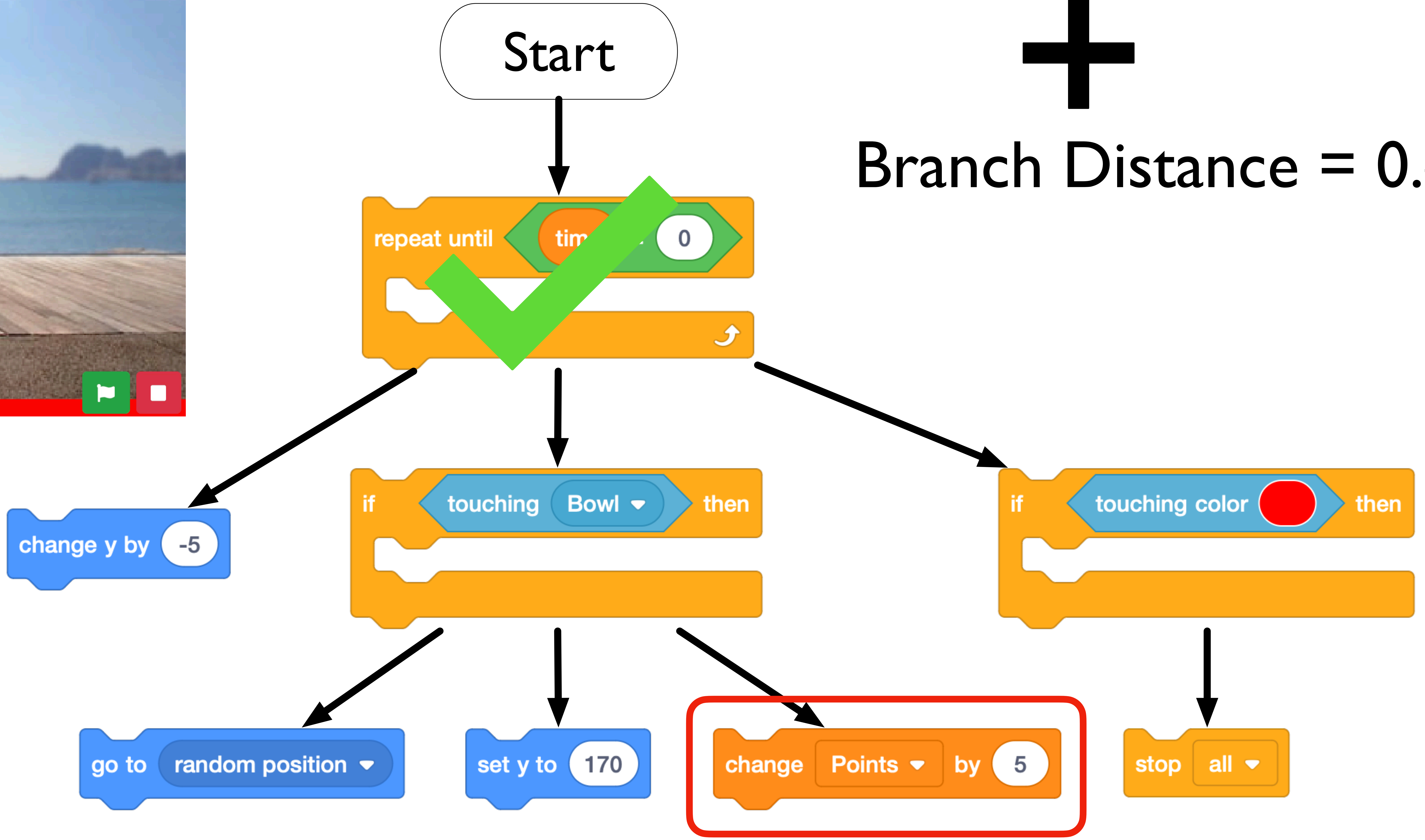
Fitness = Distance to Target



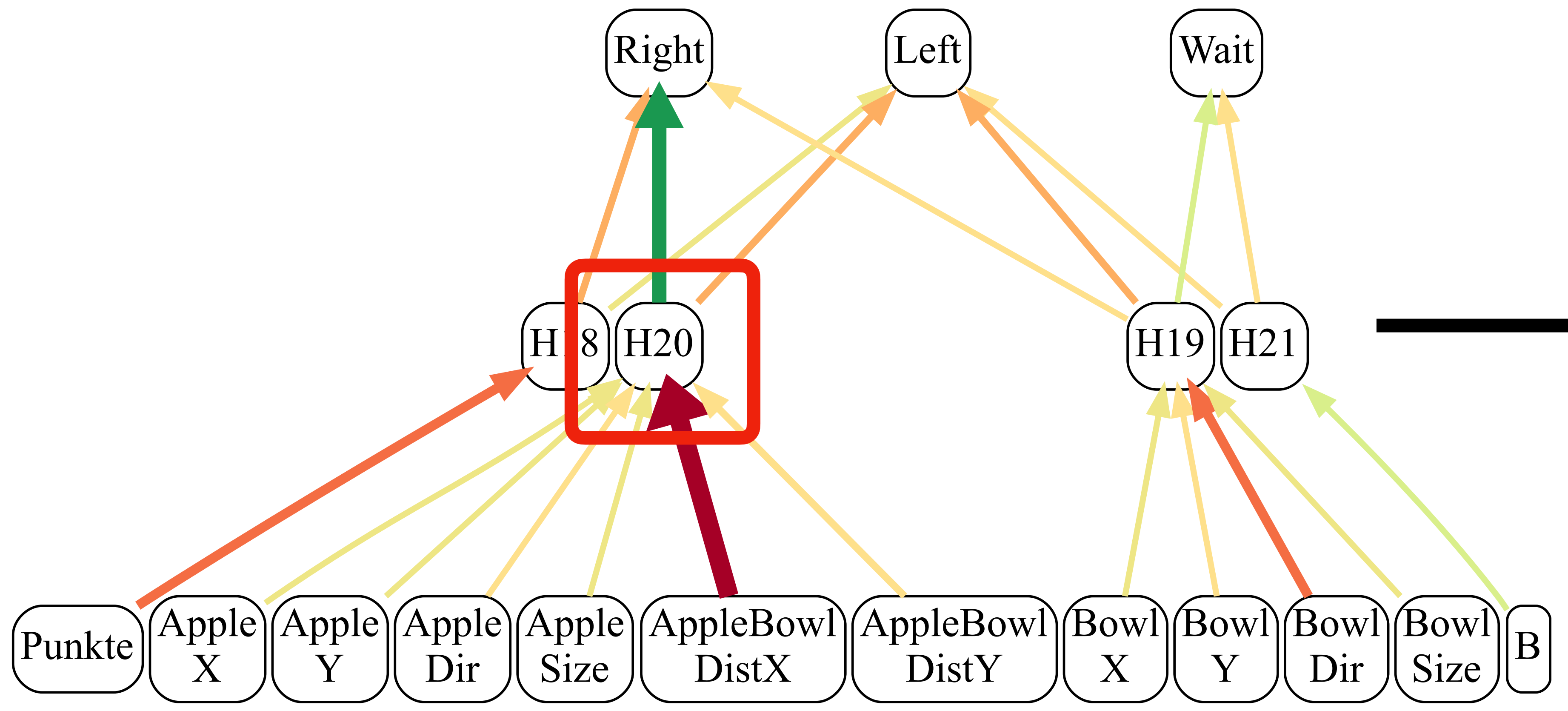
Approach Level = 0

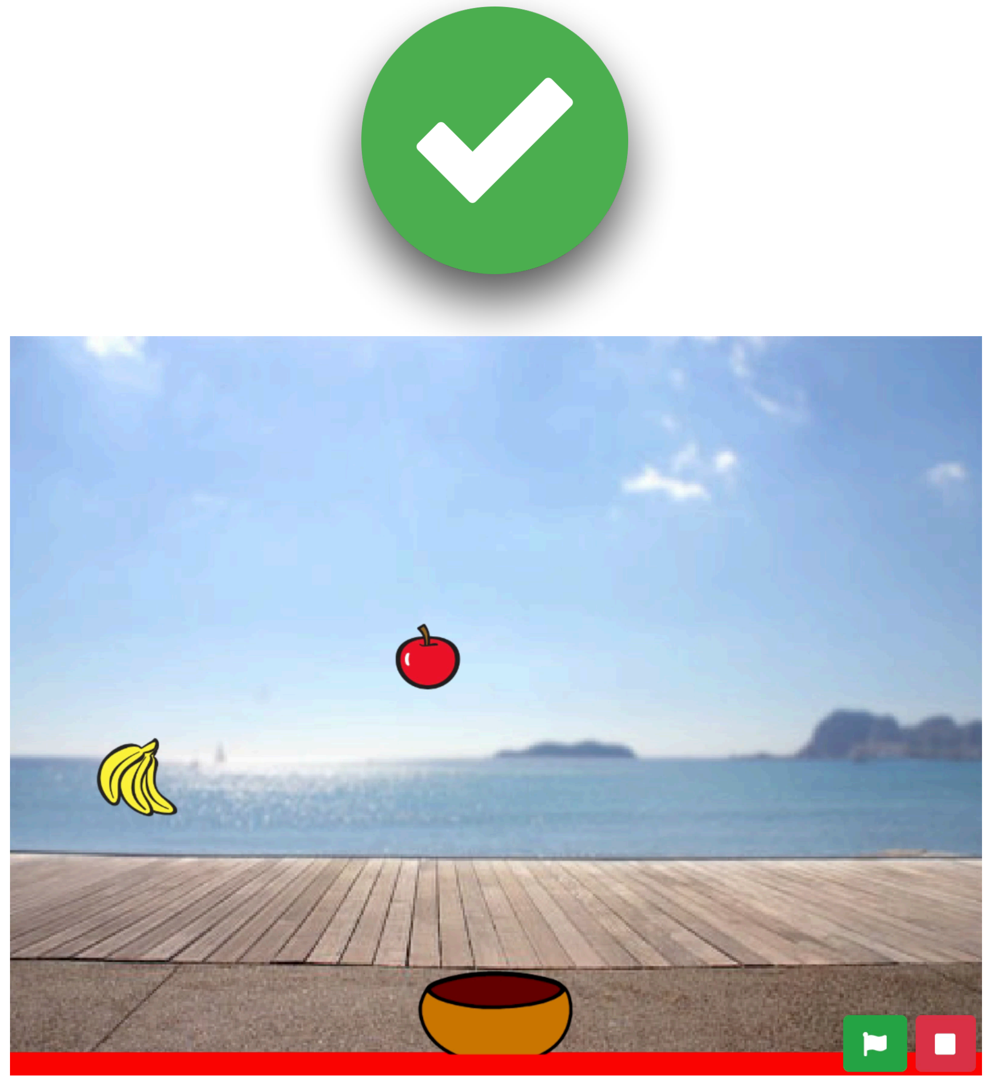
+

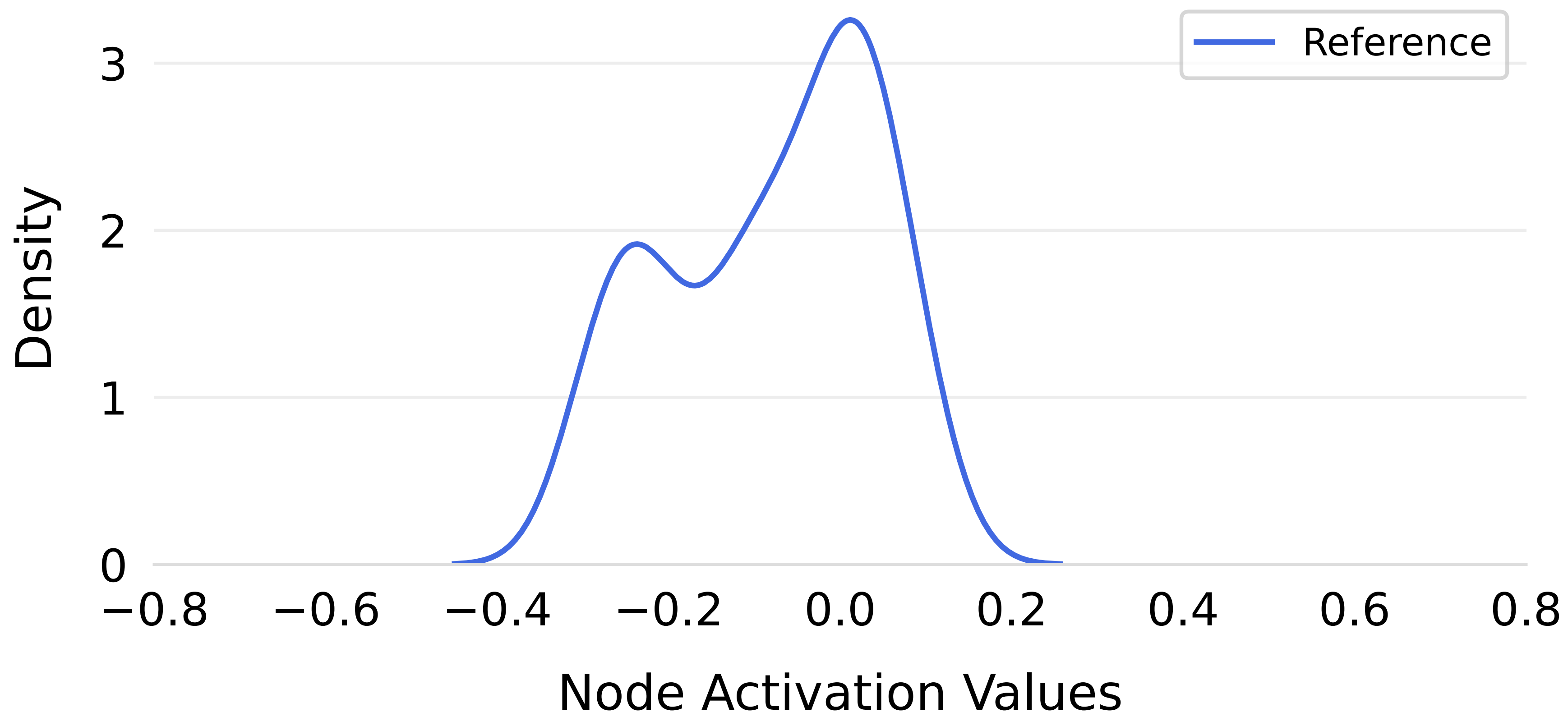
Branch Distance = 0.4



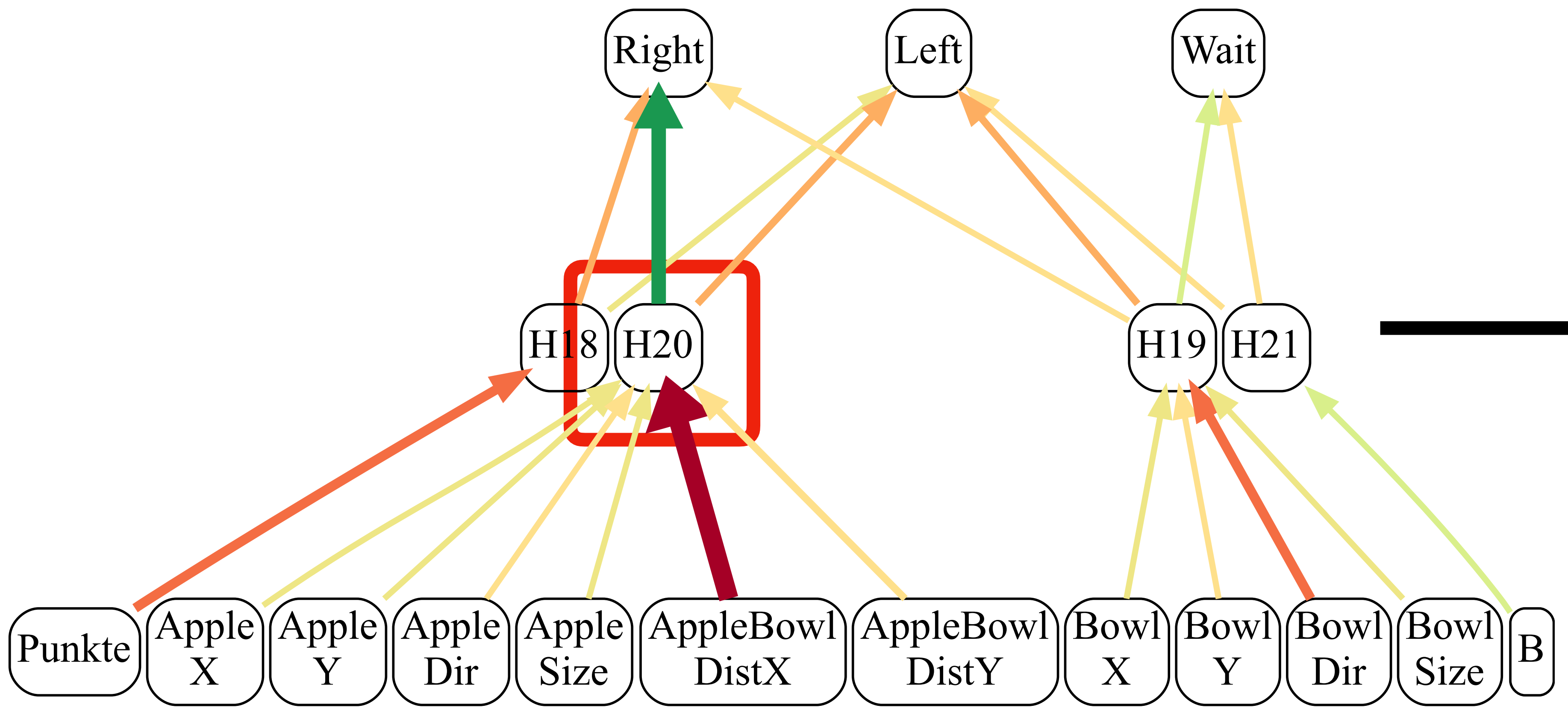
Test Oracle Based on Surprise Adequacy

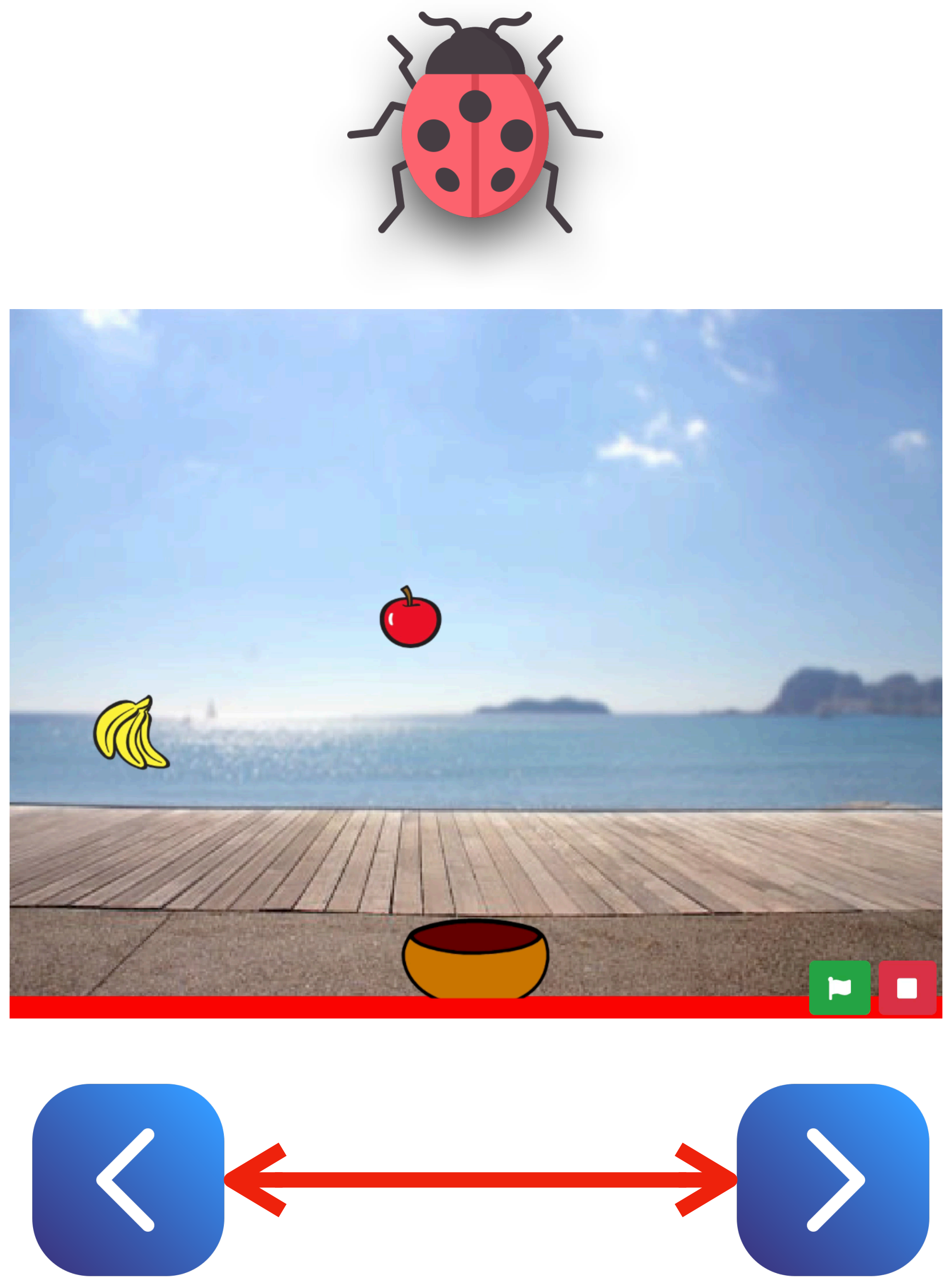


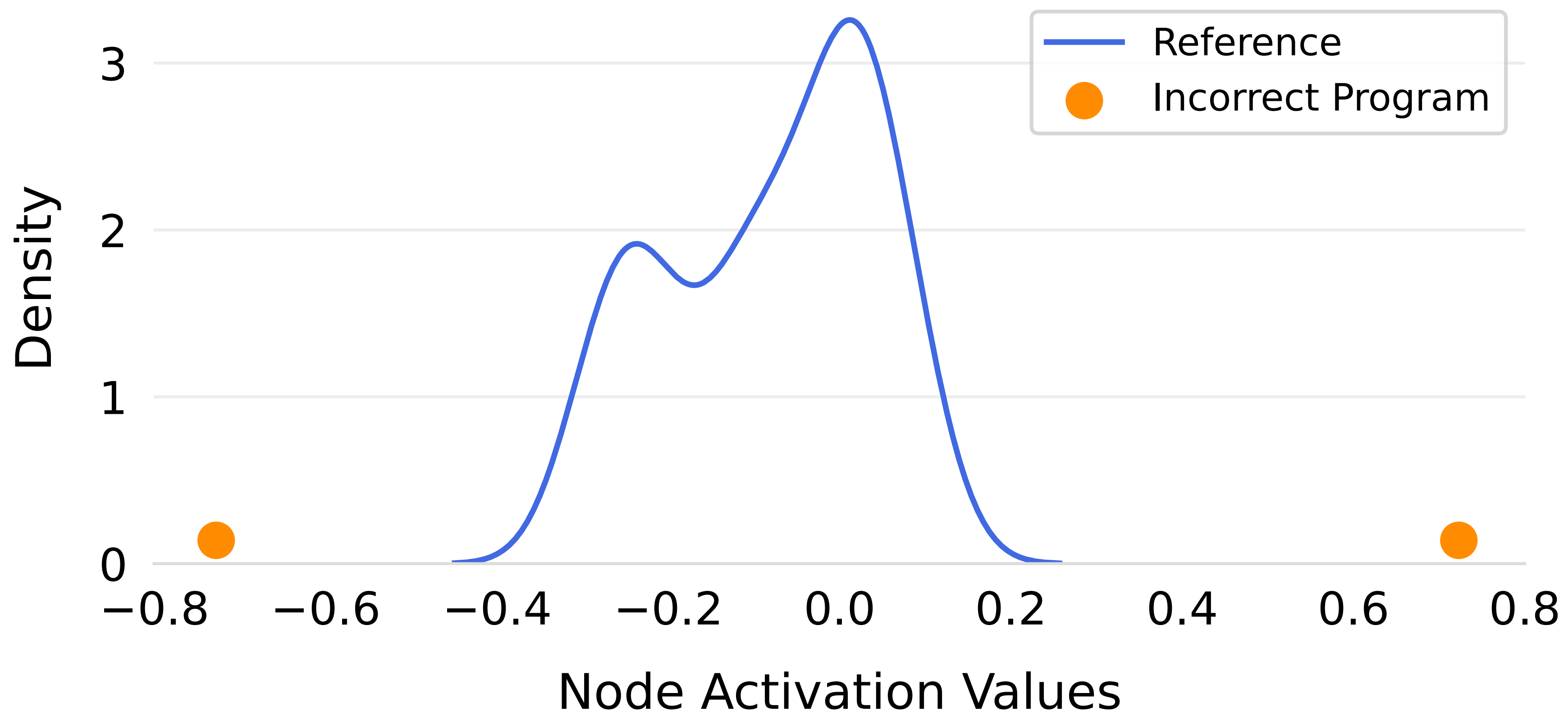




Test Oracle Based on Surprise Adequacy







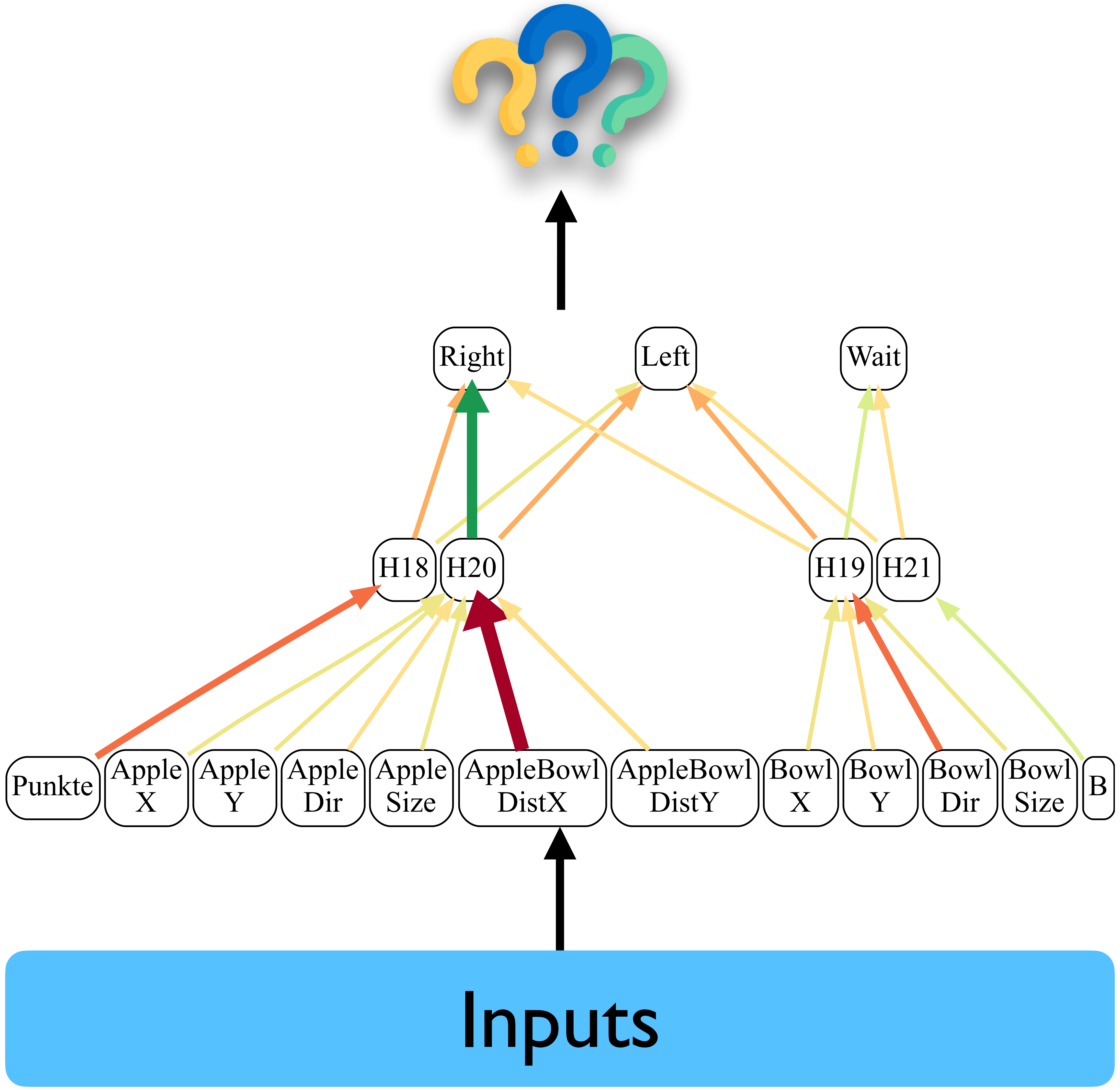
Test Oracle Based on Surprise Adequacy

- Surprise Adequacy measures how much networks are surprised by the input they receive compared to previous inputs

