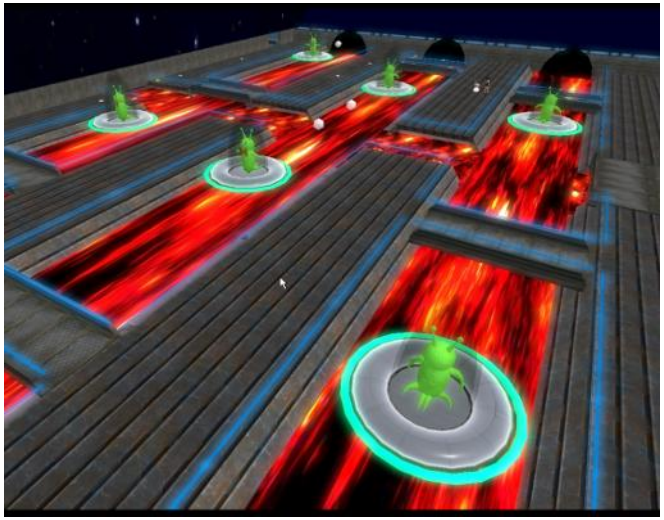


Game-based Data Capture for Player Metrics

Aline Normoyle, John Drake
University of Pennsylvania

Maxim Likhachev
Carnegie Mellon University

Alla Safonova
Disney Research, Pittsburgh

Player Metrics

- What are they?
  - Player statistics
  - Objective description of player behaviors
  - Examples
    - Deaths per level
    - Percentage of time spent in an activity
- Why collect them?
  - Playtesting and design
  - Train Bots and Non-Player Characters
  - Customize gameplay

Gathering Metrics

- Passively record players
- Potential drawbacks
  - Desired player behaviors may be infrequent
  - Data acquisition costly, laborious, memory intensive
- Can we reduce the number of data capture sessions we need to run?

Our approach

- Investigate ***active*** approach for acquiring metrics
  - Make automated data capture more efficient
  - Focus on metrics which most require data
  - Model correspondence between player metrics and game configurations
    - Use model to choose the best game configuration to run next
    - No need to know player behaviors *a priori*

Related Work – Playtesting

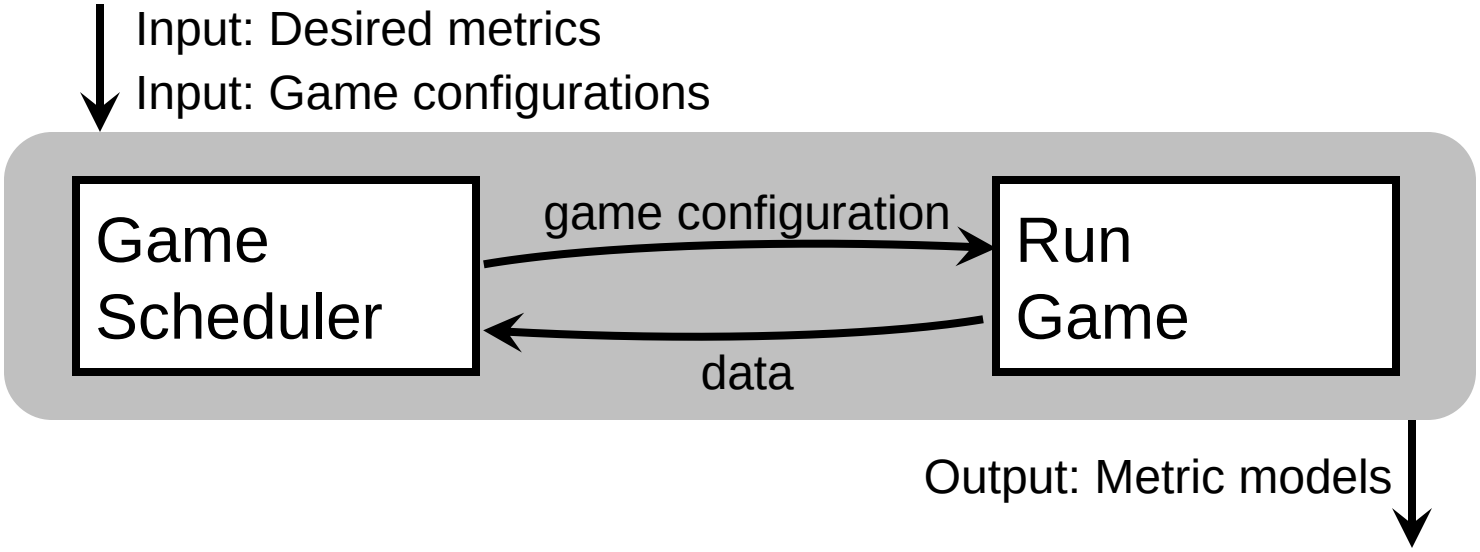
- Identifying problems and improving gameplay [Kennerly 2003; DeRosa 2007; Luban 2009; Ambinder 2009]
- Designing more effective development/test cycles [Medlock et al. 2002]
- Designing automated methods for logging, tracking, and reporting [Kennerly 2003; DeRosa 2007; Kim et al. 2008]
- Organizing and visualizing the large data sets collected with logging [Chittaro, Ranon, and Ieronutti 2006; Andersen et al. 2010; Moura, el Nasr,and Shaw 2011]
- Designing better models for understanding player data [Tychsen and Canossa 2008; Drachen and Canossa 2009; Mahlmann et al. 2010]

Related Work – Active Learning

- Goal: Reducing manual effort or cost [Chaloner and Verdinelli 1995; Settles 2012]
- Reducing manual labeling
  - text classification [Roy and Mccallum 2001; Hoi, Jin, and Lyu 2006]
  - image classification and retrieval [Tong and Chang 2001; Hoi et al. 2006]
  - speech recognition [Tur, Hakkani-Tur, and Schapire 2005; Riccardi and Hakkani-Tur 2005]
- Optimal experiment design
  - "robot-scientist" [King et al. 2004]
  - animation controller [Cooper, Hertzmann, and Popovic 2007]

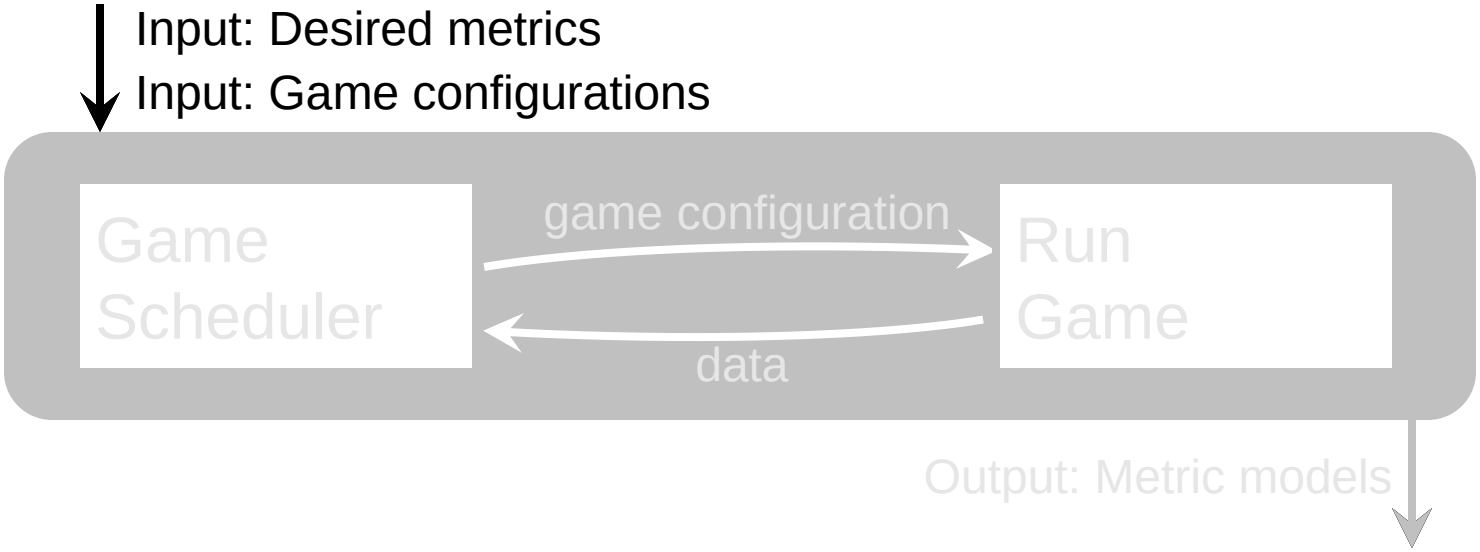
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



