Soar as a High-Level AI with SGIO

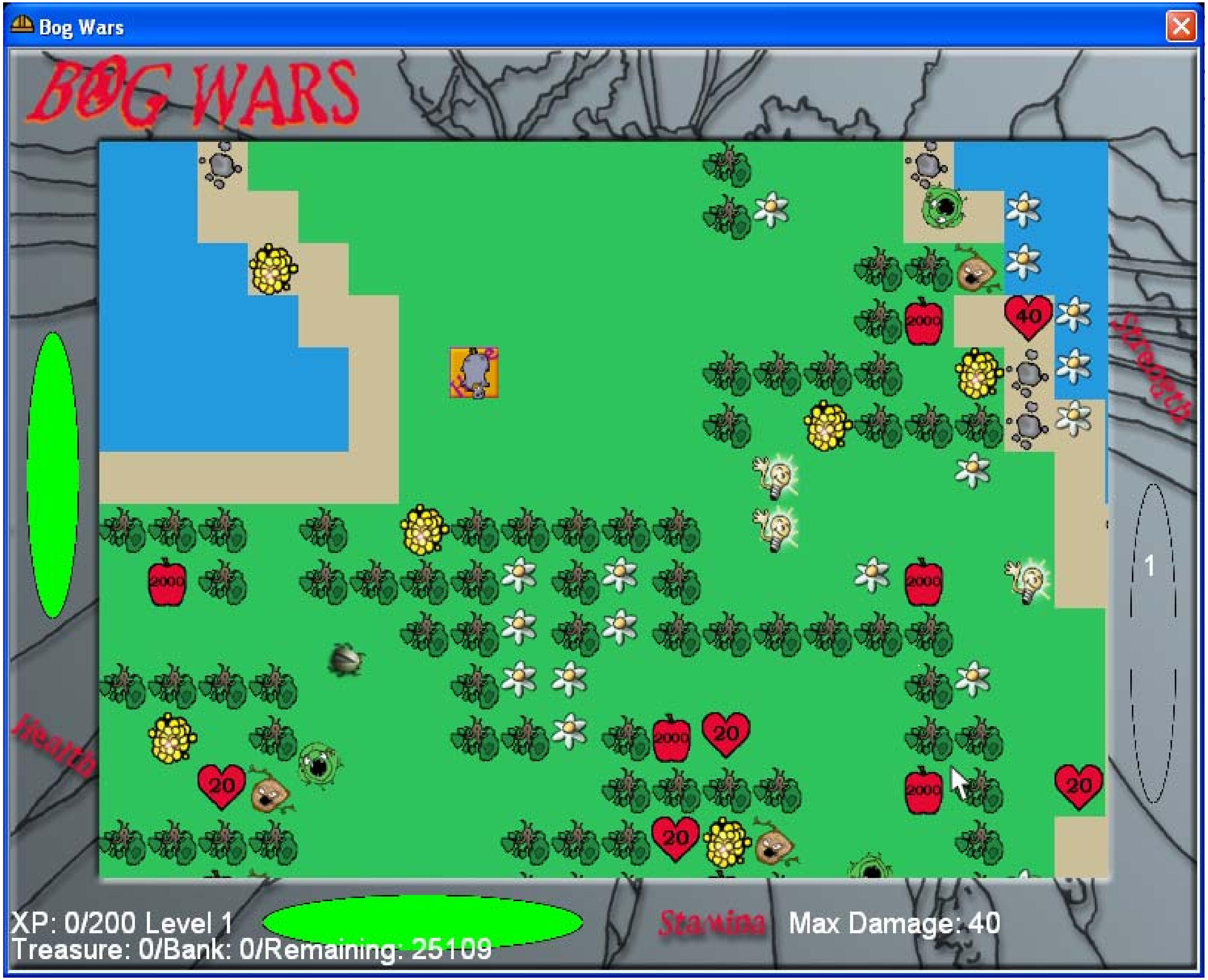
Alex Kerfoot

Overview

- Case study by Matt Gilgenbach, Mike Rodehorst, and Myself
- Overview of Environment
  - BogWars by John Voight
- C-Side functionality
- Role of Soar
- SGIO

