The CIG 2007 Car Racing Competition

Julian Togelius
and Simon M. Lucas
University of Essex

The aim of the competition

- Provide a way of comparing controller creation methods (evo + RL + others)
- Depth, yet smooth learning curve
- Make it fun and engaging
- Build on research we've already done
- Make it simple to enter
- Make it simple to enter, seriously!

What we came up with

- The point-to-point car racing task with one or two cars simultaneously
- A cross-platform Java package with no external dependencies
  - “Clean” Java interface
  - Simple TCP/IP interface for people who loathe Java
- Example code for controllers, networks, EAs…

What it looks like

How much skill is needed?

1. Maneuvering the car to the current wp
2. Doing it fast (skidding, orbiting)
3. Anticipating how to reach the next wp (angle and speed of wp intersection)
4. Predicting whether the competitor will reach the current wp first
    1. Go for the next wp (slow down enough)
    2. Block the competitor
5. Avoid being blocked by the competitor

Etc…

Interface

- Information about the state of the game is fed to the controller…
  - Either first- or third-person
- …and the controller outputs driving commands
  - bang-bang control, 9 commands in all
- 20 hz in simulated time
- Either via a Java interface or over TCP/IP

Interface: sensor model

- First-person sensors:
  - Speed
  - Angle and distance to current wp
  - Angle and distance to next wp
  - Angle and distance to competing car
- Third person sensors:
  - x and y positions for current and next wp
  - x, y, rotation for own car and other car
  - Linear and rotational velocities for both cars
  - Presence of other car (boolean)

Rules

- Basic score: number of wp reached / time
- Overall winner: reaches more wp than runner-up *in direct competition*
- Any kind of software allowed…
- …but must connect to unmodified simplerace package via Java or TCP
- No more than 50 ms per time step allowed

Computing the competition score

- When computing the competition score of a controller, it is tested for 500 trials in each of the following conditions:
  - On its own
  - Against a weak heuristic controller
  - Against an average evolved MLP controller
- Competition score is the average of all this
- …but this score does not necessarily reflect how well a controller performs against good competitors!

Did anyone notice?

- Six contestants including the author
- A fair bit of media coverage (Slashdot, New Scientist, various blogs)
  - …but only just before the deadline!
- A number of people declared their interest, but didn’t submit in time

The contestants

- Julian Togelius, Uni Essex
- Hugo Marques, Uni Essex
- Matthew Simmerson, Ipswich
- Aravind Gowrisankar, UT Austin
- Thomas Haferlach, Uni Edinburgh
- Pete Burrow, Cambridge

Julian Togelius

- Hard-coded heuristic controller
  - Turns left or right when to point straight to the next wp
  - Accelerate if driving slower than a set speed, otherwise decelerate
    - Without braking, it could “orbit” waypoints
- MLP-based controller
  - 8 inputs (first-person), 6 hidden, 2 outputs
  - Evolved for 200 gens with 50+50 ES

Matthew Simmerson

- Controller based on Ken Stanley's NEAT (NeuroEvolution of Augmenting Topologies)
- Using his own implementation, NEAT4J

Aravind Gowrisankar

- Used another NEAT implementation, jNeat.
- By far the slowest-evaluating controllers…
- Pop size 200, 500 generations
- Uses all the first person sensors
- Fitness mean of solo fitness and against simple neural net
- Still working on it, apparently…

Thomas Haferlach

- Controller based on CTRNNs (Continuous-Time Recurrent Neural Networks)
- Modular structure: one network controlling steering, one controlling driving
- Evolved with a modified version of CoSyNE (Faustino Gomez)
- Pop size 100, 2000 gens (approx.)

Peter Burrow

- Modular controller based on two networks
- Upper layer: MLP for waypoint selection
  - Inputs: distance and angle to current and next wp, and quotient between both car's distances to current wp, and between speeds
  - Output: binary wp choice
- Wp choice selects inputs to lower layer
- Lower layer: Elman net for driving
  - Inputs: distance, angle to selected wp, speed
- Incrementally evolved with standard ES

Hugo Marques + Julian Togelius

- Controller based on internal modelling of itself and the opponent
- Every 10 time steps, the next 100 time steps is internally simulated to see who will reach the current wp first
- Target wp selected accordingly
- Own controller used as model of opponent behaviour (could be learned on-line)
- Promising, but needs much more work

The results…?

Ranking table at deadline (March 26)

- Peter Burrow 18.2
- Thomas Haferlach 16.9
- Marques and Togelius 15.3
- Matthew Simmerson 13.5
- Aravind Gowrisankar 13.4
- Julian Togelius 11.0, 9.4

- But these are just competition scores…

The results…?

- However, in this case the competition score also provided a strict hierarchy in car-to-car races
- Examples

The winner

Peter Burrow

General wisdom

- The importance of modularity
  - Though not obvious which sort
- …and of input representation
- Familiarity of the experimenter with the method important (NEAT not winning)
- The competition score a good predictor…
  - …or the controllers not good enough?
- Room for improvement

The future

- CEC 2007 / FUZZ-IEEE 2007
  - Details not final
  - New entries already made
  - Possible tradeoff: complexity versus describability and robustness of dynamics and collisions (cf. pole balancing)
- Possibly making it into something more permanent
- Getting other communities involved (RL)
- Your input highly appreciated