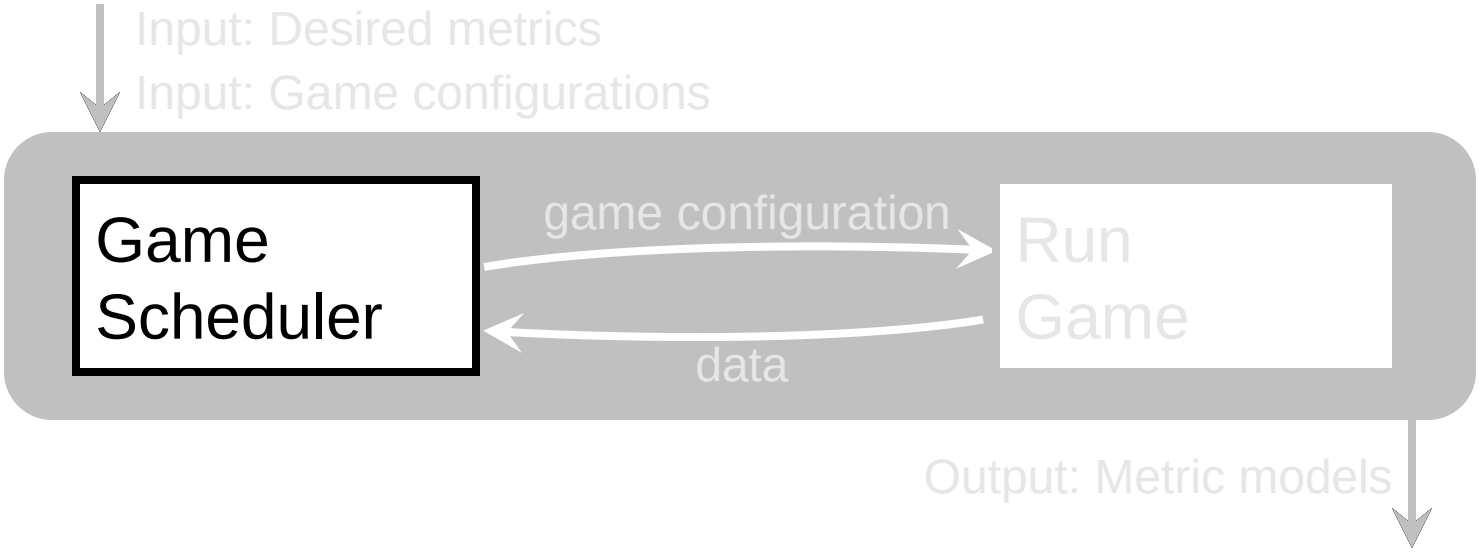
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



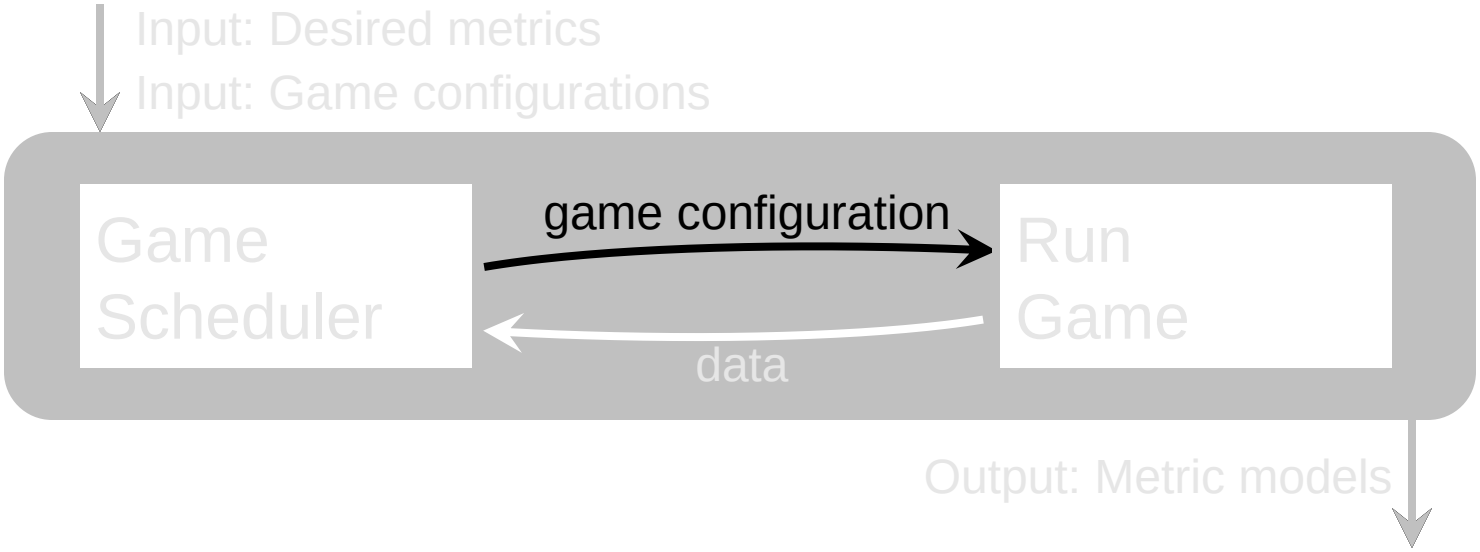
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