➡ Low ~ similar behaviour ~ correct

➡ High ~ suspicious behaviour ~ incorrect

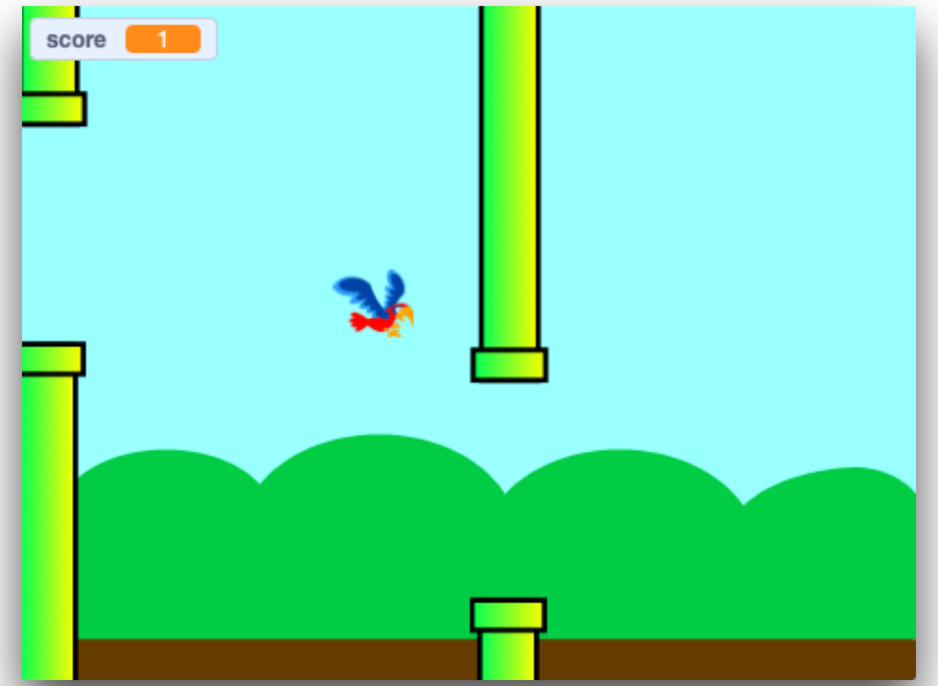
➡ Regression testing approach

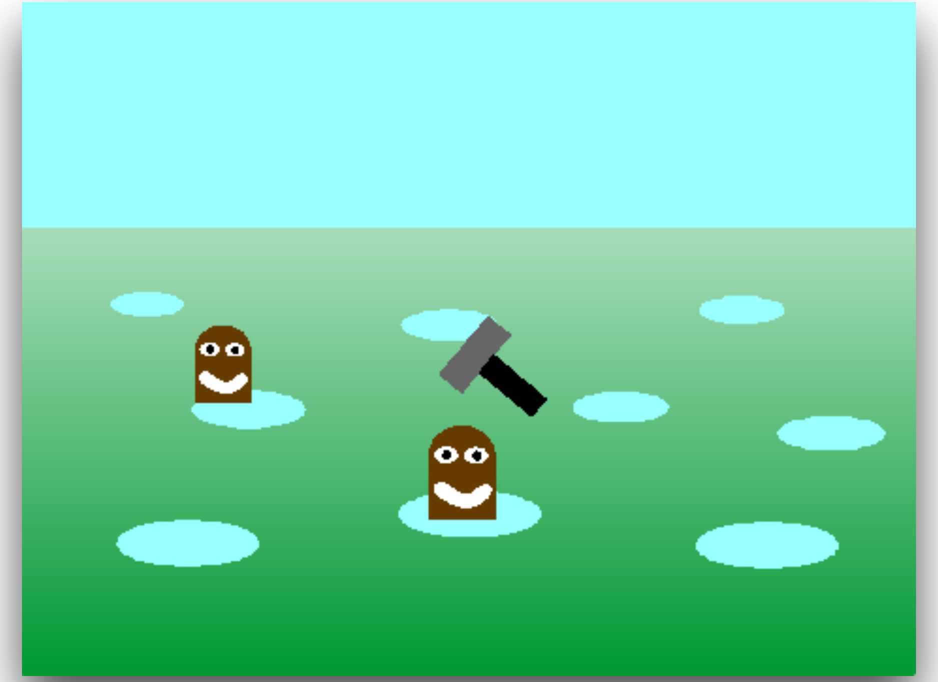
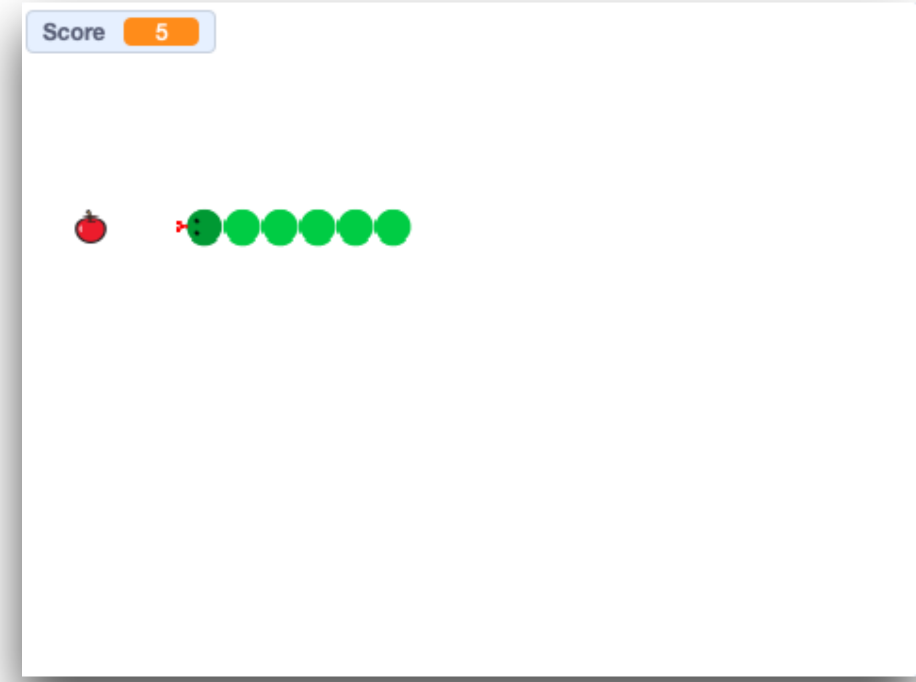


Evaluation of Neatest

Dataset of 25 Scratch games



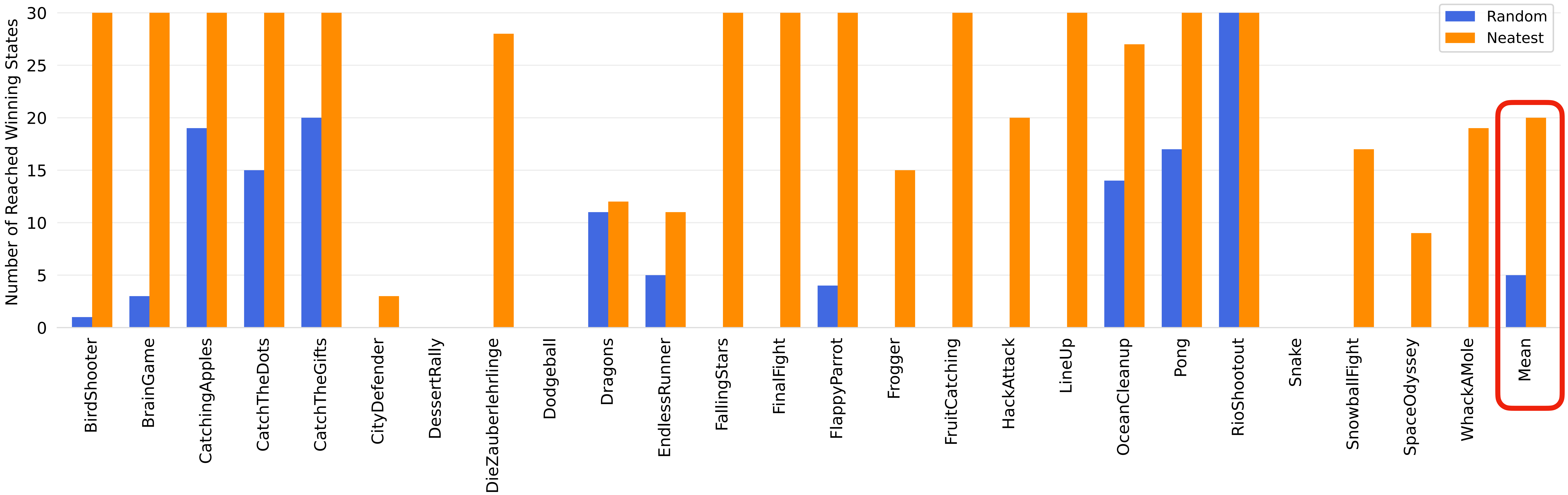


Neatest Covers Scratch Games Reliably

- Compares Neatest with random test generation baseline
- Statements are covered if generated test passes the robustness check 10 times

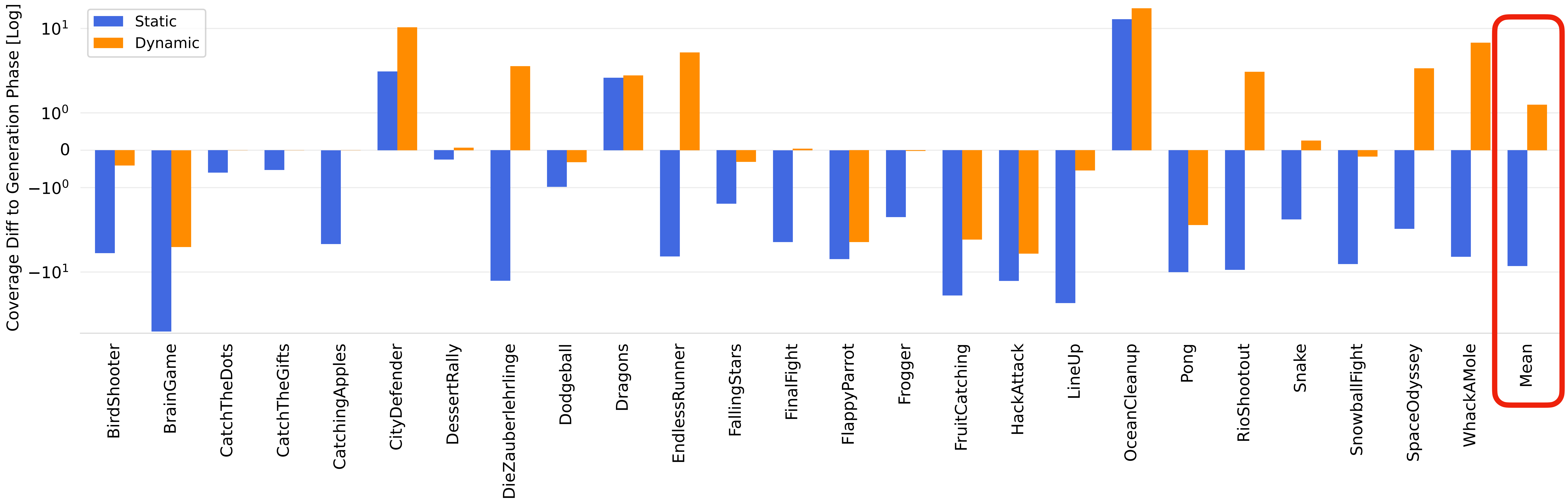
➡ Neatest wins games on average 20/30 times



Dynamic Tests are Robust Against Randomisation

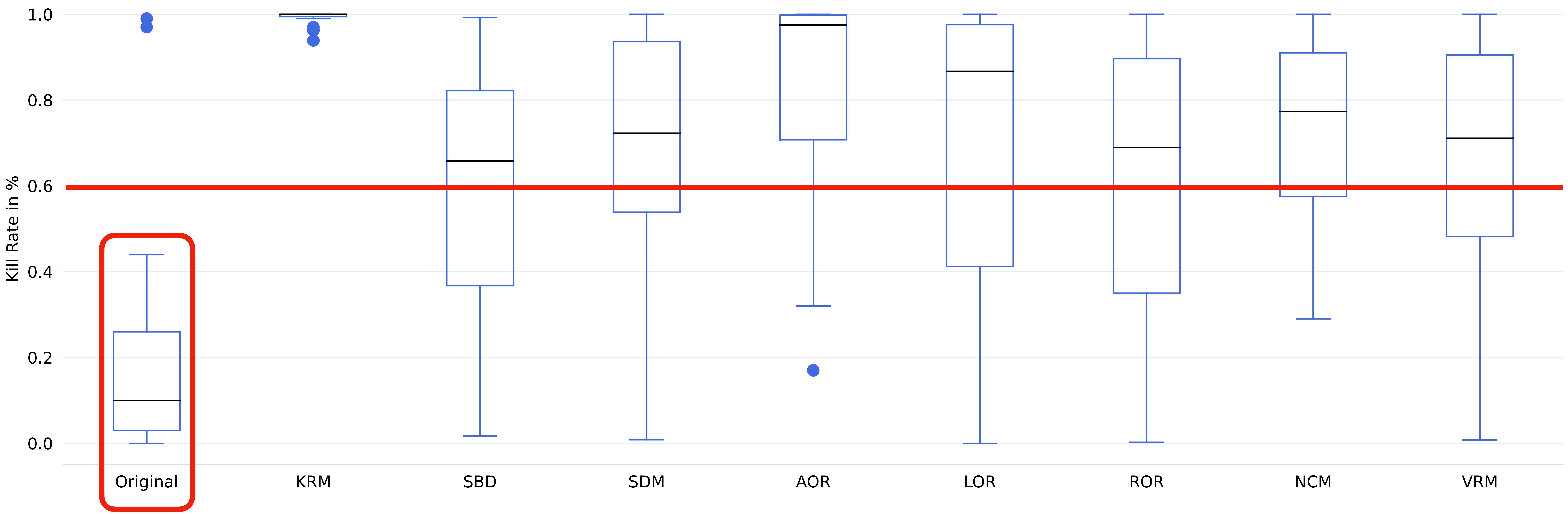
- Execute generated static and dynamic tests
- No robustness check

➡ Contrary to static suites, dynamic suites do not lose in coverage

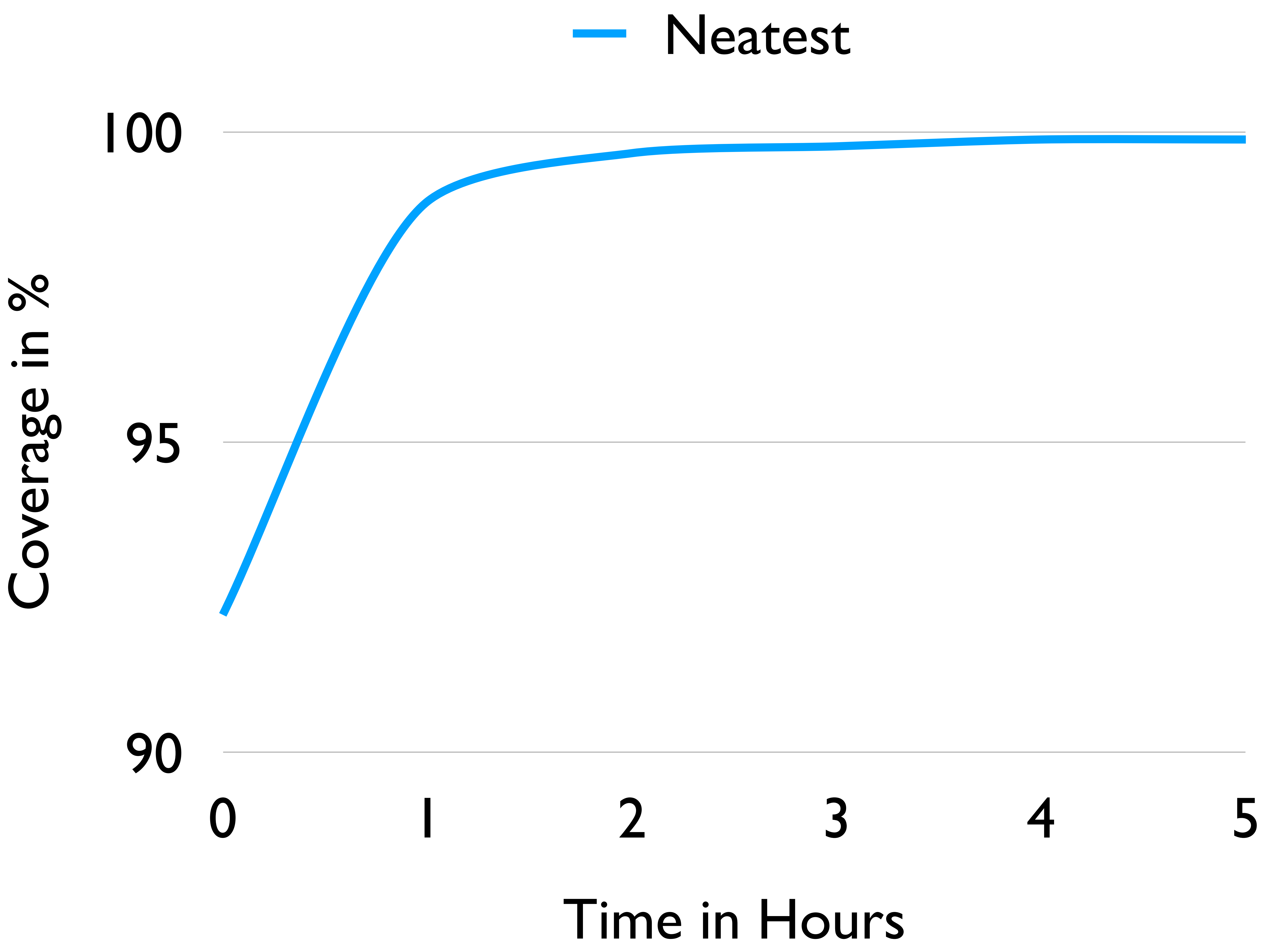


Dynamic Networks as Test Oracles

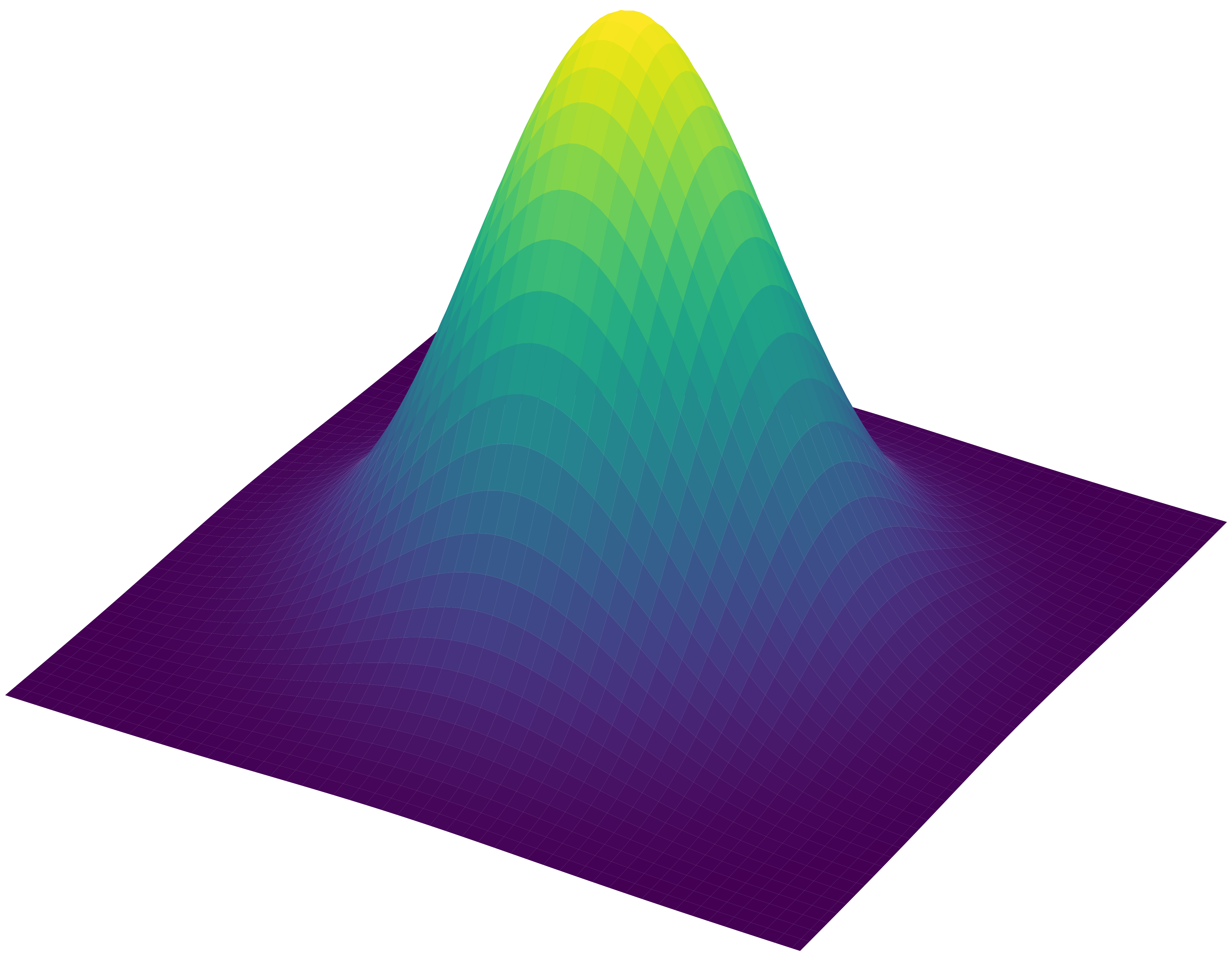
- Mutation analysis on 243835 mutants using 8 mutation operators
  - ➡High true-positive median of > 60%
  - ➡Low false-positive median of 10%



