Procedural Content Generation: Player Models

2016-07-12

OOB

- HW 6, Hero Agents
- Capstones -- Pitches
- 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

Questions

1. How can we describe decision making?
2. What do the algorithms we’ve seen share?
3. What are the dimensions we tend to assess?
4. FSMs/Btrees: ____ :: Planning : _____
5. For the 2nd blank, we need m_____s.
6. When is reactive appropriate? Deliberative?
7. What is the ‘hot-potato’ passed around (KE)?
8. H______ have helped in most approaches.
9. Which approach should you use?

Questions

1. What are the 2 most “complex” decision making techniques we’ve seen?
2. What are their strengths? Weaknesses?
3. What is the key (insight) to their success?
4. What is typically necessary to support this insight (hint: used in Planning + RBS)?
5. What does Planning have that (forward chaining) RBS do not?
6. When do we need a communication mechanism?

PROCEDURAL CONTENT GENERATION: RECAP

Procedural Content Generation

- Costs exponentially increasing: levels, maps, tracks, missions, characters, weapons…
- CONTENT IS KING
- Use of computation to produce elements of gameplay (instead of manual effort)
  - Design aspects of the game
    - Save development cost
    - Save storage or main memory (“infinite games”)
  - Adapt aspects of the game (player models)

Why industry is looking at PCG

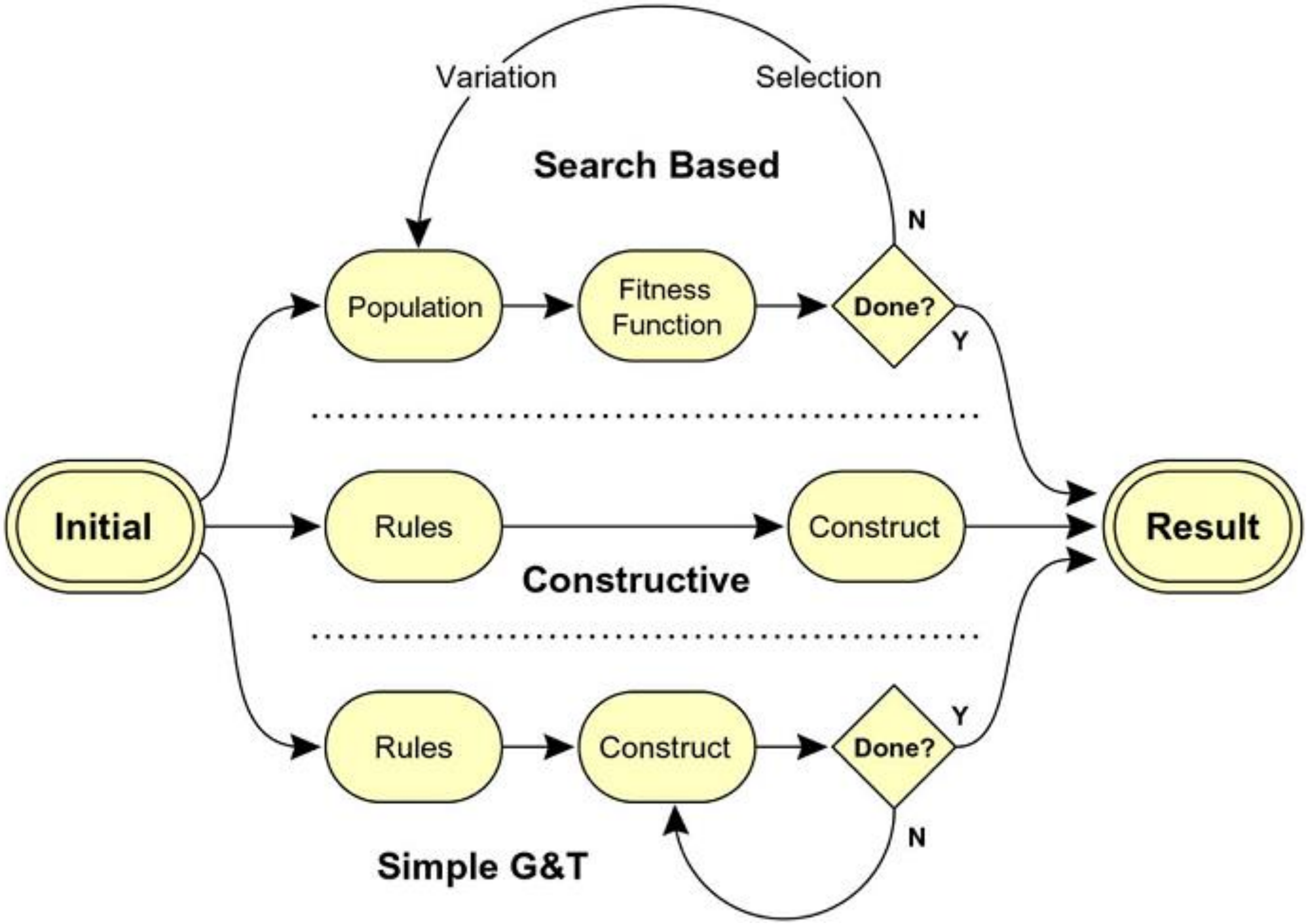
- Development cost savings
- Replayability
- Customization/adaptation
- Dynamic difficulty adjustment
  - Flow theory
    - C.f. http://en.wikipedia.org/wiki/Flow_(psychology)
    - http://jenovachen.com/flowingames/designfig.htm
    - http://en.wikipedia.org/wiki/Flow_(video_game)