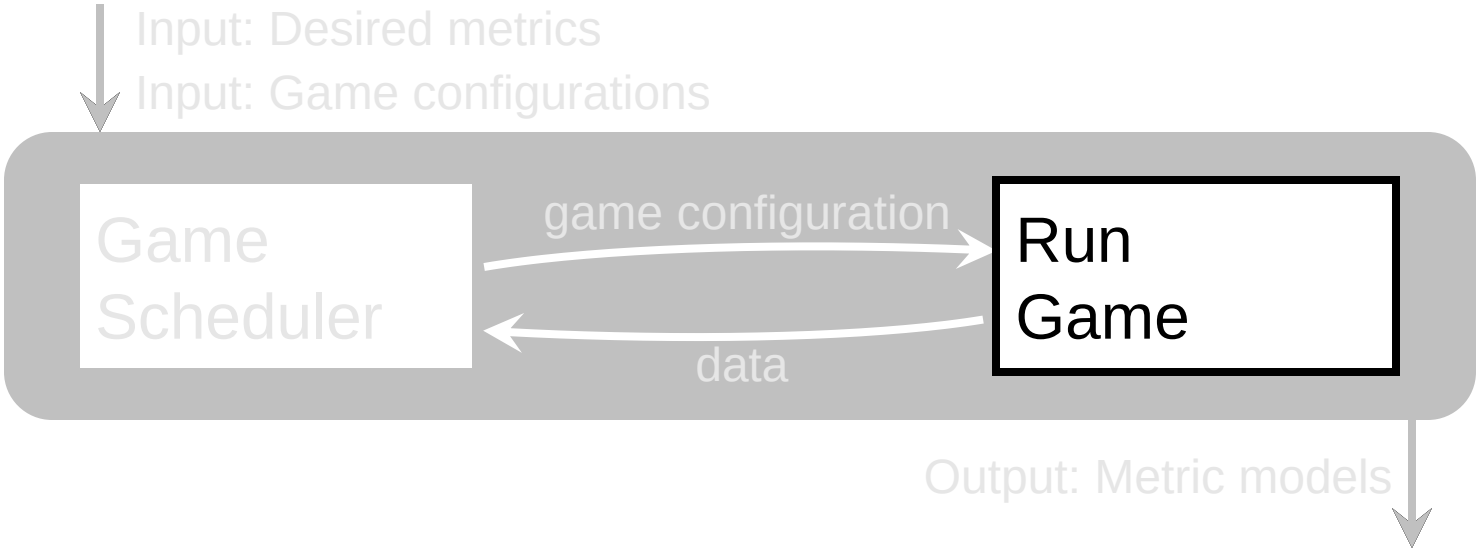
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



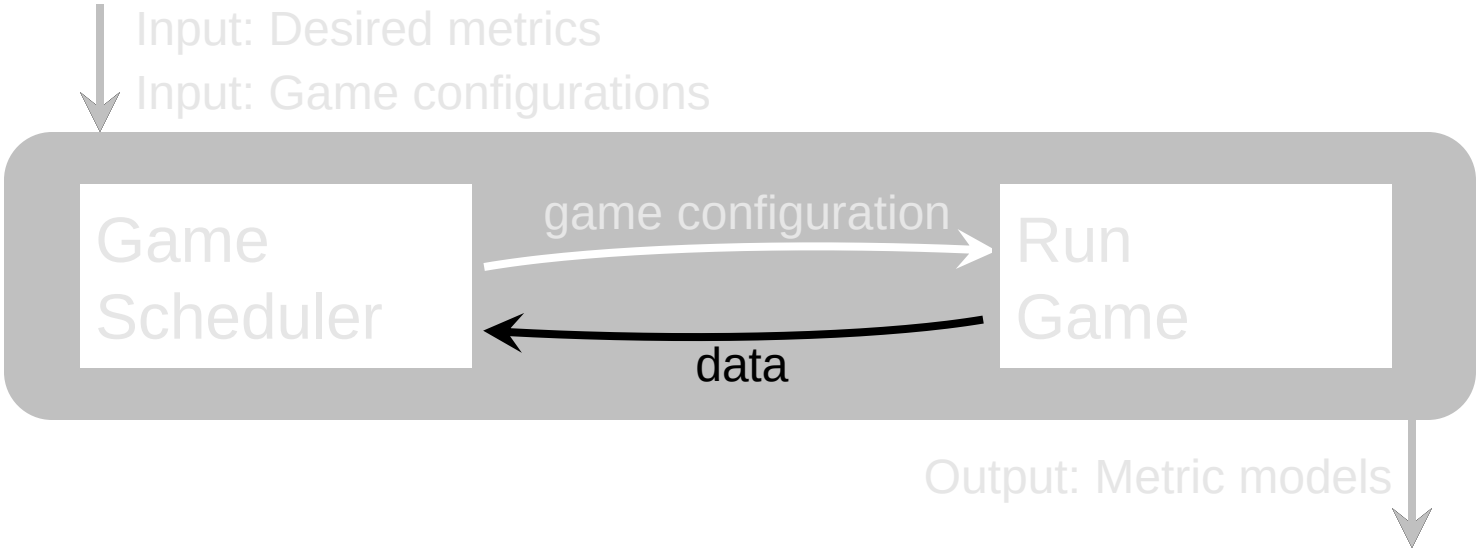
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



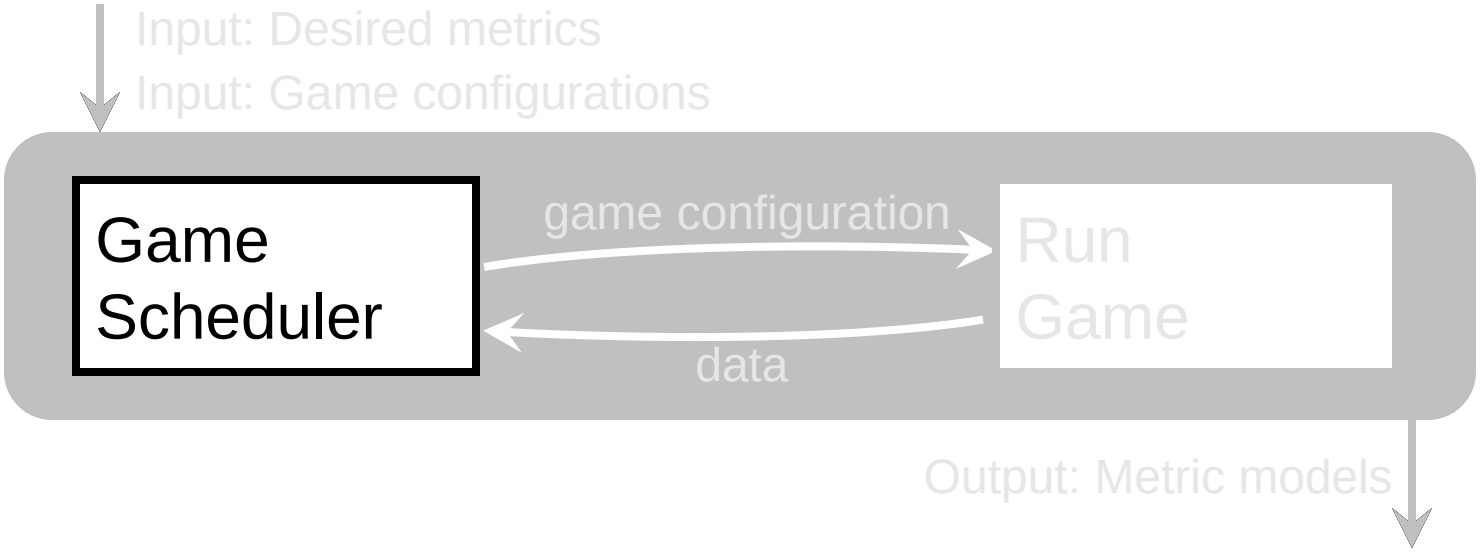
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