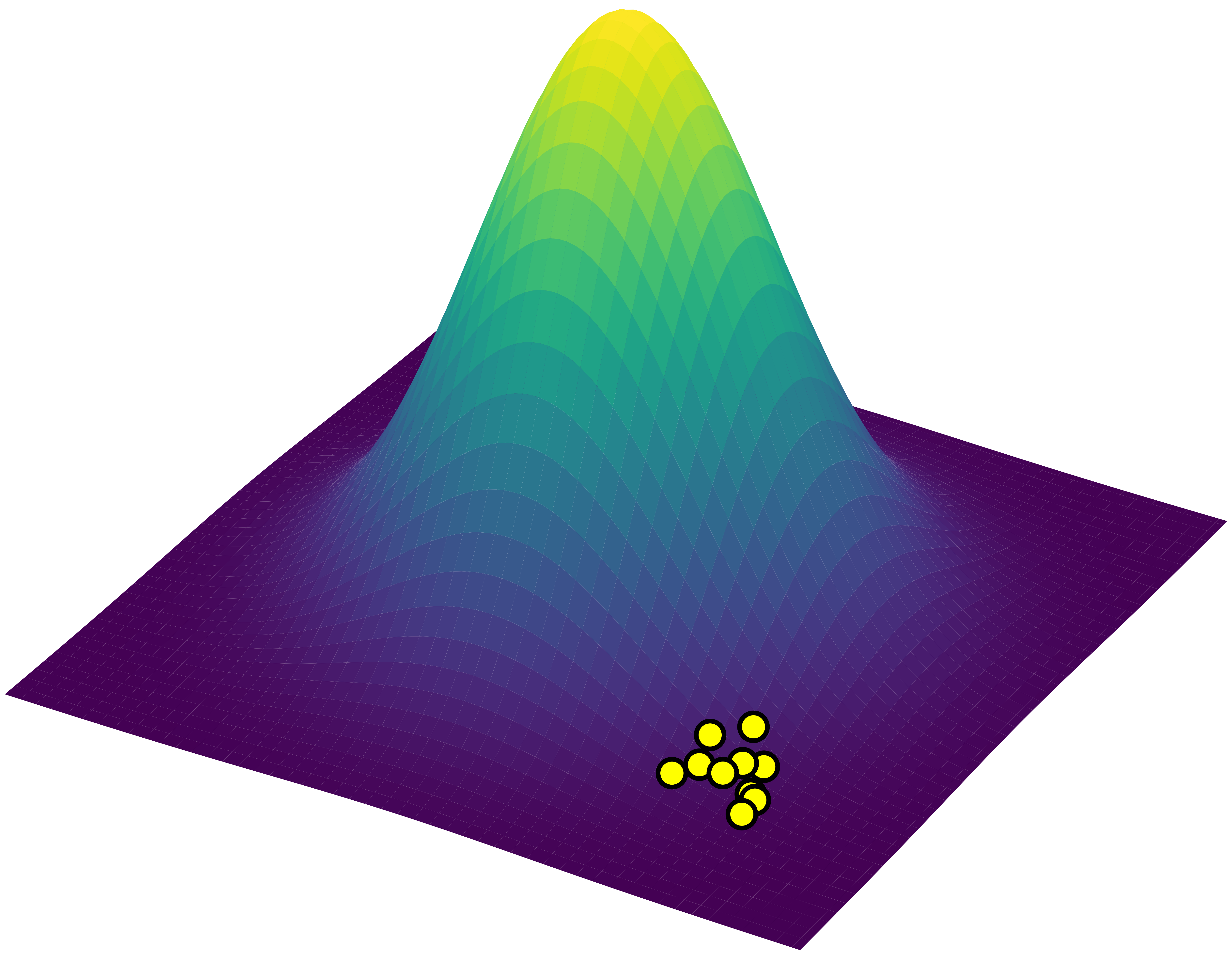
Good but Slow Performance of Neatest



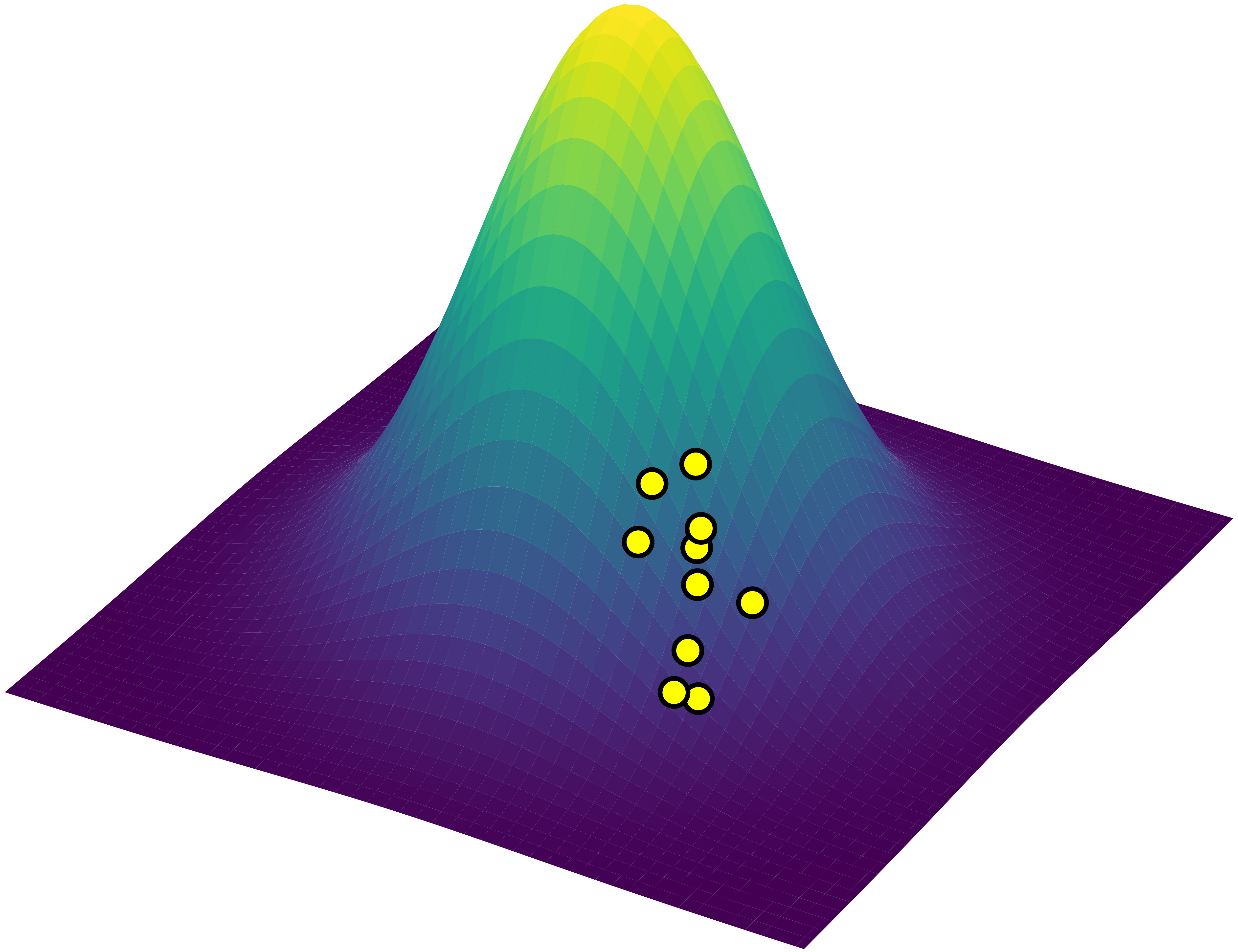
Slow Progress of Stochastic Weight Mutation



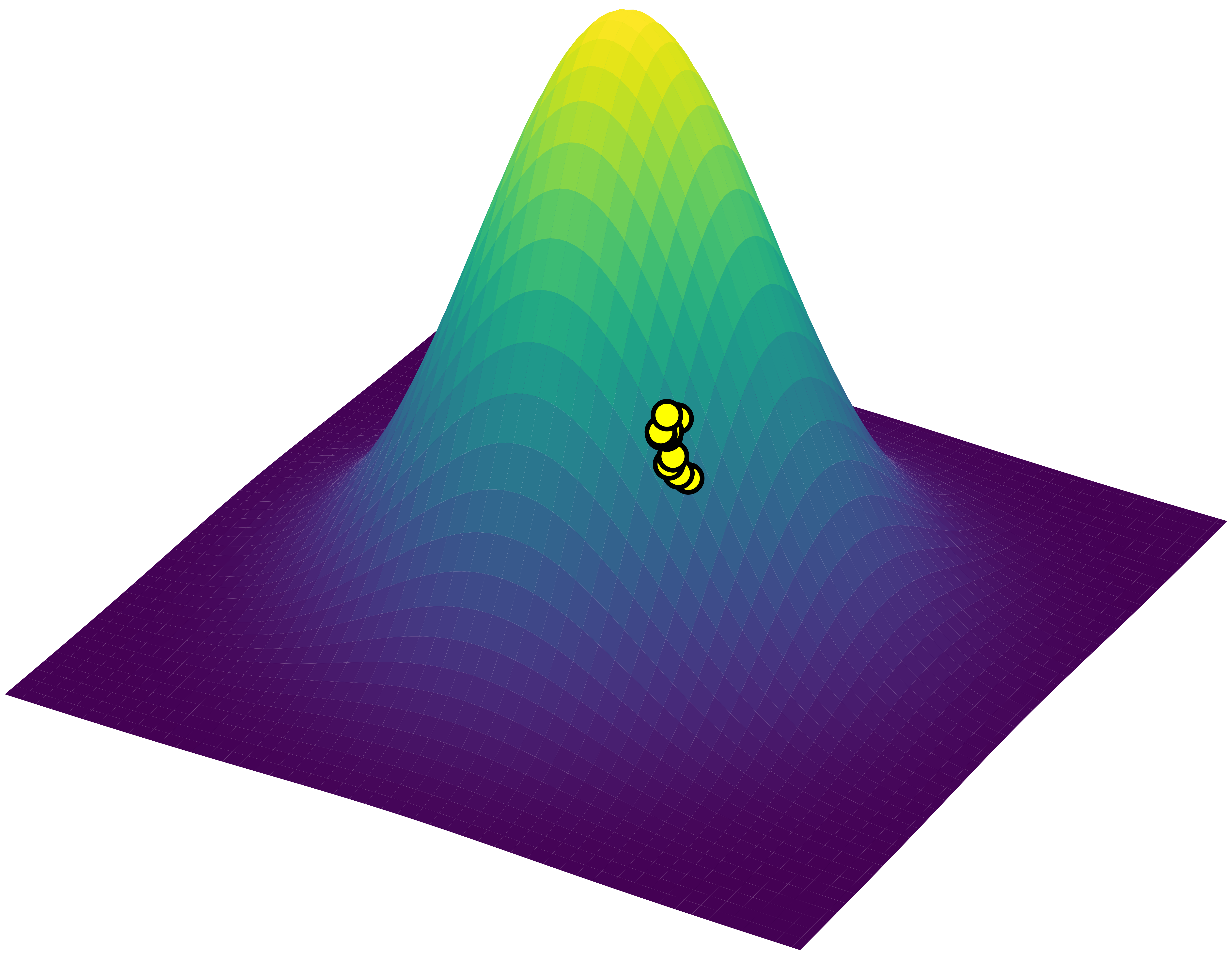
Slow Progress of Stochastic Weight Mutation



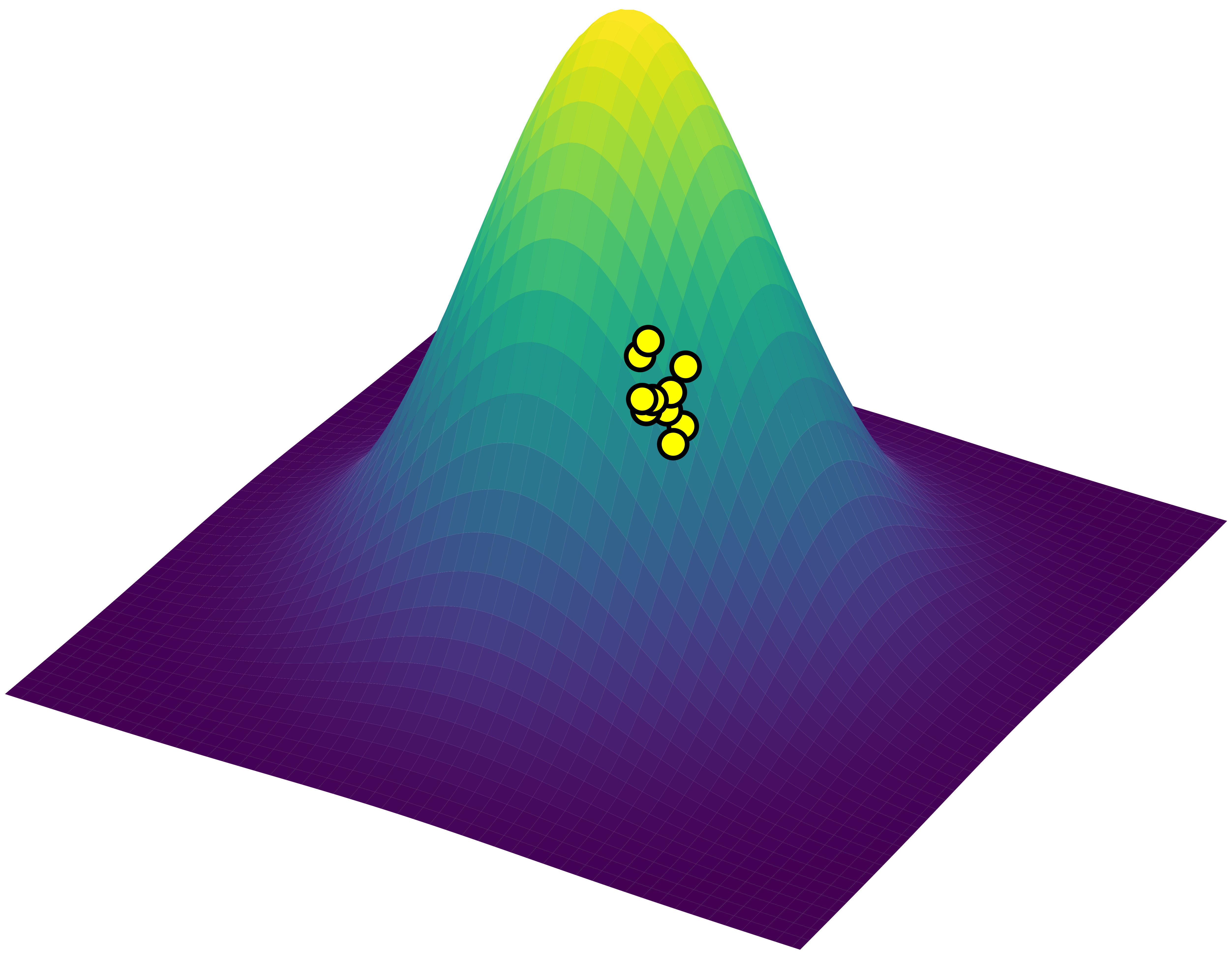
Slow Progress of Stochastic Weight Mutation



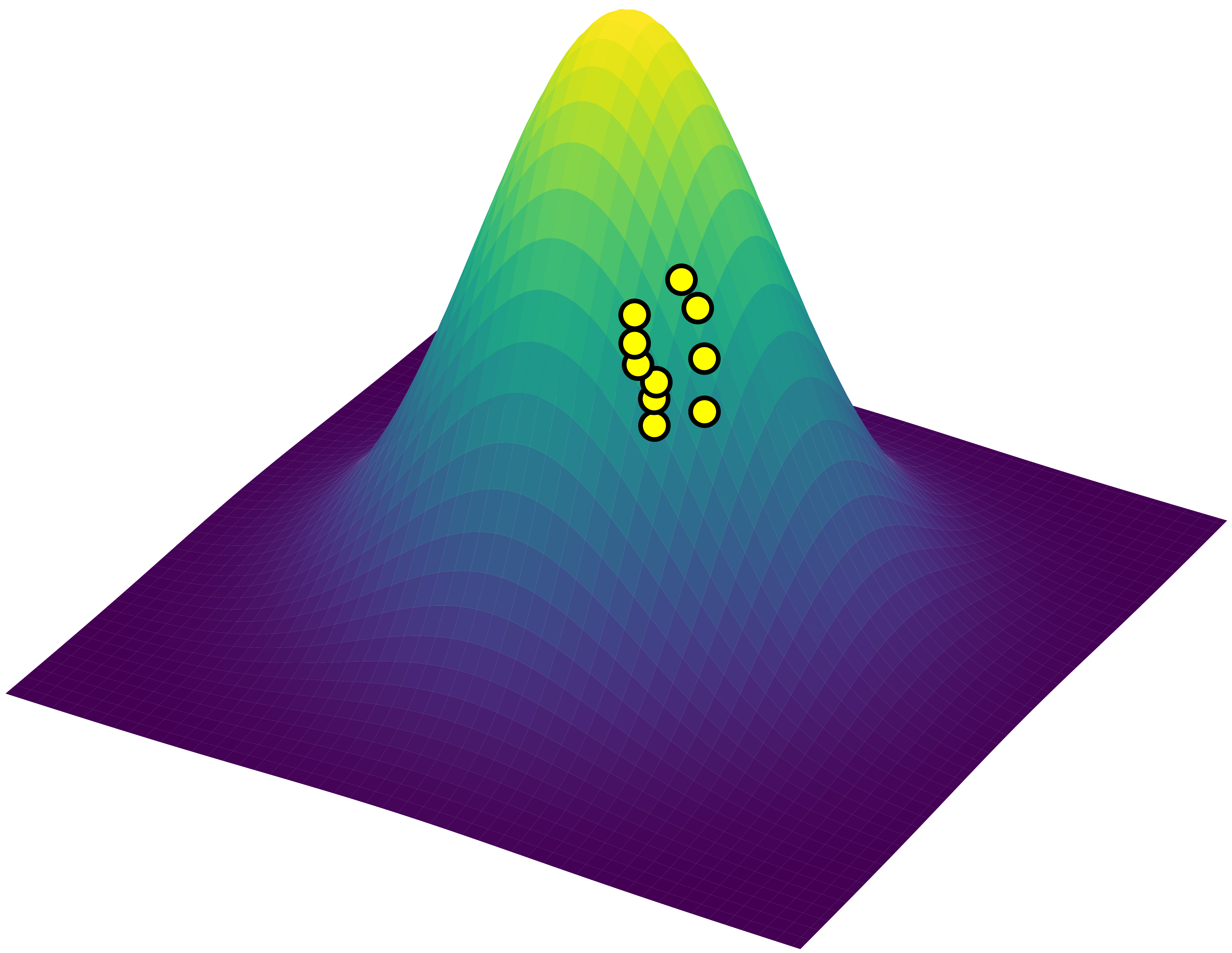
Slow Progress of Stochastic Weight Mutation



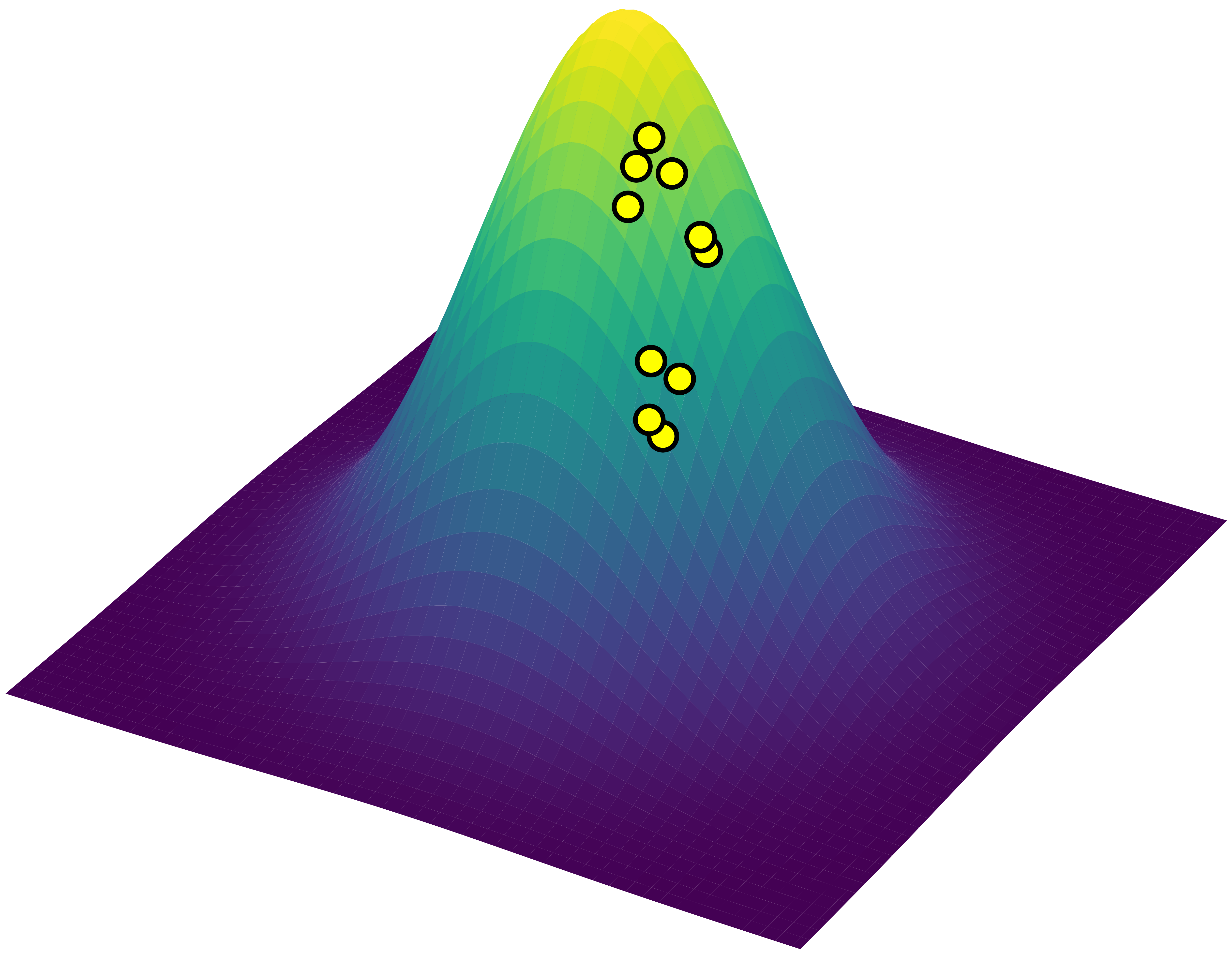
Slow Progress of Stochastic Weight Mutation



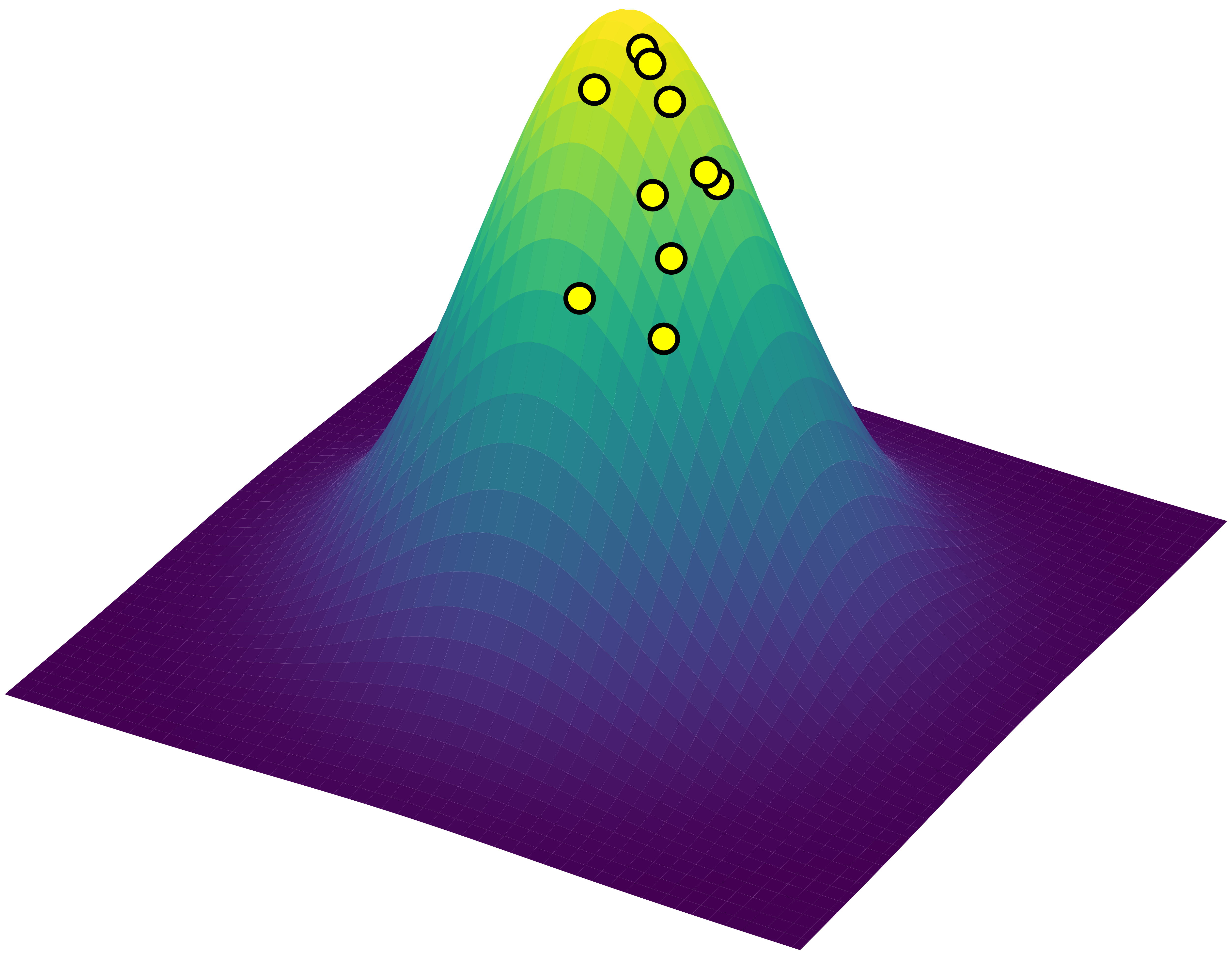
Slow Progress of Stochastic Weight Mutation



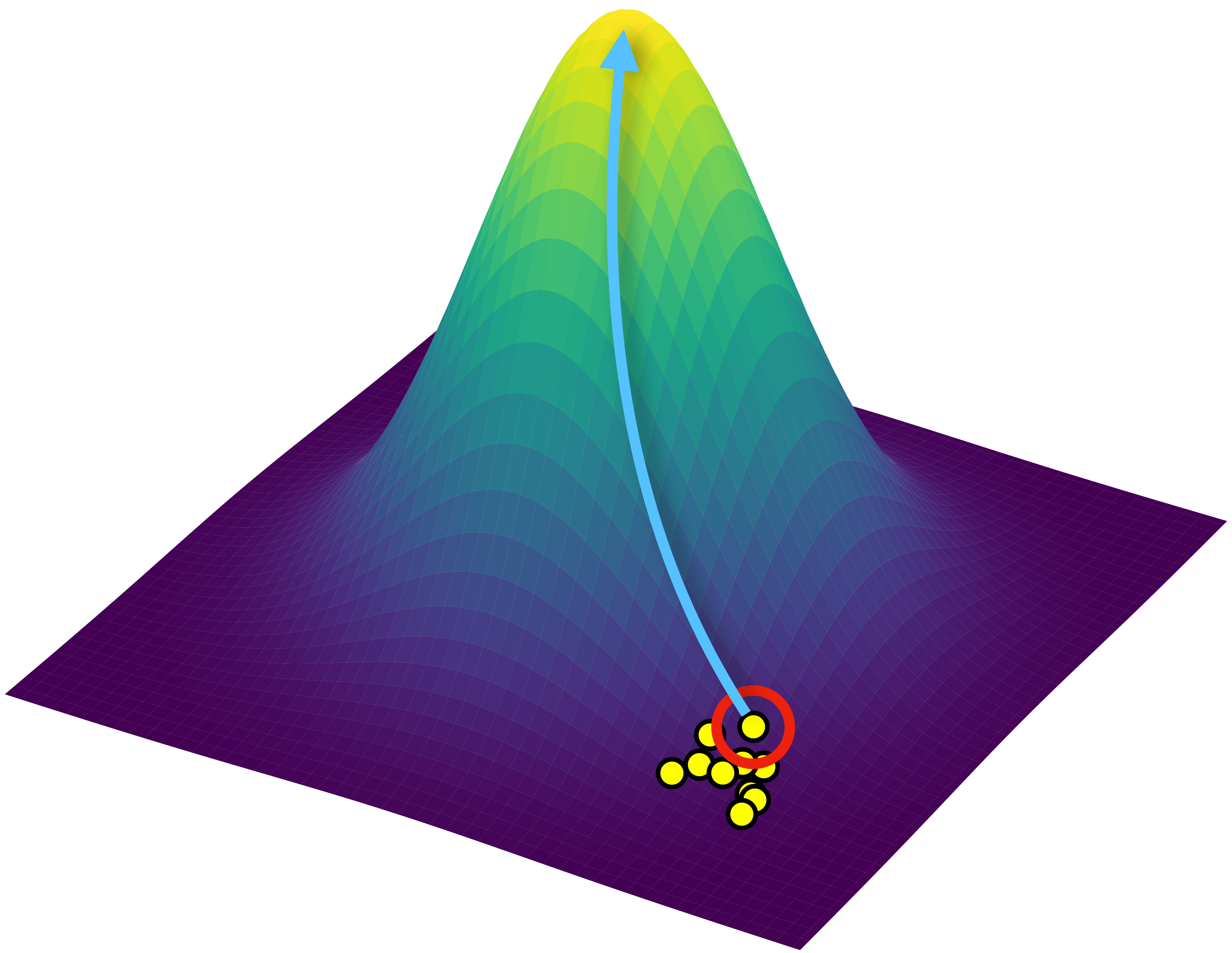
Slow Progress of Stochastic Weight Mutation