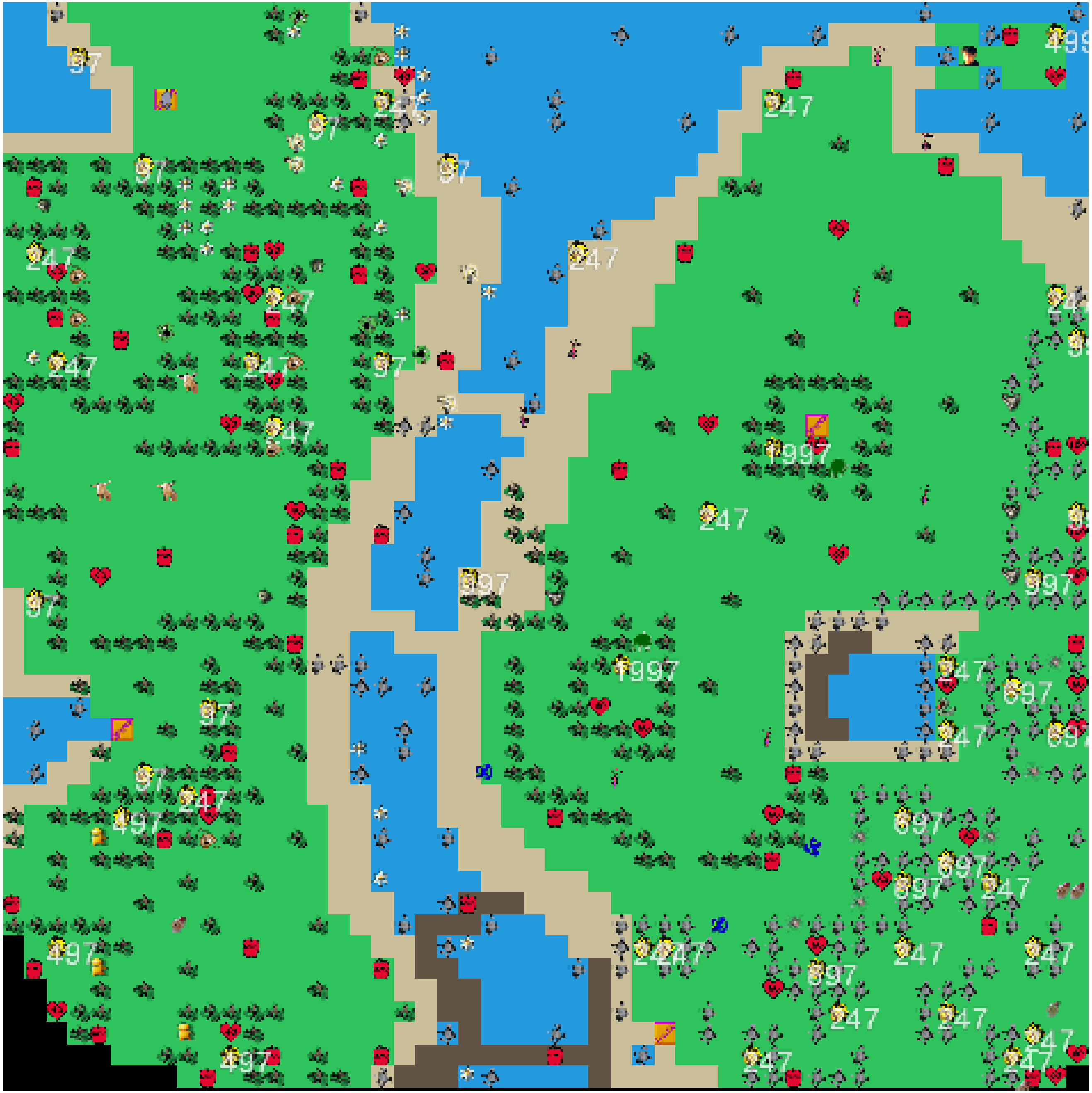
Environment: Bog Wars

- Map – Large compared to TankSoar
- Goal – Collect gold, bring to Home
- Obstacles
  - Tiles with different speeds
  - Enemies of varying strength
  - Gold decays over time
  - Gold lost forever if you die while carrying it



XP: 0/200 Level 1
Treasure: 0/Bank: 0/Remaining: 25109

Max Damage: 40


97
499
247
247
97
97
97
247
247
247
24
9
247
247
97
247
1997
247
997
997
97
1997
247
697
97
247
69
97
247
497
697
697
697
247
497
2247
247
247
697
247
247
47
497
247
247

Why High-Level?