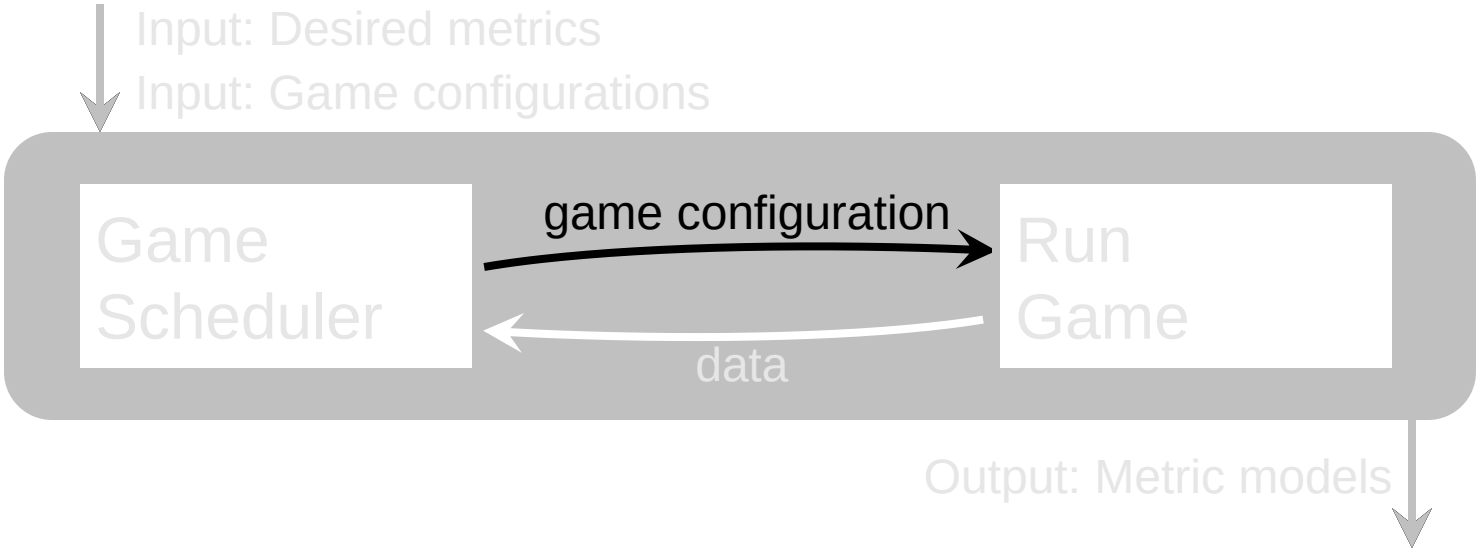
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



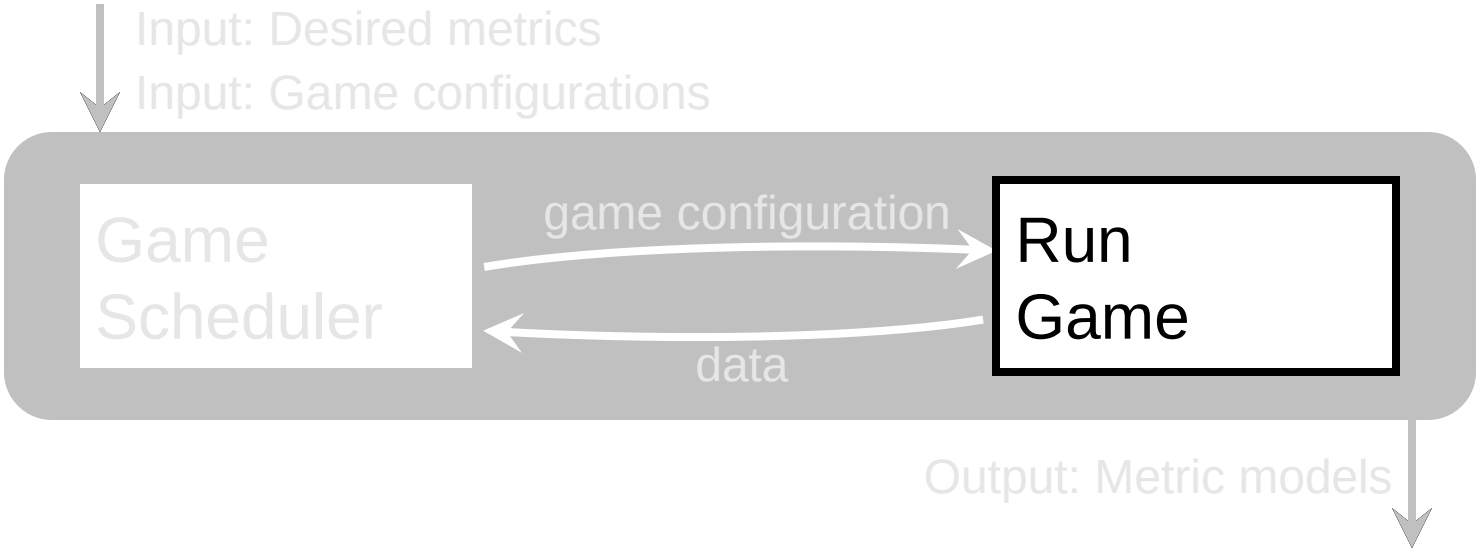
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



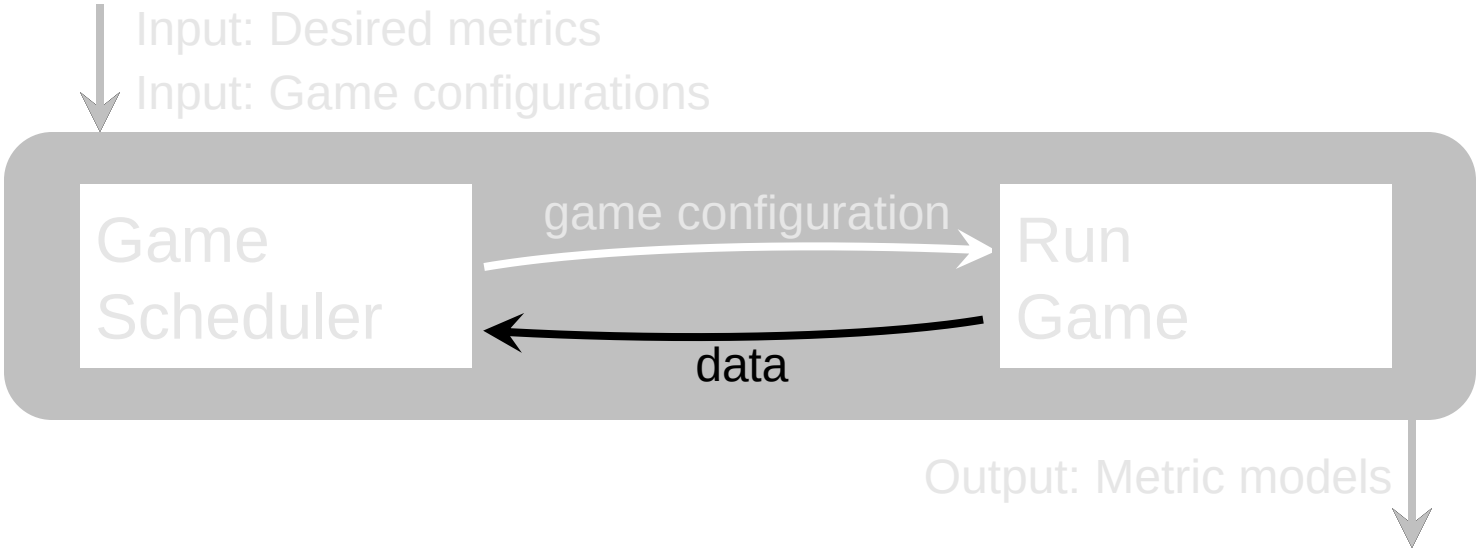
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