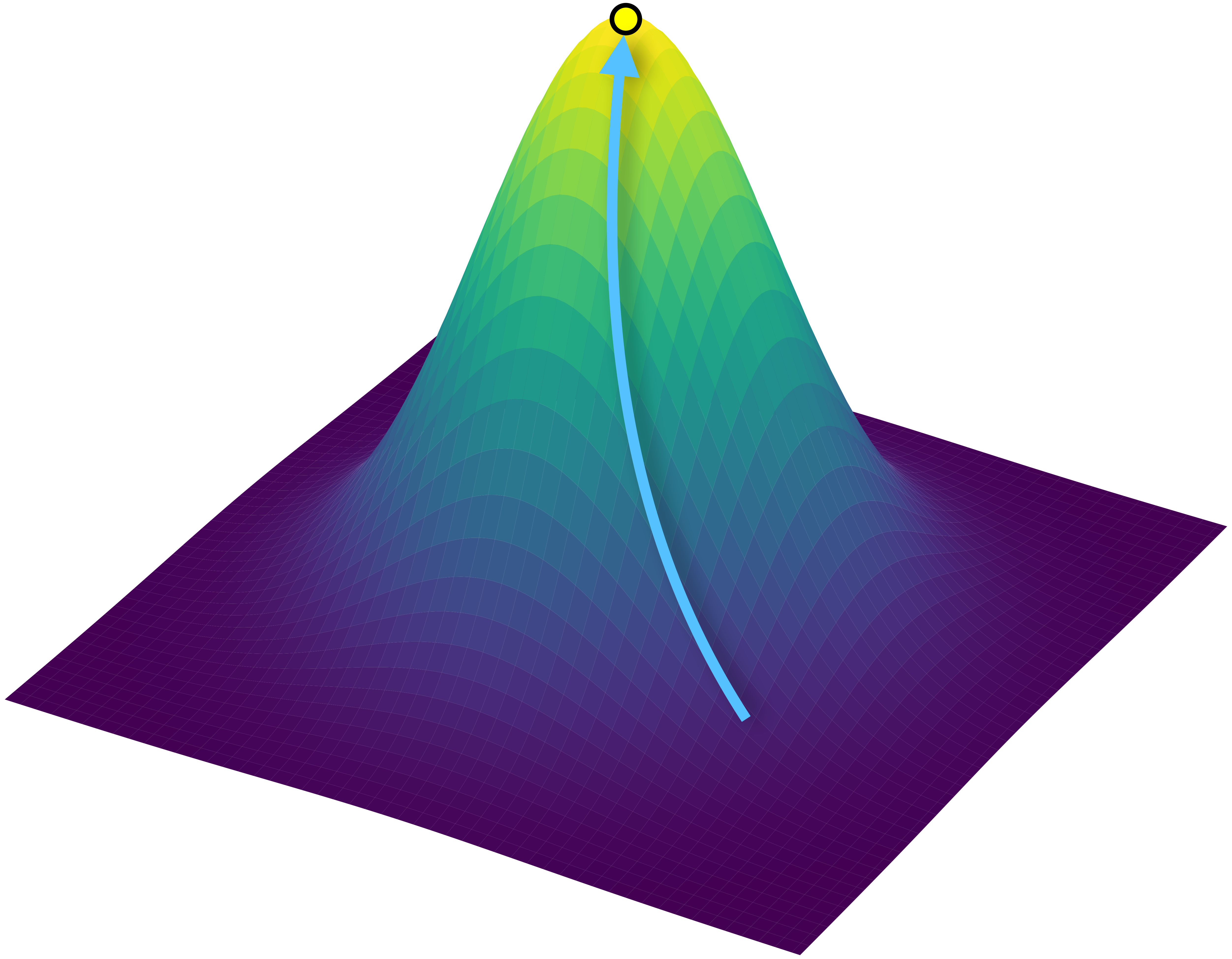
Slow Progress of Stochastic Weight Mutation



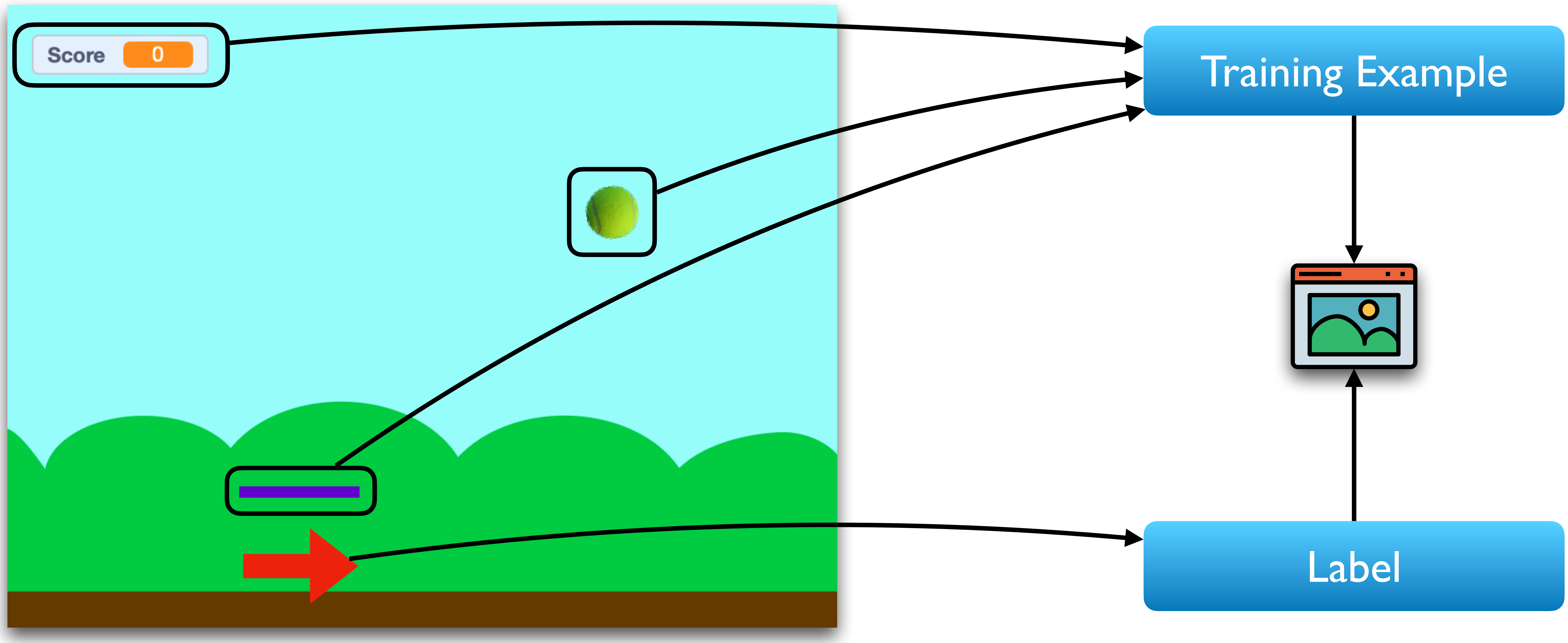
Gradient-Descent as Systematic Optimiser



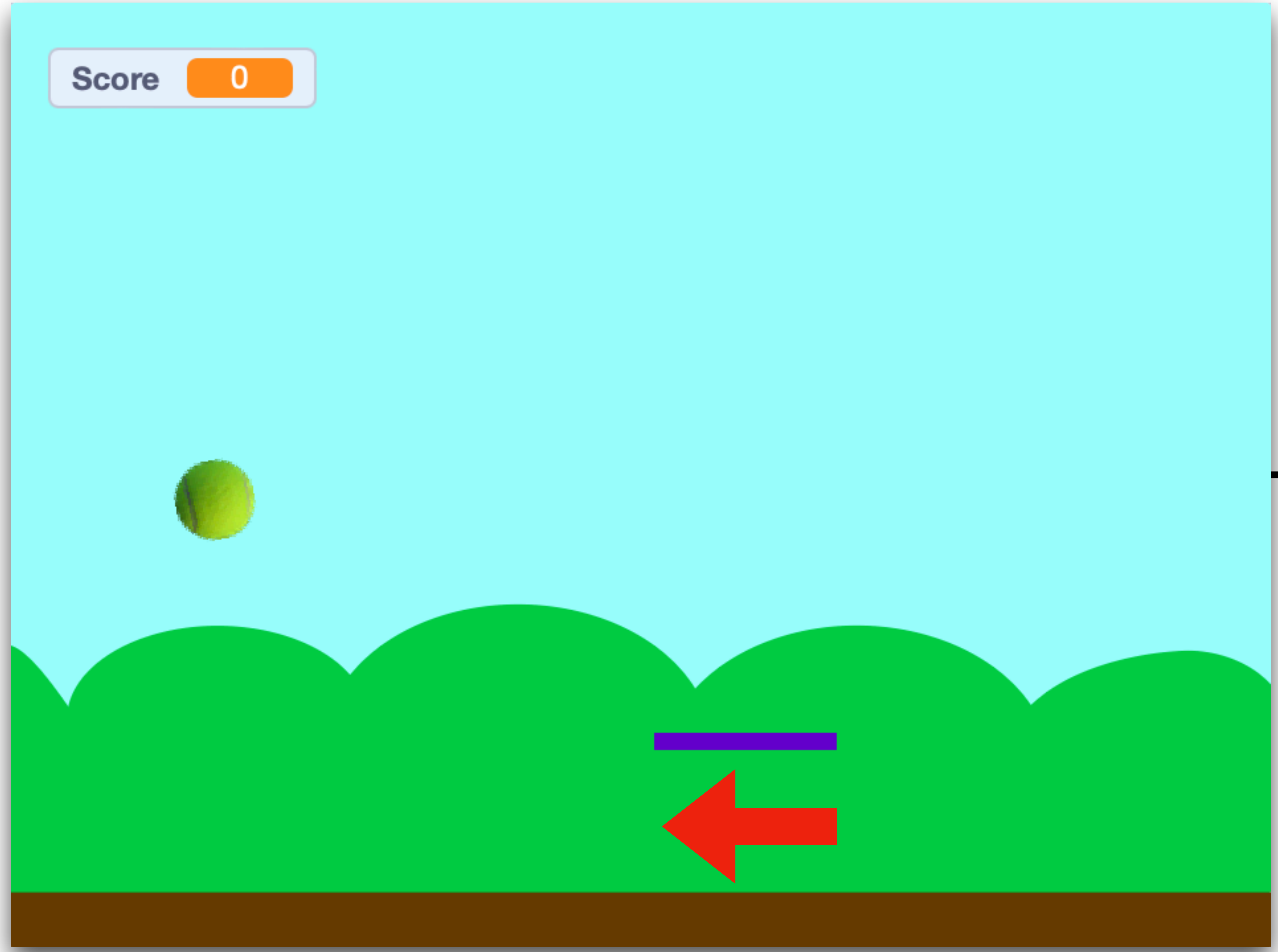
Gradient-Descent as Systematic Optimiser

