Procedural Content Generation: Player Models

2019-11-04
OOB

• HW 7: SMB level generation
• Trajectory update
  – Intro: Game AI vs Academic AI; Graphs + Search
  – Taking action in a game:
    • Basic Physical Acts: Paths, Movement, Steering
    • Decisions: FSMs, D&B Trees, RBS, Plan, Meta {BB, Fuzzy}
  – Creating & Adapting content (sans NPC decisions):
    • PCG, Player Modeling
  – Adapting NPC decisions (& content) → Learning
  – Support Technologies; Design GAI; AI Based Games; Special topics (e.g. narrative); Revisits
PCG as Local Search

1. What “search” is happening? Do we seek a path to goal?
2. What is the state space? How many states do we save?
3. How memory efficient is this search?
4. Hill climbing: (PCG as parameter search part 1)
   1. L____ search
   2. What is the “landscape”?
   3. Need a function that maps p____ to f_____
5. GAs: (PCG as parameter search part 2)
   1. Good in ______ domains, where _D.K.__ is scarce or hard to encode
   2. Can also be used for ____ search
   3. Also needs a f______ function (maps c____ to f____)
6. Other local search techniques
   1. Gradient Descent
   2. Simulated annealing
   3. Local beam
   4. Tabu
   5. Ant Colony Optimization
PCG and GAs

1. Create a random set of $n$ chromosomes (population)
2. Assign a fitness score to each chromosome (fitness function)
3. Remove the $m\%$ ($m < 100$) worst chromosomes
4. Cycle through remaining pairs of chromosomes and cross-over (with some probability)
5. Randomly mutate (during?) cross-over (with some probability)
6. Reduce new population to size $n$
7. Repeat steps 2-6 until [stepwise improvement diminishes || one individual is fit enough || # generations reached]

• Tuning parameters
  – Population size
  – Number of generations
  – Fitness function
  – Representation
  – Mutation rate
  – Crossover operations
  – Selection procedure
  – Number of solutions to keep
Search-based approach to PCG

• Core components
  – Search algorithm
  – Content representation/encoding
    • defines (and thus also limits) what content can be generated, and determines whether effective search is possible
  – Evaluation function(s)
    • f(artifact) → number indicating quality

• Very small search space or lots of time: exhaustive search
• Easy to find good solutions and important to maintain high diversity: random search
• Otherwise: search-based optimization
On content representation: Locality v Dimensionality

- Representation should have high *locality*
  - a small change to the genotype should on average result in a small change to the phenotype and a small change to the fitness value
- Lack of locality means what?
- “curse of dimensionality”
  - the larger the search space, the harder it is (in general) to find a certain solution

- Same space of content phenotypes can be represented in several different ways in genotype space
- Continuum: Direct $\leftrightarrow$ Indirect
  - Direct, e.g. level map
  - List of positions and properties of game entities
  - Repo of reusable patterns and how they are distributed
  - List of properties, e.g. #gaps, #enemies
  - Random seed
Three class of evaluation functions

- **Direct**: base fitness calculations directly on phenotype representation of the content
  - Theory-driven: guided by intuition and/or qualitative theories of player experience
  - Data-driven: based on quantitative measures of player experience

- **Simulation-based**: use AI agents to play content and estimate quality
  - Usually calculate statistics about the agents’ behavior/playing style to score game content

- **Interactive**: evaluate content based on interaction with a human
  - Explicit / reported preference
  - Implicit / inferred preference: e.g. based on how often and how long the player uses weapon
Creating computational models of a player’s behavior, preferences, or skills is called player modelling. With a model of the player, we can create algorithms that create content specifically tailored to that player.

