Genetic Programming of Autonomous Agents

Scott O'Dell

Advisors

Dr. Joel Schipper

Dr. Arnold Patton

Outline

- Introduction to Genetic Programming
- Project Summary
- Project Description
- Preliminary Results
- Schedule

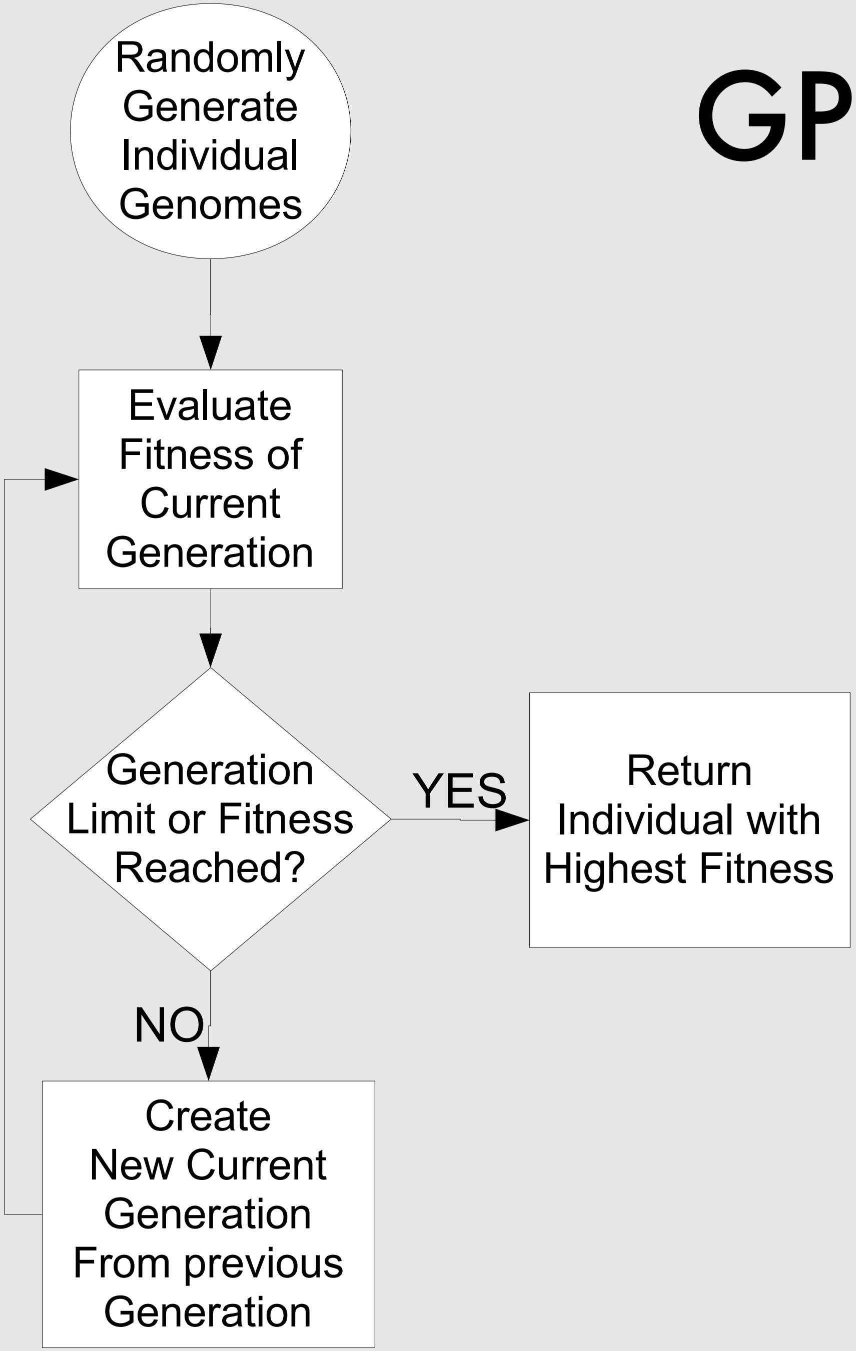
Outline

- **Introduction to Genetic Programming**
- Project Summary
- Project Description
- Preliminary Results
- Schedule

What is Genetic Programming?

- Machine learning technique
- Evolution: Survival of the fittest
- Leverages Randomness
- Program evolved to solve a task

GP Flowchart



Randomly Generate Individual Genomes

Evaluate Fitness of Current Generation

Generation Limit or Fitness Reached?

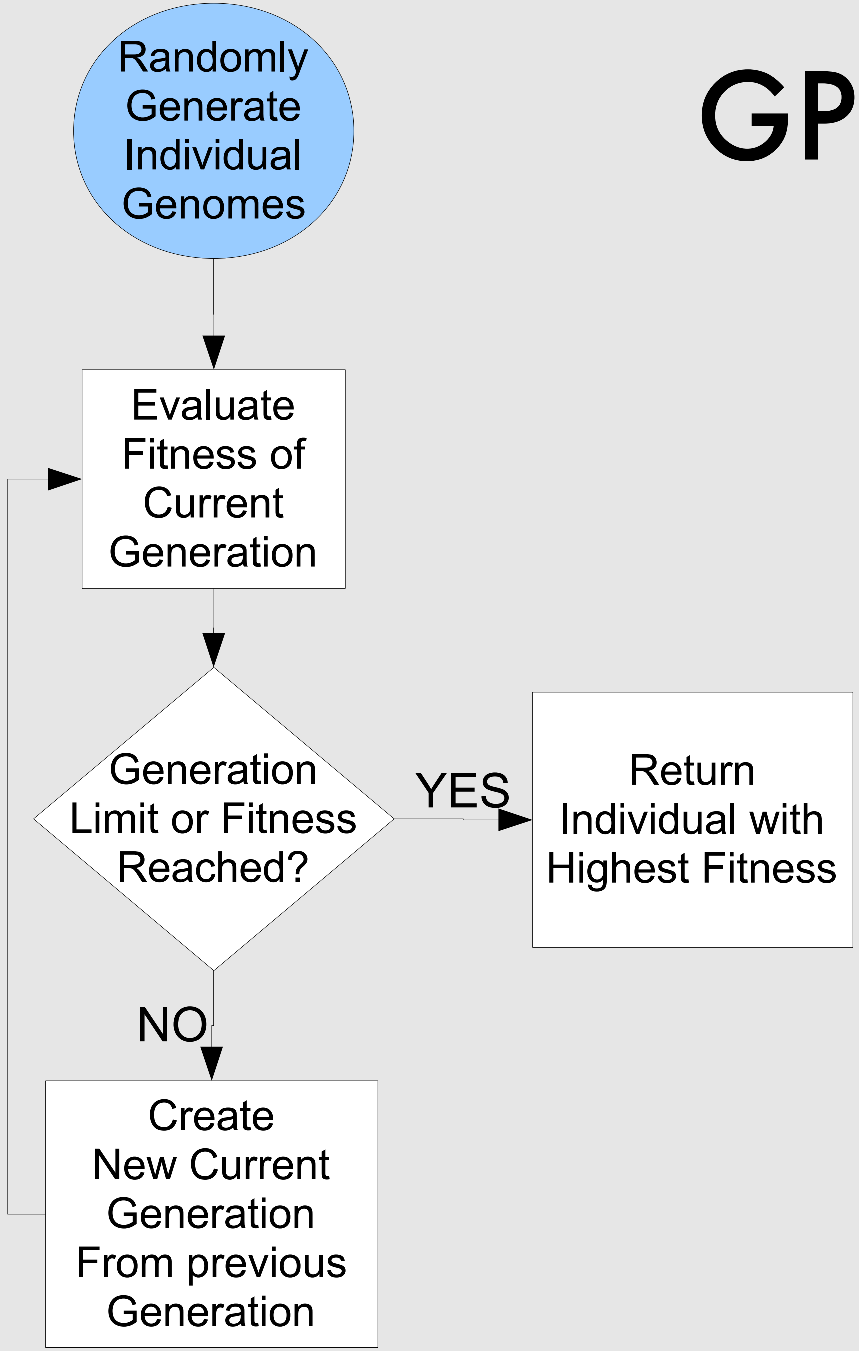
YES

Return Individual with Highest Fitness

NO

Create New Current Generation From previous Generation

GP Flowchart



Randomly Generate Individual Genomes

Evaluate Fitness of Current Generation

Generation Limit or Fitness Reached?