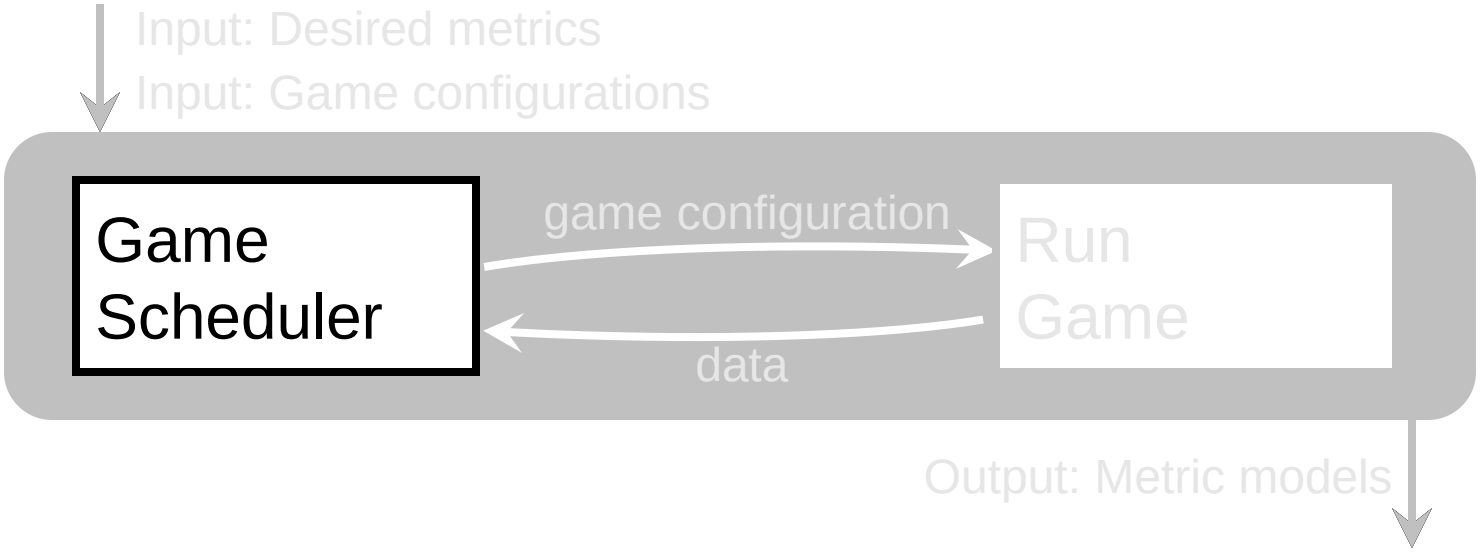
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



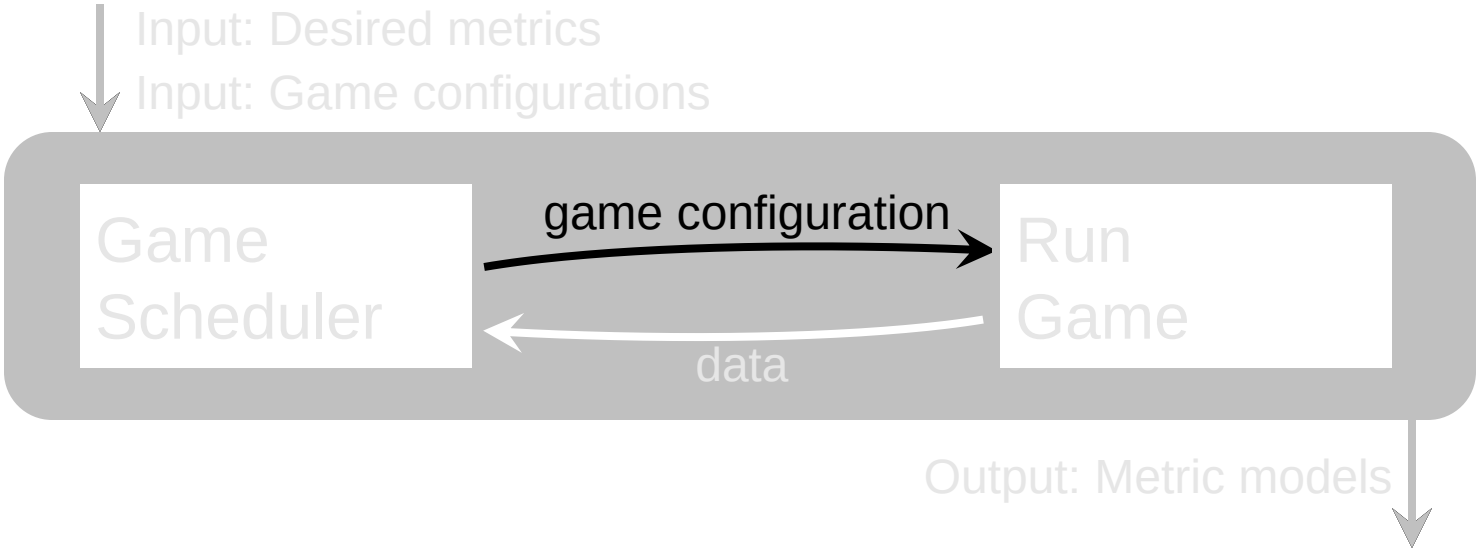
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



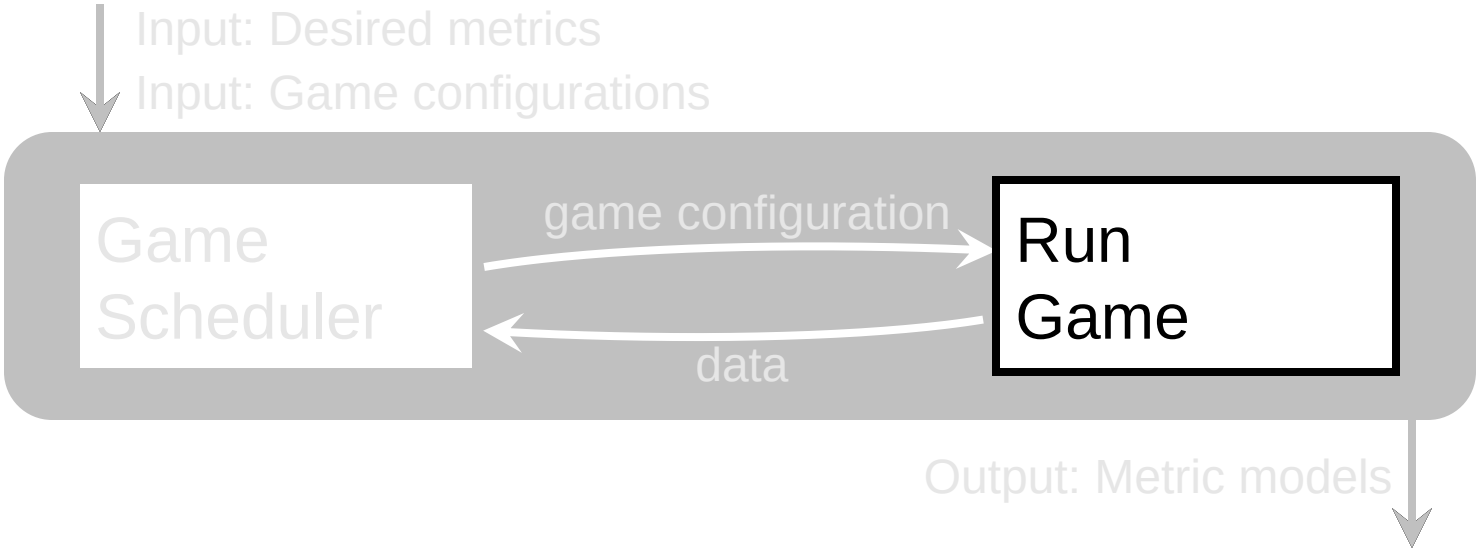
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



