General Video Game AI Tutorial

www.gvgai.net

QMULGRADFEST

Raluca D. Gaina
19 February 2018
Who am I?

Raluca D. Gaina

2nd year PhD Student
Intelligent Games and Games Intelligence (IGGI)

PhD Topic: Rolling Horizon Evolutionary Algorithms in General Video Game Playing

Why am I here doing this?

My supervisor forced me

GVGAI is cool!

The new QMUL Games AI group is cool too!

r.d.gaina@qmul.ac.uk
rdgain.github.io
@b_gum22
<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:00 - 11:15</td>
<td>Arrival &amp; coffee</td>
</tr>
<tr>
<td>11:15 - 12:00</td>
<td>Introductory talk</td>
</tr>
<tr>
<td>12:00 - 12:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>12:30 - 14:30</td>
<td>Practical</td>
</tr>
<tr>
<td>14:30 - 15:00</td>
<td>Final evaluation private league</td>
</tr>
<tr>
<td>15:00</td>
<td>Winners announced</td>
</tr>
</tbody>
</table>
Outline

• Introduction - General Game Playing
• General Video Game AI Research
• GVGAI Framework and Competition
• Sample agents
• Guidelines for practical
• Prizes!
Research on games

- Pure
- AlphaGO
- GO | AlphaGO
- Visual Doom | IntelAct, '16 Full Deathmatch winner
- Starcraft II

Raluca D. Gaina | GVGAI tutorial | 19.02.2018 | QMUL | gvgai.net | rdgain.github.io
General game playing

First AIII GGP Competition, by the Stanford University Logic Group 2005

http://ggp.stanford.edu
http://tiltyard.ggp.org/


Peter Keevash and Liana Yepremyan, “Rainbow matchings in properly-coloured multigraphs” [1]
Levine et al. propose the creation of a new benchmark for GVGP [Levine et al., 2013]

Compliments ALE in two ways:
- Creation of games in a more general framework.
- No screen capture analysis needed, information via encapsulated objects.

Video Game Description Language (VGDL) [Tom Schaul, 2013]
- Benchmark for learning and planning problems.
- Base for the GVGAI Framework

Arcade Learning Environment (ALE), Evaluation of AI agents in 55 games of the Atari 2600 Collection

One agent plays all (real-time, arcade) games
Believable Characters in Video Game AI

• Believability Assessment

• Believable Agents
General Video Game AI

- GVGAI -> A competition for game-playing agents, level and rule generators
- Agents are tested on unseen videogames
- 160+ games
- Games are implemented in the Video Game Description Language

Real time -> 40ms as budget time (per action)
- If returns action in 40-50ms
  - NIL is applied
- If returns action after 50ms
  - disqualification and game loss

At most 51.2% wins for the best algorithm
- knowledge transfer?
- game identification?
How GVGAI differs...

From General Game Playing:
• We do videogames! Real-time constraints!
• We don’t make the ruleset available
How GVGAI differs...

From the Arcade Learning Environment (ALE):
• Structured API (information via Java objects)
• Potentially infinite supply of games
• Agents tested on unseen games
GVGAI for game design

- Use AI agents to evolve game parameters

Evolved Game (20-5) Sample

Evolved Game (5-20) Sample
The GVGAI Competition – www.gvgai.net

GVGAI 2016 Competition - Level Generation Track:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Username</th>
<th>Country</th>
<th>Percentage Preferred (%)</th>
<th>Description</th>
<th>Generator</th>
<th>Screenshots</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>easblade</td>
<td>New Zealand</td>
<td>36</td>
<td>Description</td>
<td>Download</td>
<td>easblade</td>
</tr>
<tr>
<td>2</td>
<td>bhorn82287</td>
<td>Afghanistan</td>
<td>12</td>
<td>Description</td>
<td>Download</td>
<td>bhorn82287</td>
</tr>
<tr>
<td>2</td>
<td>Jnicho</td>
<td>United Kingdom</td>
<td>12</td>
<td>Description</td>
<td>Download</td>
<td>Jnicho</td>
</tr>
<tr>
<td>4</td>
<td>Number13</td>
<td>Germany</td>
<td>8</td>
<td>Description</td>
<td>Download</td>
<td>Number13</td>
</tr>
</tbody>
</table>