PCG Desiderata

- Fast, Reliable, High-quality
- Novelty, Structure, Interest
- Controllable (parameterized)
  - Geometric aspects (e.g. length of track)
  - Gameplay aspects (e.g. difficulty)
  - Adapted to player(s) (e.g. preference, type, fun)
  - Designer-centric as well

See: http://www.marioai.org/levelGenerationCompetition.pdf for excellent overview

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172-186.



http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5756645&tag=1

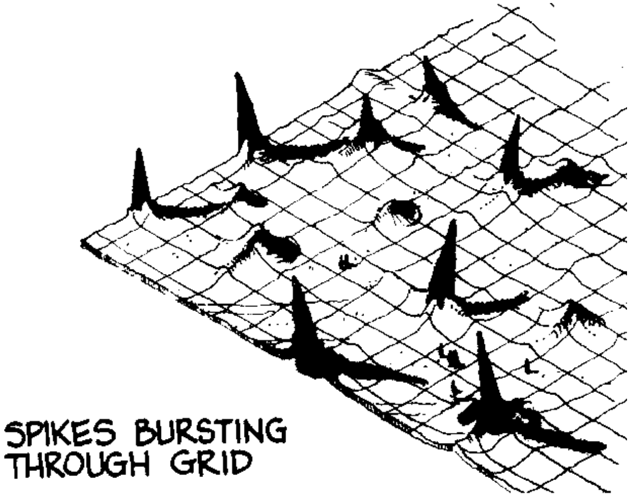
PCG NoNos

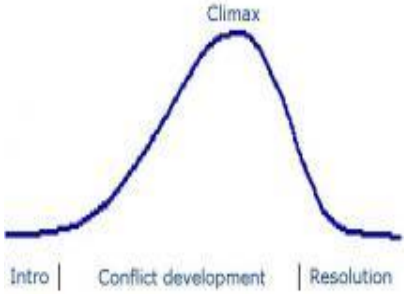
- PCG can be NP-hard for anything non-trivial
  - But you can’t always make players wait
- Thorough testing of run-time PCG is impossible
  - Best you can do is statistical sampling
- Offensive material
- Bad content
- Algorithm crash
- Meaningless activities
  - Easy to create quests, but if they don’t connect to the larger game, no one cares

PCG Gos

- Terrain, cities, trees, landscaping
- Levels/maps/dungeons
- Quests
- Monsters & NPCs
- Weapons
- Stories (?)
- Music (?)

L4Ds Legacy





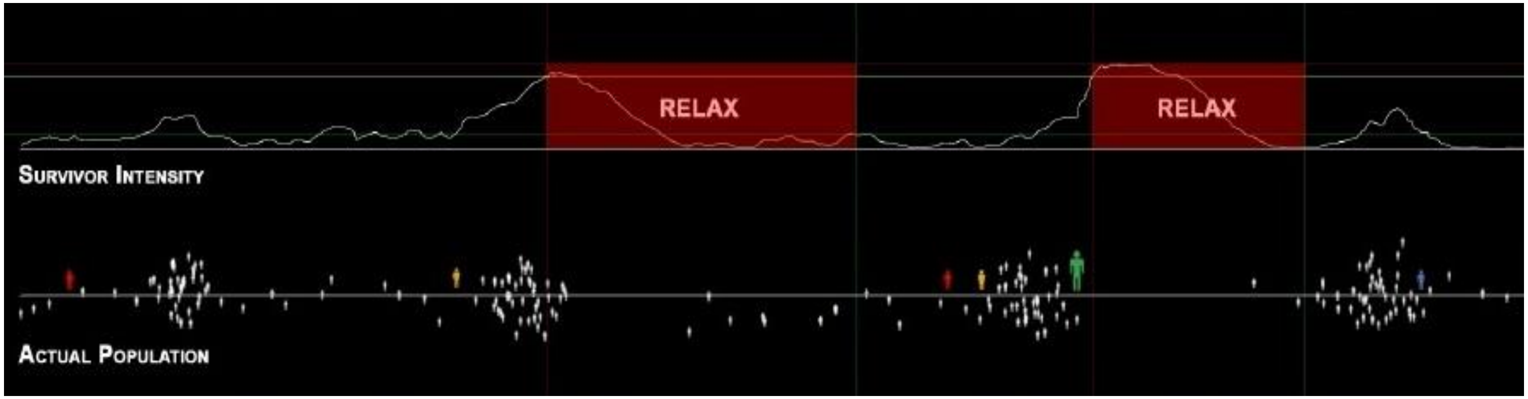


Photo Credit: http://dankline.wordpress.com/2011/06/29/the-ai-director-of-darkspore/