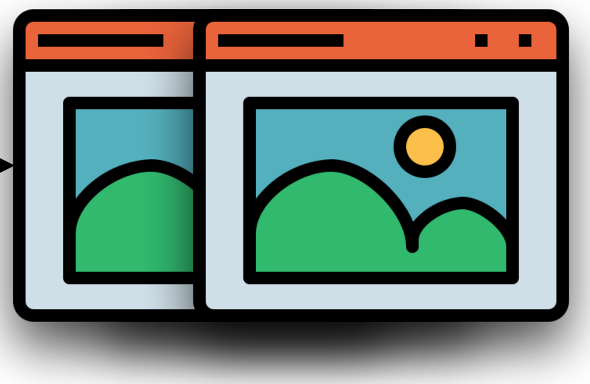


Human Gameplay Traces as Training Set

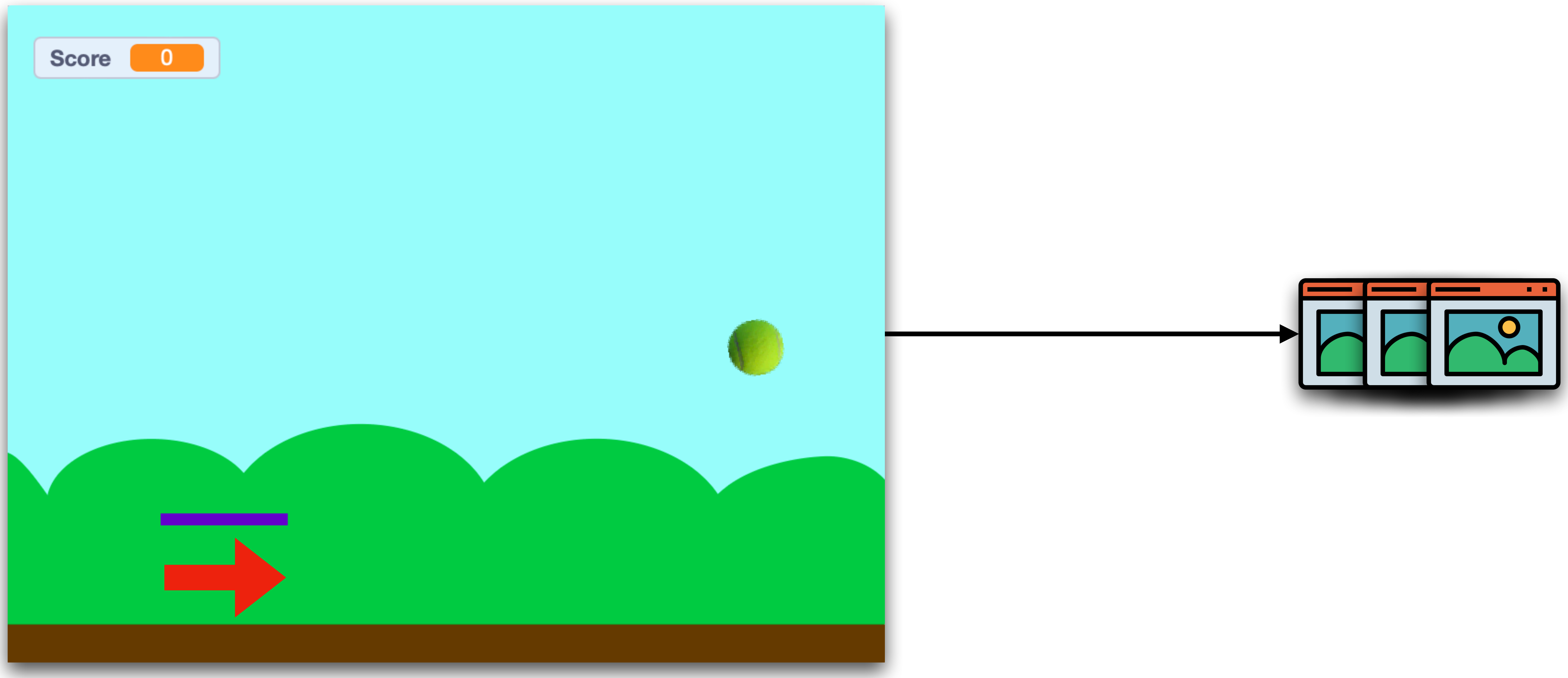


Human Gameplay Traces as Training Set

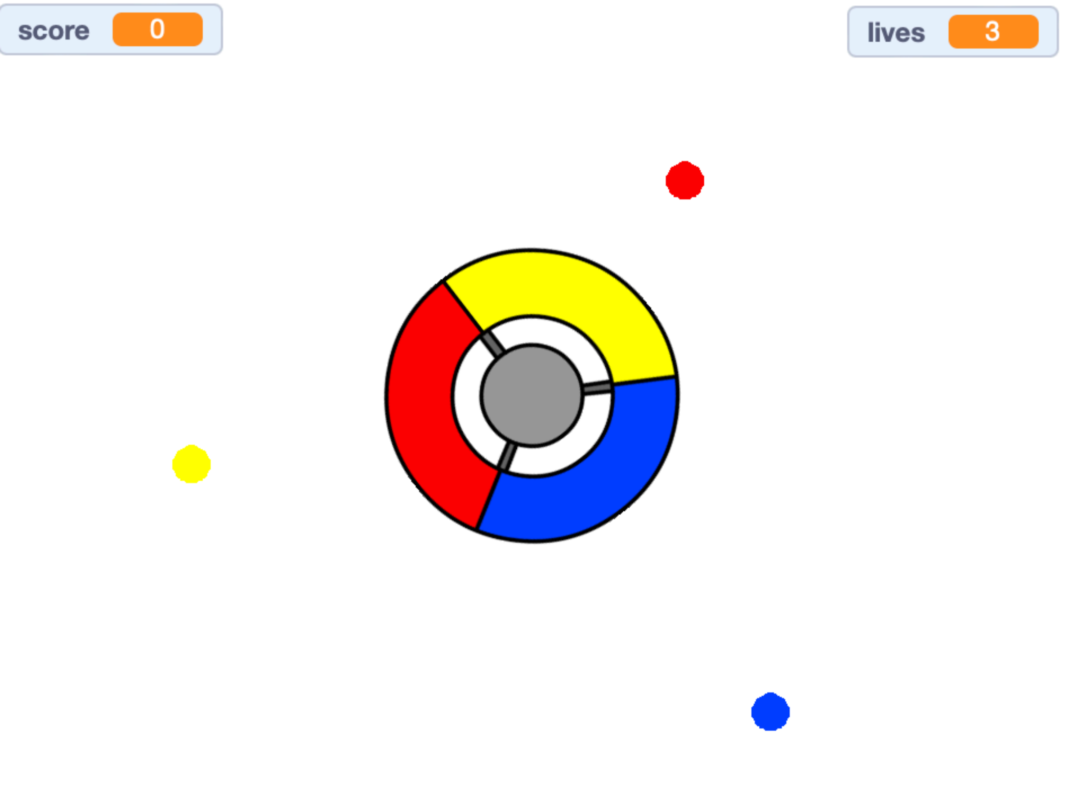


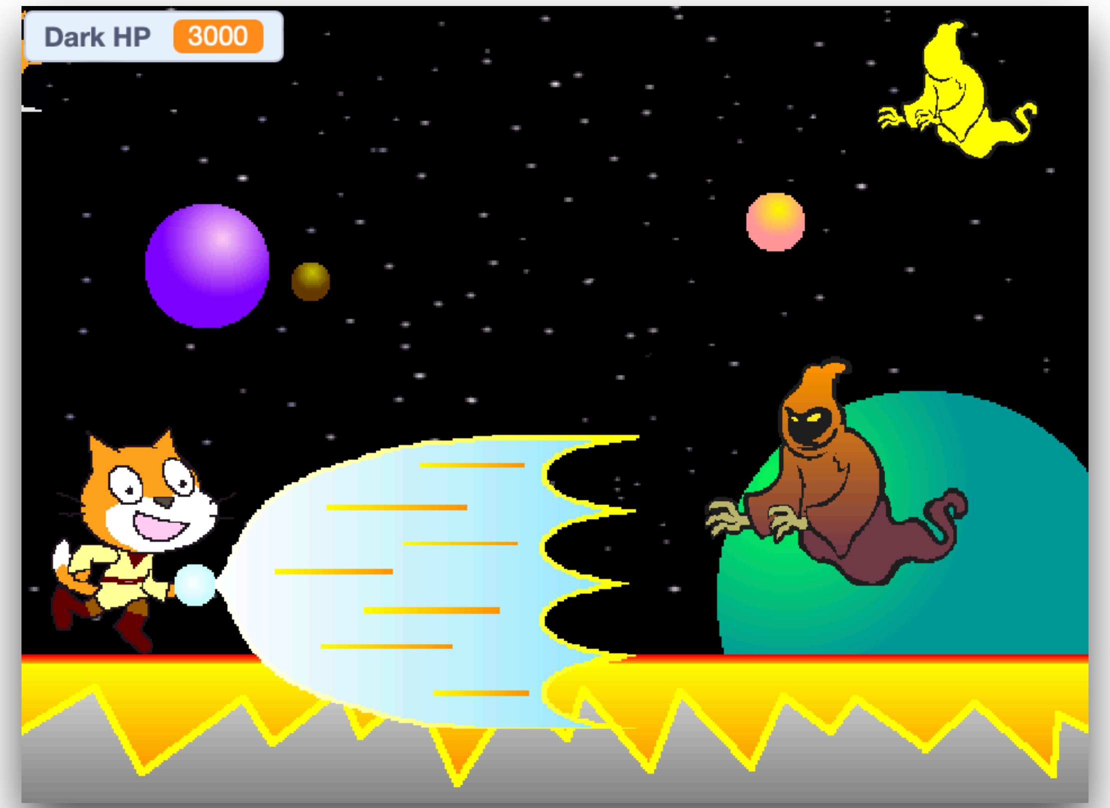


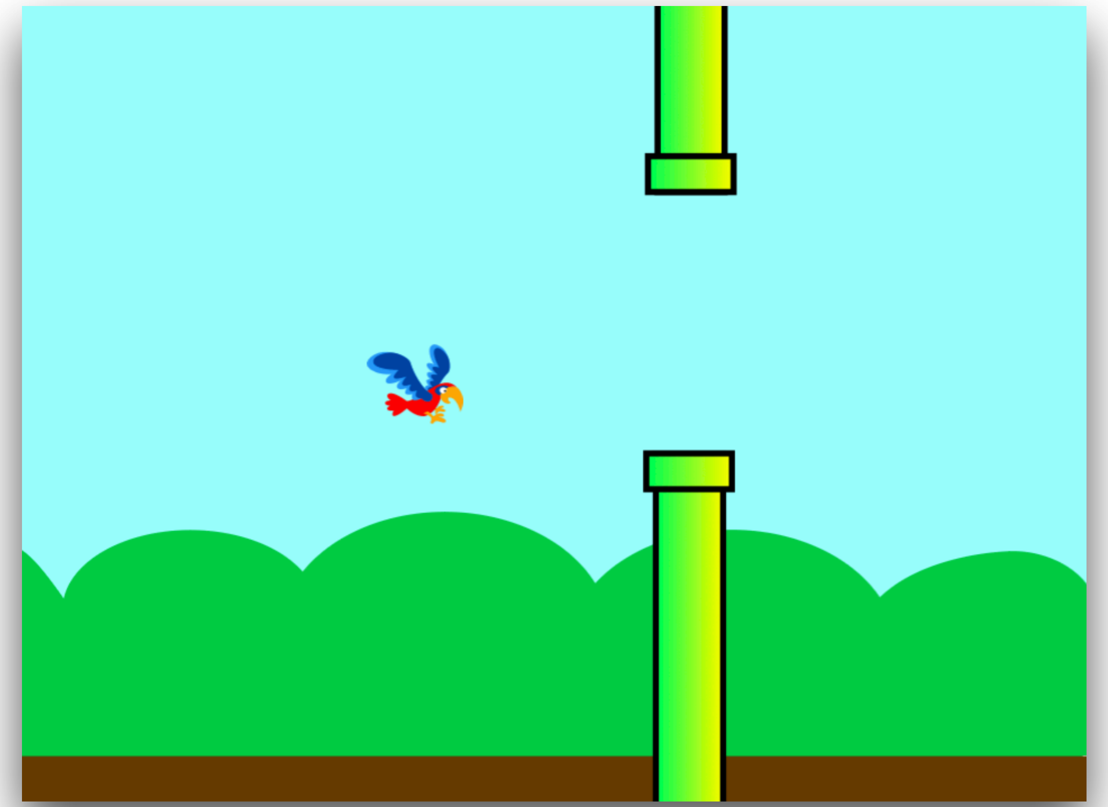
Human Gameplay Traces as Training Set

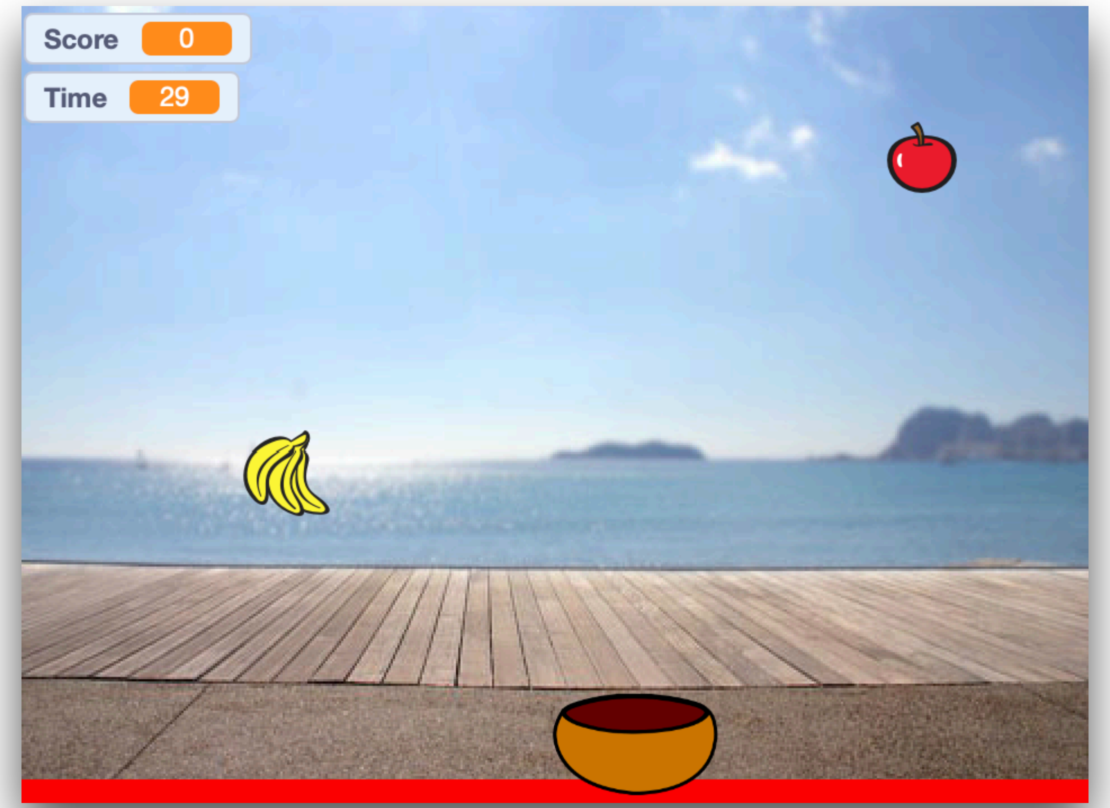


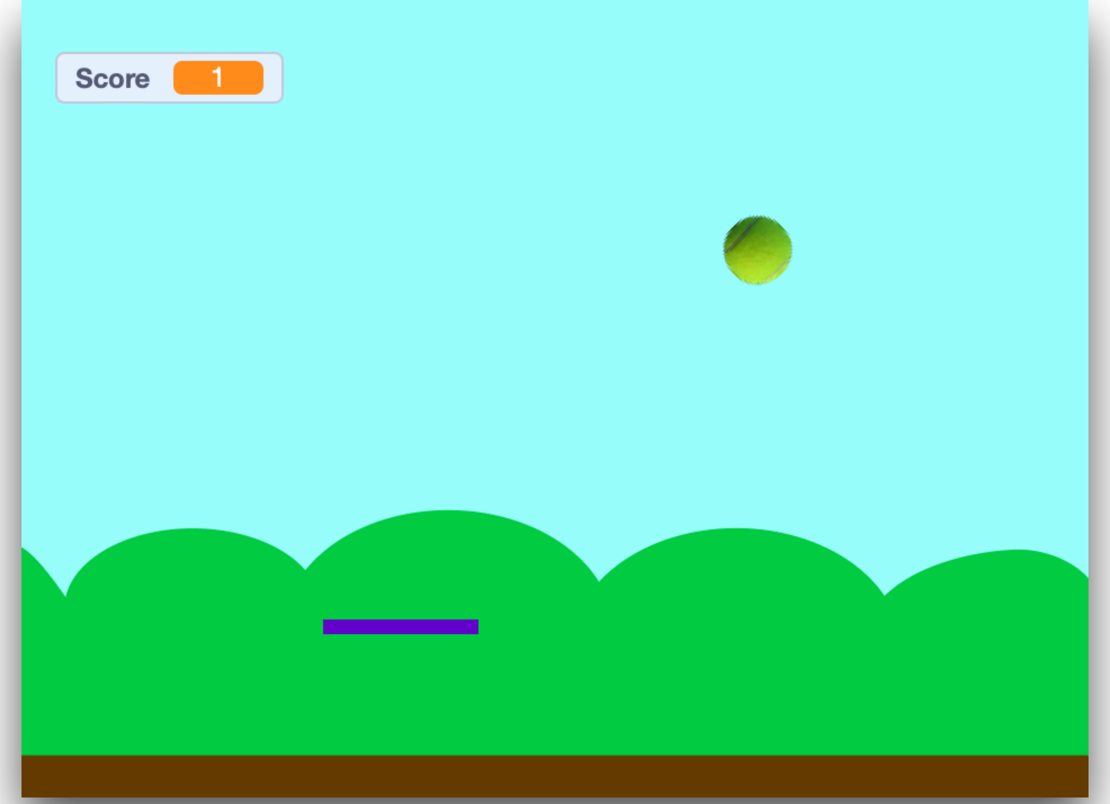
Evaluation Dataset of 8 Scratch Games

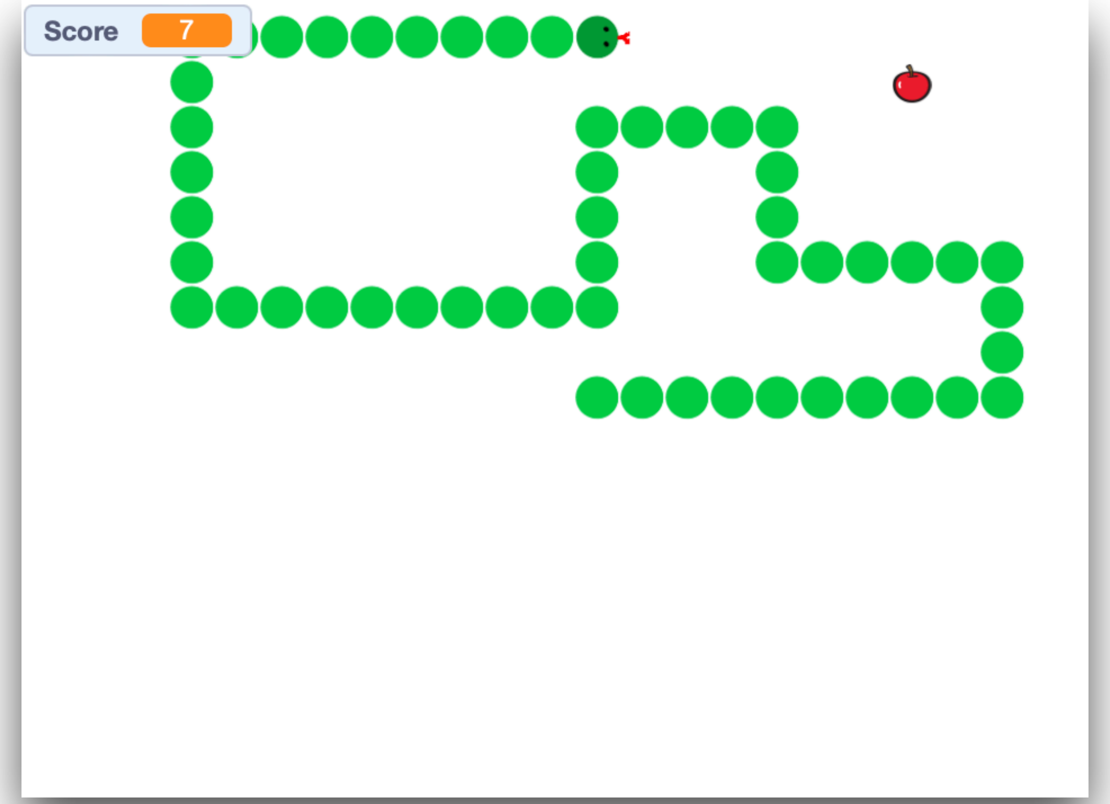


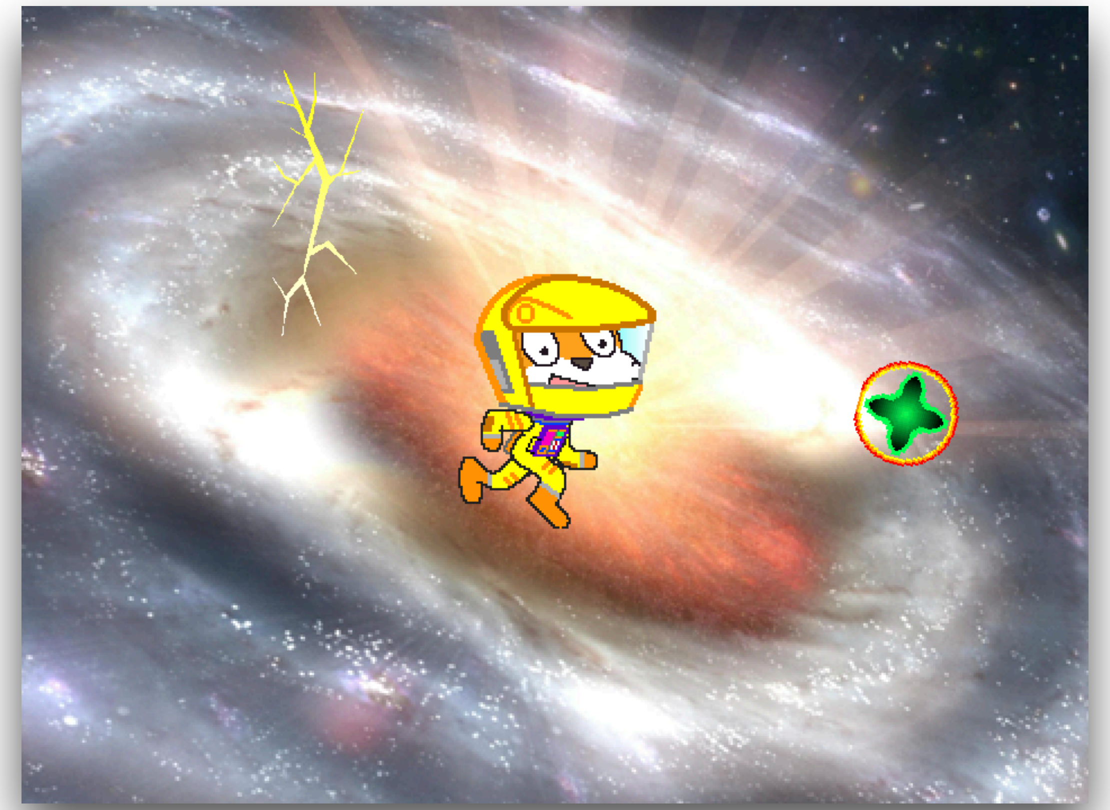




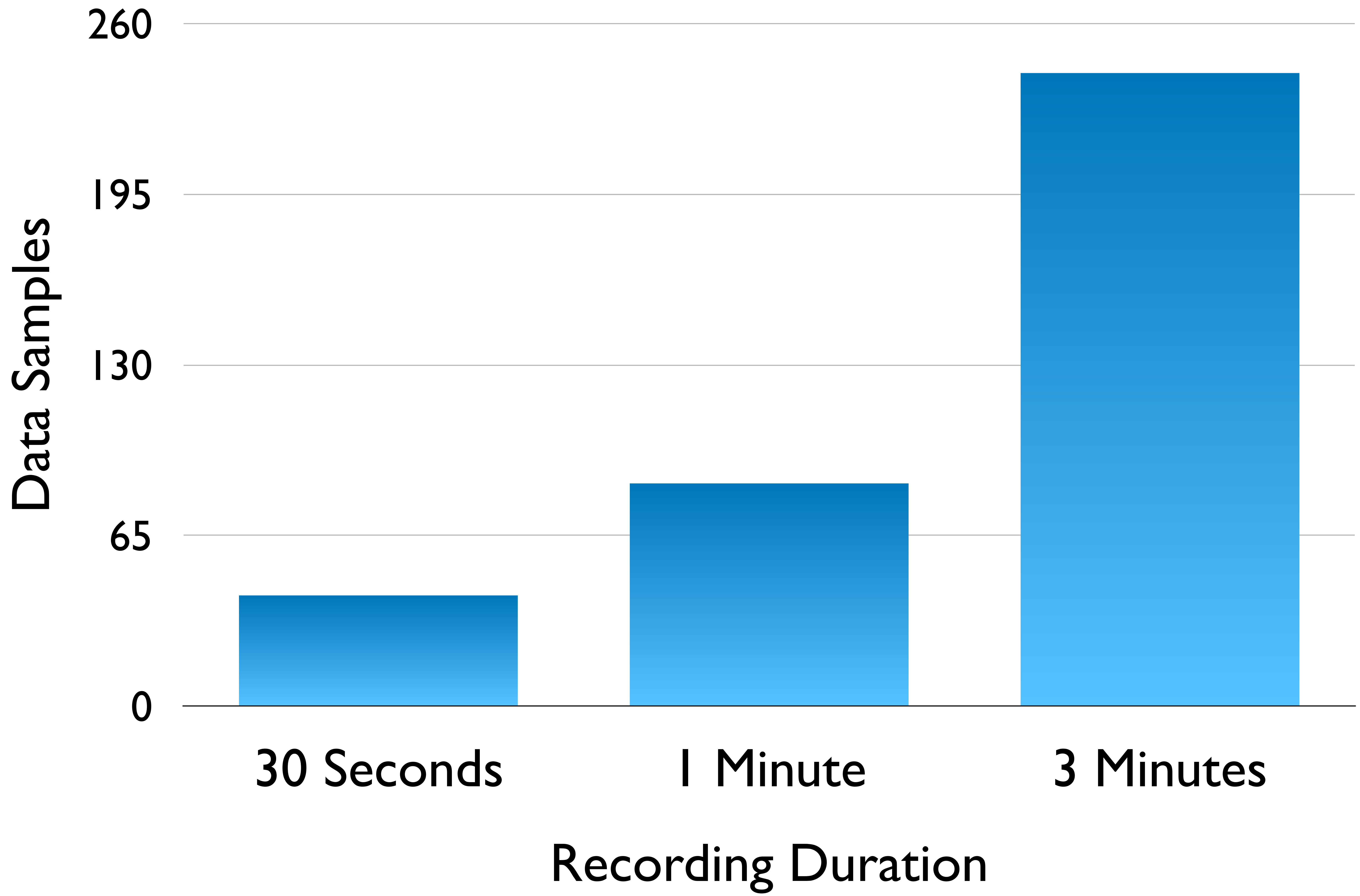




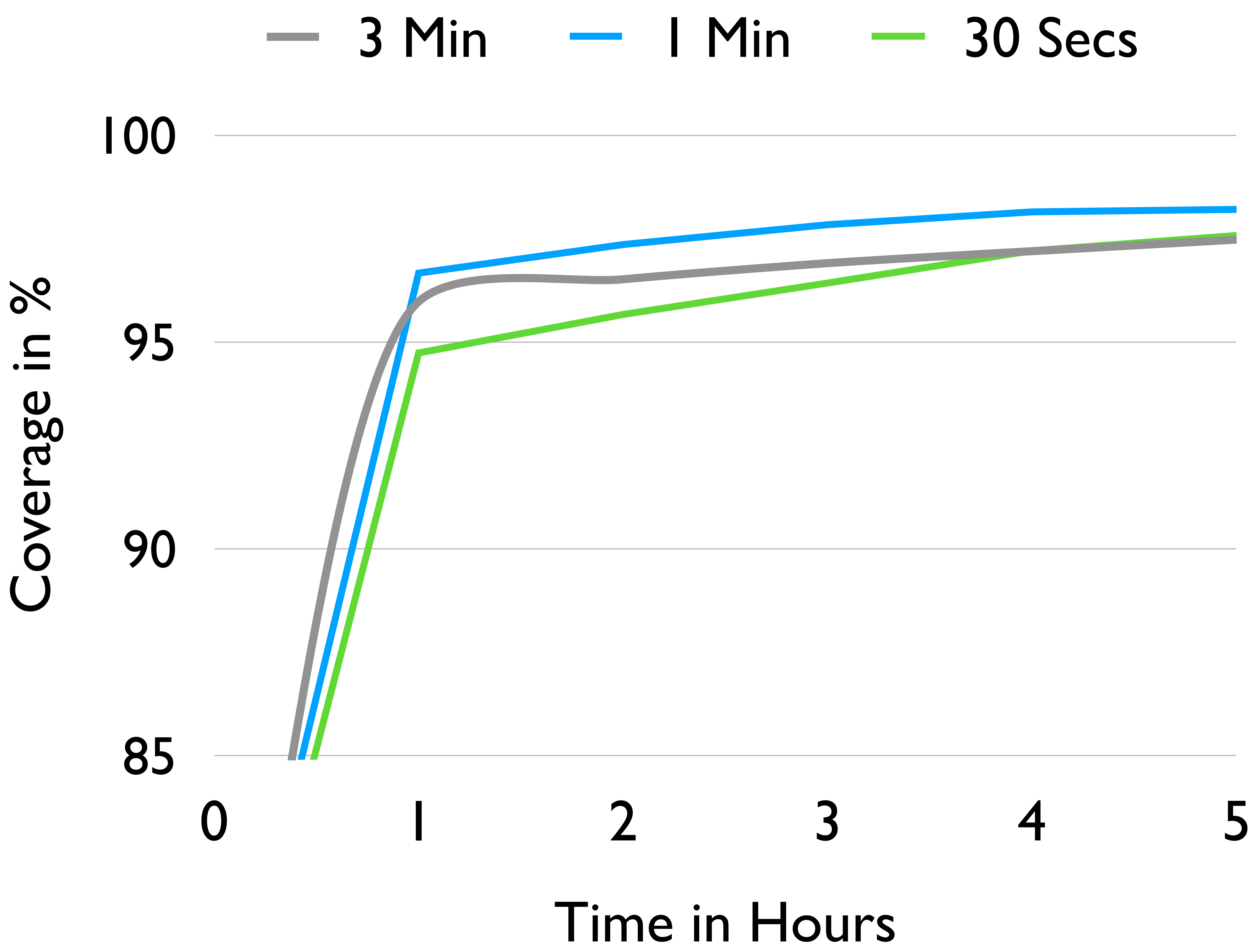




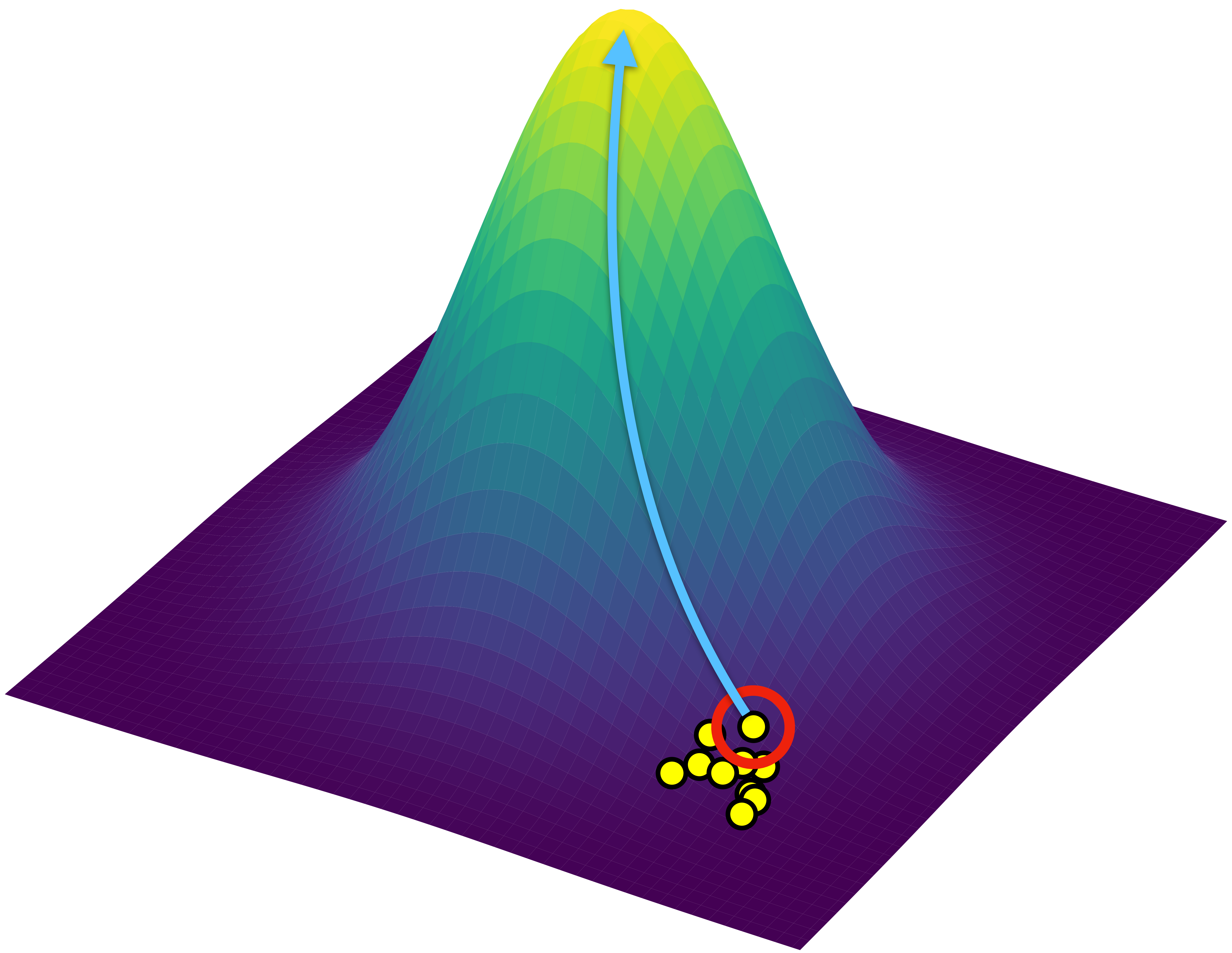
How Many Data Samples Are Required?



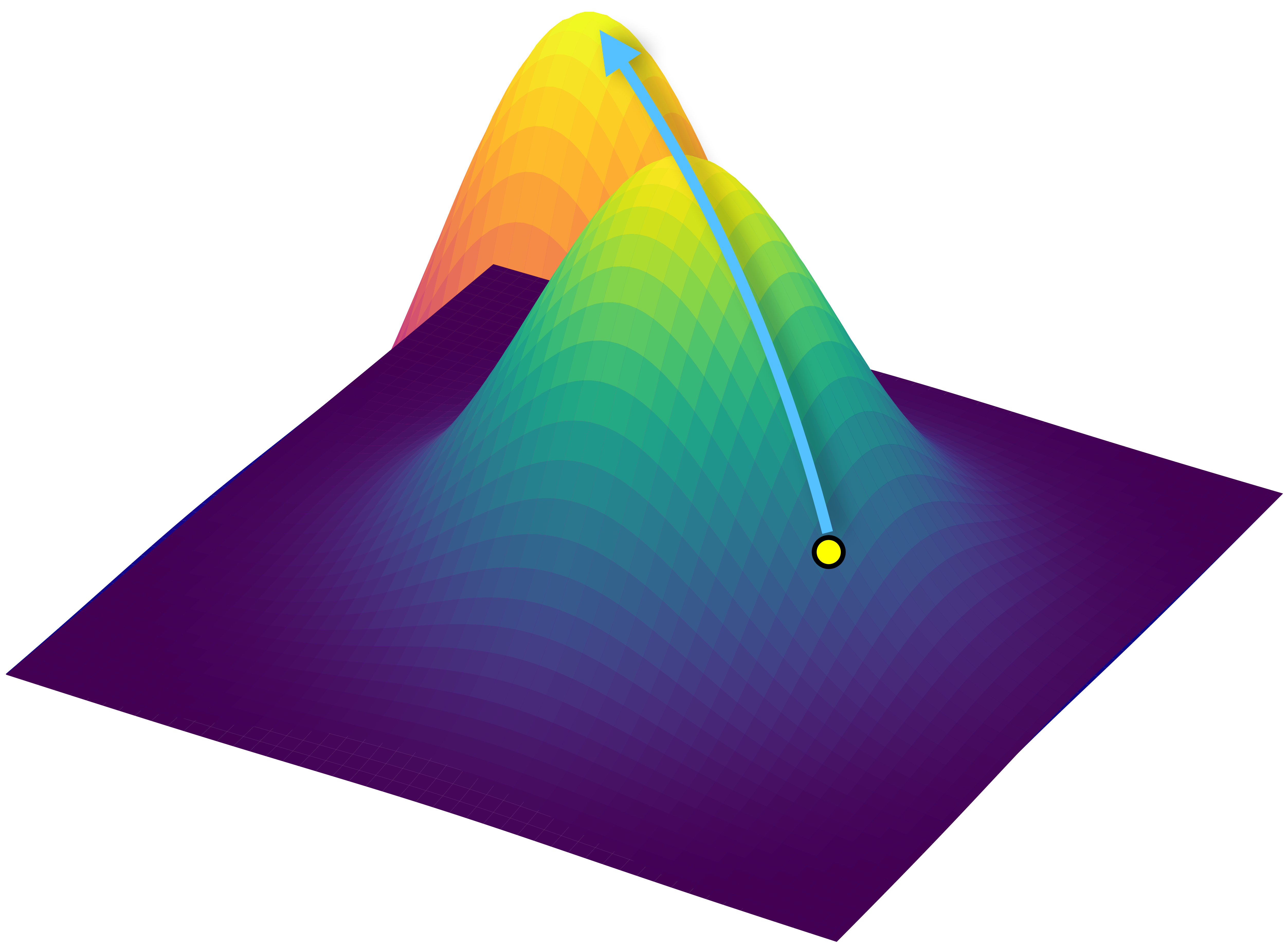
Too Many Data Samples Impair the Search



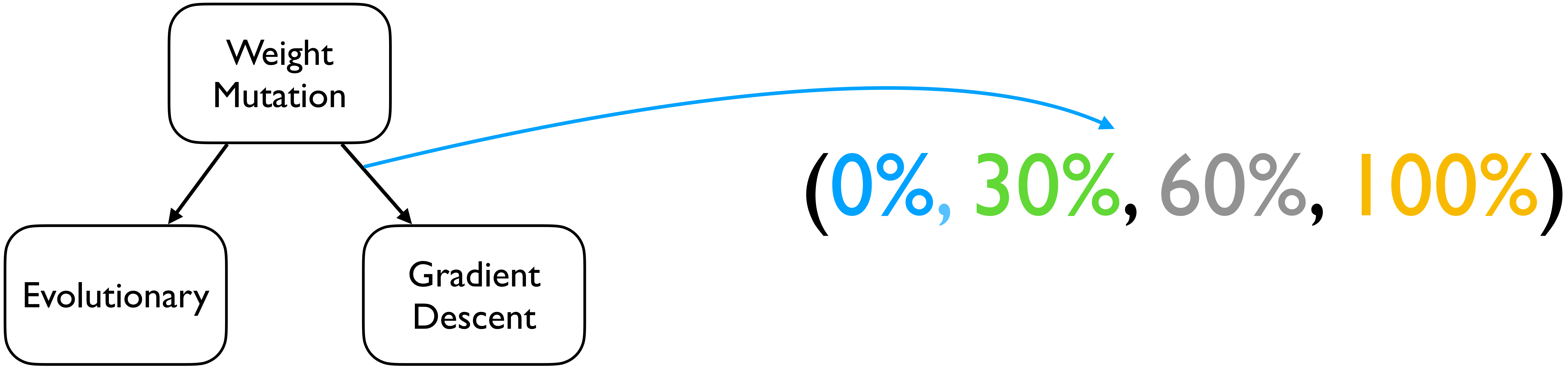
Human Traces Can Only Approximate Optimisation Goal



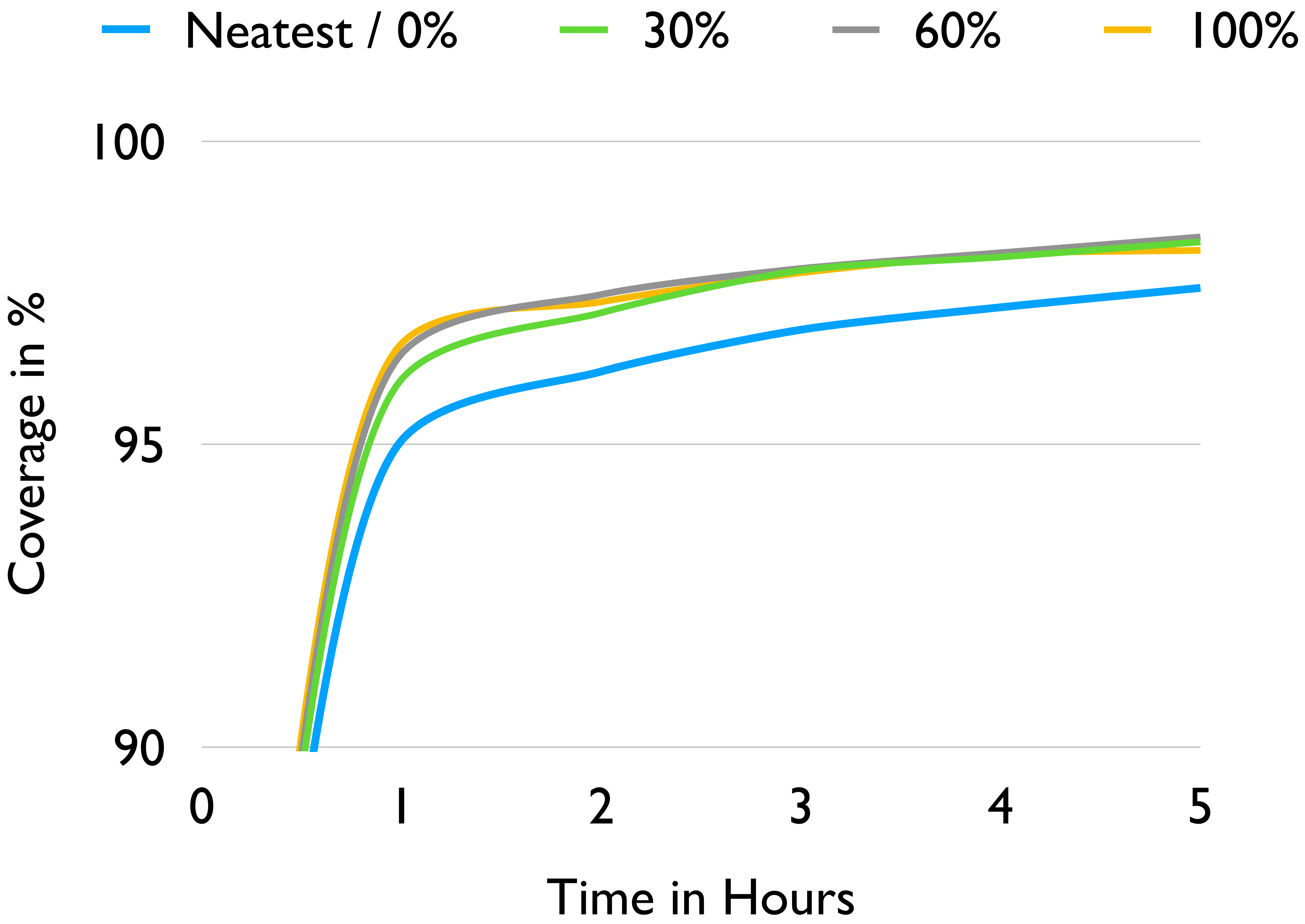
Human Traces Can Only Approximate Optimisation Goal