GVGAI for level & rule generation
GVGAI for learning agents

• No forward model simulations, learn to play from experience
• Game state observation
  • JSON game objects
  • Screen capture (PNG file)
• Java or Python
• 5 minutes for training per game in first 3 levels
  • 10 runs per game in 2 new levels -> validation
• Winner 2017: Q-learning (but barely better than random...)
Most successful approaches so far:

- Monte Carlo Tree Search
- Evolutionary Algorithms
- Hybrids
- Breadth-first search in deterministic games
- Value maps
- Interesting target identification
RHEA in GVGAI

- Rolling Horizon Evolutionary Algorithm [2][3][4]
- Individuals as sequences of actions
- Great in puzzle games
- Not so great in quick reaction games
- Can be enhanced in various ways
  - Parameter optimization
  - Initialization methods
  - Shift buffer
  - Statistical tree
  - Monte Carlo rollouts
Evolving Game Playing Agents

As humans evolved immersed in the environment and the got better at surviving

It is possible to evolve game-playing AI Agents that improve their performance throughout generations
initial formula

evolution

improved formula
Beyond playing to win [5]

Winning Maximization

Exploration Maximization

Knowledge Discovery

Knowledge Estimation
Deceptive games [6]
GVGAI environments

• Single vs two-player
• Full vs partially observable
• Puzzles vs racing games
• Action/adventure games
• Collecting, shooting, navigation
• Deceptive games
• Competitive vs cooperative
GVGAI gameplay analysis

- Record more data about agent processing and game events
- Adjust strategy dynamically based on gameplay analysis
GVGAI Framework and competition

—— www.gvgai.net ——

https://github.com/GAIGResearch/GVGAI

https://github.com/GAIGResearch/GVGAI/wiki
## Game description

### SpriteSet
- **floor** > **Immovable**\(\text{img=newset/floor6}\) \(\text{hidden=True}\)
- **human** > **annoyed** > **RandomNPC**\(\text{speed=0.25}\)\(\text{img=newset/cursedman}\)\(\text{cons=2}\)
- **citizen** > **quiet** > **RandomNPC**\(\text{speed=0.25}\)\(\text{img=newset/man2}\)\(\text{cons=1}\)
- **avatar** > **ShootAvatar**\(\text{stype=cigarette}\)\(\text{img=newset/girl1}\)\(\text{rotateInPlace=False}\)
- **george** > **Chaser**\(\text{stype=citizen}\)\(\text{speed=0.25}\)\(\text{img=newset/man4}\)\(\text{frameRate=8}\)
- **cigarette** > **Flicker**\(\text{limit=5}\)\(\text{singleton=True}\)\(\text{img=newset/cigarette}\)
- **wall** > **Immovable**\(\text{img=oryx/wall6}\)

### TerminationSet
- **SpriteCounter**\(\text{stype=avatar}\)\(\text{win=False}\)
- **SpriteCounter**\(\text{stype=quiet}\)\(\text{win=False}\)
- **Timeout**\(\text{limit=1000}\)\(\text{win=True}\)

### InteractionSet
- **quiet george** > **transformTo**\(\text{stype=annoyed}\)
- **avatar george** > **killSprite**\(\text{scoreChange=-1}\)
- **annoyed cigarette** > **transformTo**\(\text{stype=quiet}\)\(\text{scoreChange=1}\)
- **human wall wall** > **stepBack**

### LevelMapping
- \(g > \text{floor george}\)
- \(c > \text{floor quiet}\)
- \(A > \text{floor avatar}\)
- \(> \text{floor}\)
public class Agent extends AbstractPlayer{

/**
 * constructor called once at the beginning of each game.
 * @param stateObs Observation of the current state.
 * @param elapsedTimer Timer when the action returned is due.
 */
public Agent(StateObservation stateObs, ElapsedCpuTimer elapsedTimer){
}

/**
 * act method called at every game tick.
 * @param stateObs Observation of the current state.
 * @param elapsedTimer Timer when the action returned is due.
 * @return ACTION_NIL all the time
 */
@Override
public ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer) {
    return Types.ACTIONS.ACTION_NIL;
}
}
StateObservation objects