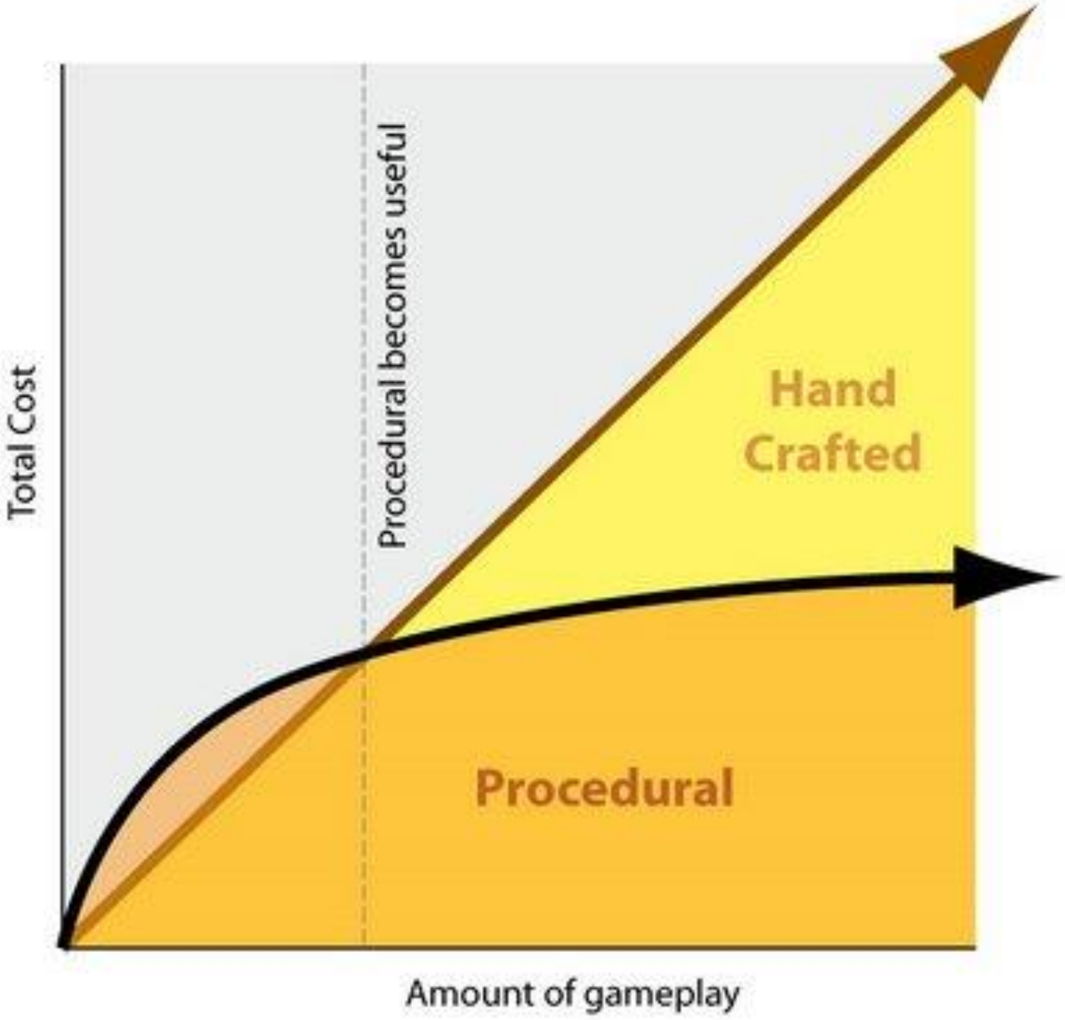
Design time vs. Run time PCG

- Design time: Speed up design of static content
  - # of unique objects in the world
  - Players expect non-repetitive
  - Game dev times now 100s of man-years with huge design teams
  - Cost savings big motivation
  - RISKS: quality (designer) control, stupidity, magic circle
- Run-time: customization, dynamic adjustment

Design time vs. Run time PCG

- Design time: Speed up design of static content
- Run-time: customization, dynamic adjustment
  - Players are different: Preferences for pace + playstyle
    - Moderate challenge levels (e.g. help avoid getting stuck)
    - Adjust to play style
    - Detect/avoid player exploits
  - When to use run-time PCG
    - When decisions can only be made at run-time
    - When pre-compute exceeds storage/memory limits
    - Replayability; story/quest generation; pacing;
  - Optimization problem
    - What is the set of content that delivers the optimal experience to the player given individual differences?

Economies of scale



Hard problems in PCG

- Procedures structuring player’s experience
  - Narrative: It’s easy to have infinite levels, but dungeon crawling gets boring without a sense of progression / achievement
- Avoiding (dealing with) occasional catastrophic failure
- Healing a broken level
- Social factors + story/language generation
- How do you know you are generating something interesting?
  - Mental & statistical models of players
  - Personalized content

PLAYER MODELING

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

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

ROBIN, BARTLE, YEE

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

Robin’s Laws of Good Gamemastering

- Suppose actions, decisions, and/or choice points are tagged with types and strengths
  - e.g. Action X has *method-actor* = 8
- Player model is vector of accumulated strengths (normalized?)

$$<F, PG, MA, T, S>$$

$$<1, 141, 81, 1, 1>$$

- 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?