YES

Return Individual with Highest Fitness

NO

Create New Current Generation From previous Generation

Creating Random Genomes

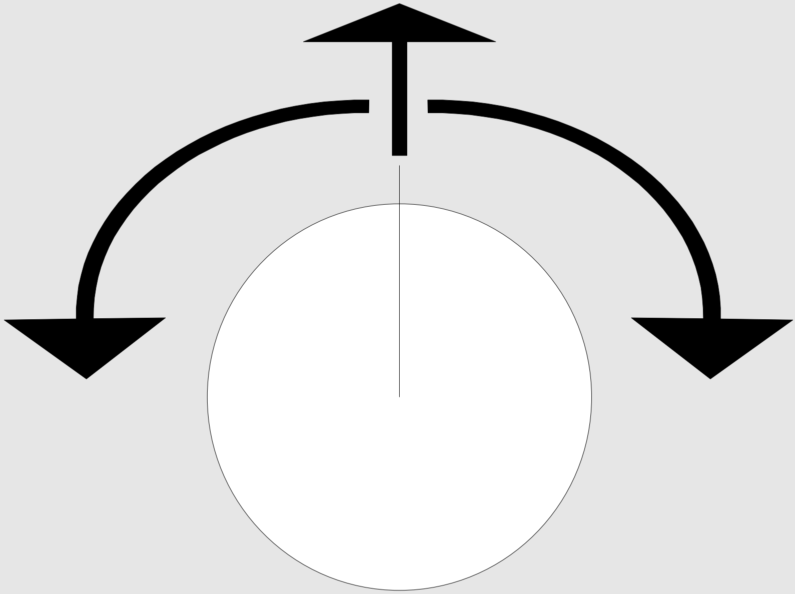
- Primitive Set
  - Function Set
    - Accepts arguments
    - Returns value
  - Terminal Set
    - No arguments
    - Represents value
    - May have side effects

- Requirements
  - Sufficiency
  - Closure

Creating Random Genomes

Wall-Following Robot

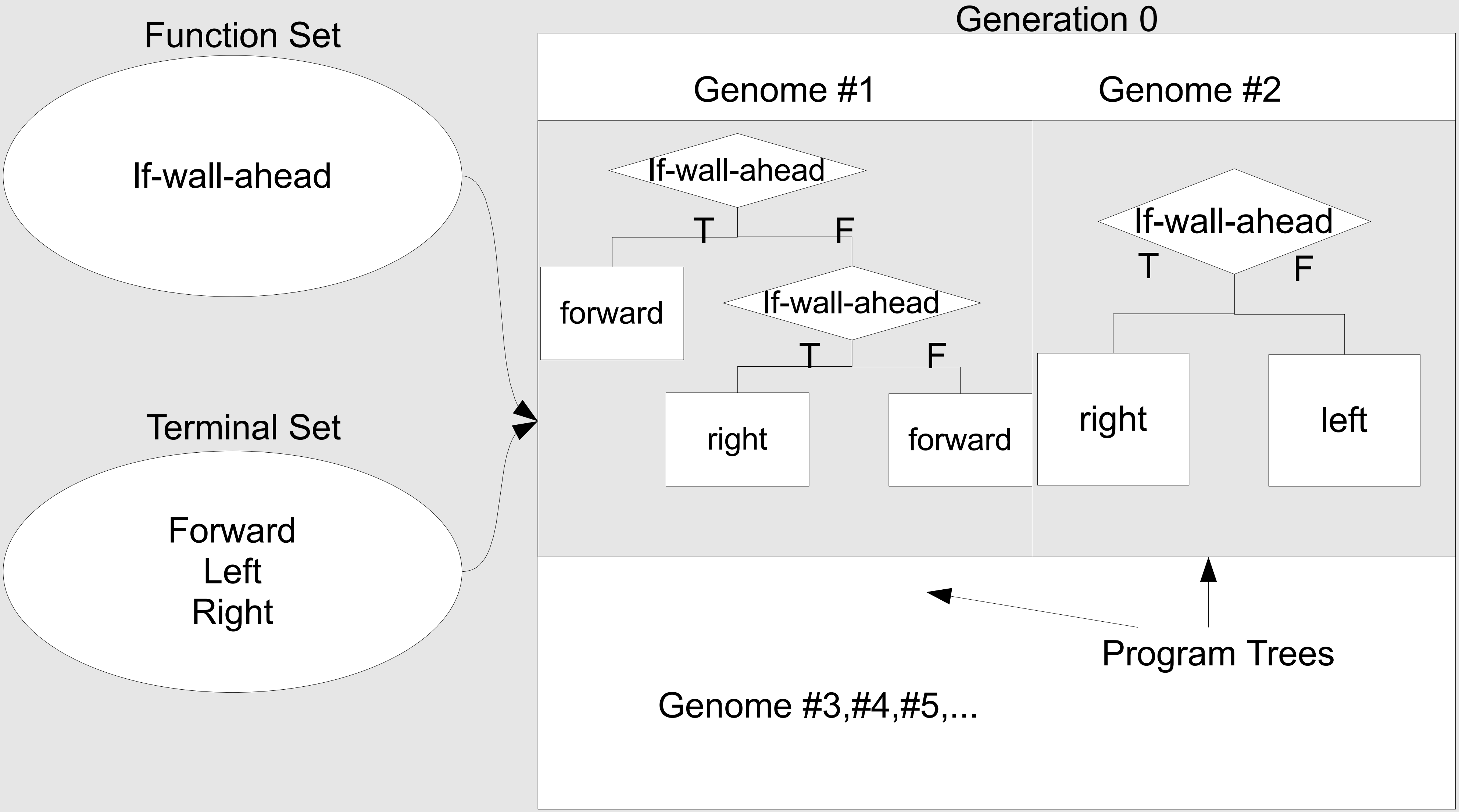
- Single proximity sensor on front
- Independent wheels



Primitive Set

- Function Set
  - If-wall-ahead
- Terminal Set
  - Forward
  - Left
  - Right

Creating Random Genomes



Function Set

If-wall-ahead

Terminal Set

Forward
Left
Right

Generation 0

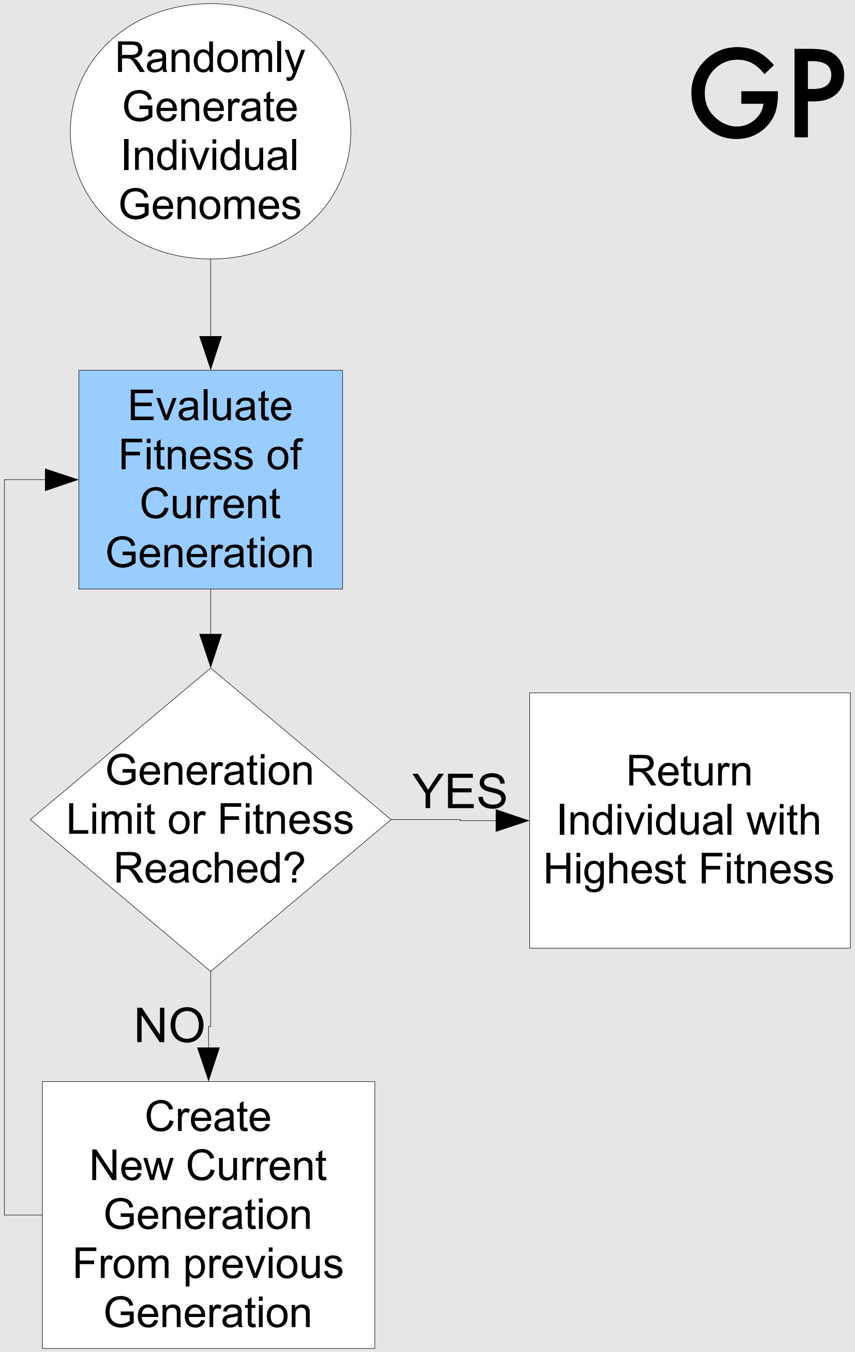
Genome #1

Genome #2

Program Trees

Genome #3,#4,#5,...