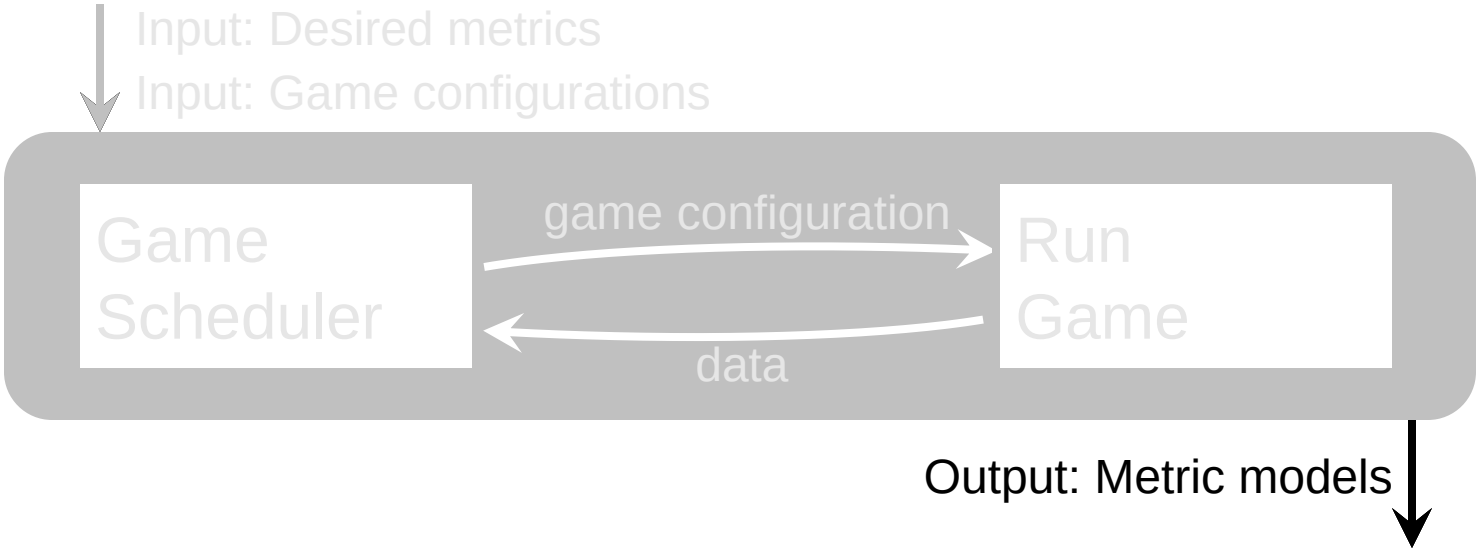
Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



Active data acquisition

- GAMELAB: autonomously collects player metrics
- Iteratively collect data to improve the estimates for each metric



Game Scheduler – Overview

- Goal: Minimize confidence intervals across all metrics using as few games as possible

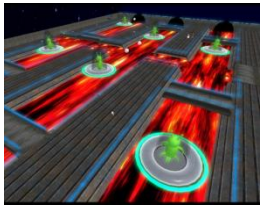
- Idea:
  - Estimate how accurate our current metrics are
    - Confidence intervals
  - Model of how player metrics correspond to game configurations
    - Game configurations = {(Game type) x (Environment) x (Game Settings)}
  - Model the decision of which game to run next as a Markov Decision Process (MDP)

Game Scheduler – Markov Decision Process

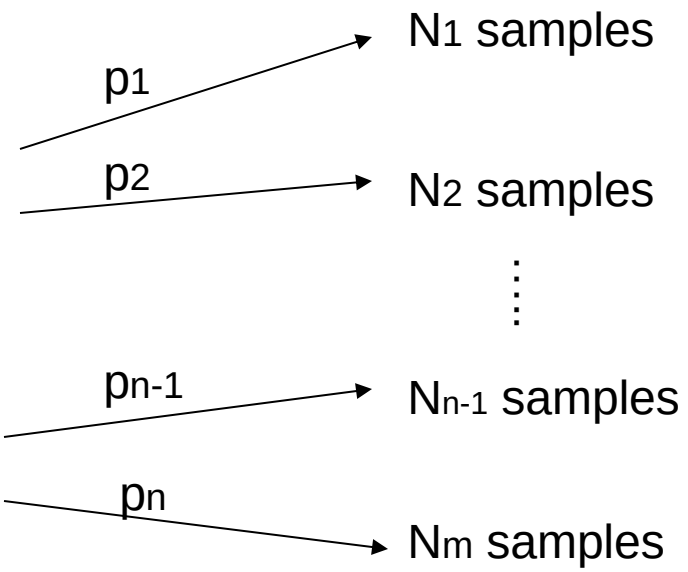
- Allows us to reason about decisions when outcomes are uncertain

Metric: Standing in narrow spaces

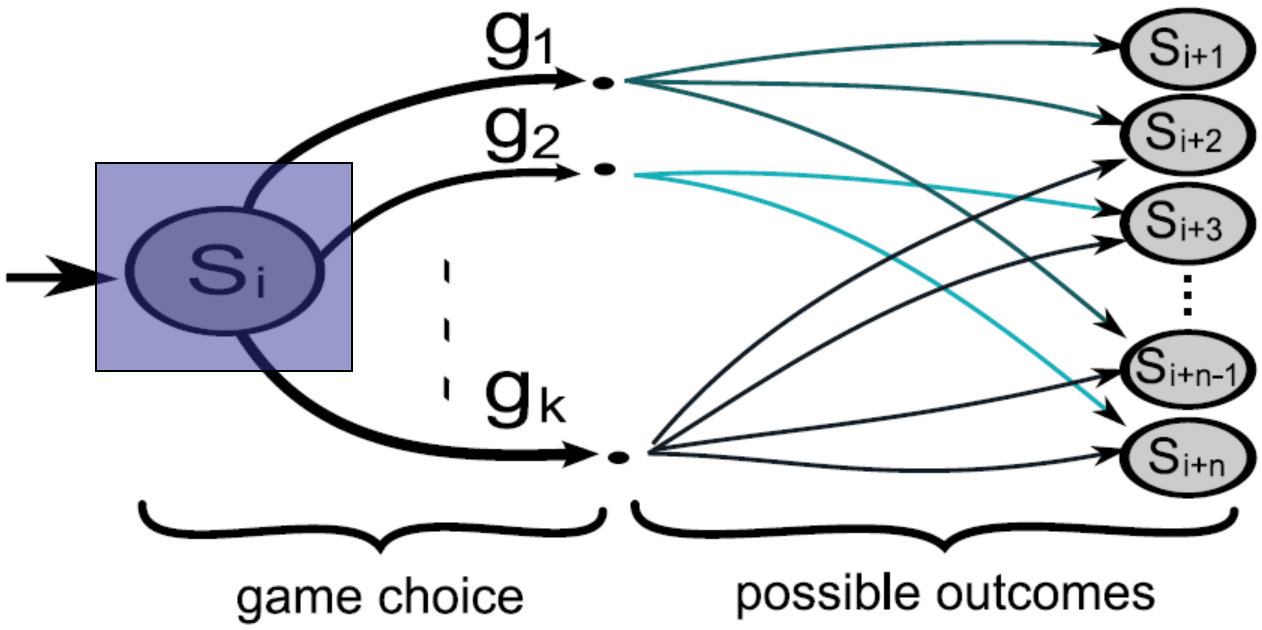
?