ACTIONS ---> MODEL ---> ACTIONS

FEATURES ---> MODEL ---> FEATURES

Robin’s Laws of Good Gamemastering

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

ACTIONS ---> MODEL ---> ACTIONS

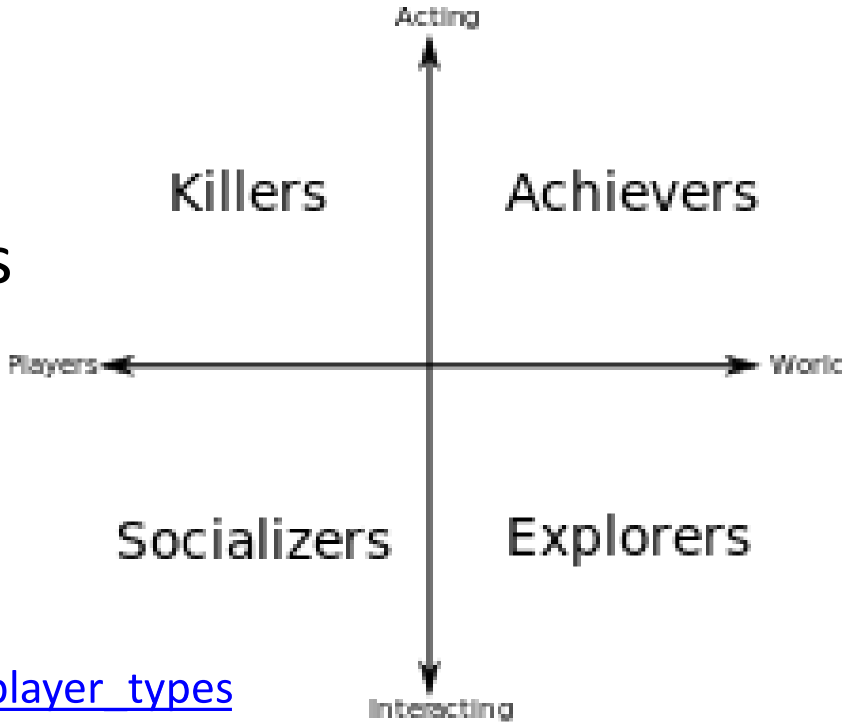
FEATURES ---> MODEL ---> FEATURES

- Player simulation

- See http://www.sjgames.com/robinslaws/

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
  - Imposing on other players



https://en.wikipedia.org/wiki/Bartle_taxonomy_of_player_types

MUD Player Modeling

- Everyone has a primary type
  - Bit of each, tendency towards one
- Take the Bartle Test!
- Suggests a feature vector

<*killer*, *achiever*, *explorer*, *socializer*>

< 0.0, 0.7, 0.1, 0.2 >

- See (no really, follow the 1st link)
  - http://www.mud.co.uk/richard/hcds.htm
  - http://en.wikipedia.org/wiki/Bartle_Test

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
  - http://www.nickyee.com/daedalus/archives/001298.php

Yee Player Model: Factors

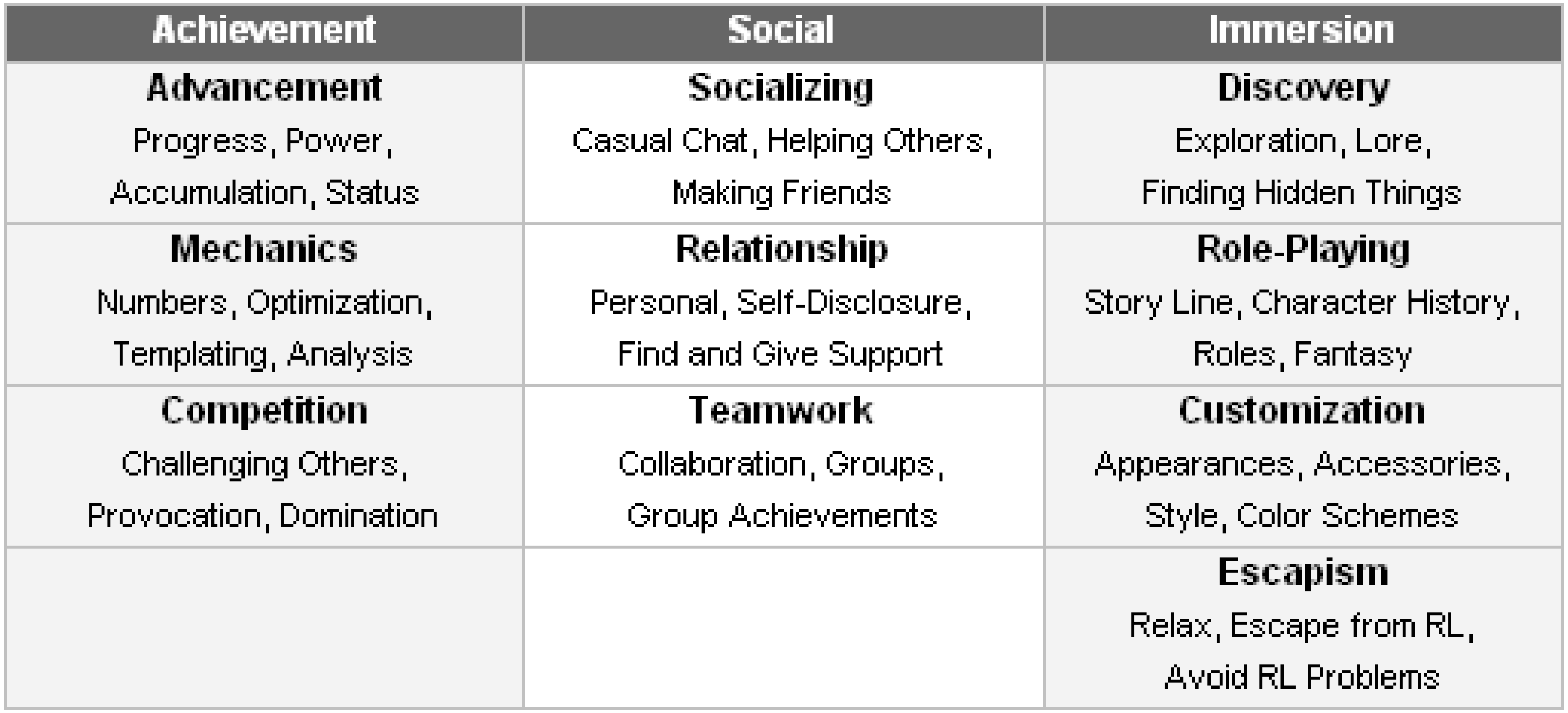
- Relationship
- Manipulation
- Immersion
- Escapism
- Achievement

