Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI

Chin Hiong Tan, Kay Chen Tan, and Arthur Tay

Presented by Nick Brusso
11/28/2012

In IEEE Transactions on Computational Intelligence and AI in Games, Volume 3 Issue 4 (2011), pp. 289-301
Objective

- Create an entertaining game AI
  - Satisfying to play against for a wide audience
  - High replay value

- Adapt game AI as the player plays
  - Dynamically scale difficulty in real-time
  - Two implementations are presented; I focus on the simpler one.
Game Environment

- Simulated car racing
  - Current is worth 1 point
  - Second is worth 0 points
  - After Current is passed, Second → Current, NewWaypoint → Second
- Objective: gain the most points in a set time.
- Cars can move outside window boundaries
  - Advantageous for the AI
Game Environment

- Control actions (on/off):
  - Accelerate, Decelerate, Left Turn, Right Turn, Neutral
  - A player would use the arrow keys
- Car physics:
  - Collisions between cars
  - Side skidding
AI Behavior Components

- **Driving Behavior**
  - Speed Regulator
  - Reversing
  - Direction Switching Compensation
  - Tight Angle Turning

- **Tactical Behavior**
  - Waypoint Prediction
  - Time Wasting
  - Blocking
Adaptive Controllers

● Satisfying gameplay experience
  ○ Over $n$ games, $|\text{Wins} - \text{Losses}|$ and $\text{Draws}$ minimized
  ○ $|p1\text{Score} - p2\text{Score}|$ minimized and $\max(p1\text{Score}, p2\text{Score})$ maximized

● Artificial Stupidity
  ○ Force the AI to make deliberate mistakes
    ■ Selectively activate/deactivate behavior components.
    ■ Requires that the AI is overdesigned (small window for the player in this case)
Adaptive Uni-Chromosome Controller (AUC)

- Stores one chromosome which encodes seven real numbers (probabilities of activating each behavior)
  - Expected behavior set encoded by the chromosome represents a "winning" strategy
- Chromosome is initialized to random values when the game begins.
- Chromosome is updated whenever a waypoint is passed, and a new behavior set is selected using probabilities.
  - If we lost the previous waypoint, probabilities are used as-is
  - If we won, probabilities are complemented before selection

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
<td>0.9</td>
<td>0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Adaptive Uni-Chromosome Controller (AUC)

- **AUC Update Algorithm**
  - $\text{win}_i$: probability that behavior $i$ is activated in the next phase.
  - $\text{myDist}$, $\text{otherDist}$: distances from each car to the waypoint.
  - $\text{sgn}($behavior$_i)$: 1 if activated, -1 if not activated
  - $l$ and $m$: learning and mutation rates ($l = 0.1$, $m$ is unused)

1) If AUC win
   for each behavior component ($i = 1$ to $7$)
   if $(\text{rand}() < \text{myDist}/(\text{myDist} + \text{otherDist}))$
   $\text{win}_i = (\text{win}_i + \text{sgn}($behavior$_i$$) \times l) \times m$;

2) If AUC lose
   for each behavior component ($i = 1$ to $7$)
   if $(\text{rand}() < \text{otherDist}/(\text{myDist} + \text{otherDist}))$
   $\text{win}_i = (\text{win}_i - \text{sgn}($behavior$_i$$) \times l) \times m$;

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.8</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
<td>0.9</td>
<td>0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Testing

The full controller (all behaviors enabled) and AUC were both tested against five static controllers:

- **Heuristic Controller (HC)**
  - Uses simple rules to collect as many waypoints as possible; ignores opponent

- **Neural Network Controller (NNC)**
  - 9 Inputs: own orientation, opponent orientation, own speed, angle to current, angle to second, distance to current, distance to second, angle to opponent, distance to opponent
  - 2 Outputs: steering control, driving control

- **Reverse Enabled Controller (RC)**
  - Behavior controller with only reversing and direction switching behaviors active (subset of full controller)
  - Constant speed used instead of speed regulator behavior
Testing

The full controller (all behaviors enabled) and AUC were both tested against five static controllers:

● Predictive Slow Controller (PSC)
  ○ Same as the Heuristic Controller with the Waypoint Prediction behavior activated
  ○ Slow constant speed used (5px per time step); This prevents skidding and overshooting the waypoint

● Predictive Fast Controller (PFC)
  ○ Same as PSC, with a speed of 8px per time step.
  ○ Reaches the waypoint faster, but might overshoot
Results (Full Controller)

The graphs show the performance of different controllers in two scenarios:

1. **Full Controller**: This plot displays the box plots for various controllers, including Heuristic, NeuralNetwork, Reverse, PredictiveSlow, and PredictiveFast. The horizontal axis represents the type of controller, while the vertical axis shows the score range.

2. **Full Controller (Win-Lose-Draw)**: This bar chart illustrates the percentage distribution for the same controllers but with a focus on win, lose, and draw outcomes. The vertical axis represents the percentage, and the horizontal axis lists the types of controllers.
Results (AUC)
Conclusions

● AUC performed well in creating an entertaining experience.
  ○ Achieved a score difference of <= 4 for at least 70.22% of games played.
  ○ Wins/losses were well-distributed

● Deals well with a variety of opponents