PLAYER MODELING

Player Modeling

A set of approaches that represent/quantify the player’s experience or preferences with the goal of improving future experiences. “experience-driven procedural content generation”

Improving the experience either:

1. Within the game, altering game elements or deciding between elements (picking which Mario Kart item)
2. Outside the game, giving feedback to designers/developers (analytics)
We can separate player modeling approaches into two more general groups:

– **Model-based**: Roughly corresponds to player modeling within games. It presupposes we have a set of categories or value range to assign to some part of a player’s experience.

– **Model free**: Roughly corresponds to player modeling outside of games. Looks for patterns of behavior without pre-existing hypothesis, drawing on statistical/machine learning techniques.
Model-based: 4 Major Models

• **Note:** This is not to say that there exists only four models, but that vast majority can be understood as derivatives of these four

1. Csikszentmihalyi's Flow (Position in space)
2. Bartle’s player types (Categories of play style)
3. Elo’s performance rating (Continuous value)
4. Drama Management (Difference between target value and .)
Model free Approaches

• Start with some data about a playerbase and game world

• **Categorize** learn what types exist

• **Regression analysis** how does x relate to y?
  – X: Input about the playerbase/game world
  – Y: Something we want to predict/understand

• **Classify** which y is x a member of?
MODEL-BASED PCG
Player Modeling

• What is optimal in PCG?
• Need a player model
  – Tells you something about the player
  – Makes predictions about player
• Challenges in player modeling:
  – What do you model?
  – Where do you get the model?
  – How do you use the model?
Player Model

• What do we want to model?
What do we want to model?

• Demographics
• Game play traces
• Stats
• Features of world when actions are performed
  – Eg: more likely to jump when X is true
• Sensors
• Preferences
  – Ask before, during, after → ratings
• Personality
Example Statistics

CAT 1: (FPS – L4D)
- Ave player health
- Shot accuracy
- # times damaged
- # times killed

CAT 2: (platformer -- mario)
- Power up blocks collected
- # deaths
- # deaths from gaps
- # times ducked
- Time between deaths
- Con blocks pressed
- # gaps jumped
- Avg width of gaps jumped

CAT 3: (rpg)
- Unique area visits
- Time spent exploring (straying)
- Win/loss ratio
- Time to complete level
- Items found
- Item usage
Player Model

• What do we want to model?
• Where do we get the model?
Where do we get the model?

• Observe player
Where do we get the model?

- Observe player
- Questionnaire
Where do we get the model?

- Observe player
- Questionnaire
- Social media?
Player Model

• What do we want to model?
• Where do we get the model?
• How do we use the model?
How do we use the model?

• Must be input to algorithm
How do we use the model?

- Must be input to algorithm
- Must actually make a difference
How do we use the model?

• Must be input to algorithm
• Must actually make a difference
• Machine Learning
  – Learn to make decisions based on observed data
  – Optimization seeks best mapping
    • Genetic algorithms
    • Simulated annealing
Model-based: 4 Major Models

- **Note:** This is not to say that there exists only four models, but that vast majority can be understood as derivatives of these four.

1. Csikszentmihalyi's Flow (Position in space)
2. Bartle etc's player types (Categories of play style)
3. Elo’s performance rating (Continuous value)
4. Drama Management (Difference between target value and .)
FLOW
Input: Skills (player placement in race), and Challenges (power-ups of enemies)
Output: A position in space.
Decision: What do we do about it?
Flow

• Advantages
  – Intuitive
  – Matches player experience

• Disadvantages
  – Need to design reliable measures for player challenge and skill, not possible in all games
  – Difficult for non-competitive multiplayer games
MODELS: ROBIN, BARTLE, YEE, ELO, TRUESKILL
Robin’s Laws of Good (tabletop RPG) Gamemastering

Five player archetypes:
• Fighter
  – prefers combat
• Power Gamer
  – gaining special abilities and riches
• Method Actor
  – taking dramatic actions
• Tactician
  – prefers thinking creatively & solving problems
• Storyteller
  – prefers complex plots
Player Types within a game Example: Dragon Age Prototype

Five types, what motivates?