Available in all agent methods: **constructor, act, result**

```java
ArrayList<Types.ACTIONS> getAvailableActions()
int getNoPlayers()
double getGameScore()
int getGameTick()
Types.WINNER getGameWinner()
boolean isGameOver()
Dimension getWorldDimension()
int getBlockSize()

ArrayList<Observation>[] getObservationGrid()
TreeSet<Event> getEventsHistory()
ArrayList<Observation>[] getNPCPositions()
ArrayList<Observation>[] getImmovablePositions()
ArrayList<Observation>[] getMovablePositions()
ArrayList<Observation>[] getResourcesPositions()
ArrayList<Observation>[] getPortalsPositions()
ArrayList<Observation>[] getFromAvatarSpritesPositions()
```
StateObservation objects

Available in all agent methods: constructor, act, result

- Vector2d getAvatarPosition()
- double getAvatarSpeed()
- Vector2d getAvatarOrientation()
- HashMap<Integer, Integer> getAvatarResources()
- Types.ACTIONS getAvatarLastAction()
- int getAvatarType()
- int getAvatarHealthPoints()
- int getAvatarMaxHealthPoints()
- int getAvatarLimitHealthPoints()
- boolean isAvatarAlive()

void advance(Types.ACTIONS action)
StateObservation copy()
Simulate possible next game states.

```c
void advance(Types.ACTIONS action)
```

- **UP**
  - Safe
  - Noisy game state
  - Evil NPC
  - Player killed
Testing a 1P planning agent

1. Find class src.tracks.singlePlayer.Test.java
2. Set gameIdx and levelIdx variables
3. Create a String variable containing the path to your Agent class, such as:
   • String sampleMCTSSController = "tracks.singlePlayer.advanced.sampleMCTS.Agent";
4. Select running mode:
   • Play as human
   • Play one game with a controller (pass your agent String to the method)
   • Replay a recorded game
   • Play N games in L levels repeated M times (visuals off)
5. Run Test.java!
Sample agents in the framework

In packages `src.tracks.singlePlayer.simple` and `src.tracks.singlePlayer.advanced`

- Do Nothing
- Random
- Simple One Step Look Ahead
- Monte Carlo Tree Search
- Rolling Horizon Evolutionary Algorithm
- Random Search
Monte Carlo Tree Search

In class sampleMCTS.SingleTreeNode

\[ \text{mctsSearch} \rightarrow \text{mostVisitedAction} \]

\[ \text{treePolicy} \rightarrow \text{uct} \]
\[ \text{expand} \]
\[ \text{rollOut} \rightarrow \text{value} \]
\[ \text{backUp} \]

Repeat while time budget
Rolling Horizon Evolutionary Algorithm

Population

Individual 0

Elitism

Individual 1

Individual 2

Individual x0

Individual xn

Mutation

[0, N. ACTIONS -1]

Evaluation

State

H

State value = fitness

FM

Next Population

Individual x1

Individual x0

Individual x0

Individual 0

Crossover

Individual 0

Individual x0

Individual 0
Rolling Horizon Evolutionary Algorithm

- init_pop
- runIteration
- crossover
- mutate
- evaluate
- sort

get_best_action

Repeat while time budget
QMUL GVGAI tutorial private league!

Best AI Agent

£20 Voucher

Best Game

£20 Voucher
QMUL GVGAI tutorial private league!

Best AI Agent

- Register on the GVGAI website
- Submit a **functioning 1P planning agent** to the QMUL private league before 14:30
- Agent with most F1 points according to GVGAI ranking system wins!

£20 Voucher

Best Game

- Make a game in VGDL which compiles (either 1P or 2P)
- Send your game and level (1 level is enough) to me via email before 14:30
- Game with highest skill depth wins!