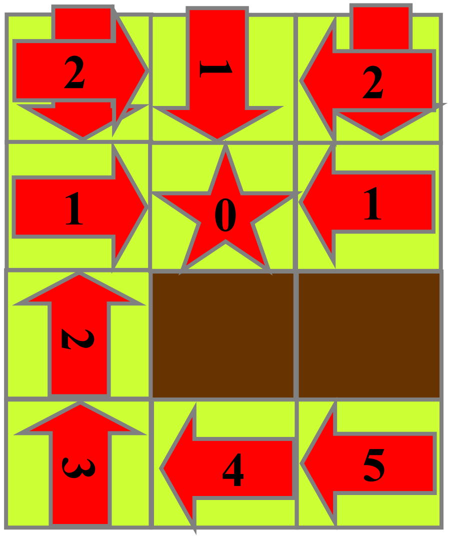
- Large, complex environment
  - 50 x 50 Map or larger
  - Non-discrete positions and directions
- Soar not designed for heavy computation
  - Mapping - 2500 tiles = lots of WME’s
  - Path-finding
- Simplify Soar rules

C Side Implementation

- Paths to Gold
- Paths to Home
- Attacking Enemies
- Avoiding Enemies

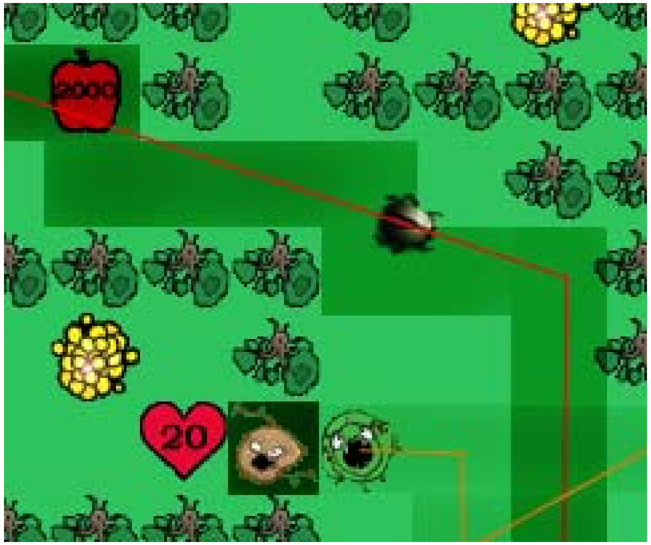
Paths

- Pre-computed UCS to and from each home base
- Return path weighted to avoid enemies
- Quickly computes and smoothes any path to or from any home base at runtime
- Interruptible (can be resumed)
  - Avoid enemies or get power ups
- Restorable (can start over)
  - If you die before you reach the goal



Monster Map

- Generated by observing enemy movements
- Monster Behavior
  - Stationary
  - Random
  - Pursuant



- Weight UCS map for heading to home
  - Enemy behavior and level
  - Tile speed

Attacking and Avoiding

- Attack
  - If at home, take path from home to enemy
  - Otherwise, use repeated application of A* towards enemy's location
  - If still not in the same tile, then A* towards their current tile
  - If in the same tile, then repeatedly charge
- Avoid
  - Head in direction directly away from enemy
  - Simple, but worked quite well

Soar Input

- ^agent
- ^objects

    ^monsters
    ^homes
    ^power-ups

- ^paths

    ^type {gold, home}
    ^monsters

- ^commands

Soar Output

- ^take-path

    ^path-id

- ^go-to-home

    ^home-id

- ^attack-enemy

    ^enemy-id

- ^avoid-enemy

    ^enemy-id

- ^sit-still

- ^cancel

    ^command-id

Major Soar Operators

- Hunt: Go get gold or experience
  - Holding no gold
  - There exist gold paths from your home
- Return-Gold: Come home with gold
  - When you are holding gold
- Change-Home: Go to a new home
  - Holding no gold
  - There are no gold paths from your home

Hunt Operator

- Get-Gold: Follow a path to gold
  - There exist “safe” gold paths from your home
  - Choose “safe” path of least cost
  - Must start from home; if elsewhere, come home
- Level-Up: Attack monsters for experience
  - There are no “safe” gold paths from your home
  - Choose the closest monster of appropriate level

SGIO: The Interface

- Bog Wars ↔ Soar Kernel
- Environmental data sent to Soar (i-link)
  - Only modified by C-side
- Decisions sent back to Bog Wars (o-link)
  - Only modified by Soar

SGIO Input/Output

- Keep a copy of input-link on C-side
- Important data updated every game cycle
- Add new WME's as needed
- Remove WME of nonexistent data
- Commit changes to Soar
- Soar runs until output
- C-side polls for commands on o-link

SGIO Output Complications

- Latent actions take more than one cycle to complete
- Keep command list on C-side
- SGIO commands act like push/pop
- C-side command list on i-link
- Carry out, complete, and cancel commands, as necessary