“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

- Relationship
- Manipulation
- Immersion
- Escapism
- Achievement

- Leader
- Solo/Group
- Learn

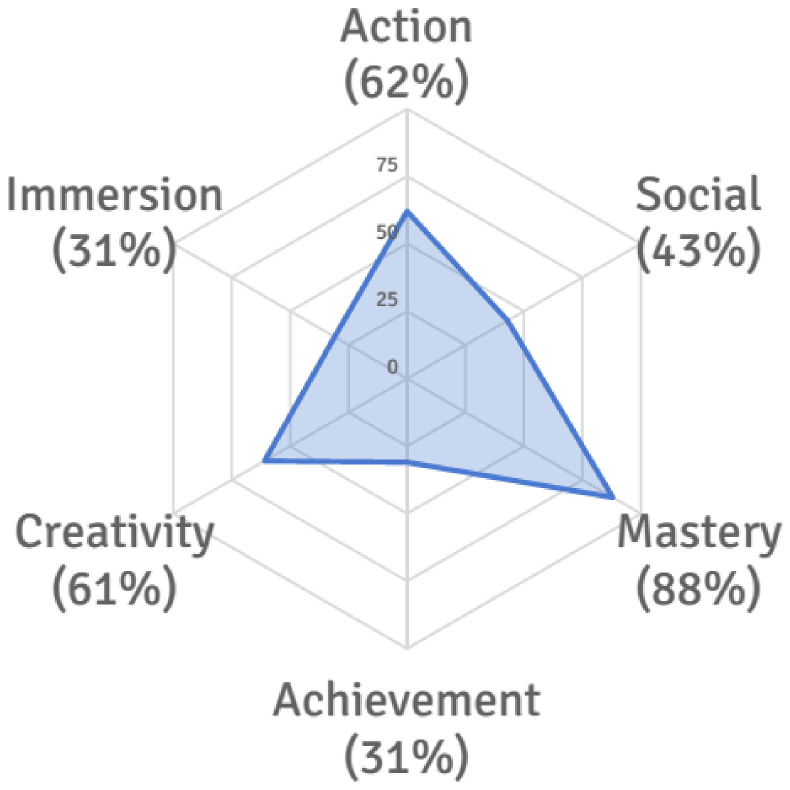


Achievement	Social	Immersion
Advancement Progress, Power, Accumulation, Status	**Socializing** Casual Chat, Helping Others, Making Friends	**Discovery** Exploration, Lore, Finding Hidden Things
Mechanics Numbers, Optimization, Templating, Analysis	**Relationship** Personal, Self-Disclosure, Find and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

http://www.nickyee.com/daedalus/archives/001298.php?page=4

Where do you fit?