GP Flowchart



Randomly Generate Individual Genomes

Evaluate Fitness of Current Generation

Generation Limit or Fitness Reached?

YES

Return Individual with Highest Fitness

NO

Create New Current Generation From previous Generation

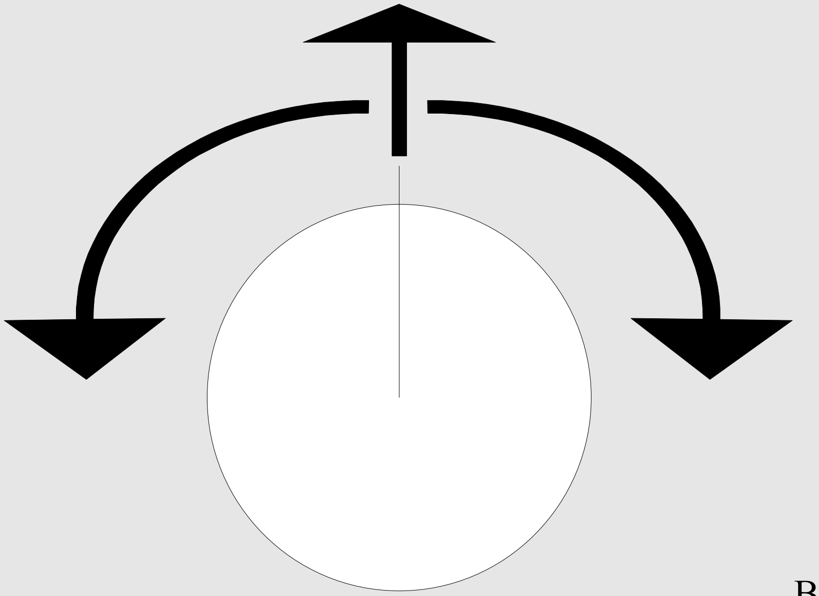
Evaluating Fitness

- Fitness Function
  - Evaluates effectiveness of programs
  - Assigns fitness score
  - Must differentiate “poor” and “very poor” performance
  - Determines likelihood of “reproduction”
- Solutions optimized for fitness function
  - NO MATTER WHAT!

Evaluating Fitness

<u>Wall-Following Robot</u>

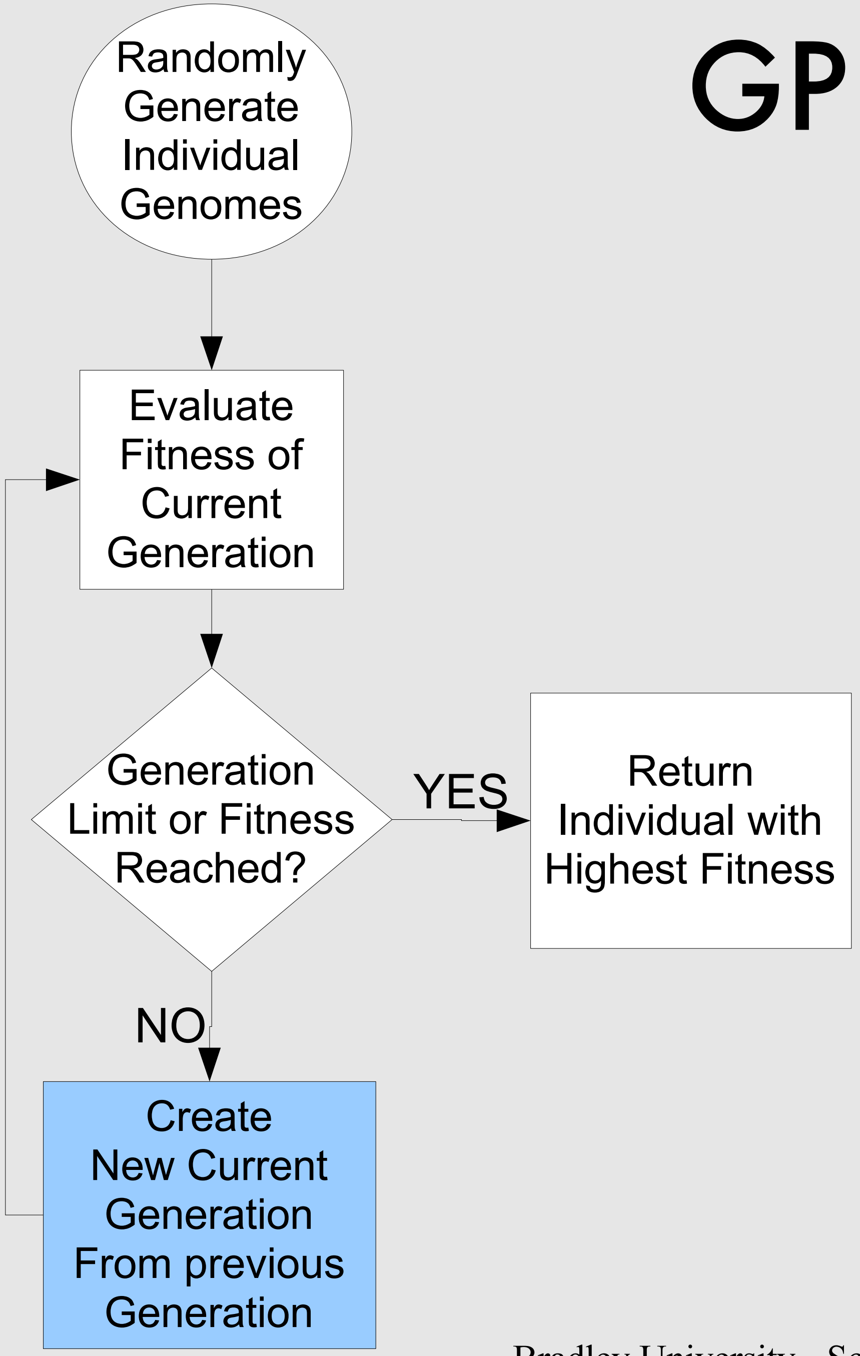
- Find a wall
- Follow wall w/o extra movements



<u>Fitness Function</u>

- Simulate robot
- Score: # of wall adjacent spaces occupied
- Limited time

GP Flowchart


Randomly Generate Individual Genomes
Evaluate Fitness of Current Generation
Generation Limit or Fitness Reached?
YES
Return Individual with Highest Fitness
NO
Create New Current Generation From previous Generation