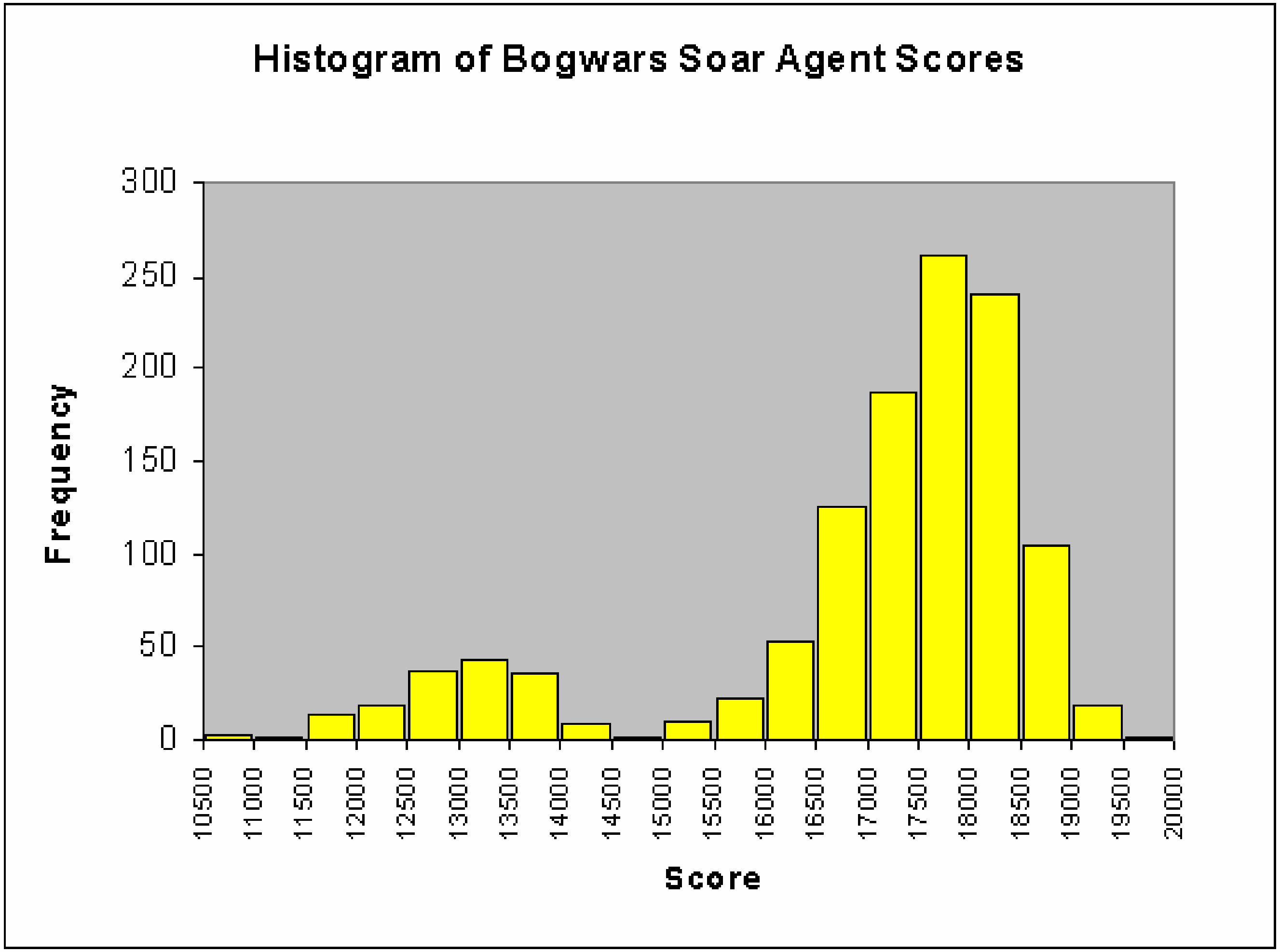
Alternate Output Method

- Keep commands on o-link
  - ^Status Accepted
  - ^Status Processing
  - ^Status Cancelled
  - ^Cancel True
- Problems
  - SGIO only sees new commands
  - Would require substantial changes of SGIO

Results

- 41 Rules
- 15 Operators
- Performed 1190 runs total on 3 computers
- Gold Score
  - Mean: 17007, Max: 19726, Min: 10659
- Bog Wars website
  - http://winter.eecs.umich.edu/eecs-498-1/

Gold Score Histogram



Nuggets and Coal

- Well-defined, predictable behavior
- Soar code is much simpler
- Easy to change/expand
- Works well with larger, more complex environments

- Soar has less power
- Latent commands require more effort
- Non-adaptive (not suited for chunking)
- Precomputation specific to map

Any Questions?