University of Essex

iGGi

# Beyond Playing to Win:

## Diversifying Heuristics for GVGAI

Cristina Guerrero-Romero, Annie Louis and Diego Perez-Liebana

*Conference on Computational Intelligence and Games (CIG) (2017)*

Ultimate Goal

> Use of General Video Game (GVG) agents for evaluation.

> Create system to analyse levels and provide feedback.

> Pool of agents capable of understanding a level without having prior information about it.

First Step

> Diversifying Heuristics in General Video Game Artificial Intelligence (GVGAI).

# Motivation

What?

> JAVA based open source framework.

> Arcade-style 2D 1 or 2 player games.

> Games described in Video Game Description Language (VGDL).

> Used for the General Video Game Artificial Intelligence Competition (GVGAI).

```
BasicGame key_handler=Pulse square_size=40
    SpriteSet
        floor > Immovable img=newset/floor2
        hole   > Immovable color=DARKBLUE img=oryx/cspell4
        avatar > MovingAvatar img=oryx/knight1
        box    > Passive img=newset/block1 shrinkfactor=0.8
        wall > Immovable img=oryx/wall3 autotiling=True
    LevelMapping
        0 > floor hole
        1 > floor box
        w > floor wall
        A > floor avatar
        . > floor
    InteractionSet
        avatar wall > stepBack
        box avatar  > bounceForward
        box wall box  > undoAll
        box hole    > killSprite scoreChange=1
    TerminationSet
        SpriteCounter stype=box    limit=0 win=True
```

```
wwwwwwwwwwwww
w........w..w
w...1.......w
w...A.1.w.0ww
www.w1..wwwww
w.......w.0.w
w.1........ww
w..........ww
wwwwwwwwwwwww
```

Why?

> Tool for General Artificial Intelligence algorithms benchmarking.

> Sample agents available.

> 150+ games available.

> It would be possible to apply the idea to GVGP.

## GVGAI Framework

> 20 games from the GVGAI platform (10 deterministic, 10 stochastic).
> 5 controllers (OLETS, OLMCTS, OSLA, RHEA and RS).
> 4 heuristics (WMH, EMH, KDH and KEH).

> 1 level per game played 20 times for each 20 different configurations.

> By heuristic, agents ranked by performance for that heuristic criteria.
> F1 ranking system.

> Rankings comparison and analysis.

# Experimental setup

Sample controllers

> OLETS (*Open-Loop Expectimax Tree Search*)

Developed by *Adrien Couetoux* , winner of the 2014 GVGAI Competition.

> OLMCTS (*Open-Loop Monte-Carlo Tree Search*)

> OSLA (*One Step Look Ahead*)

> RHEA (*Rolling Horizon Evolutionary Algorithm*)

> RS (*Random Search*)

Common ground modifications

> Depth of the algorithms set to 10.

> Evaluation function isolated to be provided when instantiating the algorithm.

> Cumulative reward implemented.

# Controllers

> Heuristics define the way a state is evaluated

> 4 heuristics with different goals

- Winning
- Exploration
- Knowledge Discovery
- Knowledge Estimation

# Heuristics

**Goal**: To win the game

> Winning.

> Maximizing score.

> All sample agents original strategy.

```
if is EndfTheGame() and is Loser() then
          return H-
else if is EndOfTheGame() and is Winner() then
          return H+
return new score - game score
```

**Criteria**

1> Number of wins.

2> Higher average score.

3> Less time steps average.

| WMH Stats (overall games) | | |
|---|---|---|
| Controller | F-1 Points | Average % of Wins |
| OLETS | 449 | 59.00 (5.43) |
| RS | 356 | 51.00 (4.24) |
| OLMCTS | 333 | 41.50 (3.69) |
| OSLA | 283 | 34.00 (4.95) |
| RHEA | 224 | 10.00 (3.29) |

**Goal**: To maximize the exploration of the level

> Maximizing visited positions.

> Use of exploration matrix.

> Not visited/visited positions.

```
if is EndfTheGame() then
          return H−
else if is outOfBounds(pos) then
          return H−
if not hasBeenBefore(pos) then
          return H+/100
else if is SameAsCurrentPos(pos) then
          return H−/200
return H−/400
```

**Criteria**

1> Percentage of level explored.