– Fighters (combat)
– Power Gamers (gaining special items)
– Tacticians (creative thinking)
– Storytellers (complex plot)
– Actors (dramatic actions)

How it worked

• Each player has a vector
  – (Fighter, MethodActor, Storyteller, Tactician, PowerGamer)

• Every action is tagged with one or more of these roles.

• Every time a player takes an action, update the value of the vector index of the associated role(s).

• System chooses reward to incentivize the maximal role(s)
Example

- Player A has vector \((F: 0, MA: 0, S: 0, T: 0, PG: 0)\)
- Player A asks for a reward for a quest, an action which is strongly tagged with the PowerGamer role (value of 100)
- Player A now has a vector \((F: 0, MA: 0, S: 0, T: 0, PG: 100)\)
- Later on Player A will be offered more powerful items if their vector remains unchanged
Player Types

• Advantages
  – Can allow designers to understand player base
  – Can broaden the appeal of a game

• Disadvantages
  – Different games/genres need their own player types designed, a tricky task
    • E.g. “Whales” in free-to-play mobile games
  – In-game usage of this model requires a lot of extra knowledge authoring
Robin’s Laws of Good Gamemastering

• Suppose actions, decisions, and/or choice points are tagged with types and strengths
  – e.g. Action X has \textit{method-actor} = 8

• Player model is vector of accumulated strengths (normalized?)
  \[ \langle F, PG, MA, T, S \rangle \]
  \[ \langle 1, 141, 81, 1, 1 \rangle \]

• Incrementally build model
Robin’s Laws of Good Gamemastering

• How does world/actions translate to model?
Robin’s Laws of Good Gamemastering

- How does world/actions translate to model?
- How does model map to design?
Robin’s Laws of Good Gamemastering

• How does world/actions translate to model?
• How does model map to design?

\[ \text{ACTIONS} \rightarrow \text{MODEL} \rightarrow \text{ACTIONS} \]
\[ \text{FEATURES} \rightarrow \text{MODEL} \rightarrow \text{FEATURES} \]
Robin’s Laws of Good Gamemastering

• How does world/actions translate to model?
• How does model map to design?

\[ ACTIONS \rightarrow MODEL \rightarrow ACTIONS \]
\[ FEATURES \rightarrow MODEL \rightarrow FEATURES \]

• Player simulation

• See [http://www.sjgames.com/robinlaws/](http://www.sjgames.com/robinlaws/)
Bartle: MUD Player Modeling

• MUD = Multi-user Dungeon
• 4 things people typically enjoy doing in MUDs:
  – Achievement (preset in-game goals eg kill, aquire)
  – Exploration
  – Socializing
  – Killers (Imposing on other players)

Bartle’s Player Types

• Most commonly used today in the design of MMORPG’s, to motivate these different “types” of players

• Example: World of Warcraft
  – Experience for killing mobs (killer)
  – completing quests (achievers)
  – exploring (explorers)
  – battlegrounds (socializers)
MUD Player Modeling

- Everyone has a primary type
  - Bit of each, tendency towards one
- Take the Bartle Test!
- Suggests a feature vector
  \[
  \langle \text{killer, achiever, explorer, socializer} \rangle
  \]
  \[
  < 0.0, 0.7, 0.1, 0.2 >
  \]
- See (no really, follow the 1\textsuperscript{st} link)
  - [http://www.mud.co.uk/richard/hcds.htm](http://www.mud.co.uk/richard/hcds.htm)
Yee Player Model

• Used iterative process to validate, expand and refine a player motivation model **empirically**
• Based on open-ended questions about motivation
  – “I feel powerful in the game.”
  – “I like to be immersed in a fantasy world.”
• See
Yee Player Model: Factors

- Relationship
- Manipulation
- Immersion
- Escapism
- Achievement

“Performed factor analysis on data to separate the statements into clusters where items within each cluster were as highly correlated as possible while clusters themselves were as uncorrelated as possible” – Yee
Yee Player Model