Game choices

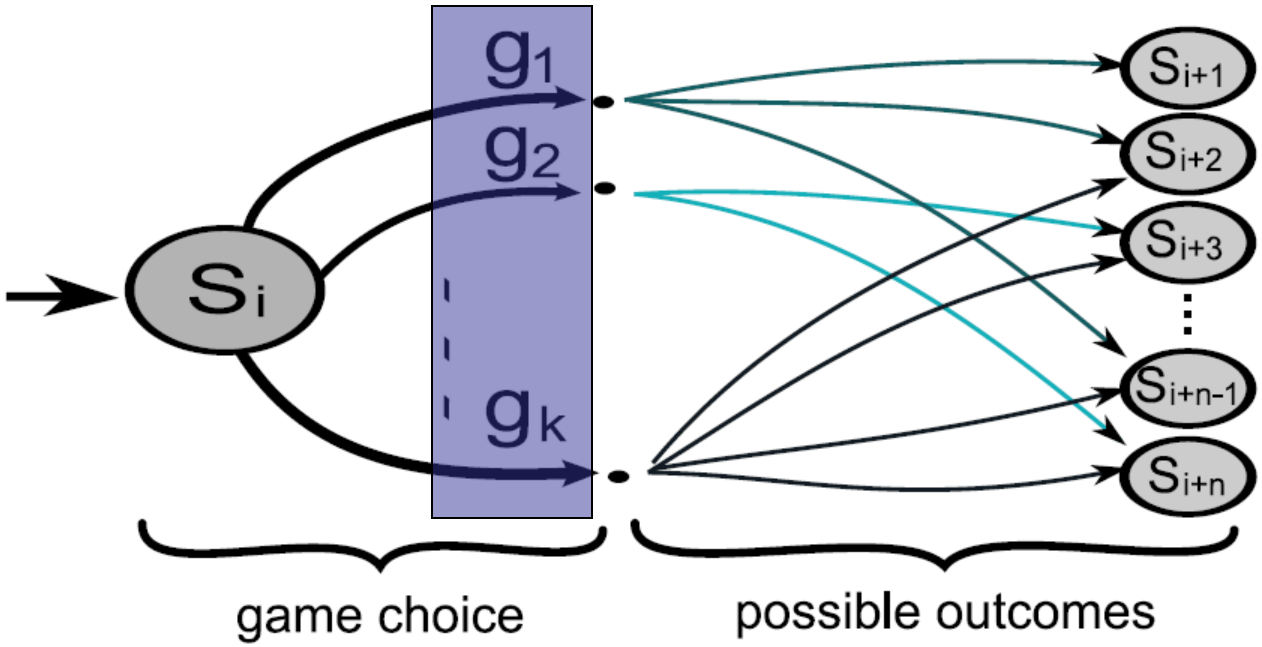


Game Scheduler – MDP Design



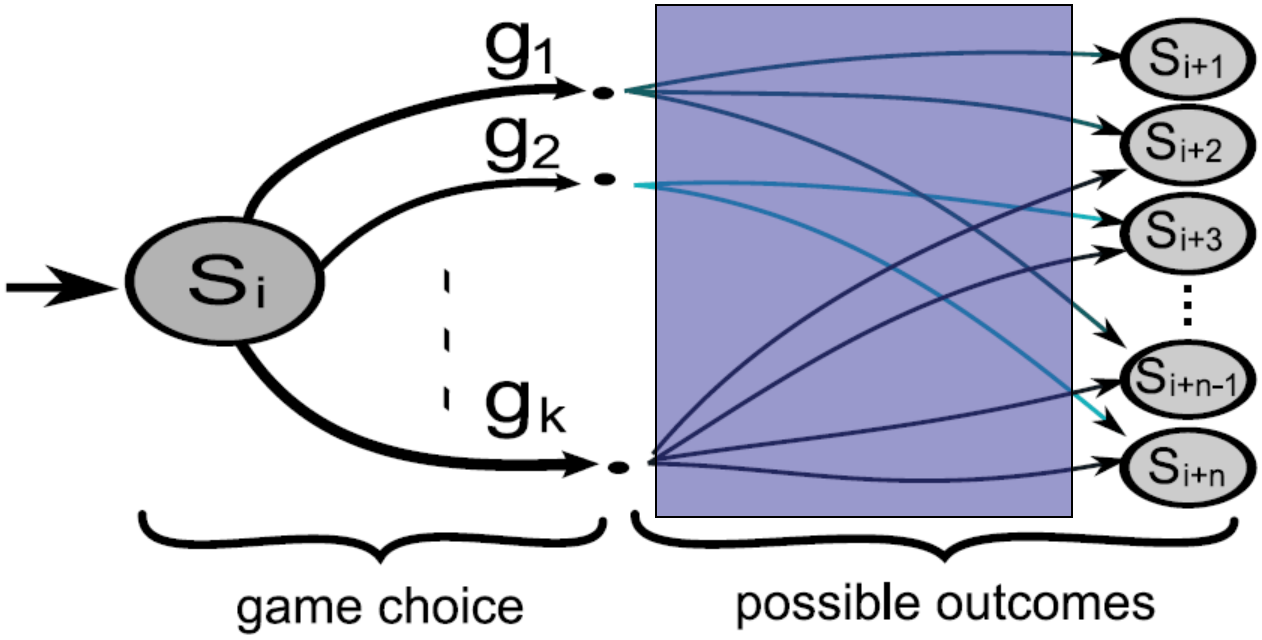
- States s_i:
  - Numbers of samples we have so far for each metric

Game Scheduler – MDP Design



- Actions g_k:
  - Game configuration to run
  - configurations = {(type) x (settings) x (environment)}