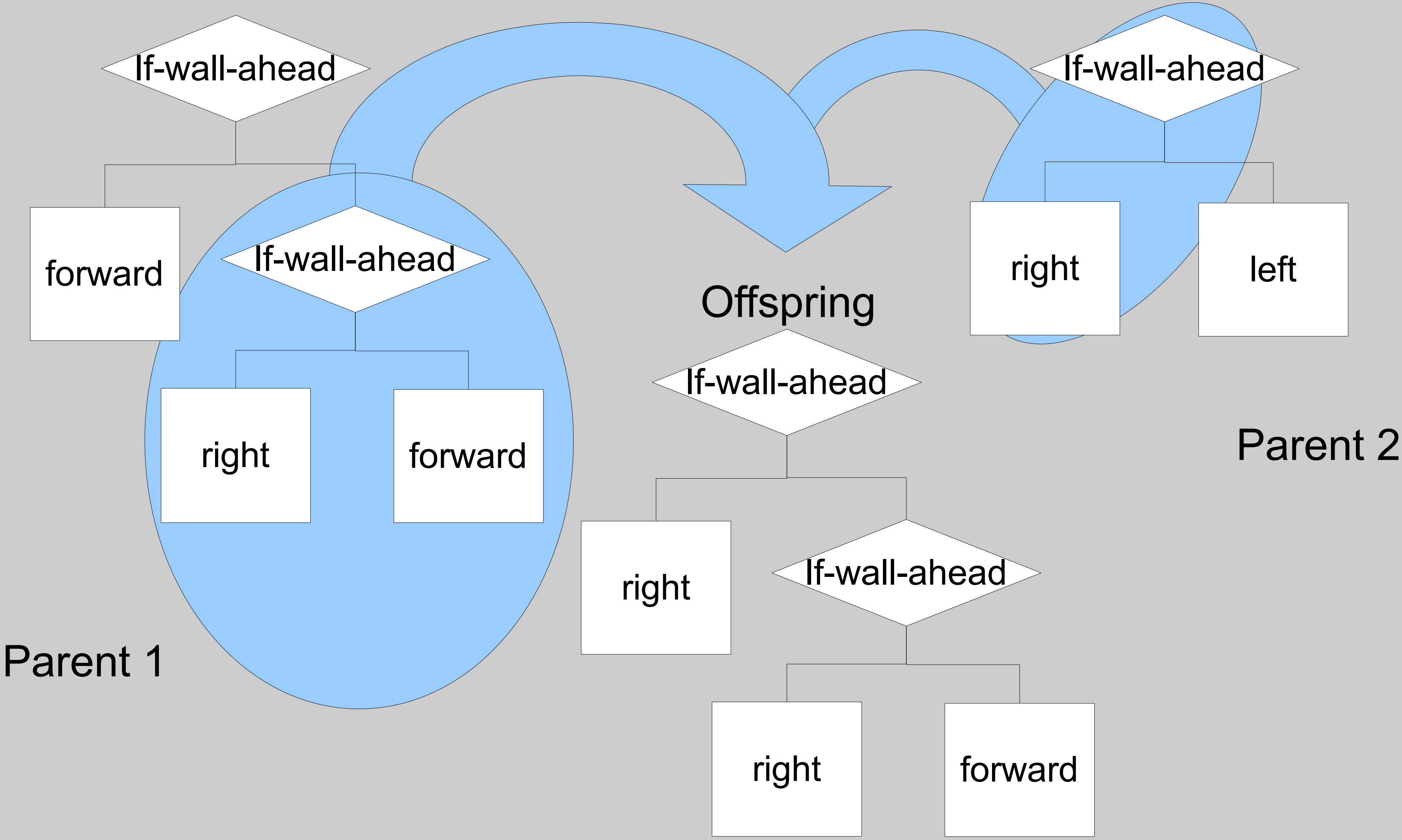
Creating a New Generation

- Selection methods
  - Fitness Proportional
    - Chance of being chosen proportional to fitness score
  - Tournament
    - Group chosen at random, highest score wins

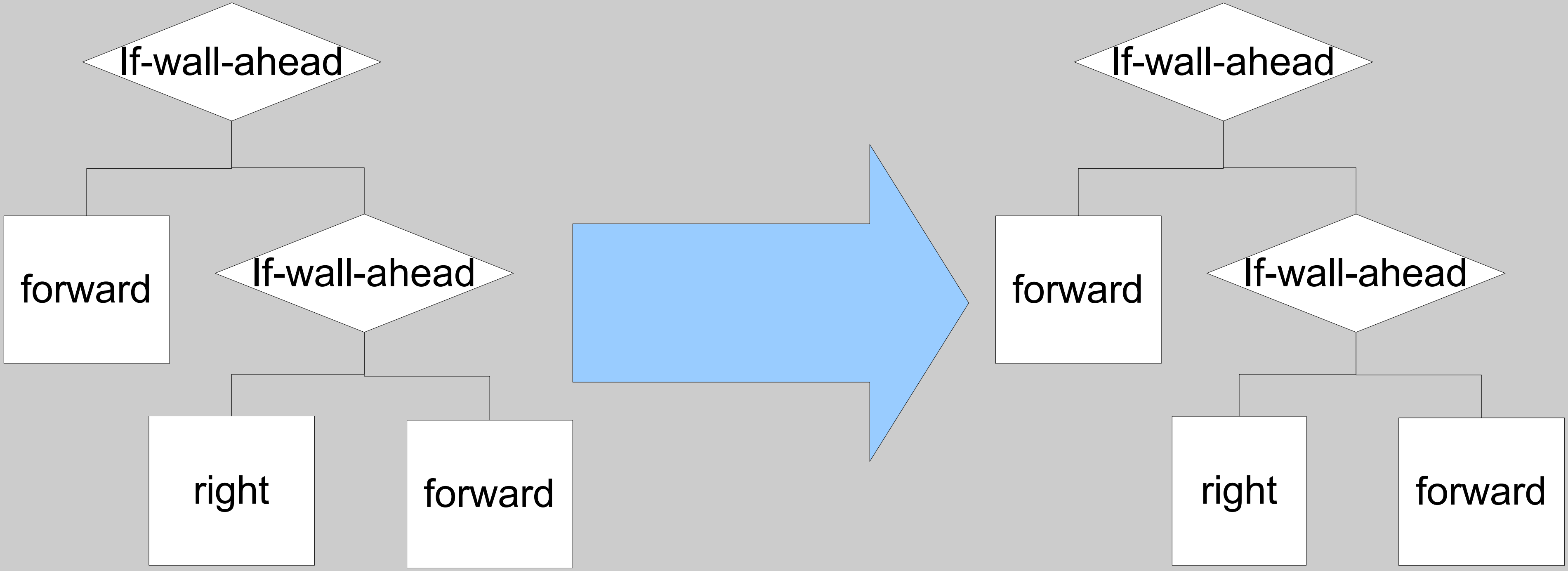
Creating a New Generation

- Genetic Operators
  - Crossover (sexual reproduction)
  - Reproduction (asexual reproduction)
  - Mutation