<table>
<thead>
<tr>
<th>Achievement</th>
<th>Social</th>
<th>Immersion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advancement</strong></td>
<td><strong>Socializing</strong></td>
<td><strong>Discovery</strong></td>
</tr>
<tr>
<td>Progress, Power,</td>
<td>Casual Chat,</td>
<td>Exploration,</td>
</tr>
<tr>
<td>Accumulation, Status</td>
<td>Helping Others,</td>
<td>Lore,</td>
</tr>
<tr>
<td></td>
<td>Making Friends</td>
<td>Finding Hidden</td>
</tr>
<tr>
<td><strong>Mechanics</strong></td>
<td><strong>Relationship</strong></td>
<td><strong>Role-Playing</strong></td>
</tr>
<tr>
<td>Numbers, Optimization</td>
<td>Personal, Self-Disclosure,</td>
<td>Story Line,</td>
</tr>
<tr>
<td>Templating, Analysis</td>
<td>Find and Give</td>
<td>Character History,</td>
</tr>
<tr>
<td></td>
<td>Support</td>
<td>Roles, Fantasy</td>
</tr>
<tr>
<td><strong>Competition</strong></td>
<td><strong>Teamwork</strong></td>
<td><strong>Customization</strong></td>
</tr>
<tr>
<td>Challenging Others,</td>
<td>Collaboration,</td>
<td>Appearance,</td>
</tr>
<tr>
<td>Provocation, Domination</td>
<td>Groups,</td>
<td>Accessories,</td>
</tr>
<tr>
<td></td>
<td>Group Achievements</td>
<td>Style, Color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Schemes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Escapism</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relax, Escape</td>
</tr>
<tr>
<td></td>
<td></td>
<td>from RL, Avoid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RL Problems</td>
</tr>
</tbody>
</table>

Motivations do not “suppress”

“The subcomponents generated by the factor analysis are NOT player types.

• It is NOT the case that we have come up with 10 boxes that we can put players in, but rather, we have revealed 10 subcomponents that co-exist and together reveal the motivations of a player.

• Bartle assumed that your underlying motivations "suppressed" each other. In other words, the more of an Achiever you were, the less of a Socializer, Explorer and Killer you could be, but just because you like ice-cream doesn't mean you will hate pasta.

• The assumption of polarized motivations is also not supported by the correlations of the current data set.”

Where do you fit?

• Proficient, Relaxed, Competitive, Grounded, and Inquisitive

After filling out a brief survey (5-7 minutes), this profile tool will generate a customized report and a list of recommended games for you. The report will describe the traits that were measured, and how you compare with other gamers. [https://apps.quanticfoundry.com/lab/gamerprofile/](https://apps.quanticfoundry.com/lab/gamerprofile/)
<table>
<thead>
<tr>
<th>ICON</th>
<th>NAME</th>
<th>SKILL RATING</th>
<th>SEASON 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>🅃️</td>
<td>BRONZE</td>
<td>1-1499</td>
<td>1-29</td>
</tr>
<tr>
<td>🅃️</td>
<td>SILVER</td>
<td>1500-1999</td>
<td>30-39</td>
</tr>
<tr>
<td>🅃️</td>
<td>GOLD</td>
<td>2000-2499</td>
<td>40-49</td>
</tr>
<tr>
<td>🅃️</td>
<td>PLATINUM</td>
<td>2500-2999</td>
<td>50-59</td>
</tr>
<tr>
<td>🅃️</td>
<td>DIAMOND</td>
<td>3000-3499</td>
<td>60-69</td>
</tr>
<tr>
<td>🅃️</td>
<td>MASTER</td>
<td>3500-3999</td>
<td>70-79</td>
</tr>
<tr>
<td>🅃️</td>
<td>GRANDMASTER</td>
<td>4000-5000</td>
<td>80-100</td>
</tr>
<tr>
<td>🅃️</td>
<td>TOP 500</td>
<td>TOP 500 IN REGION</td>
<td></td>
</tr>
</tbody>
</table>