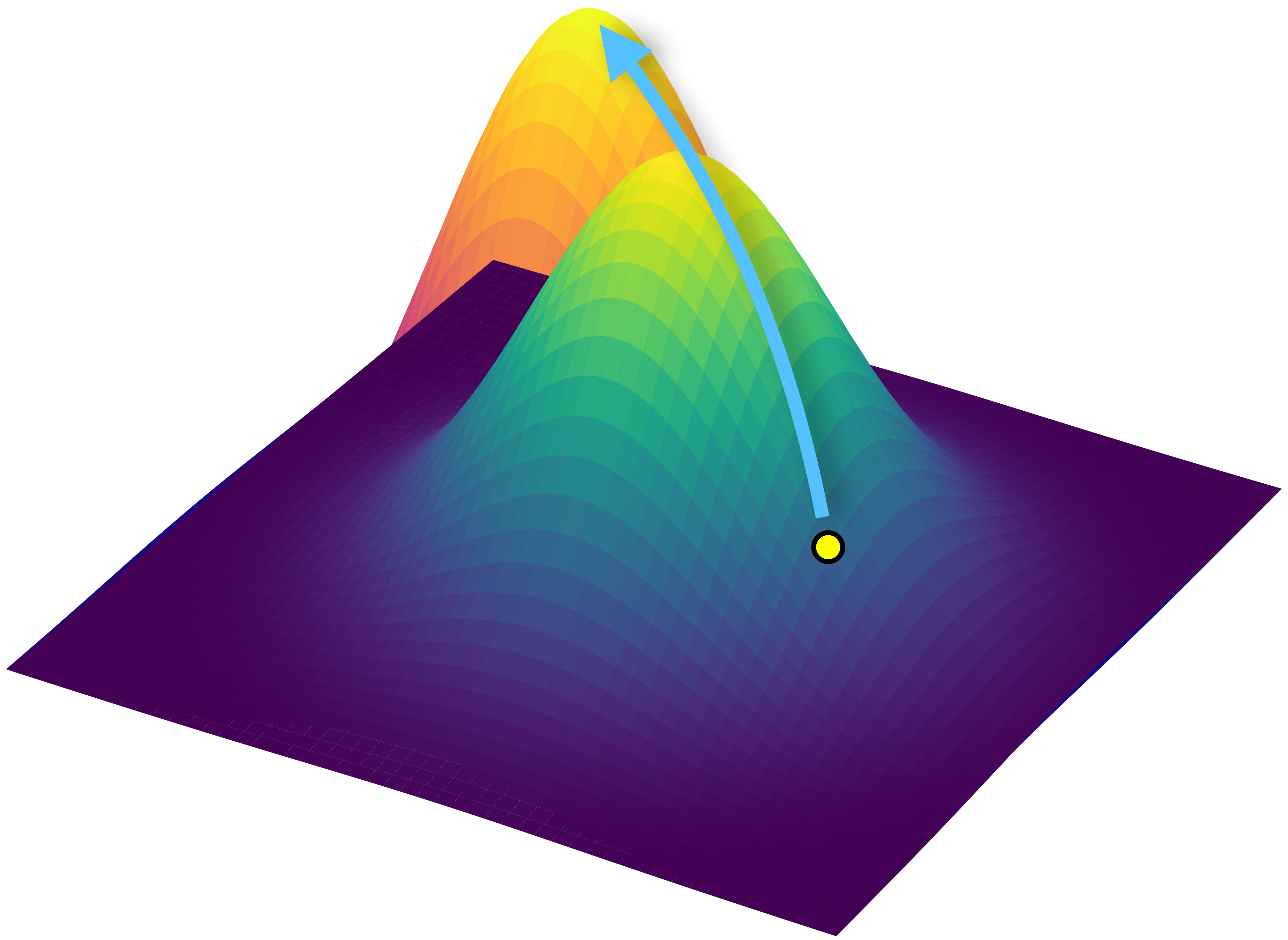
Does Neuroevolution Benefit From Gradient-Descent?

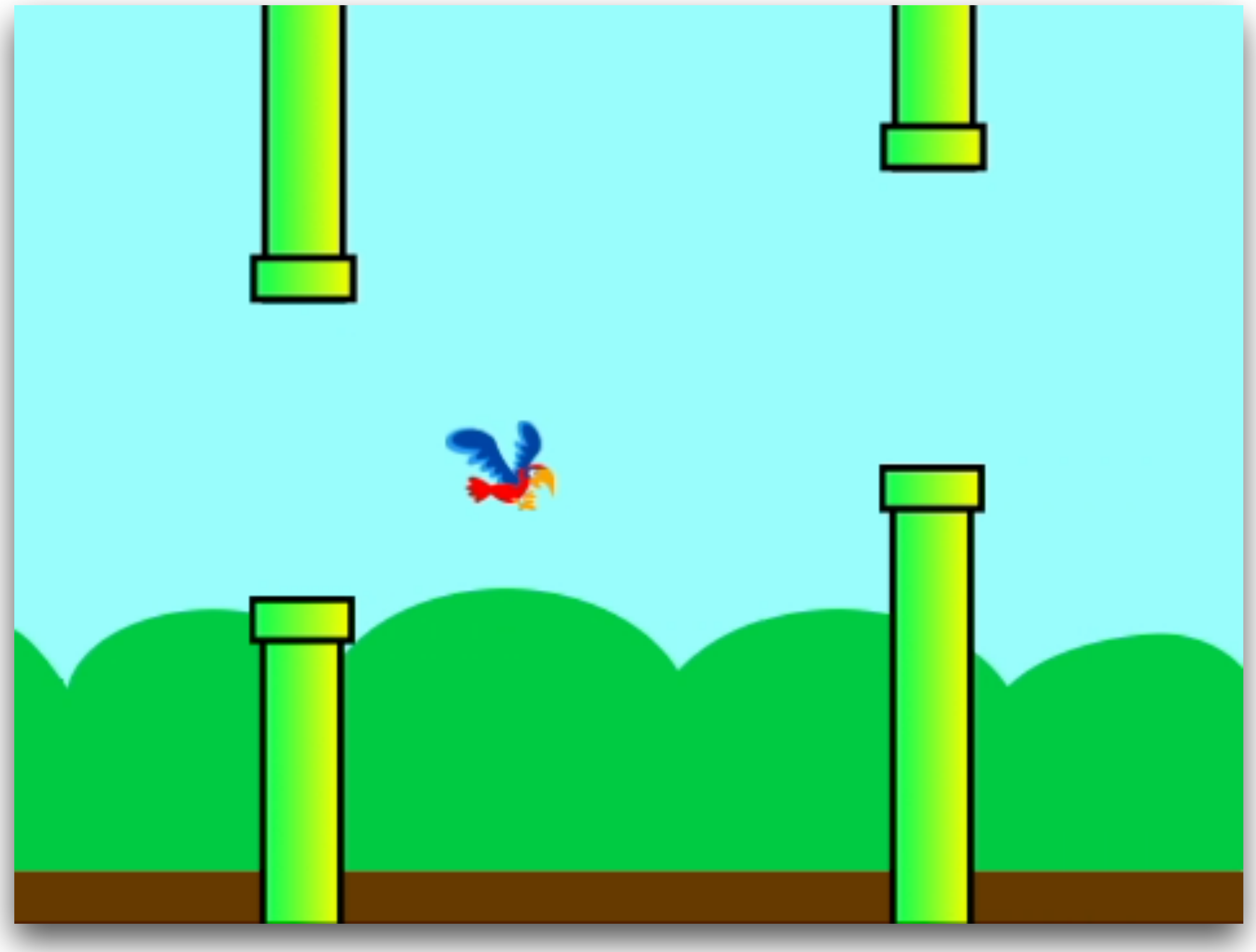


Significant Speedups through Gradient-Descent

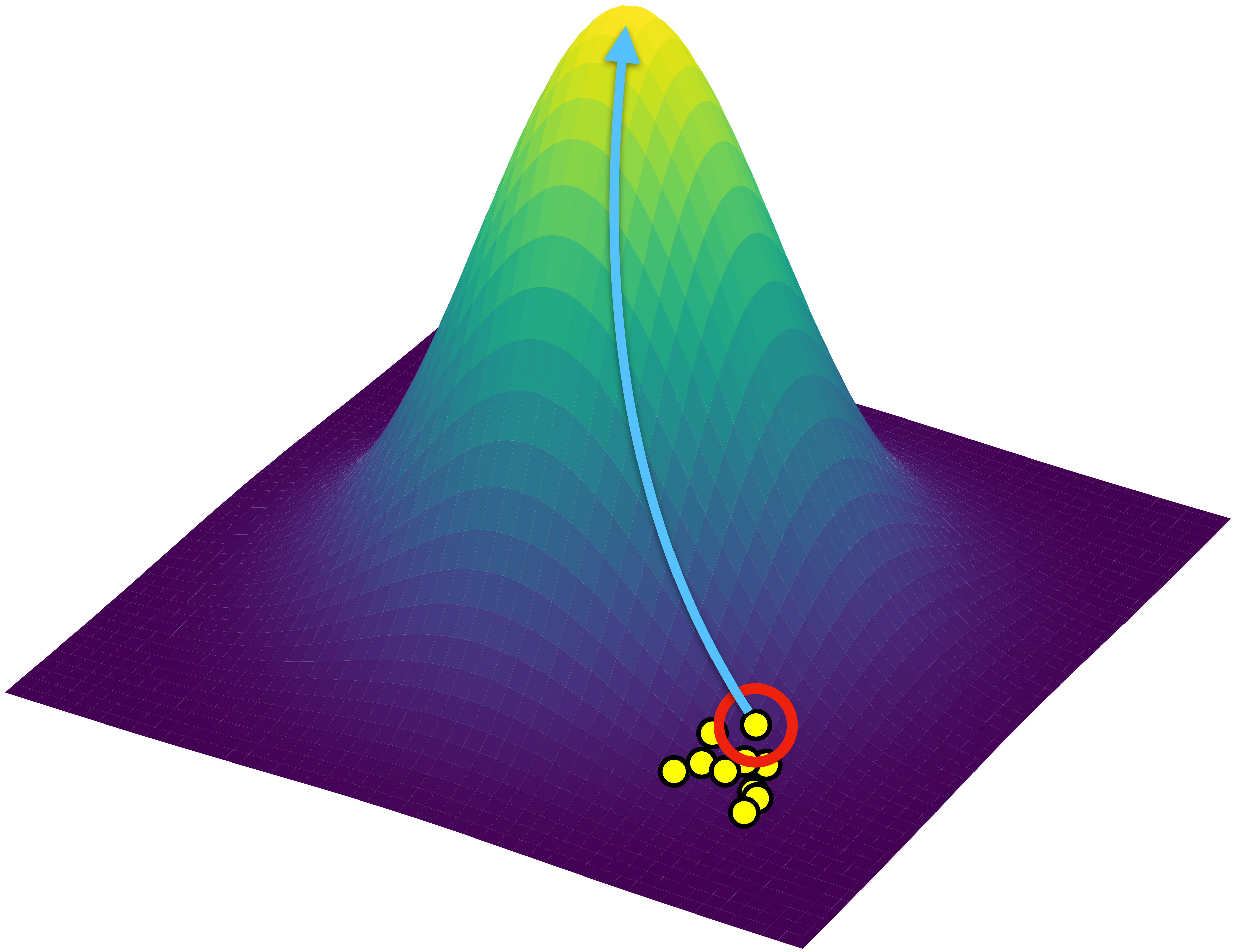


Gradient-Descent Introduces Human Bias





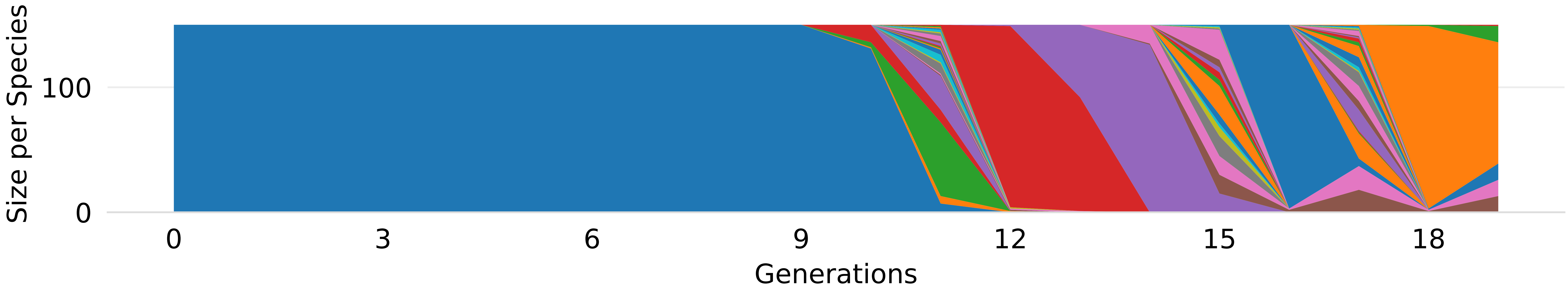
Does Gradient Descent Affect Speciation?



$$\delta = \frac{c_1 D}{N} + \frac{c_2 E}{N} + \boxed{c_3 \overline{W}}$$

Repeated Explosion in Number of Species

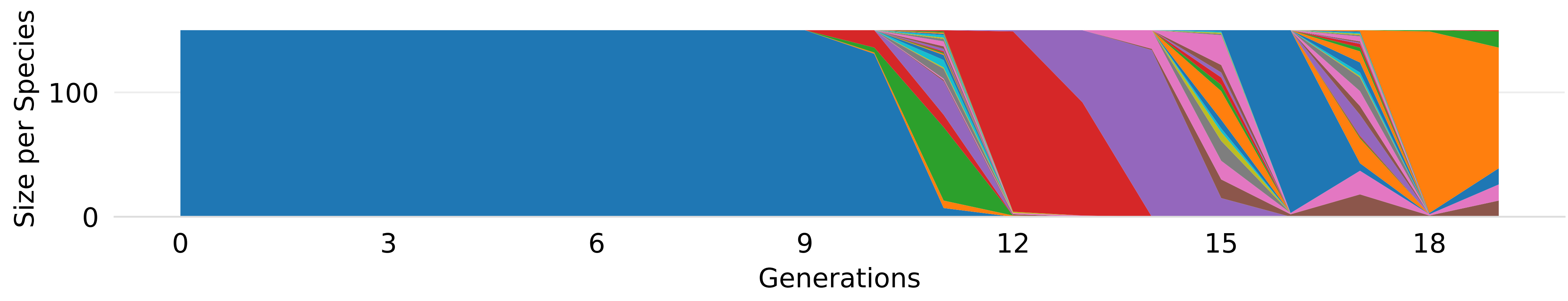
$$\delta_t = 3 \qquad c_3 = 0.5 \quad \rightarrow \quad 94.82\,\% \;\;/\;\; 5$$



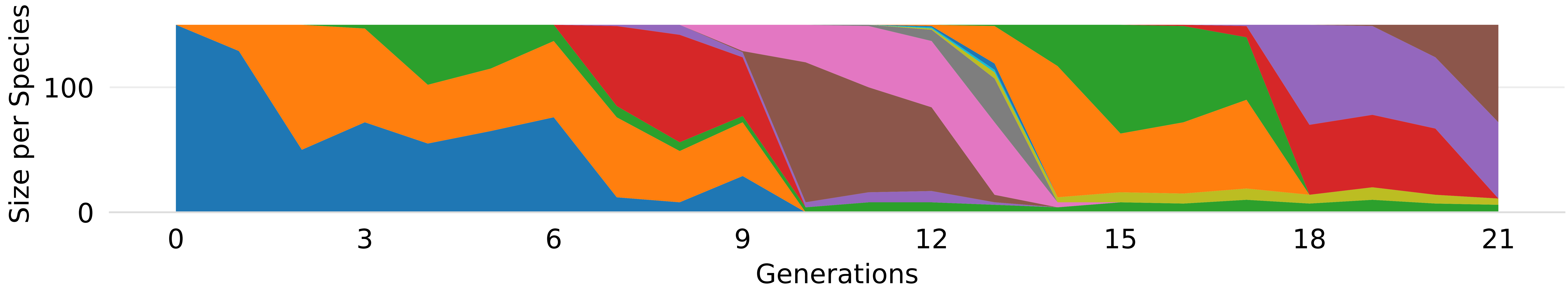
$$\delta = \frac{c_1 D}{N} + \frac{c_2 E}{N} + c_3 \overline{W}$$

Compatibility Threshold Affects Speciation

$\delta_t = 3 \qquad c_3 = 0.5 \quad \rightarrow \quad 94.82\,\% \ / \ 5$



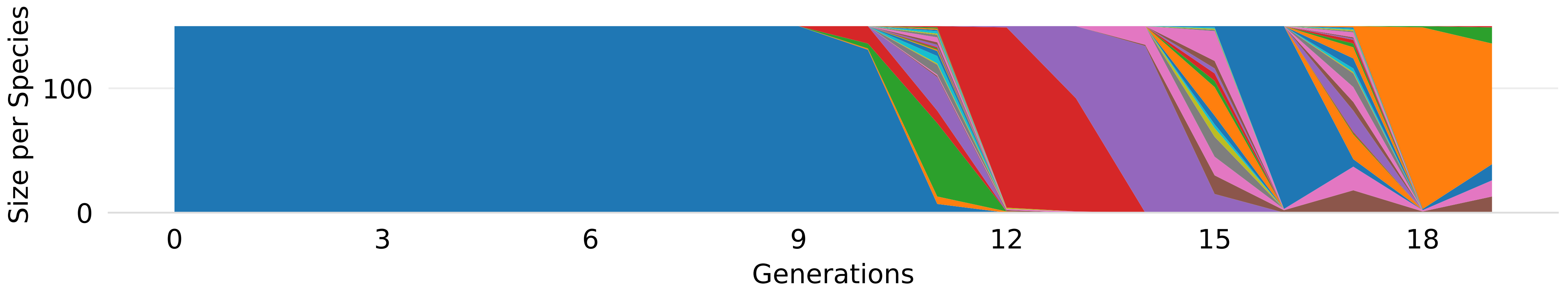
$\delta_t = 2 \qquad c_3 = 0.5 \quad \rightarrow \quad 95.61\,\% \ / \ 8$



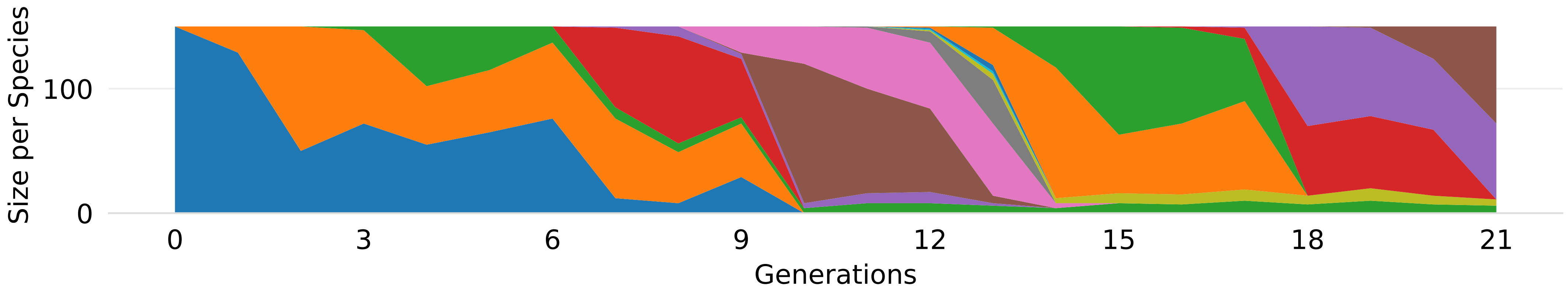
$$\delta = \frac{c_1 D}{N} + \frac{c_2 E}{N} + c_3 \overline{W}$$

Gradient Descent Affects Speciation

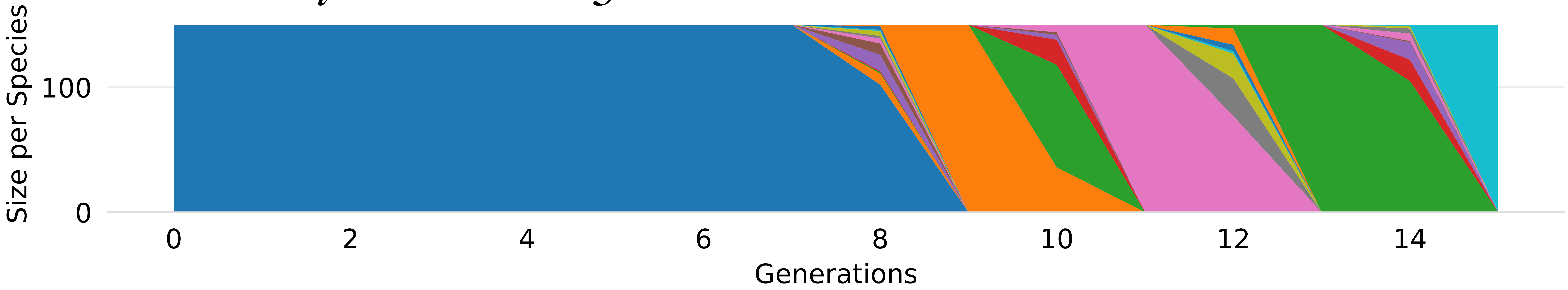
$\delta_t = 3 \qquad c_3 = 0.5 \quad \rightarrow \quad 94.82\,\% \;\;/\;\; 5$



$\delta_t = 2 \qquad c_3 = 0.5 \quad \rightarrow \quad 95.61\,\% \;\;/\;\; 8$



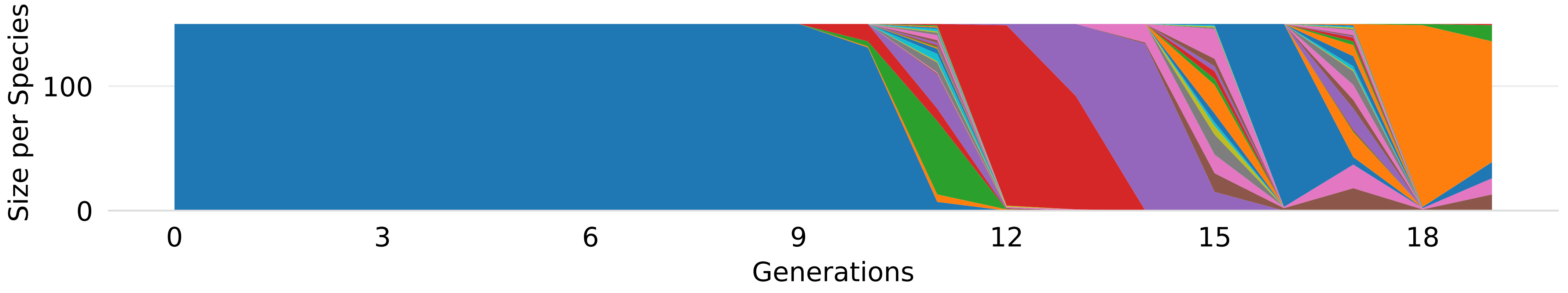
$\delta_t = 2 \qquad c_3 = 0 \quad \rightarrow \quad 95.35\,\% \;\;/\;\; 5$



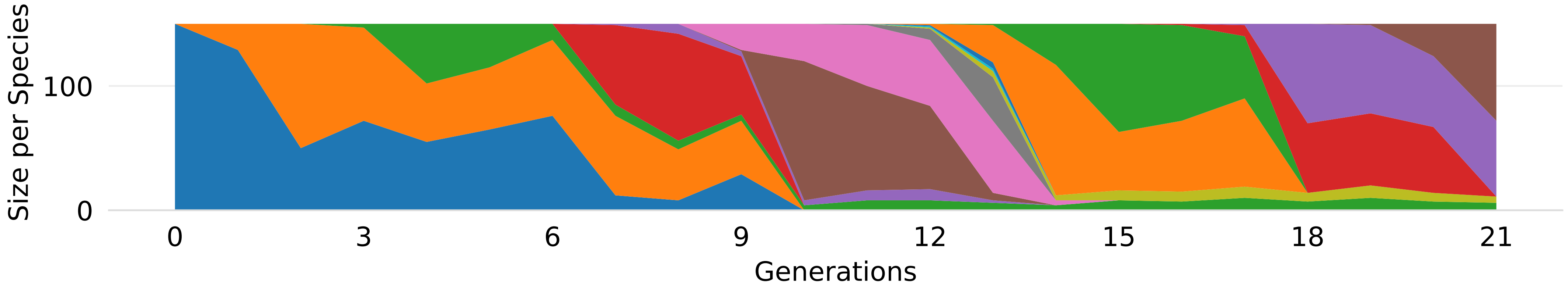
$$\delta = \frac{c_1 D}{N} + \frac{c_2 E}{N} + c_3 \overline{W}$$

Gradient Descent Affects Speciation

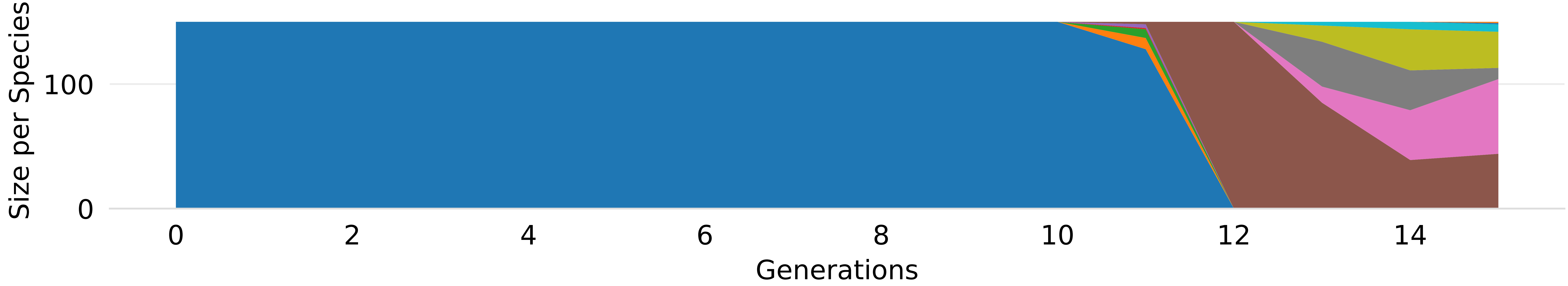
$\delta_t = 3 \qquad c_3 = 0.5 \quad \rightarrow \quad 94.82\,\% \;\;/\;\; 5$