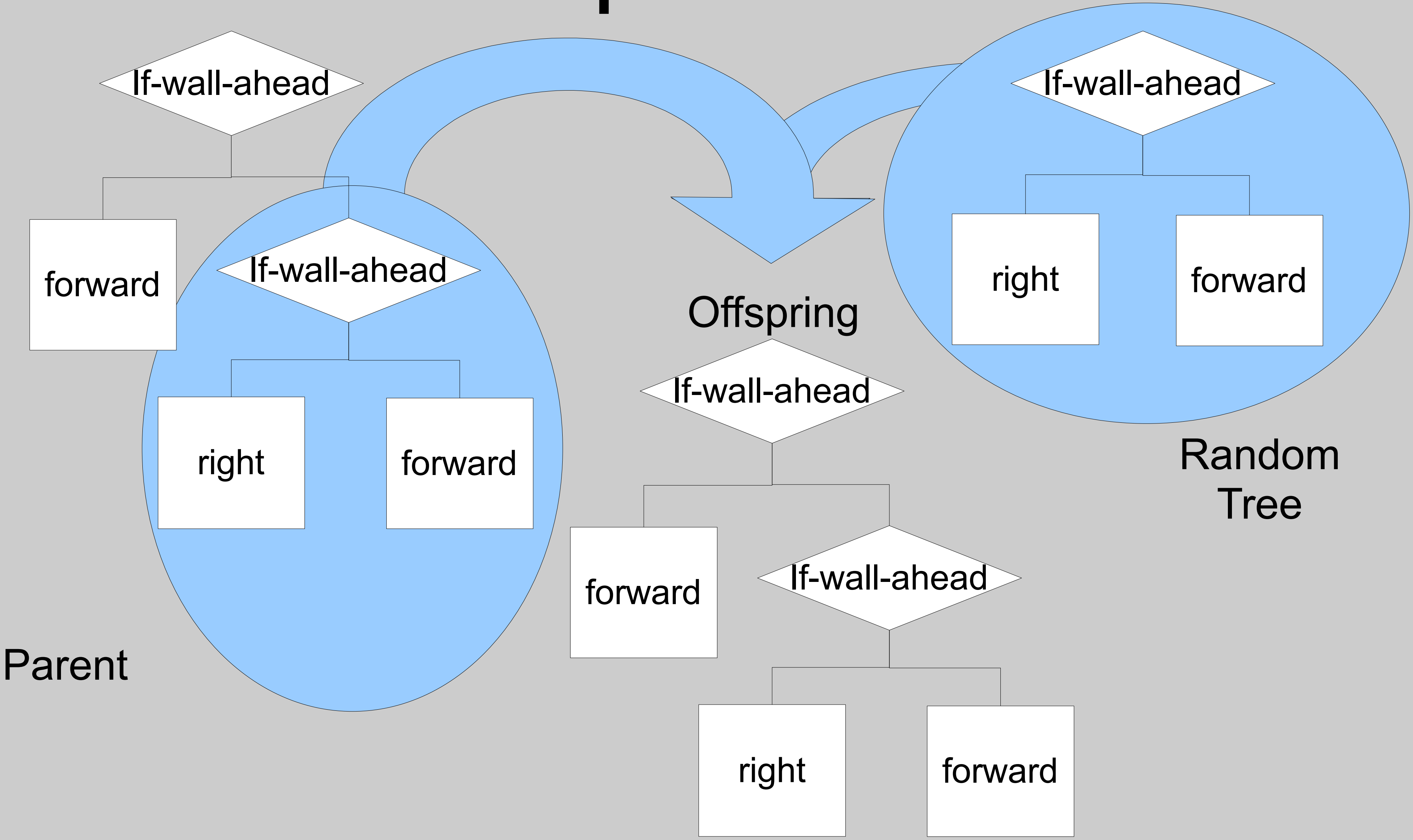
Genetic Operator: Crossover



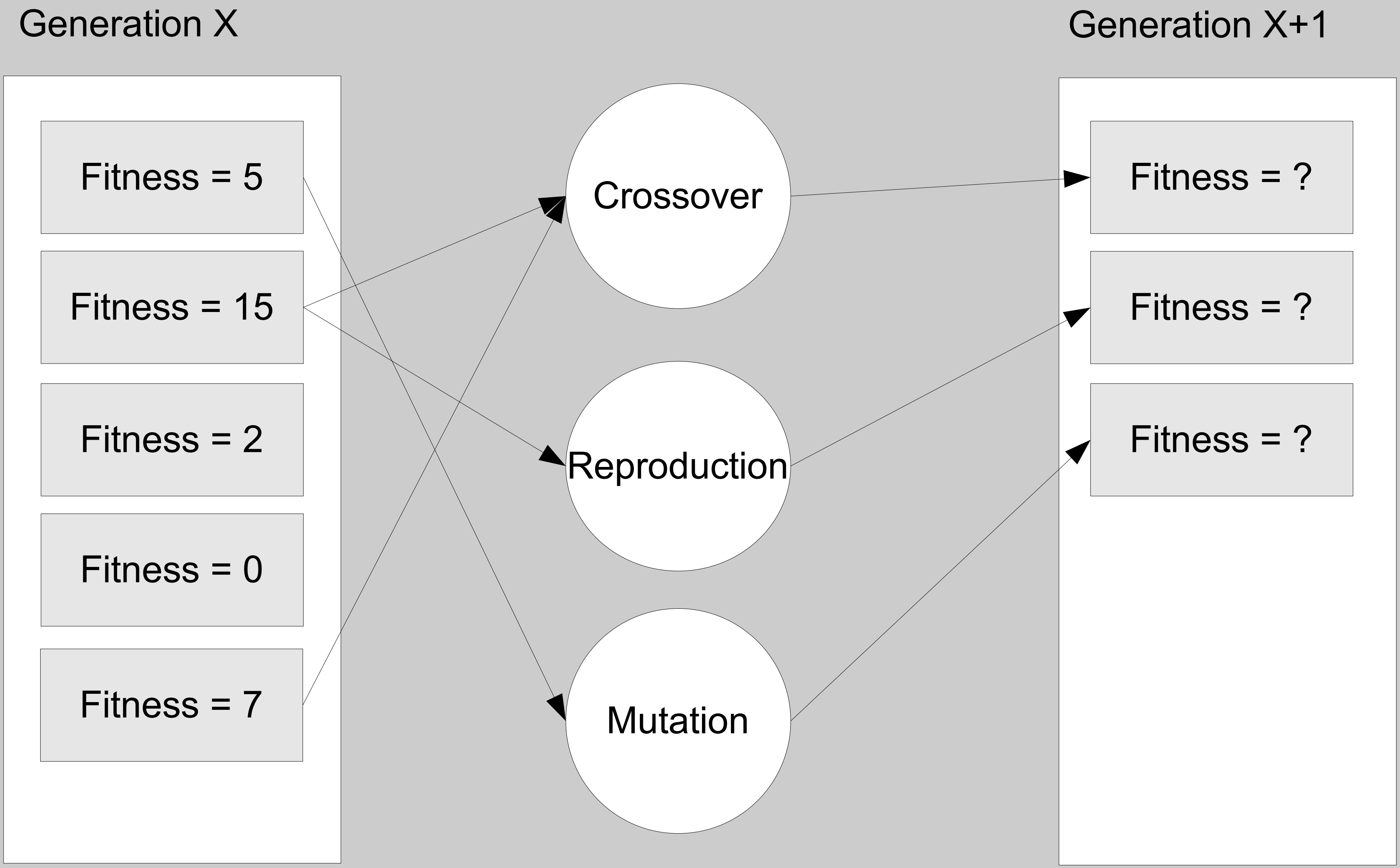
Genetic Operator: Reproduction



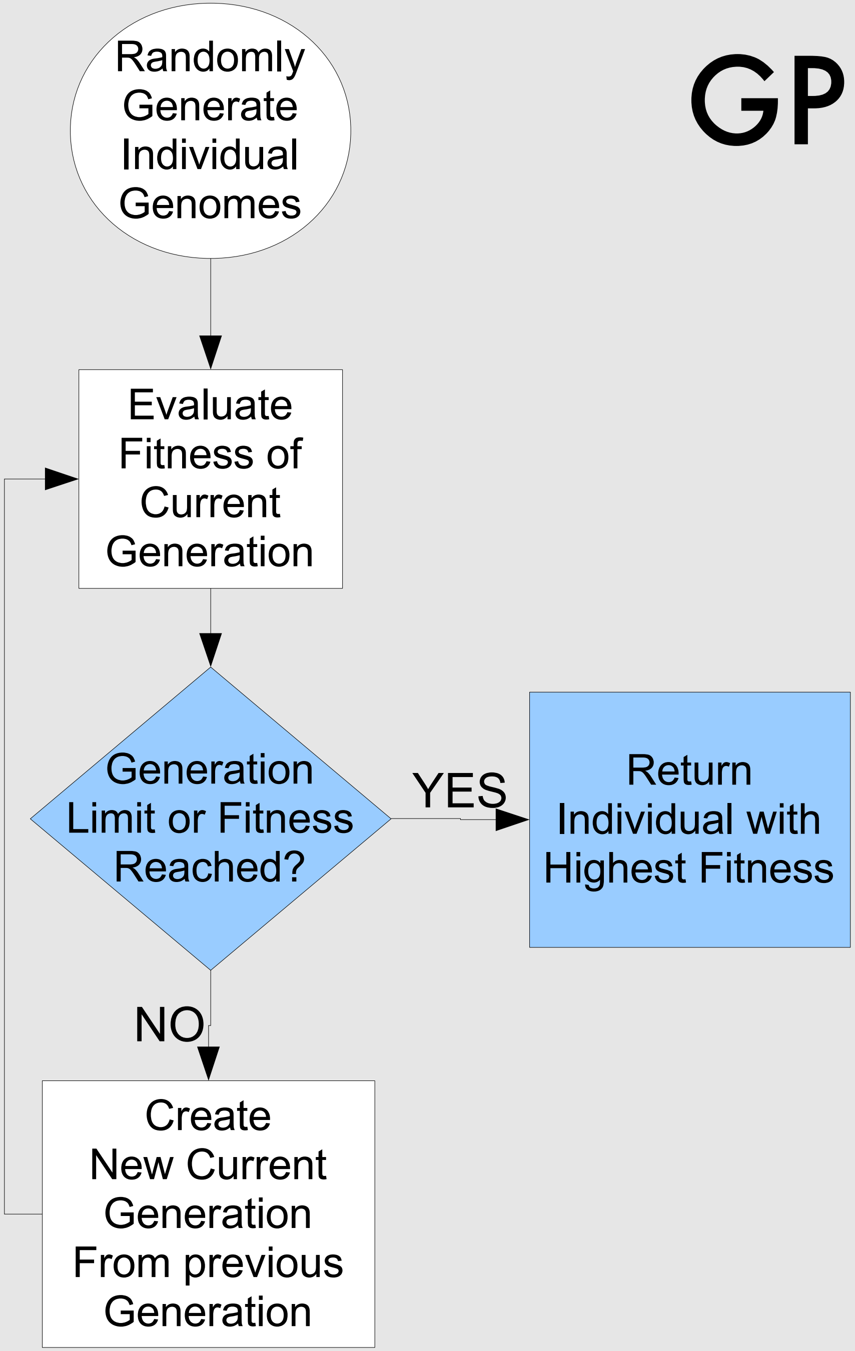
Genetic Operator: Mutation



Creating a New Generation



GP Flowchart


Randomly Generate Individual Genomes
Evaluate Fitness of Current Generation
Generation Limit or Fitness Reached?
YES
Return Individual with Highest Fitness
NO
Create New Current Generation From previous Generation