£20 Voucher
1. Sign up on gvgai.net to be able to submit your agent
2. Shout when done to collect usernames
3. Log in -> Private Leagues top menu tab -> QMUL

Training Set
- 10 games
- 5 levels/game
- Public ✓
- Website ✓
  - Local

Validation Set
- 10 games
- 5 levels/game
- Private ×
- Website ✓
  - Online

Test Set
- 10 games
- 5 levels/game
- Private ×
- No submissions ×
  - Final results
## GVGAI ranking system

### Bubble:

<table>
<thead>
<tr>
<th>Game</th>
<th>Username</th>
<th>Country</th>
<th>Points</th>
<th>Winner %</th>
<th>Avg. Score</th>
<th>Avg. Timesteps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble</td>
<td>Cyclus</td>
<td>Germany 🇩🇪</td>
<td>25</td>
<td>100</td>
<td>18 (3.54)</td>
<td>566.6 (157.68)</td>
</tr>
<tr>
<td>Bubble</td>
<td>Jaybot</td>
<td>Germany 🇩🇪</td>
<td>18</td>
<td>100</td>
<td>16.6 (3.26)</td>
<td>357 (78.45)</td>
</tr>
<tr>
<td>Bubble</td>
<td>sampleOLMCTS</td>
<td>United Kingdom 🇬🇧</td>
<td>15</td>
<td>60</td>
<td>14.8 (2.86)</td>
<td>906.2 (175.25)</td>
</tr>
</tbody>
</table>

### Total:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Username</th>
<th>Country</th>
<th>Description</th>
<th>G-1</th>
<th>G-2</th>
<th>G-3</th>
<th>G-4</th>
<th>G-5</th>
<th>G-6</th>
<th>G-7</th>
<th>G-8</th>
<th>G-9</th>
<th>G-10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>asd592</td>
<td>Germany 🇩🇪</td>
<td>Description</td>
<td>1</td>
<td>12</td>
<td>15</td>
<td>6</td>
<td>18</td>
<td>0</td>
<td>10</td>
<td>8</td>
<td>25</td>
<td>25</td>
<td>120</td>
</tr>
<tr>
<td>2</td>
<td>Jaybot</td>
<td>Germany 🇩🇪</td>
<td>Description</td>
<td>2</td>
<td>10</td>
<td>25</td>
<td>18</td>
<td>8</td>
<td>1</td>
<td>12</td>
<td>18</td>
<td>8</td>
<td>12</td>
<td>114</td>
</tr>
<tr>
<td>3</td>
<td>JACAM</td>
<td>Germany 🇩🇪</td>
<td>Description</td>
<td>25</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>8</td>
<td>25</td>
<td>18</td>
<td>0</td>
<td>109</td>
</tr>
</tbody>
</table>
Opening GVGAI in Eclipse

1. Download the GVGAI framework
2. Unzip
3. Open Eclipse and create a new workspace in any other folder.
4. File -> New -> Project... -> New Java project
5. Untick ‘default location’ and select the GVGAI-master directory location -> Finish
6. In project view, navigate to src.tracks.singlePlayer -> Test.java class -> Run!
1. Download the GVGAI framework
2. Unzip
3. Open IntelliJ and open another project / create a new one (can be deleted after)
4. File -> New -> Project from Existing Sources ...
5. Overwrite project and module files when asked  
6. Import all libraries and modules by default (Next ... Next -> Finish)  
7. In project view, navigate to src -> tracks -> singlePlayer -> Test.java class -> Run!
Let’s get hands on

---

[www.gvgai.net](http://www.gvgai.net)

https://github.com/GAIGResearch/GVGAI

https://github.com/GAIGResearch/GVGAI/wiki
Tasks for 2nd part

Root

Make a **1P planning agent** for the private league

Make a **VGDL game** (if 1P, it can be used for the private league, to get the agents tested on it)

**Experiment** with other tracks, specific game sets or game design features
- Two player planning
- Single player learning
- Level generation
- Rule generation

Work either solo or small teams (3-4 max)
Where to start?!!

Oh no this sounds too difficult...

If you can’t program ...
... find someone who can!

If you’re making a game ...
... read the wiki information on VGDL!

If you’re making an agent ...
... start from a sample agent or previous successful entries

➢ Change parameters
➢ Change heuristic (state evaluation)
➢ Change algorithm logic (e.g. mutation operator for EAs; tree policy for MCTS)
➢ Mash algorithms together (MCTS as mutation operator in EA?)
But first...

- Lunch!
- Find potential work buddies
- Talk to QMUL Game AI group members


[Schaul, 2013] Schaul, T. A Video Game Description Language for Model-based or Interactive Learning, Proceedings of the IEEE Conference on Computational Intelligence in Games, 2013, 193-200


http://gameaibook.org/
Thank you!