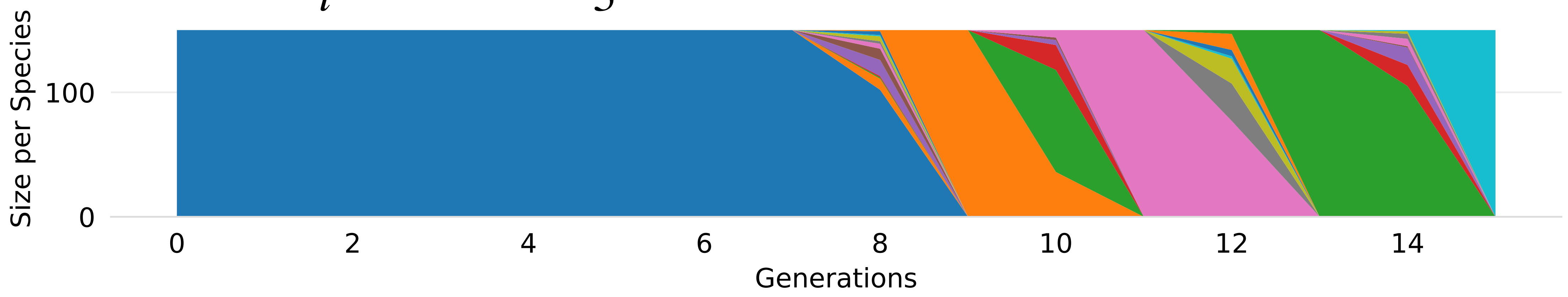
$\delta_t = 2 \qquad c_3 = 0.5 \quad \rightarrow \quad 95.61\,\% \;\;/\;\; 8$



$\delta_t = 3 \qquad c_3 = 0 \quad \rightarrow \quad 95.18\,\% \;\;/\;\; 3$

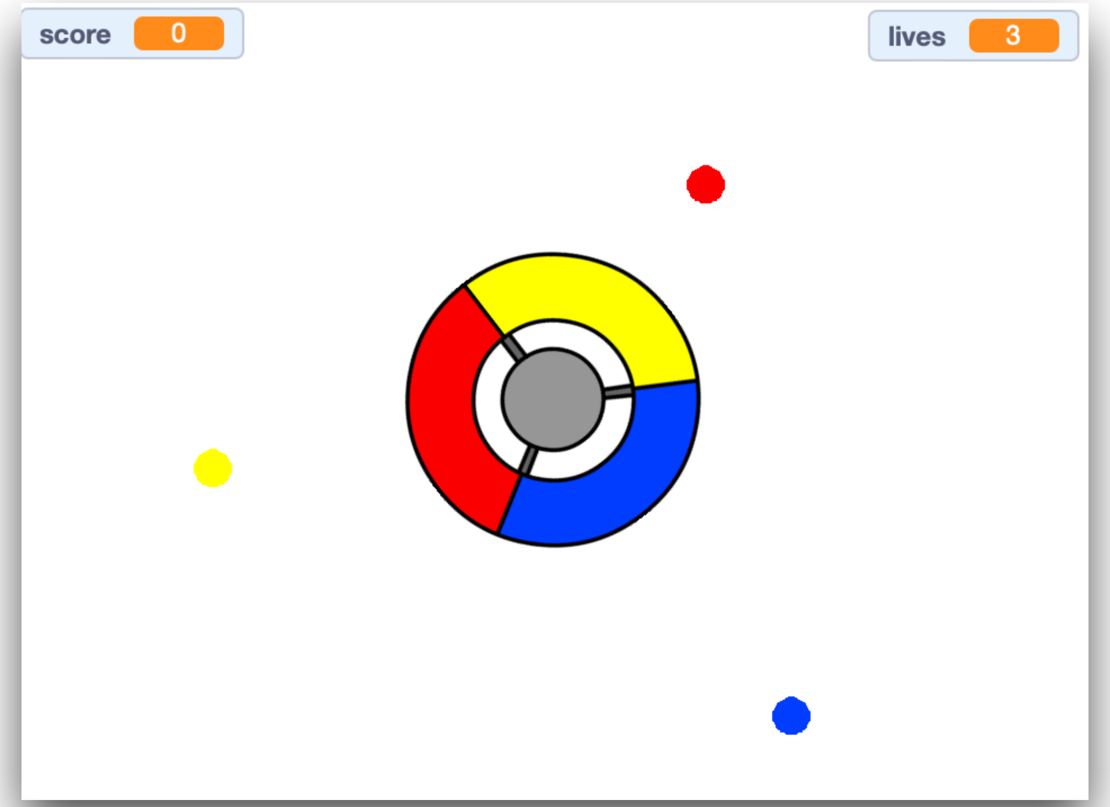


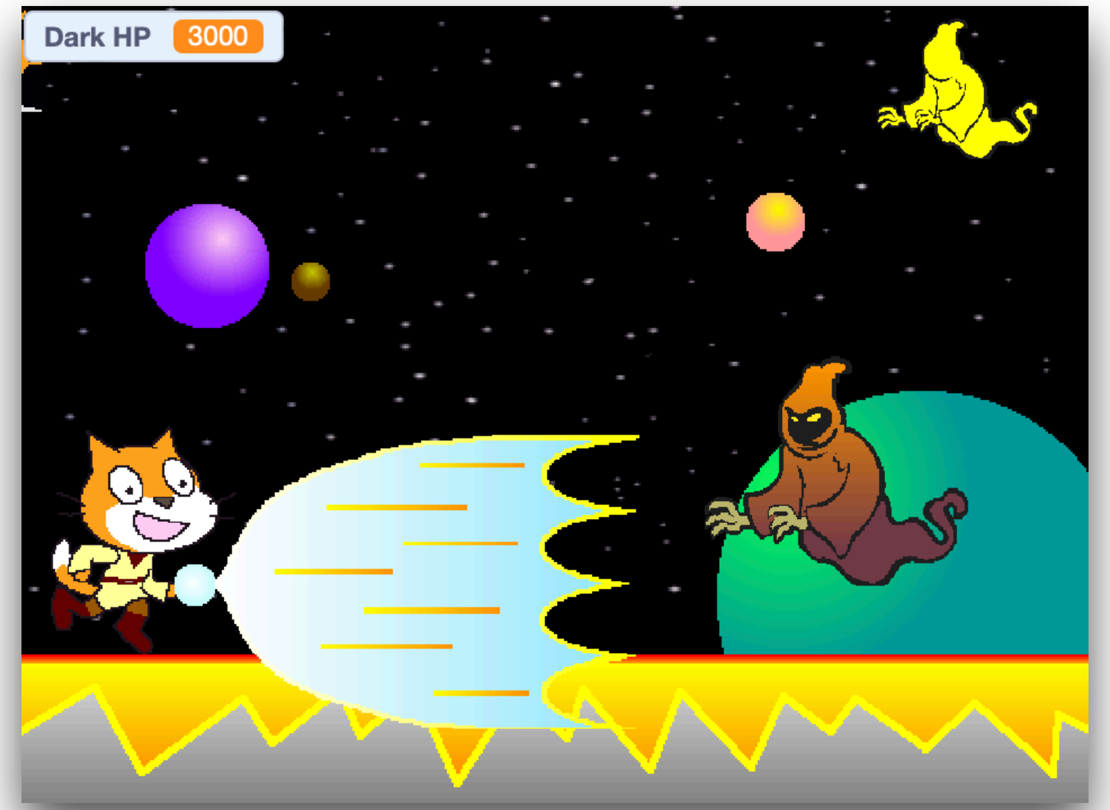
$\delta_t = 2 \qquad c_3 = 0 \quad \rightarrow \quad 95.35\,\% \;\;/\;\; 5$

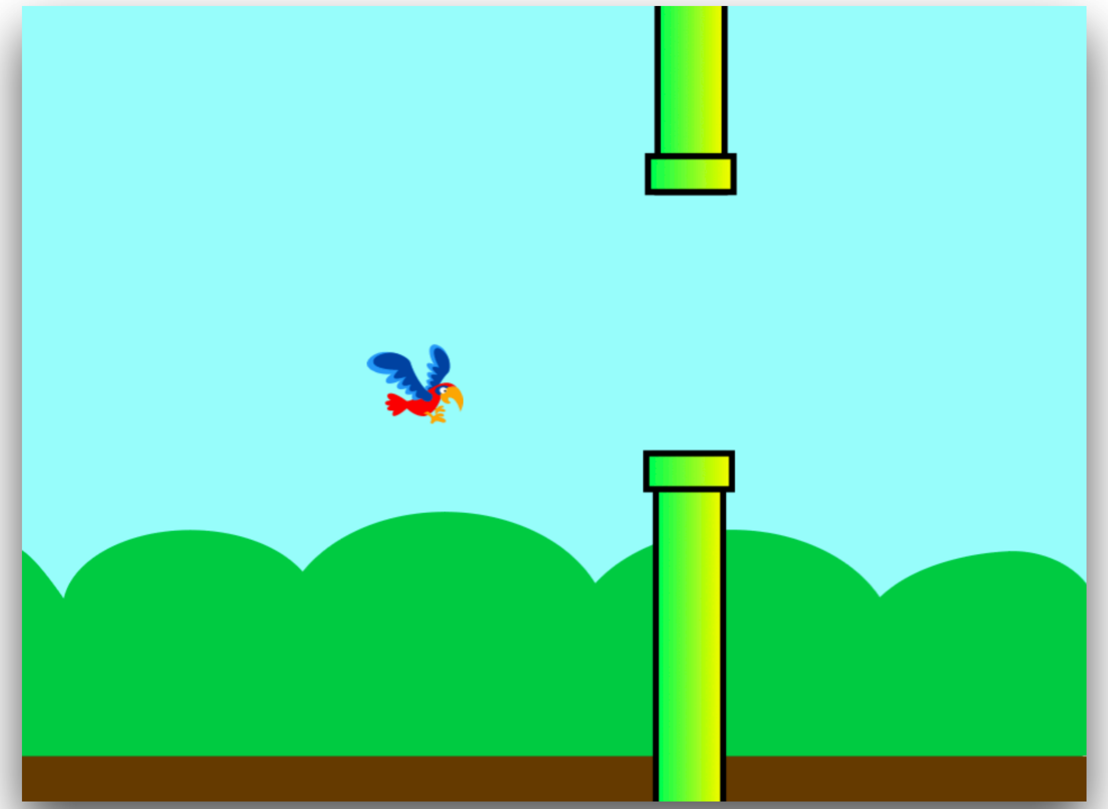


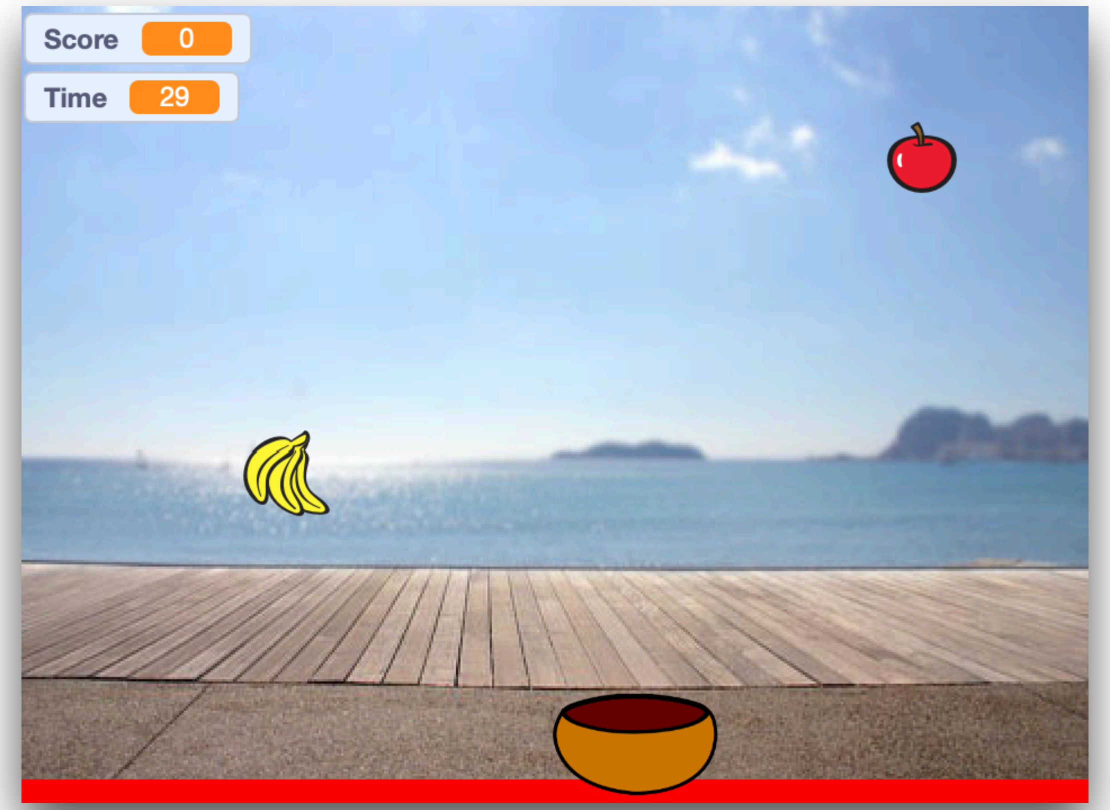
$$\delta = \frac{c_1 D}{N} + \frac{c_2 E}{N} + c_3 \overline{W}$$

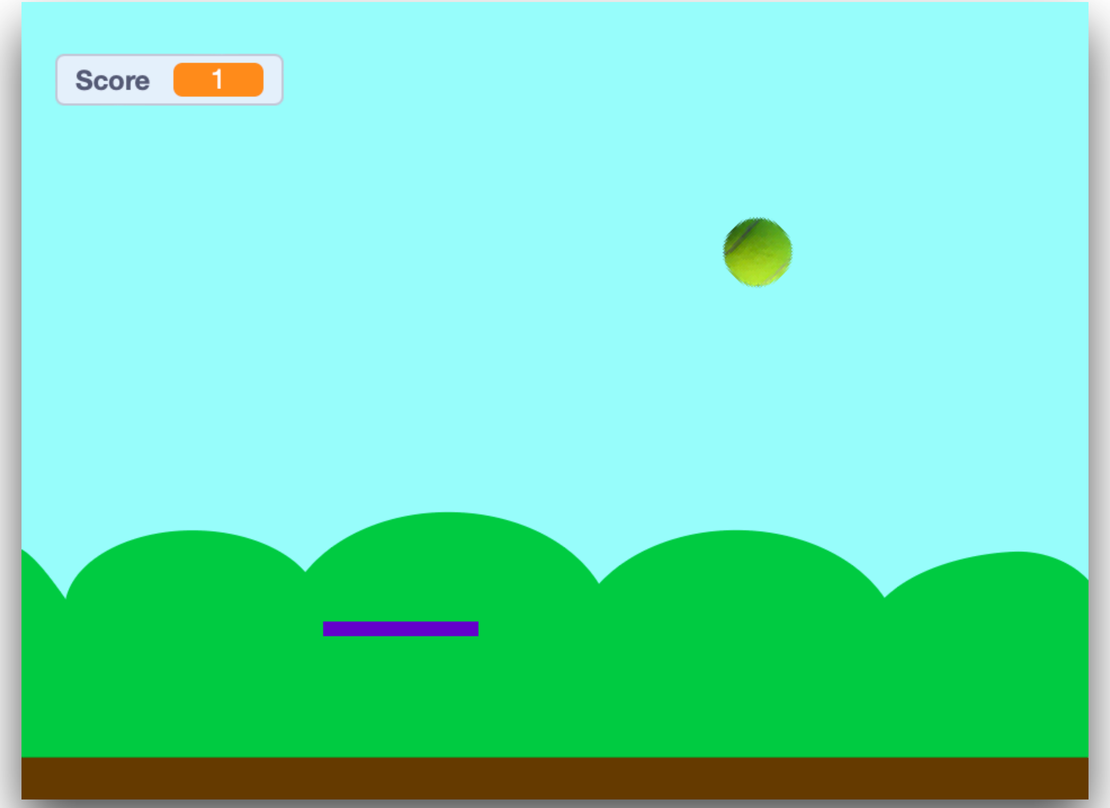
What Is the Influence of Varying Player Behaviour?

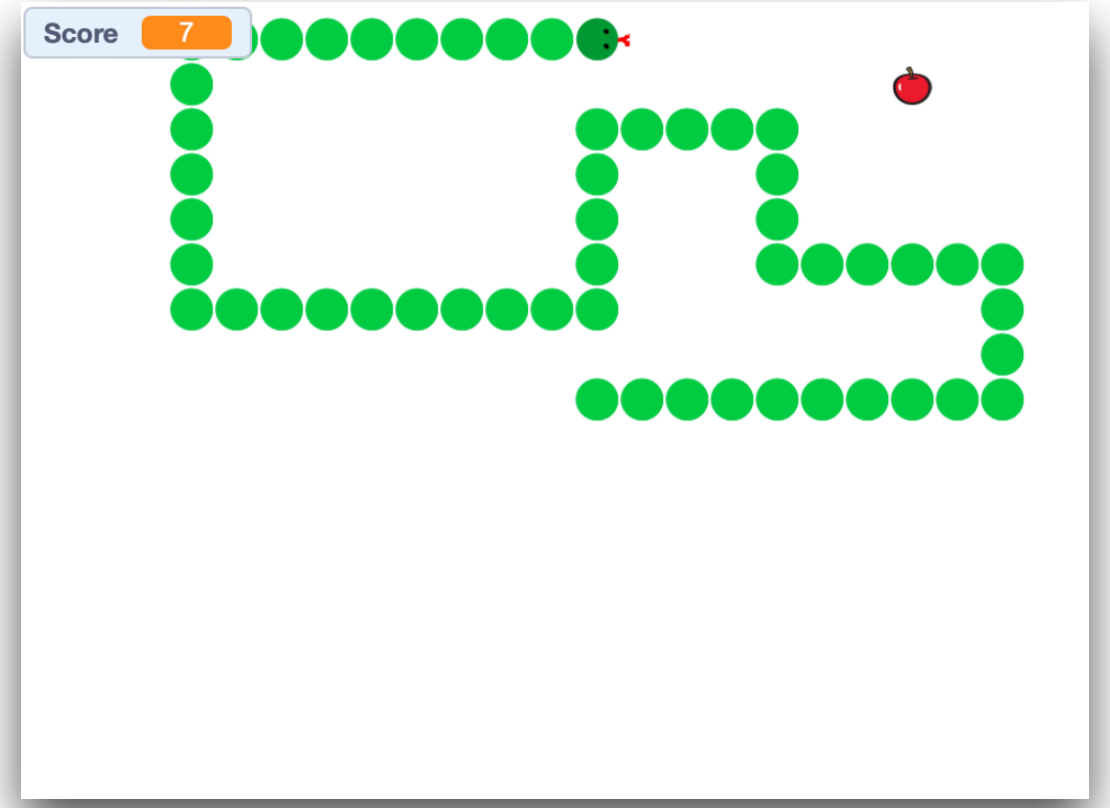


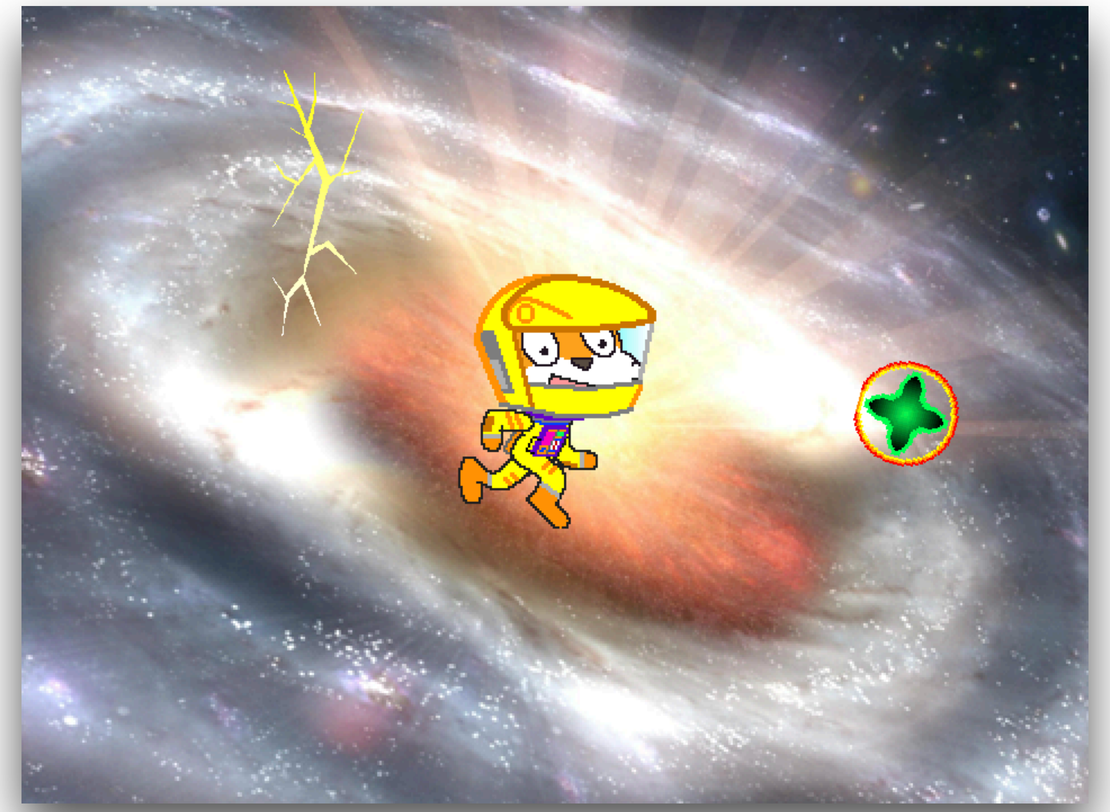




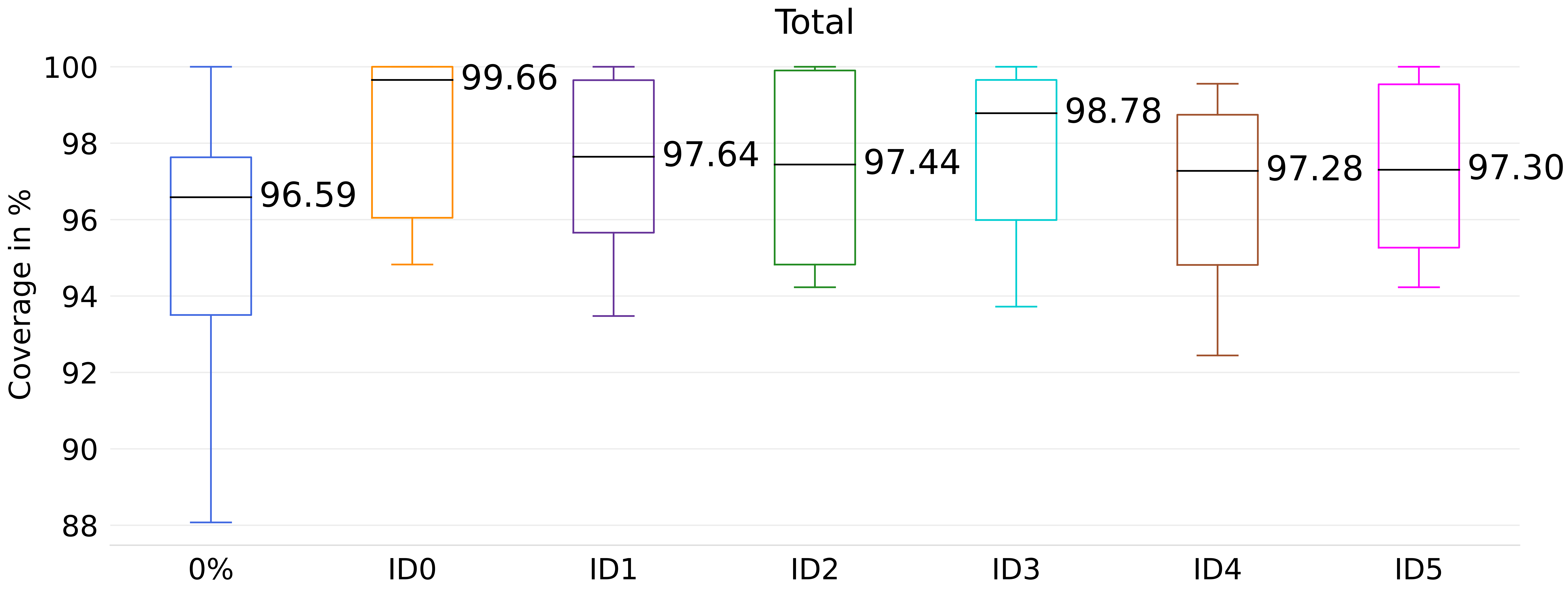




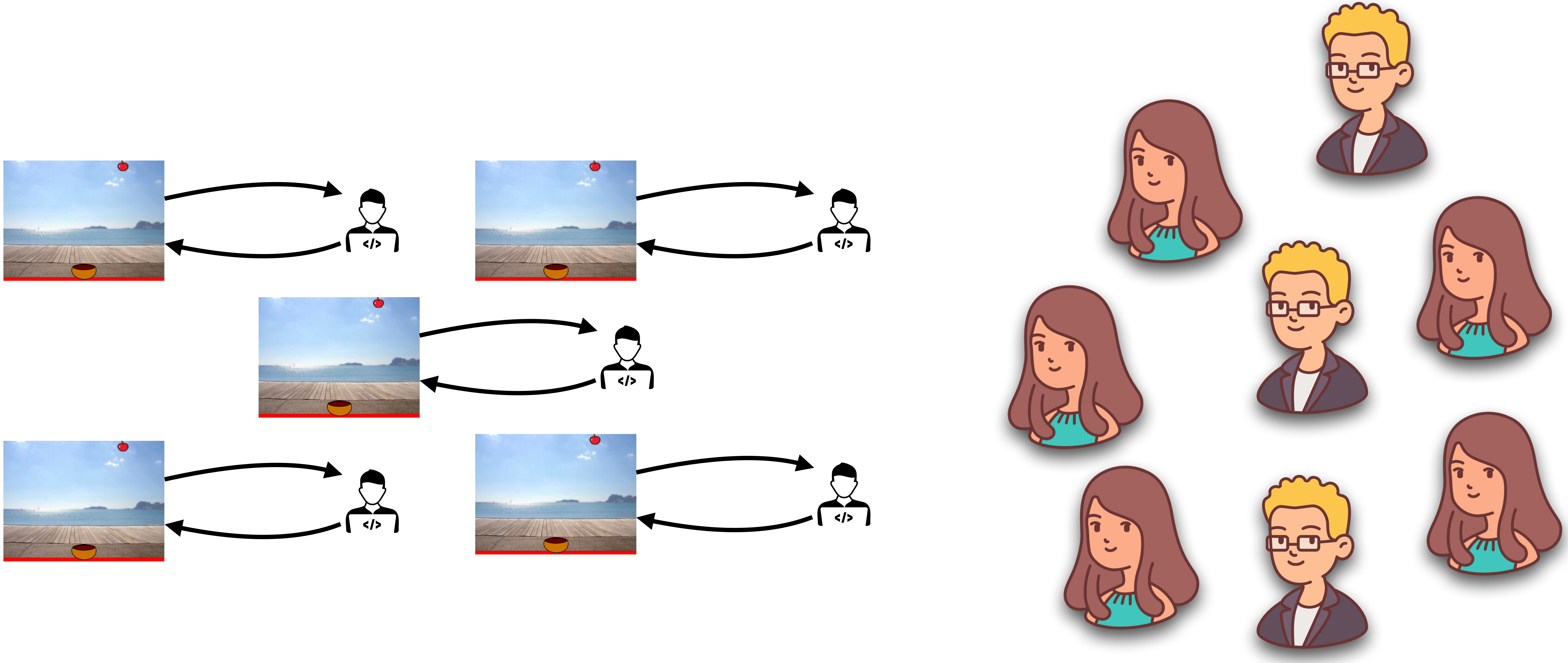




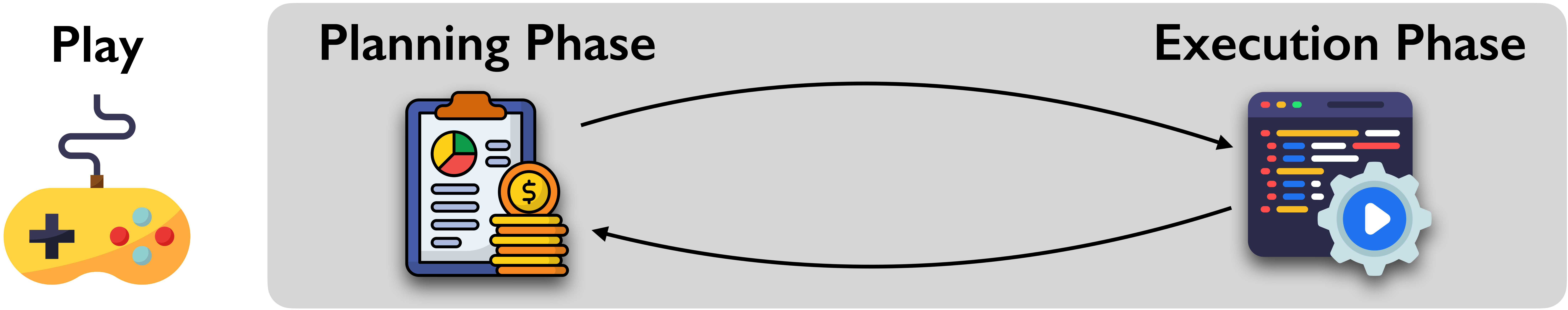
Player Behaviour Affects Network Optimisation