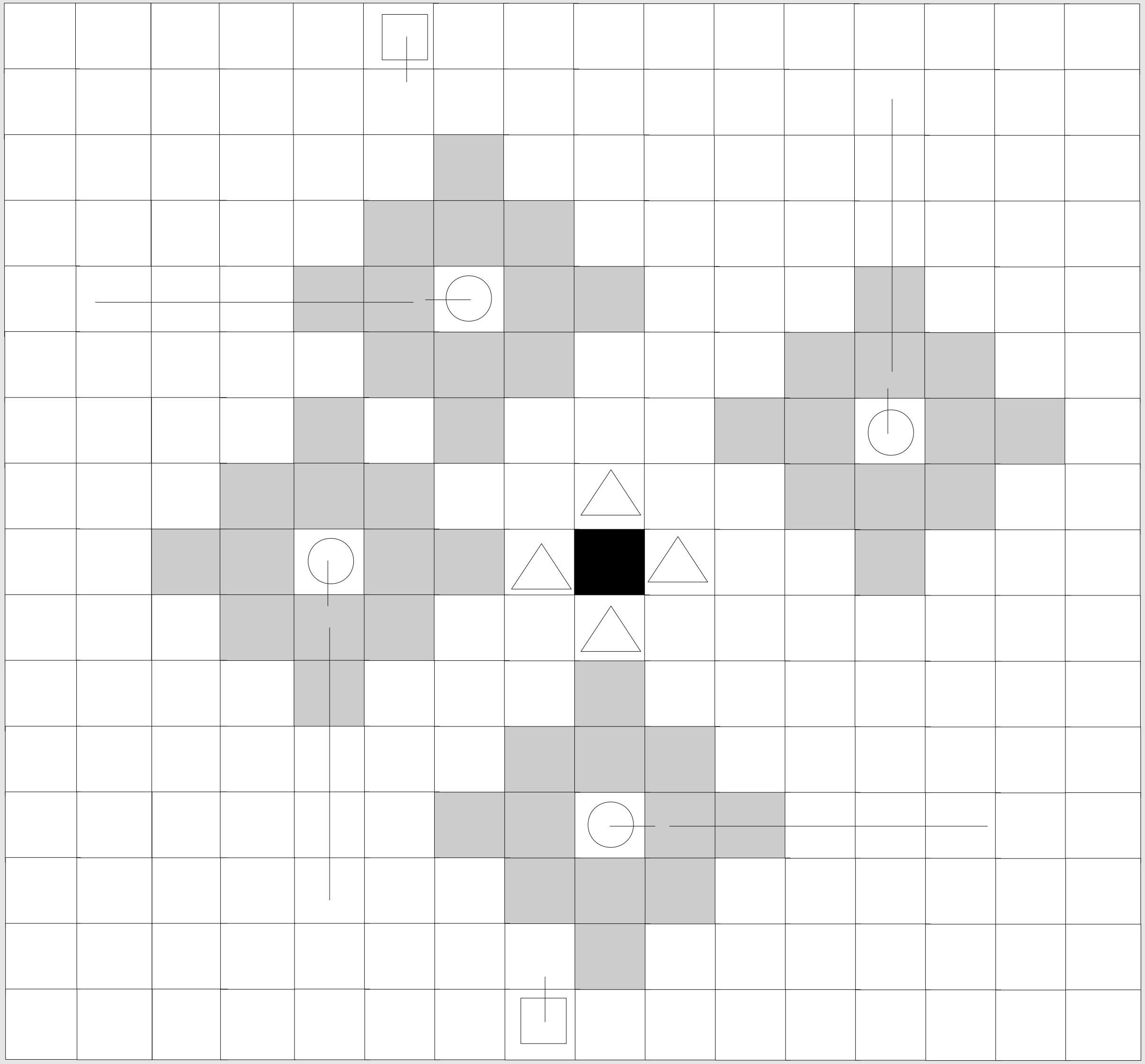
Outline

- Introduction to Genetic Programming
- **Project Summary**
- Project Description
- Preliminary Results
- Schedule

Perimeter Maintenance

- Military application
- Control programs for autonomous agents
- Maximize perfect perimeter around base
- Maximize coverage of large perimeter

Perimeter Maintenance



■ Base

△ Starting Positions

○ ○ Guard Agents

□ □ Enemy Agents

— Line of Sight

▨ Capture Areas

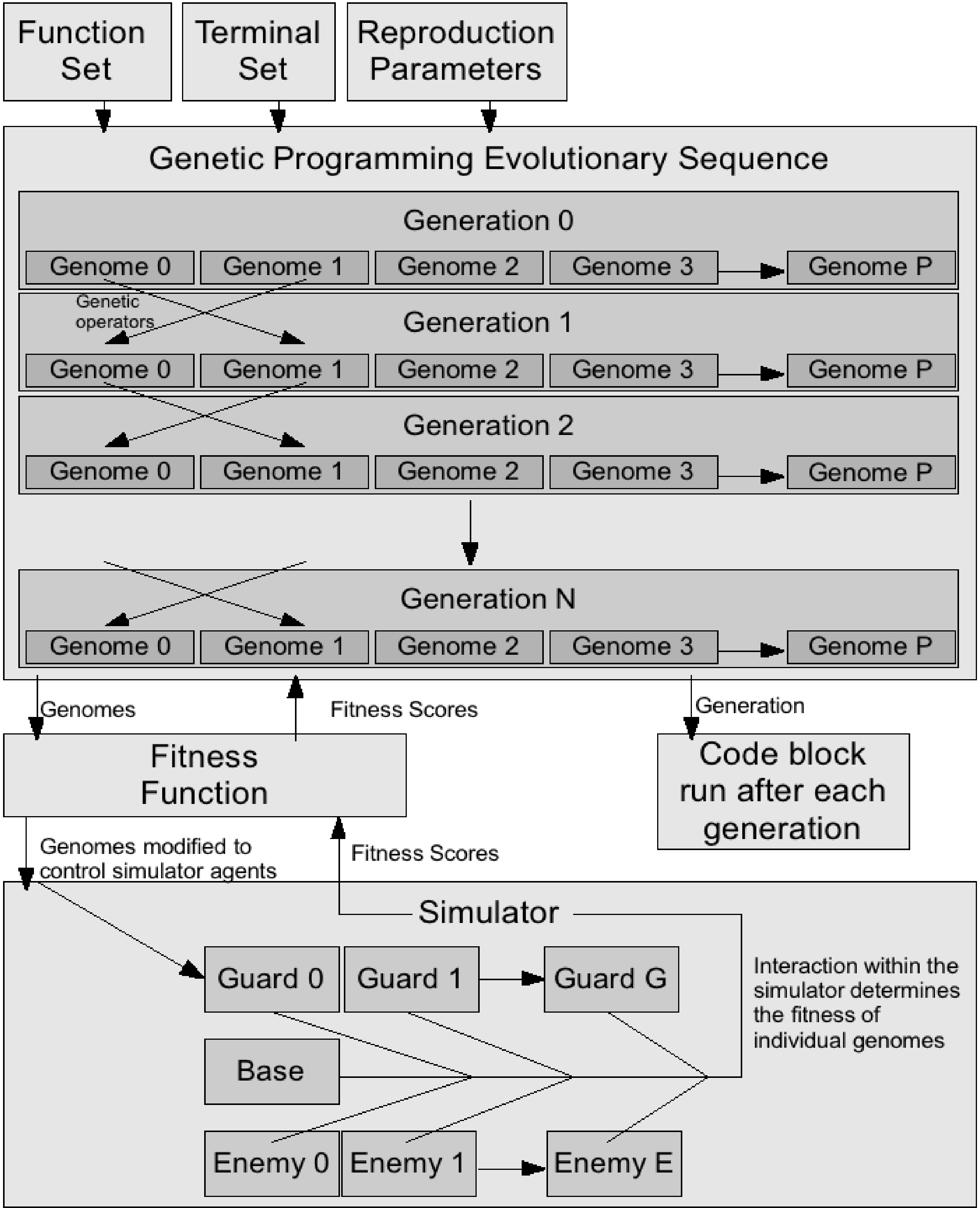
Outline

- Introduction to Genetic Programming
- Project Summary
- **Project Description**
- Preliminary Results
- Schedule