ELO
Elo’s Performance Rating

• Algorithm:
  – For each win, add your opponent’s rating plus 400
  – For each loss, add your opponent’s rating minus 400
  – Divide this sum by the number of played games

• Formula:
  \[
  \frac{\text{(Total of opponents’ ratings} + 400(\text{Wins} – \text{Losses}))}{\text{Games}}
  \]
Quick Note: Elo rating naming confusion

Many games do not make use of this exact rating algorithm, but you will still hear references to “Elo rating”/Elo hell

At a high level this phrase refers to any approach to update player skills given wins/losses and prior player skills

– Probability distribution
Elo Rating

• Advantages
  – Requires very little designer authoring
  – Does a surprisingly good job of keeping competitive games fair

• Disadvantages
  – Player’s do not like seeing this value drop
    • Solution: Many games have “hidden” true Elo ratings
  – Limited to competitive experiences of 2 players
    • Only makes sense when comparing players to others

See also https://en.wikipedia.org/wiki/TrueSkill
MS TrueSkill ™

- The TrueSkill (Bayesian) ranking system is a skill based ranking system for Xbox Live developed at Microsoft Research. The purpose of a ranking system is to both identify and track the skills of gamers in a game (mode) in order to be able to match them into competitive matches. TrueSkill has been used to rank and match players in many different games, from Halo 3 to Forza Motorsport 7.

- An improved version of the TrueSkill ranking system, named TrueSkill 2, launched with Gears of War 4 and was later incorporated into Halo 5.

- In particular, the ELO ranking system has been used successfully by a variety of leagues organized around two-player games, such as world football league, the US Chess Federation or the World Chess Federation, and a variety of others.

- In video games many of these leagues have game modes with more than two players per match. ELO is not designed to work under these circumstances. In fact, no popular skill-based ranking system is available to support these games. Many one-off ranking systems have been built and are in use for these games, but none of them is general enough to be applied to such a great variety of games.
DRAMA MANAGEMENT
Drama Management

Exposition \[\rightarrow\] Climax \[\rightarrow\] Denouement

Rising action \[\rightarrow\] Falling action
Drama Management

Intensity

Climax

Rising action

Falling action

Exposition

Time

Denouement
Left4Dead AI Director

- Constantly measures player “intensity”
  - Intensity increases when attacked/killing
  - Players hit max when incapacitated
  - Players slowly drop intensity when nothing is happening and reset at the start of a new stage

- (This is a lot like challenge in flow)
Left4Dead AI Director

• Three “modes”
  – **Build up**: AI Director attempts to increase tension
  – **Peak**: Intensity reaches max, director stops spawning enemies
  – **Relax**: Director spawns helpful items/alters path to decrease tension
AI Director as FSM

Intensity

Build Up

Peak

Falling action

Rising action

Relax

Time
Drama Management Summary

• Advantages
  – Most effective, highly praised
  – Gives players another level of strategy

• Disadvantages
  – Fair amount of authoring (less than Player Types, more than Flow)
  – All the downsides of FSMs
PCG: STEPPING BACK
Player Model Process

Player Preferences

Monitor Player Performance

Adapt the Game

Measure effectiveness of adaptation

Models of player types

Re-model player types

e.g. feature vector

Can simulate performance
Concerns
Concerns

- Player desires
Concerns

- Player desires
- Privacy
Concerns

- Player desires
- Privacy
- Testing
Concerns

• Player desires
• Privacy
• Testing
• Player experience
Summary

Authoring burden for using in a game:
player types > drama management > flow > ELO

**Ultimately**: Decision of when to use what depends on the type of game and desired player experience
THOUGHT EXPERIMENT: SMB LEVEL GENERATION
Mario Level Generation