2> Less time steps average to find last new position.

| EMH Stats (overall games) | | |
|---|---|---|
| Controller | F-1 Points | Average % Explored |
| RS | 428 | 74.94 (1.83) |
| OLETS | 377 | 76.86 (2.19) |
| OLMCTS | 309 | 65.60 (1.64) |
| OSLA | 282 | 54.14 (2.18) |
| RHEA | 204 | 27.56 (1.64) |

**Goal**: To interact with the game as much as possible, triggering sprite spawns and interactions

> Acknowledging the different elements.
> New interactions with the game.
> Curiosity: Interactions in new locations.

> Use of sprite knowledge database.
> Interaction table (*collision* & *action-onto*).

```
if is EndfTheGame() and is Loser() then
        return H−
else if is EndfTheGame() and is Winner() then
        return H−/2
else if is outfBounds(pos) then
        return H−
if newSpriteAck() then
        return H+
if eventOccured(lastTick) then
        if is newUniqueInteraction(event) then
                return H+/10
        else if is newCuriosityCollision(event) then
                return H+/200
        else if is newCuriosityAction(event) then
                return H+/400
return H−/400
```

# Knowledge Discovery (KDH)

**Criteria**

1> Sprites acknowledged.

2> Unique interactions achieved.

3> Curiosity discovered.

4> Last acknowledgement game tick.

5> Last unique interaction game tick.

6> Last curiosity discovery game tick.

| KDH Stats (overall games) | | | | | |
|---|---|---|---|---|---|
| Controller | F-1 Points | % Ack (Rel) | % Int (Rel) | % CC (Rel) | % CA (Rel) |
| RS | 414 | 100.00 | 96.18 | 85.46 | 87.42 |
| RHEA | 342 | 99.66 | 95.48 | 62.48 | 54.44 |
| OLMCTS | 330 | 99.79 | 93.53 | 84.75 | 84.06 |
| OLETS | 279 | 99.86 | 88.97 | 90.72 | 77.55 |
| OSLA | 235 | 98.48 | 84.99 | 56.37 | 51.75 |

**Goal**: To predict the outcome of interacting with sprites, changes in the victory status and in score

> Predicting the outcome of the interaction with each element.

> Acquiring knowledge: win condition & score change

> Interacting with the game uniformly.

> Use of sprite knowledge database.

> Interaction table (*collision* & *action-onto*).