Top Level

- Written in Ruby
- Easy to interface with different languages
- Processor intensive tasks, faster languages

Top Level


Function Set
Terminal Set
Reproduction Parameters
Genetic Programming Evolutionary Sequence
Generation 0
Genome 0
Genome 1
Genome 2
Genome 3
Genome P
Genetic operators
Generation 1
Genome 0
Genome 1
Genome 2
Genome 3
Genome P
Generation 2
Genome 0
Genome 1
Genome 2
Genome 3
Genome P
Generation N
Genome 0
Genome 1
Genome 2
Genome 3
Genome P
Genomes
Fitness Scores
Generation
Fitness Function
Code block run after each generation
Genomes modified to control simulator agents
Fitness Scores
Simulator
Guard 0
Guard 1
Guard G
Base
Enemy 0
Enemy 1
Enemy E
Interaction within the simulator determines the fitness of individual genomes

Function Set

- prog
  - Evaluate 2 arguments sequentially
- ifGreater
  - pseudo code: if(1st > 2nd) then 3rd else 4th
  - Perform actions based on sensors
- +, -, *, /, and %
  - standard arithmetic calculation
  - develop sensor weighting systems

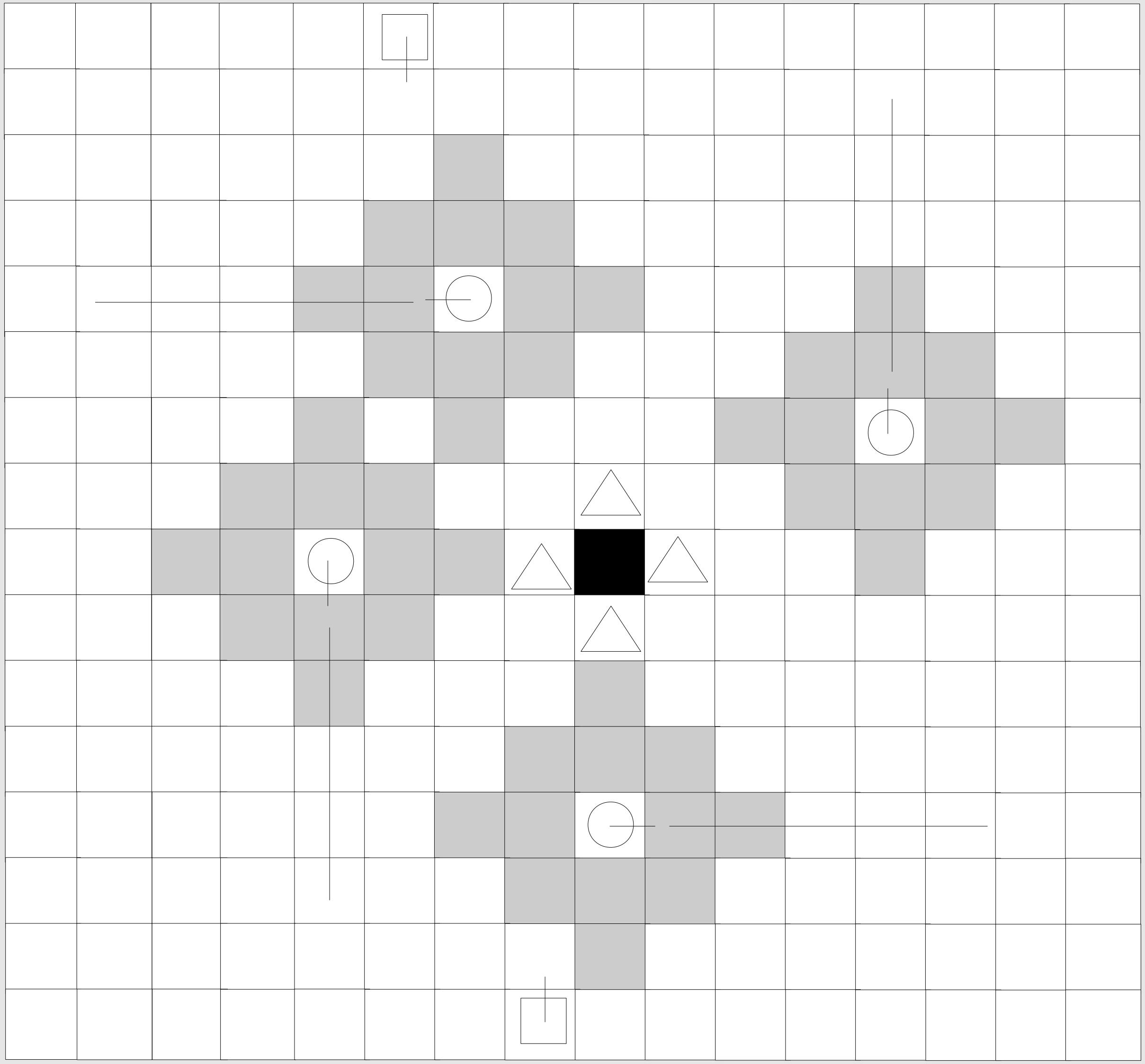
Terminal Set

- perim
  - returns Manhattan distance from the base,
  - make decisions based distance from base.
- forward, left, and right
  - moves agent
- i
  - random integer
  - generated during creation of genome

Simulator

- Genome controls agent
- Interactions determine fitness
- Initially, grid-based
- Later, continuous, add noise

Perimeter Maintenance



■ Base

△ Starting Positions

○ Guards

□ Enemies

— Line of Sight

▨ Capture Areas

Fitness Function

- Positive points
  - Guard captures enemy
  - Distance at which enemy is captured
- Negative points
  - Enemy enters perimeter
  - Guard collision

Robotic Platform

- Time permitting
- Genome interpreter
- Primitive set defines
  - Motor control routines
  - Sensor processing routine

Outline

- Introduction to Genetic Programming
- Project Summary
- Project Description
- **Preliminary Results**
- Schedule

Preliminary Results

- Completed
  - GPES block
  - Primitive set for grid navigation
  - Grid-based simulator / fitness function

Preliminary Results

- Maximizing perfect perimeter
  - Perimeter = 0 (enemy must hit base)
  - Guard sensor range = 4
  - Fitness function
    - distance from base at which enemies were captured
  - # of Generations = 50
  - Population of each generation = 1,000
  - 80% crossover, 15% reproduction, 5% mutation

Preliminary Results

```
. . . . . . . . 4 . . . . . . . . . .
. . . . . . . 4 . 4 . . . . . . . . .
. . . . . . 4 . . . 4 . . . . . . . .
. . . . . 4 . . . . . 4 . . . . . . .
. . . . 4 . . . < . . . 4 . 4 . . . .
. . . . . 4 . . . . . 4 . 4 . 4 . . .
. . . . 4 . 4 . . . 4 . 4 . . . 4 . .
. . . 4 . 4 . 4 . 4 . 4 . . . . . 4 .
. . 4 . . . 4 . 4 . 4 . . . ^ . . . 4
. 4 . . . . . 4 . x . 4 . . . . . 4 .
4 . . . v . . . 4 . 4 . 4 . . . 4 . .
. 4 . . . . . 4 . 4 . 4 . 4 . 4 . . .
. . 4 . . . 4 . 4 . . . 4 . 4 . . . .
. . . 4 . 4 . 4 . . . . . 4 . . . . .
. . . . 4 . 4 . . . > . . . 4 . . . .
. . . . . . . 4 . . . . . 4 . . . . .
. . . . . . . . 4 . . . 4 . . . . . .
. . . . . . . . . 4 . 4 . . . . . . .
. . . . . . . . . . 4 . . . . . . . .
```

Preliminary Results

- Maximizing coverage of large perimeter
  - Perimeter = 9
  - Guard sensor range = 4
  - Fitness function
    - # of enemies captured
  - # of Generations = 50
  - Generation population = 1,000
  - 80% crossover, 15% reproduction, 5% mutation