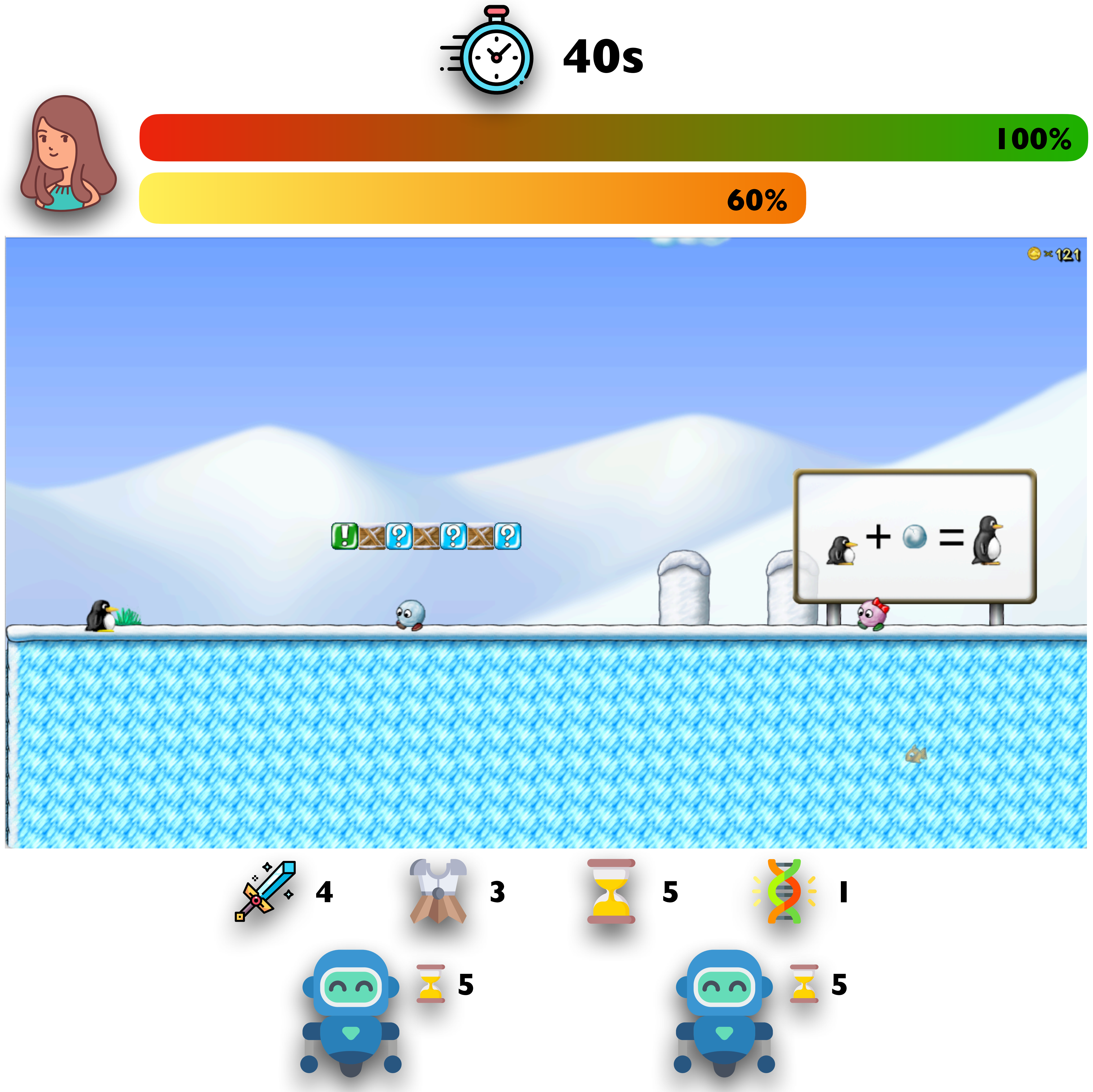
Play Test



PlayTest



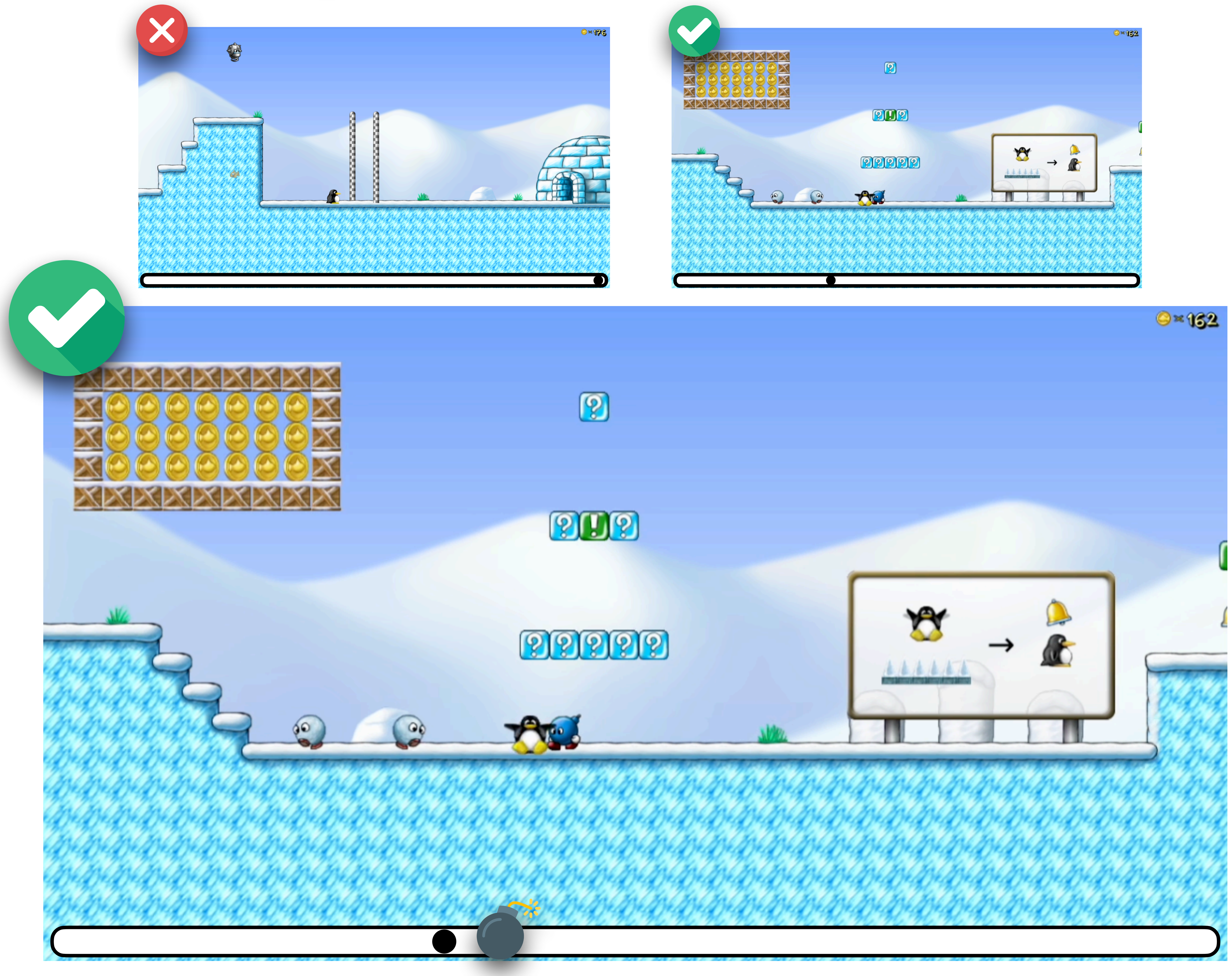
Planning Phase



Planning Phase

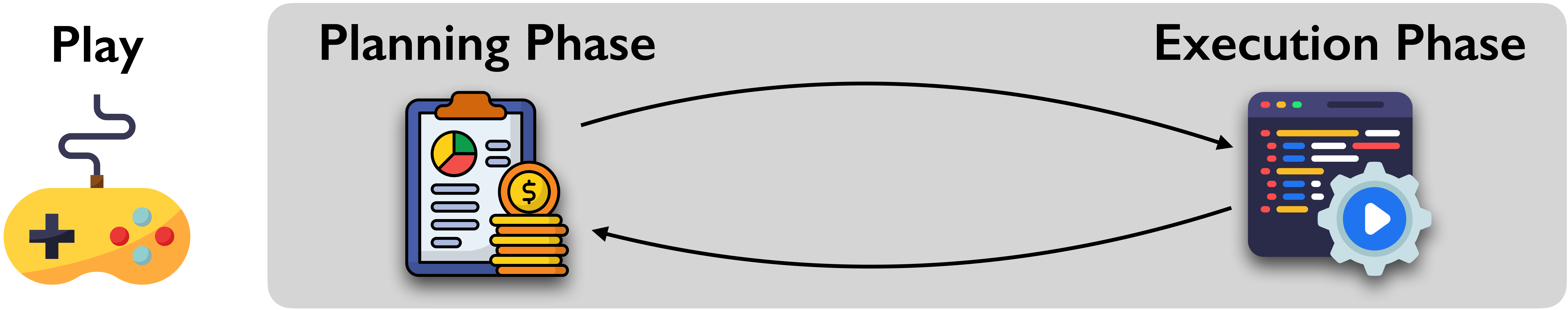
If	Penguin	Touching	Bomb	Then	Game Over		
If	Penguin	Y	<	0	Then	Game Over	
If	Penguin	Touching	Coin	Then	Increase Coins		

Execution Phase

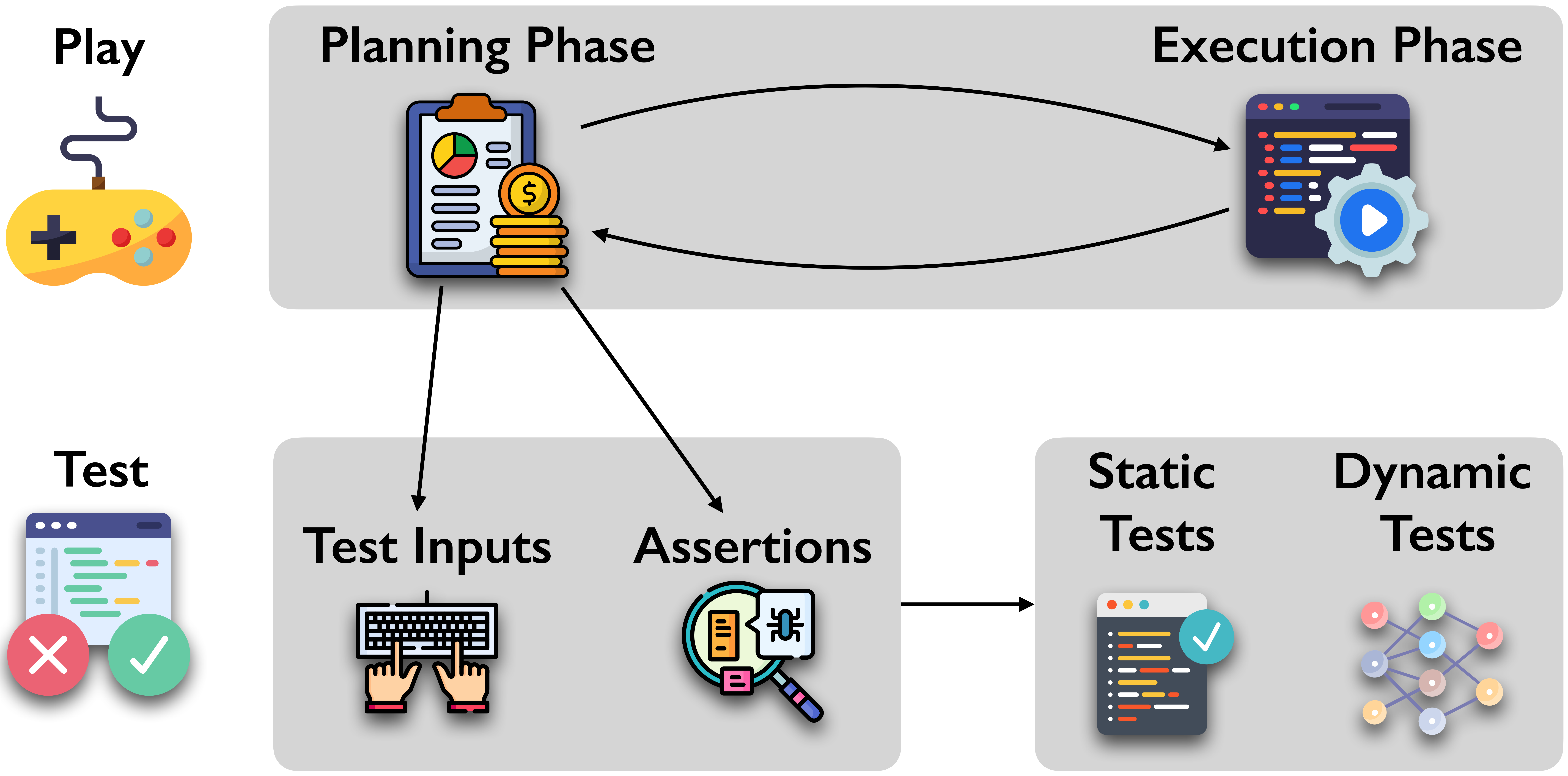


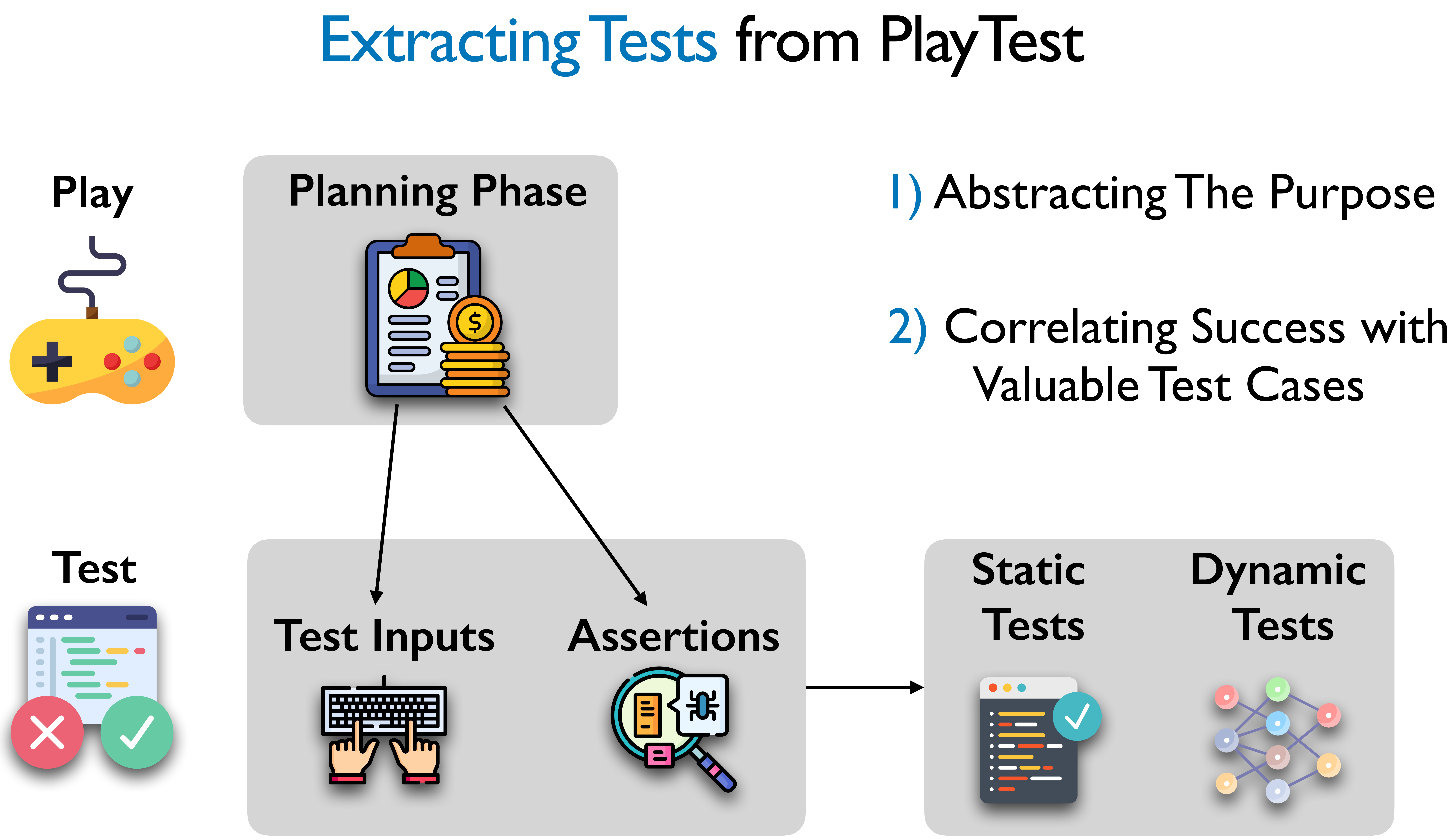
If | Penguin ▾ | Touching ▾ | Bomb ▾ | Then | Game Over ▾

Extracting Tests from PlayTest



Extracting Tests from PlayTest




Extracting Tests from PlayTest
Play
Planning Phase
1) Abstracting The Purpose
2) Correlating Success with Valuable Test Cases
Test
Test Inputs
Assertions
Static Tests
Dynamic Tests

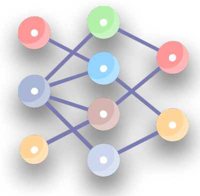
Challenges of Game Testing

Randomisation

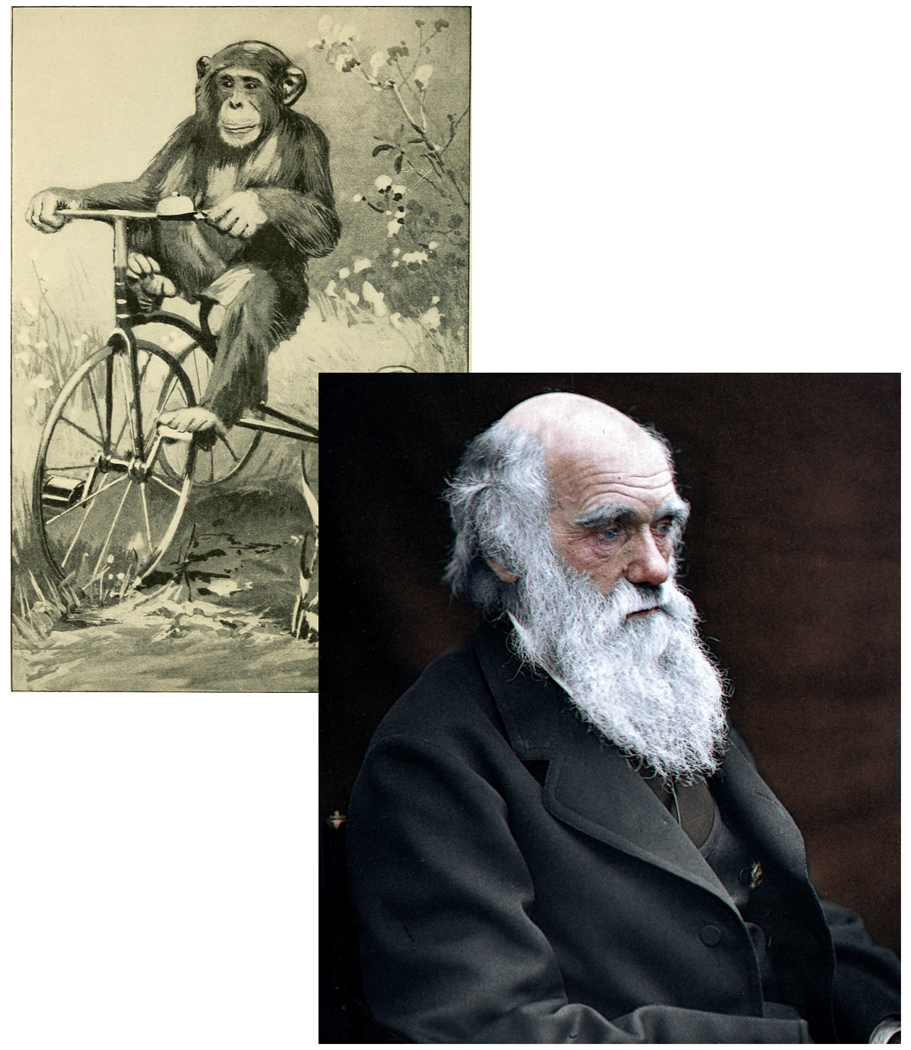
Challenging program statements

Neuro evolution

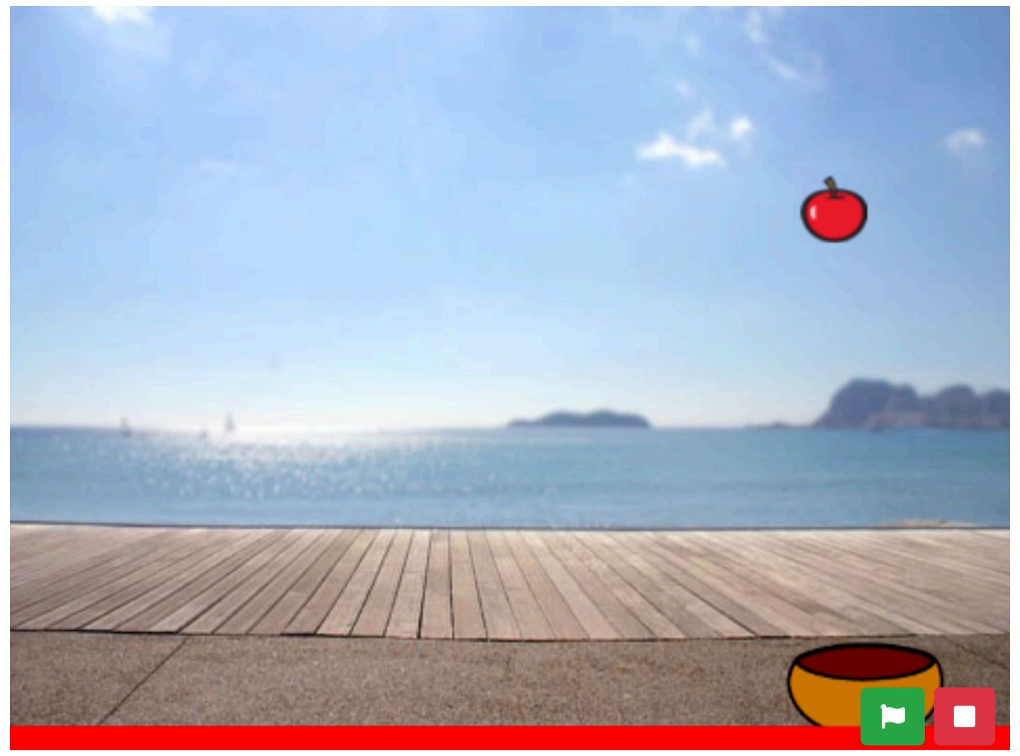
Neural networks

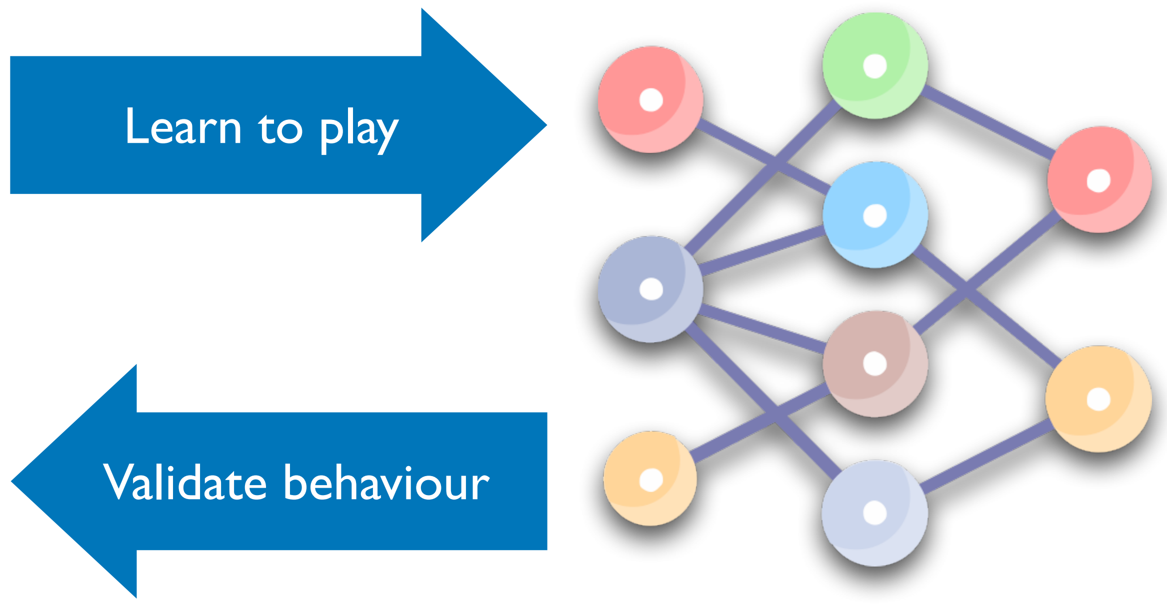


- Mimic a human brain to solve complex tasks
- Must be optimised for each task individually



Neatest





Dynamic Test Suites

Gradient-Descent as Systematic Optimiser