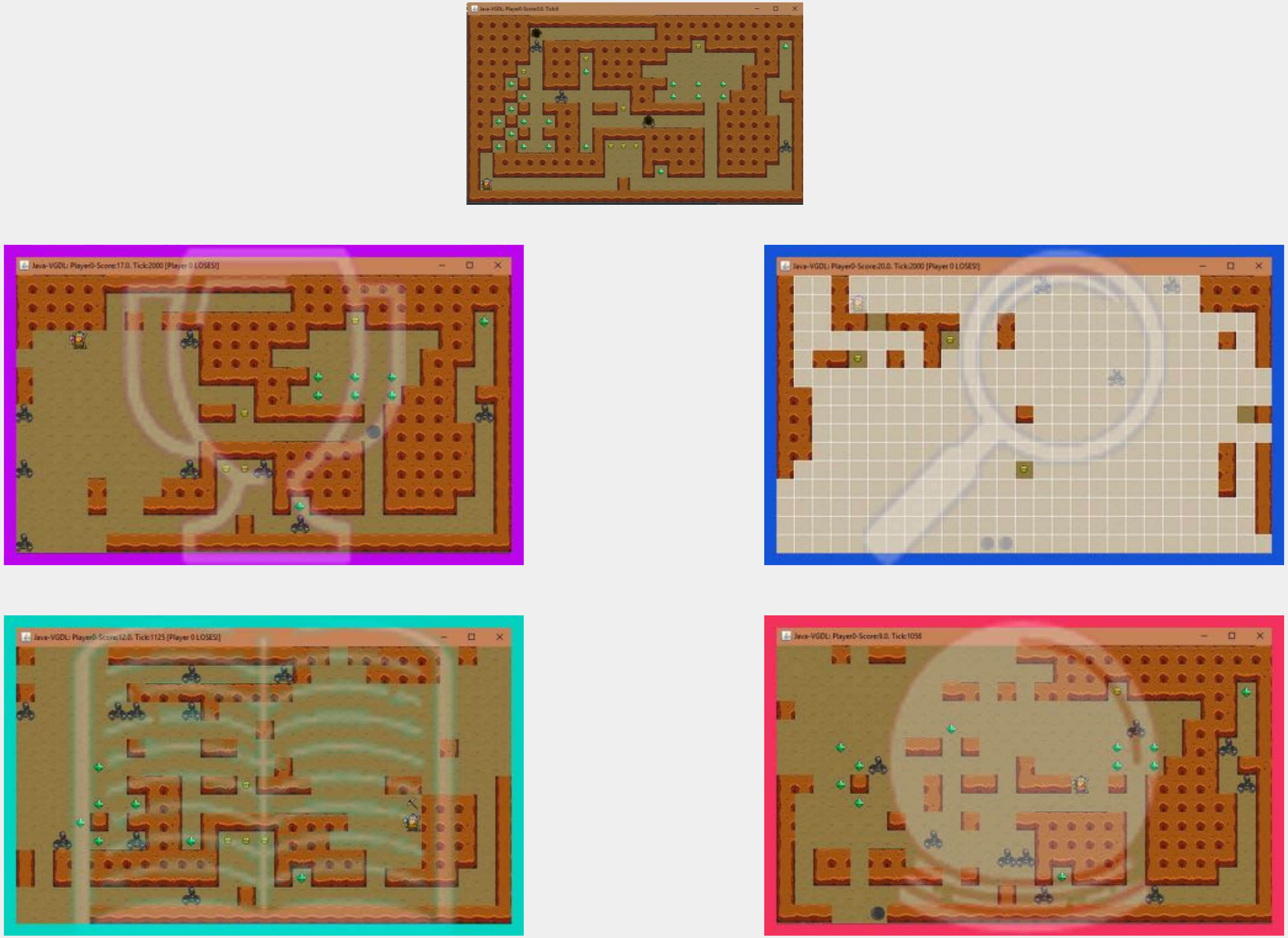
```
if is EndfTheGame() and is Loser() then
        return H–
else if is EndfTheGame() and is Winner() then
        return H–/2
else if is outfBounds(pos) then
        return H–
if newSpriteAck() then
        return H+
if eventOccured(lastTick) then
        if is newUniqueInteraction(events) then
                return H+/10
        return rewardForTheEvents(events) -> in [0; H+/100]
n_int = getTotalNStypeInteractions(int history)
if n_int == 0 then
        return 0
return H–/(200 × n_int) -> in [H-/200; 0]
```

**Criteria**

1> Smallest average for the prediction square error.

2> Number of interactions predicted.

| KEH Stats (overall games) | | | |
|---|---|---|---|
| Controller | F-1 Points | Avg Sq error average | % Int Estimated (Rel) |
| OLMCTS | 347 | 0.338 | 97.92 |
| RHEA | 330 | 0.505 | 97.50 |
| OSLA | 313 | 0.617 | 73.19 |
| RS | 310 | 0.528 | 98.33 |
| OLETS | 300 | 1.086 | 87.92 |



# Heuristics


https://www.youtube.com/watch?v=aLgPm9kbfY8

# Heuristics - Demo

| Rankings | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | WMH | | EMH | | KDH | | KEH | |
| 1 | 449 | OLETS | 428 | RS | 414 | RS | 347 | OLMCTS |
| 2 | 356 | RS | 377 | OLETS | 342 | RHEA | 330 | RHEA |
| 3 | 333 | OLMCTS | 309 | OLMCTS | 330 | OLMCTS | 313 | OSLA |
| 4 | 283 | OSLA | 282 | OSLA | 279 | OLETS | 310 | RS |
| 5 | 224 | RHEA | 204 | RHEA | 235 | OSLA | 300 | OLETS |

# Results

> First step in the possibility of enlarging GVGP techniques.

> Agent performance changes depending on the heuristic used.

> It is challenging and difficult to achieve different goals with a good performance for every game when it is generalized.

# Conclusions

> Heuristics improvement and enlargement.

> Heuristics combination.

> Repeat experiments using more levels.

> Apply idea to learning approaches (learn by repetition without forward model).

> Use GVGAI for evaluation, ultimately applied to PCG.

# Future work

# Thanks!

http://github.com/kisenshi

@kisenshi

## Questions?