• Predict users’ emotional response to level:
  – Fun
  – Challenging
  – Boring
  – Frustrating
Mario Level Generation

• Controllable features
Mario Level Generation

- Controllable features
- {#gaps, gap_width, spatial_diversity}
Mario Level Generation

• Controllable features
• {#gaps, gap_width, spatial_diversity}
• Fixed: #coins, #enemies, # ?-boxes, #powerups, etc.
Building the Model

• Option #1: Pairwise Ordering
• For each pair of levels, ask:

  * Level A is more \(<\text{adj}>\) than Level B
  * Levels A and B are equally \(<\text{adj}>\)
  * Neither Level A nor Level B are \(<\text{adj}>\)

Where \(<\text{adj}>\) = \{fun, challenging, boring, frustrating\}
Building the Model

• Option #2: Use ANN to learn to predict question-answering
Building the Model

• Option #2: Use ANN to learn to predict question-answering
• Need a function that maps features to a measure of quality:
  \[ f (\text{gaps, width, spatial\_diversity}) \rightarrow \mathbb{R} \]
Building the Model

- Other option: Learn decision tree with continuous variables
- Benefit of trained neural network: The network is your fitness function.
Using the Model

• Search for optimal feature set for a player
Using the Model

• Search for optimal feature set for a player
• State is 3-tuple \{G, W, SD\} – Could brute-force
Using the Model

- Search for optimal feature set for a player
- State is 3-tuple \{G, W, SD\} – Could brute-force
- Optimization search – Genetic algorithms, hill-climbing, simulated annealing
RHYTHM-BASED PLATFORMER LEVEL GENERATION
Rhythm based level-generation

• Patterned set of moves

- 20s, regular, low density
- 20s, regular, medium density
- 20s, regular, high density
- 15s, swing, medium density
- 10s, random, high density

move 0 5
jump 2 2.25
jump 4 4.25
move 6 10
jump 6 6.5
jump 8 8.5
Geometry grammar

- Rhythm determines what roots
- Terminals determine specific content element selections

\[
\begin{align*}
\text{Moving} & \rightarrow \text{Sloped} \mid \text{flat\_platform} \\
\text{Sloped} & \rightarrow \text{Steep} \mid \text{Gradual} \\
\text{Steep} & \rightarrow \text{steep\_slope\_up} \mid \text{steep\_slope\_down} \\
\text{Gradual} & \rightarrow \text{gradual\_slope\_up} \mid \text{gradual\_slope\_down} \\
\text{Jumping} & \rightarrow \text{flat\_gap} \\
& \mid (\text{gap} \mid \text{no\_gap}) (\text{jump\_up} \mid \text{Down} \mid \text{spring} \mid \text{fall}) \\
& \mid \text{enemy\_kill} \\
& \mid \text{enemy\_avoid} \\
\text{Down} & \rightarrow \text{jump\_down\_short} \mid \text{jump\_down\_medium} \mid \text{jump\_down\_long} \\
\text{Waiting-Moving} & \rightarrow \text{stomper} \\
\text{Waiting-Moving-Waiting} & \rightarrow \text{moving\_platform\_vert} \\
& \mid \text{moving\_platform\_horiz}
\end{align*}
\]
Grammar generation problems

- Over-generation
- Global constraints
Problems with Grammars

• Problem with grammars: global context
• Can you fix grammars?
• Generate-and-test with some evaluation function
• Backtrack?
• Planning: Search for a sequence of actions that meet a goal
  – Heuristics meant to speed up search
  – Heuristics can also be used to guide algorithm toward more preferable solutions
• HTN planning: Tasks decompose to sub-tasks
Critics

• Generate-and-test: generate a bunch of levels and pick the one that scores best
• Critics are pieces of code that evaluate and select
• Line distance critic
  – Given a path the level should follow
• Component style critic
  – Consistency of components
  – Jumping caused by gaps vs. monsters