- Proficient, Relaxed, Competitive, Grounded, and Inquisitive

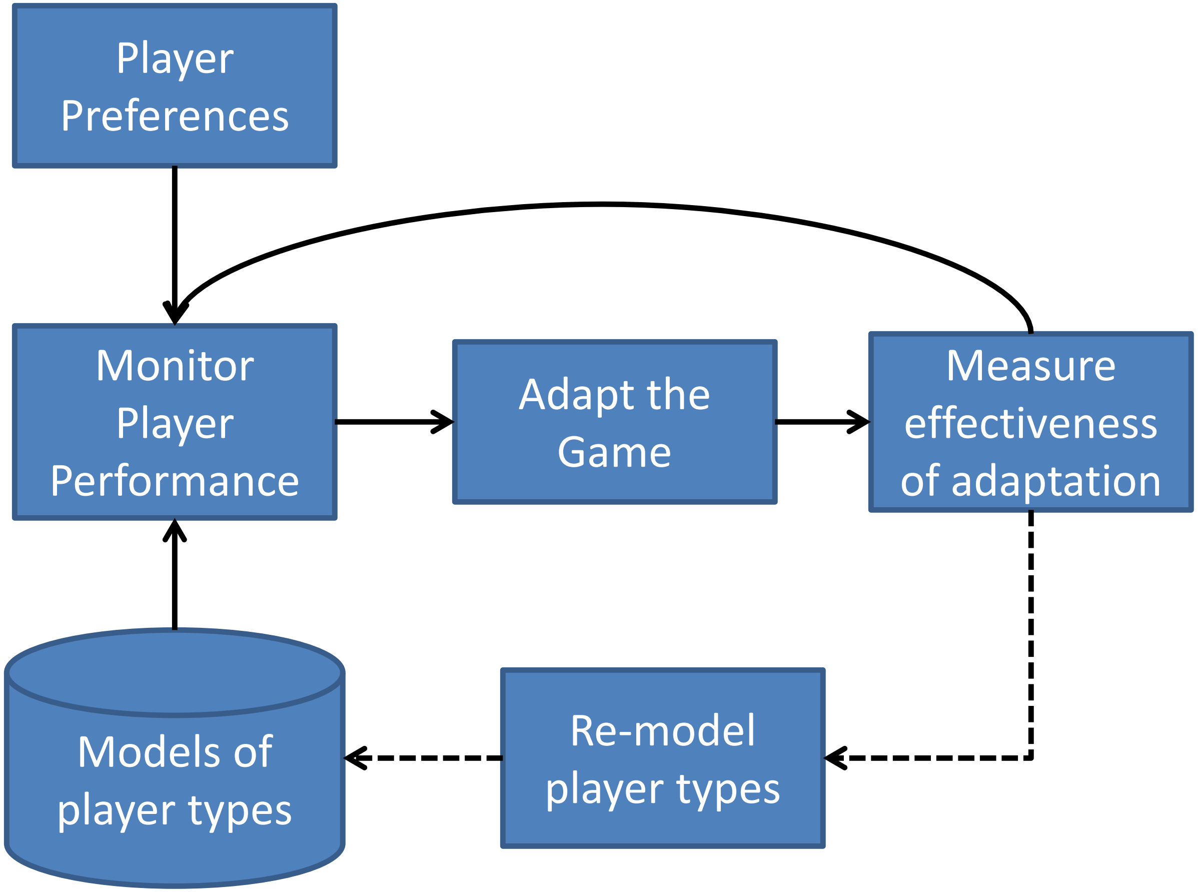


After filling out a brief survey (5-7 minutes), this profile tool will generate a customized report and a list of recommended games for you. The report will describe the traits that were measured, and how you compare with other gamers.

https://apps.quanticfoundry.com/lab/gamerprofile/

PCG: STEPPING BACK

Player Model Process



Concerns

Concerns

- Player desires

Concerns

- Player desires
- Privacy

Concerns

- Player desires
- Privacy
- Testing

Concerns

- Player desires
- Privacy
- Testing
- 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 <adj> than Level B

Levels A and B are equally <adj>

Neither Level A nor Level B are <adj>

Where <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:

$$f\,(\text{gaps, width, spatial\_diversity}) \rightarrow \mathbf{R}$$

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

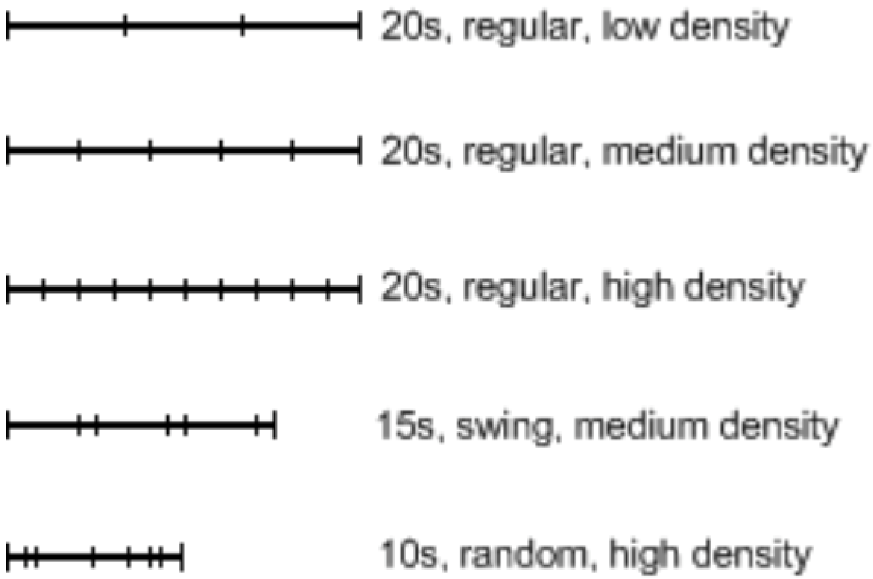
- Search for optimal feature set for a player
- State is 3-tuple {G, W, SD} – Could brute-force
- Optimization search – Genetic algorithms, hill-climbing, simulated annealing