Preliminary Results

```
. . . . . . . . . . . . . . . . 4 . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 4 . 4 . . . . . . . . . . . . . .
. . . . . . . . . . . . . . 4 . * . 4 . . . . . . . . . . . . .
. . . . . . . . . . . . . 4 . * . * . 4 . . . . . . . . . . . .
. . . . . . . . . . . . 4 . * . ^ . * . 4 . . . . . . . . . . .
. . . . . . . . . . . . . * . . . . . * . . . . . . . . . . . .
. . . . . . . . . . . . * . 4 . . . 4 . * . . . . . . . . . . .
. . . . . . . . . 4 . * . . . 4 . 4 . . . * . 4 . . . . . . . .
. . . . . . . . 4 . * . . . . . 4 . . . . . * . 4 . . . . . . .
. . . . . . . 4 . * . 4 . . . . . . . . . 4 . * . 4 . . . . . .
. . . . . . 4 . * . . . 4 . . . . . . . 4 . . . * . 4 . . . . .
. . . . . 4 . * . < . . . 4 . . x . . 4 . . . > . * . 4 . . . .
. . . . . . 4 . * . . . 4 . . . . . . . 4 . . . * . 4 . . . . .
. . . . . . . 4 . * . 4 . . . . . . . . . 4 . * . 4 . . . . . .
. . . . . . . . 4 . * . . . . . 4 . . . . . * . 4 . . . . . . .
. . . . . . . . . 4 . * . . . 4 . 4 . . . * . 4 . . . . . . . .
. . . . . . . . . . . . * . 4 . . . 4 . * . . . . . . . . . . .
. . . . . . . . . . . . . * . . . . . * . . . . . . . . . . . .
. . . . . . . . . . . . 4 . * . v . * . 4 . . . . . . . . . . .
. . . . . . . . . . . . . 4 . * . * . 4 . . . . . . . . . . . .
. . . . . . . . . . . . . . 4 . * . 4 . . . . . . . . . . . . .
. . . . . . . . . . . . . . . 4 . 4 . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . 4 . . . . . . . . . . . . . . .
```

Preliminary Results

- Find optimal points, but...
  - Boring
  - Deploy and Post
  - Shouldn't they move?
- Problem: asymmetries of grid domain
  - Found points that use asymmetry as advantage
  - Cannot move and maintain advantage

Preliminary Results

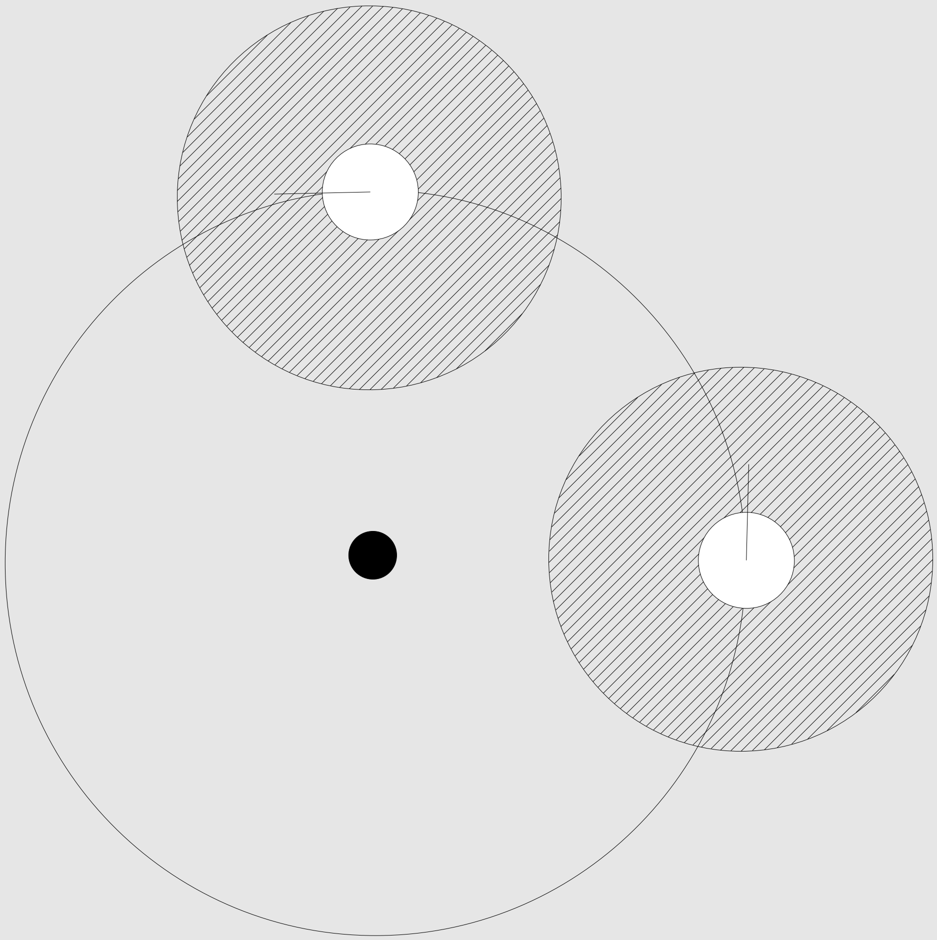
```
. . . . . . . . . . . 4 . . . . . . . . . . .
. . . . . . . . . . 4 . 4 . . . . . . . . . .
. . . . . . . . . 4 . * . 4 . . . . . . . . .
. . . . . . . . 4 . * . * . 4 . . . . . . . .
. . . . . . . 4 . * . ^ . * . 4 . . . . . . .
. . . . . . . . * . . . . . * . . . . . . . .
. . . . . . . * . 4 . . . 4 . * . . . . . . .
. . . . 4 . * . . . 4 . 4 . . . * . 4 . . . .
. . . 4 . * . . . . . 4 . . . . . * . 4 . . .
. . 4 . * . 4 . . . . . . . . . 4 . * . 4 . .
. 4 . * . . . 4 . . . . . . . 4 . . . * . 4 .
4 . * . < . . . 4 . . x . . 4 . . . > . * . 4
. 4 . * . . . 4 . . . . . . . 4 . . . * . 4 .
. . 4 . * . 4 . . . . . . . . . 4 . * . 4 . .
. . . 4 . * . . . . . 4 . . . . . * . 4 . . .
. . . . 4 . * . . . 4 . 4 . . . * . 4 . . . .
. . . . . . . * . 4 . . . 4 . * . . . . . . .
. . . . . . . . * . . . . . * . . . . . . . .
. . . . . . . 4 . * . v . * . 4 . . . . . . .
. . . . . . . . 4 . * . * . 4 . . . . . . . .
. . . . . . . . . 4 . * . 4 . . . . . . . . .
. . . . . . . . . . 4 . 4 . . . . . . . . . .
. . . . . . . . . . . 4 . . . . . . . . . . .
```

From here, 7 units on the perimeter can be protected

From here, only 5 units on the perimeter can be protected

Preliminary Results

- Less boring results
  - Co-evolution of enemies
  - Continuous domain
  - Noise



Outline

- Introduction to Genetic Programming
- Project Summary
- Project Description
- Preliminary Results
- **Schedule**

Schedule

- Main goals scheduled for before spring break
- Robotic platform work placed after spring break
  - Time permitting
  - Decision based on results with continuous simulator
  - Research of platform made in parallel with simulator work

Schedule: Completed Work

Week of	Agenda
October 3	Genome Class
October 10	Generation Class
October 17	Genetic Programming Evolutionary Sequence Class
October 24	Grid-Based Simulator
October 31	Fitness Function, Terminal Set, Function Set, and initial Simulations
November 7	Code Refactoring
November 14	Capstone Project Deliverables
November 21	*Thanksgiving Break*

Schedule: Future Work

Week of	**Agenda**
January 9	Enemy Co-evolution and Heterogeneous Teams
January 16	Continuous Simulator
January 23	Graphics for Continuous Simulator
January 30	Interface Code for Continuous Simulator and Simulations
February 6	Add Noise to Continuous Simulator
February 13	Code Refactoring
February 20	Simulations with Noise, Modification of Fitness Function

Schedule: Future Work

Week of	**Agenda**
February 27	Simulations with Modified Fitness Function
March 6	Collect Results and Create Presentation
March 13	*Spring Break*
March 20	Research Robotic Platform
March 27	Prepare Robotic Platform
April 3	Write Program Tree Interpreter for Robotic Platform
April 10	Load Evolved Program onto Robotic Platform and Debug
April 17	Evaluate Performance, Modify Simulator, New Simulations
April 24	Load Newly Evolved Program onto Robotic Platform

Questions?