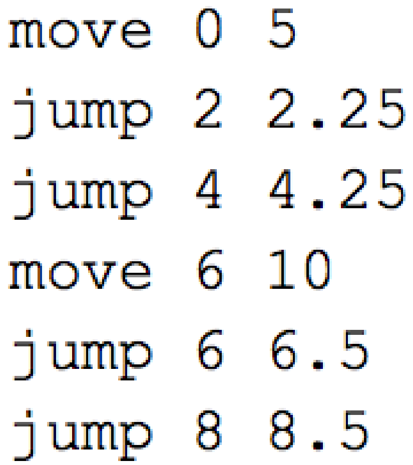
Smith, Treanor, Whitehead, Mateas {FDG, 2009}

RHYTHM-BASED PLATFORMER LEVEL GENERATION

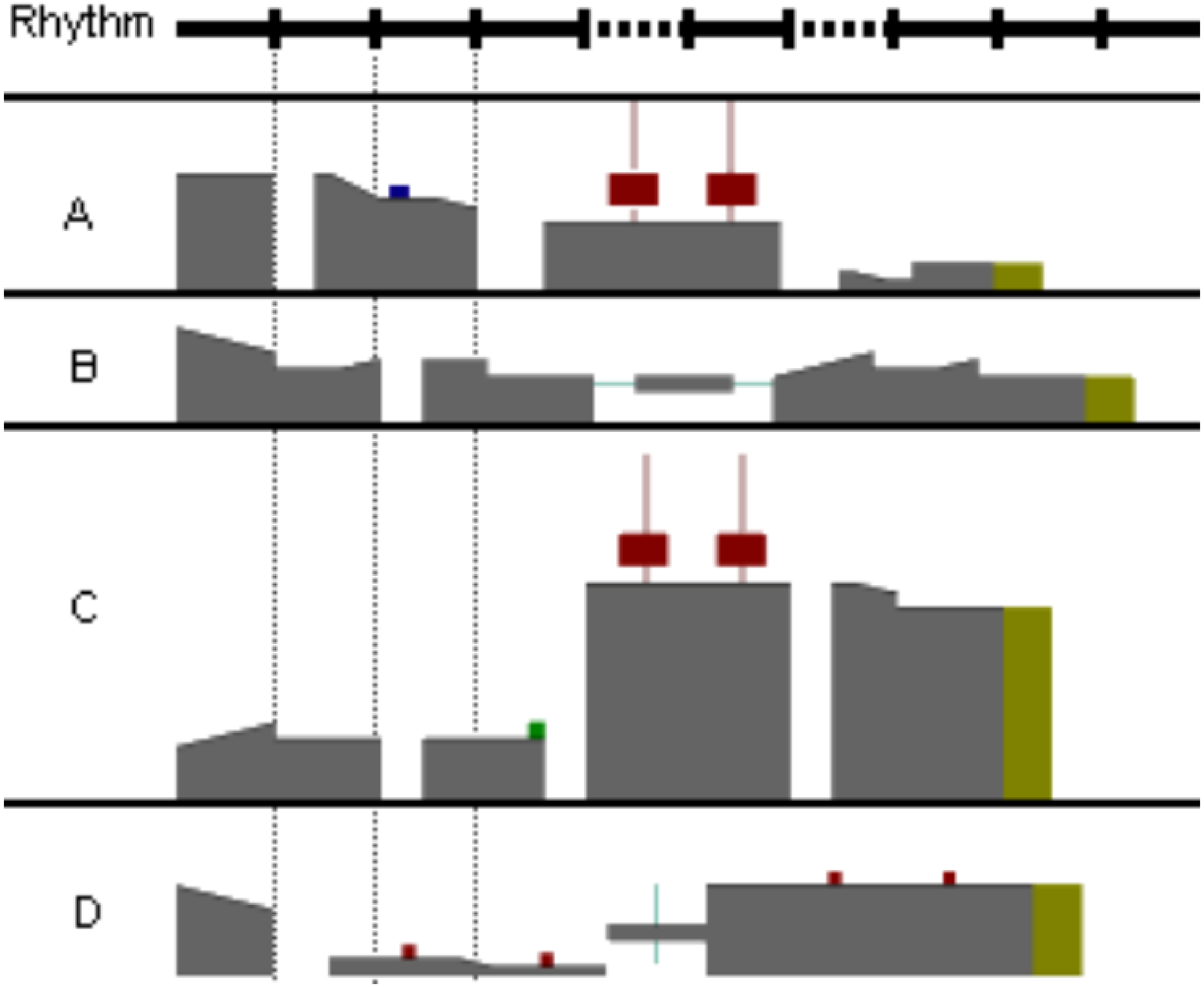
Rhythm based level-generation

- Patterned set of moves


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

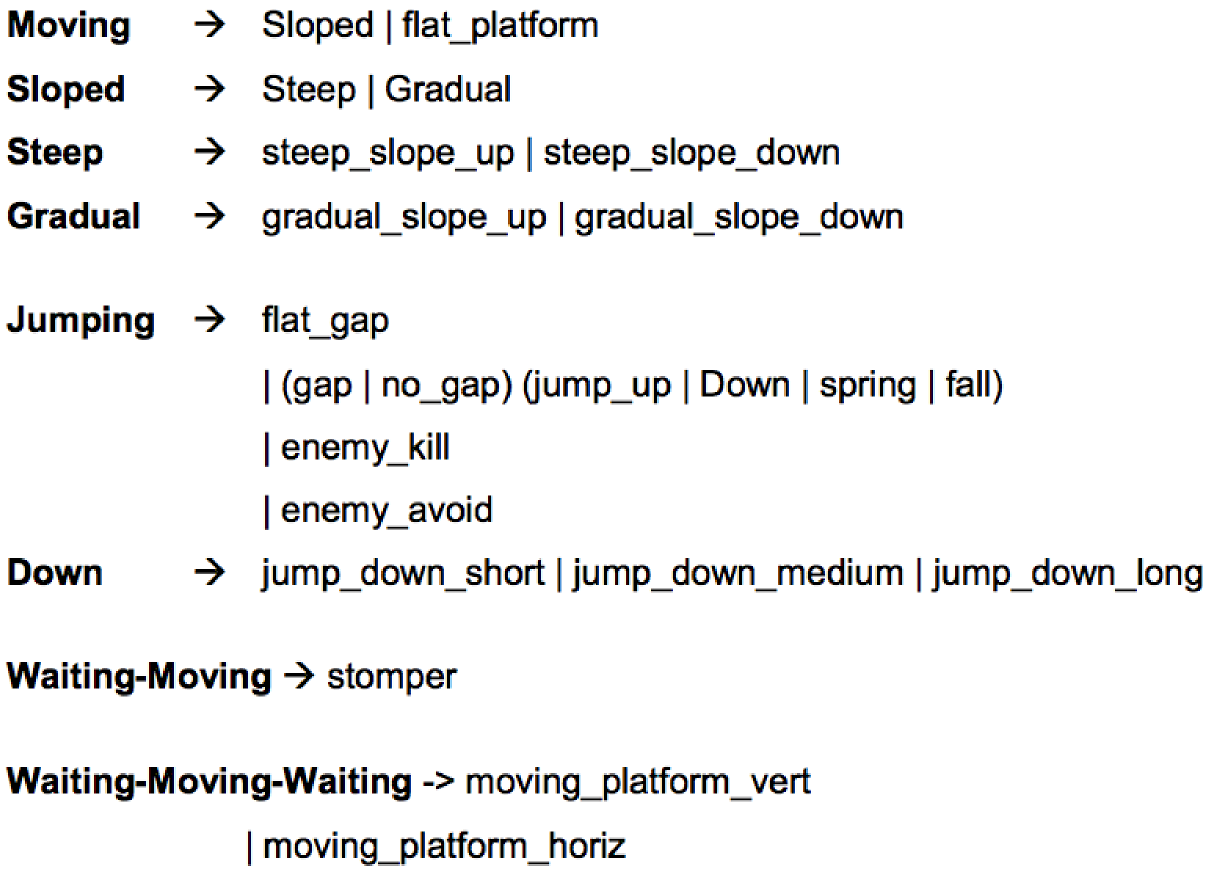


```
move 0 5
jump 2 2.25
jump 4 4.25
move 6 10
jump 6 6.5
jump 8 8.5
```


Rhythm
A
B
C
D

Geometry grammar

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



```
Moving        →  Sloped | flat_platform
Sloped        →  Steep | Gradual
Steep         →  steep_slope_up | steep_slope_down
Gradual       →  gradual_slope_up | gradual_slope_down

Jumping       →  flat_gap
                 | (gap | no_gap) (jump_up | Down | spring | fall)
                 | enemy_kill
                 | enemy_avoid
Down          →  jump_down_short | jump_down_medium | jump_down_long

Waiting-Moving → stomper

Waiting-Moving-Waiting -> moving_platform_vert
                 | moving_platform_horiz
```

Grammar generation problems

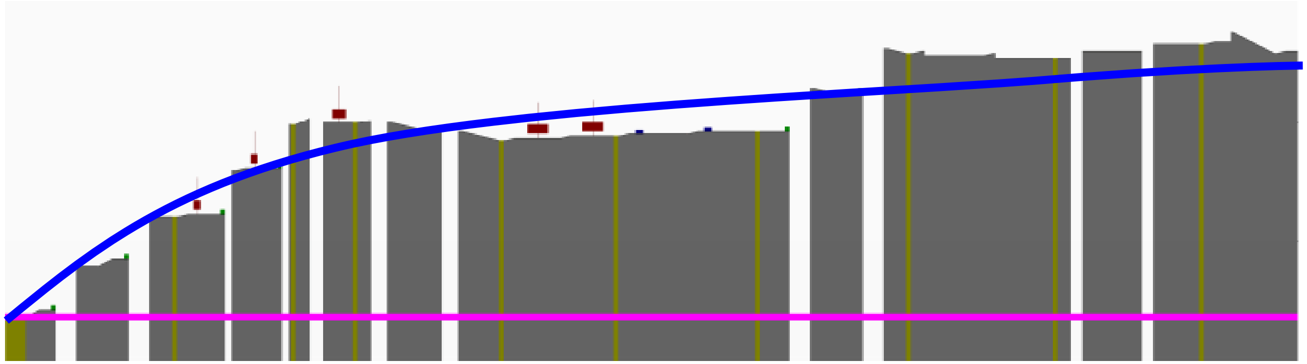
- Over-generation
- Global constraints

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



Control lines for line critic