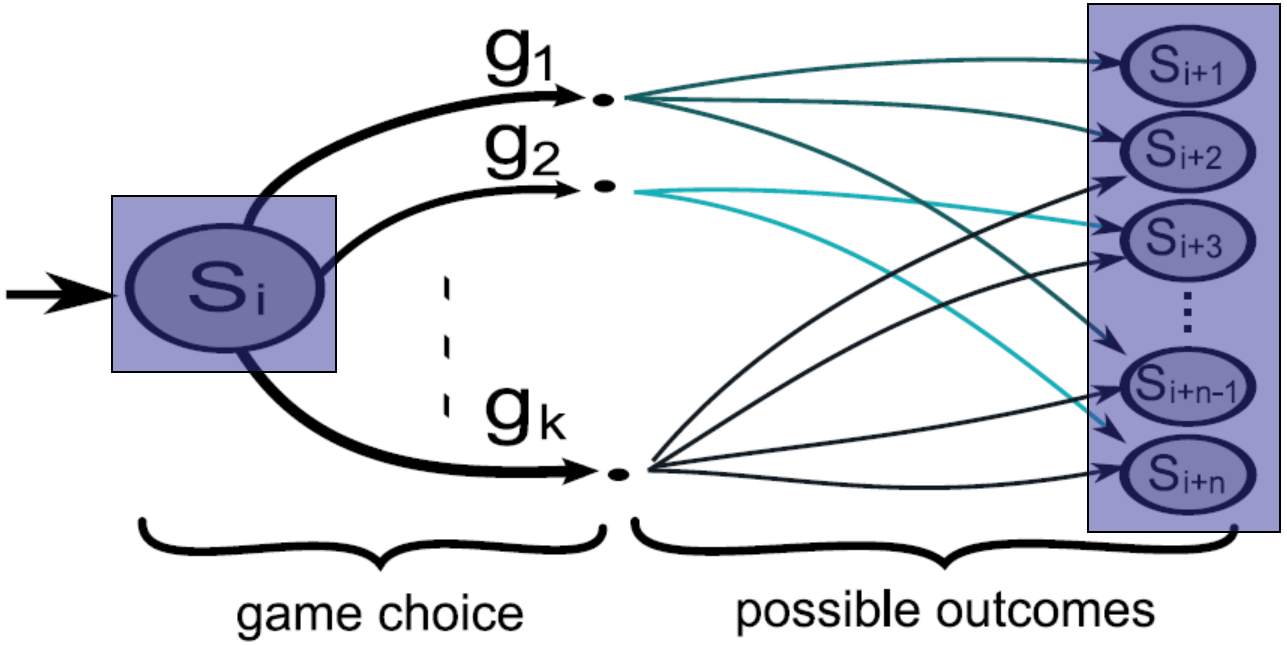
Game Scheduler – MDP Design



- Transition probabilities $P_k(s_i, s_j)$
  - Probability of receiving a set of additional samples if we run game g_k

$$P_k(s_i, s_j) = \prod_{m=1}^{M} P_k(\theta_m^a \leq p_m \leq \theta_m^b \mid \lambda_{k,m})$$

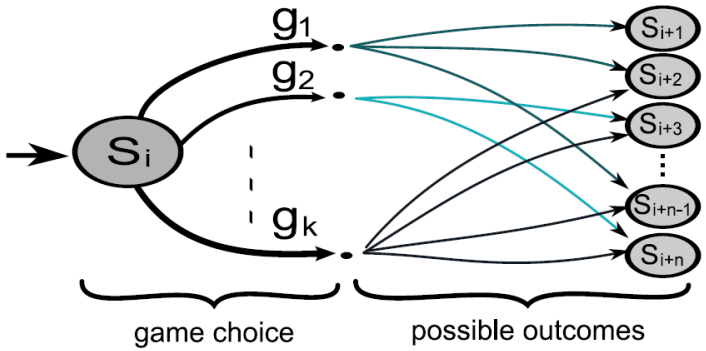
Game Scheduler – MDP Design



- Reward function R(s_i, s_j)
  - Measure of how accurate our metric estimates are in s_j

$$R(s_i, s_j) = \sum_{m=1}^{M} [CI(p_m) - CI(p_m + \theta_m)]$$

Game Scheduler – MDP Design



- Solving this MDP gives our scheduling policy
  - States are represented in a tree structure
    - Solving policy is linear in number of states
  - To keep the numbers of transitions and states tractable, we segment counts into bins
  - Depth of tree determines how far into the future we look for scheduling

Proof of Concept - Mini-games in Second Life

- Scavenger Hunts
- Configurable environments
- Collected 5 metrics

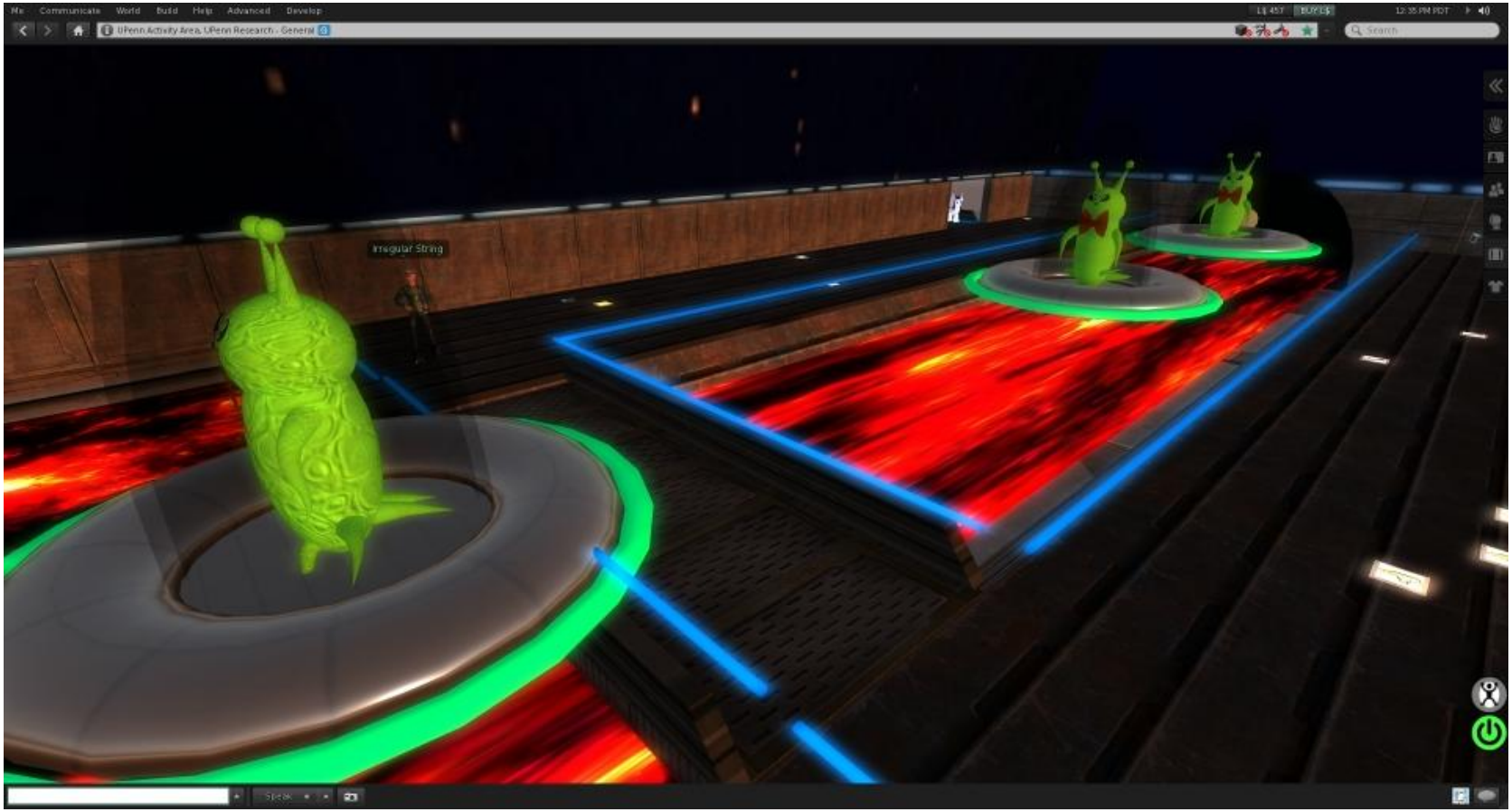
Proof of Concept - Second Life

Proof of Concept – Scavenger Hunts

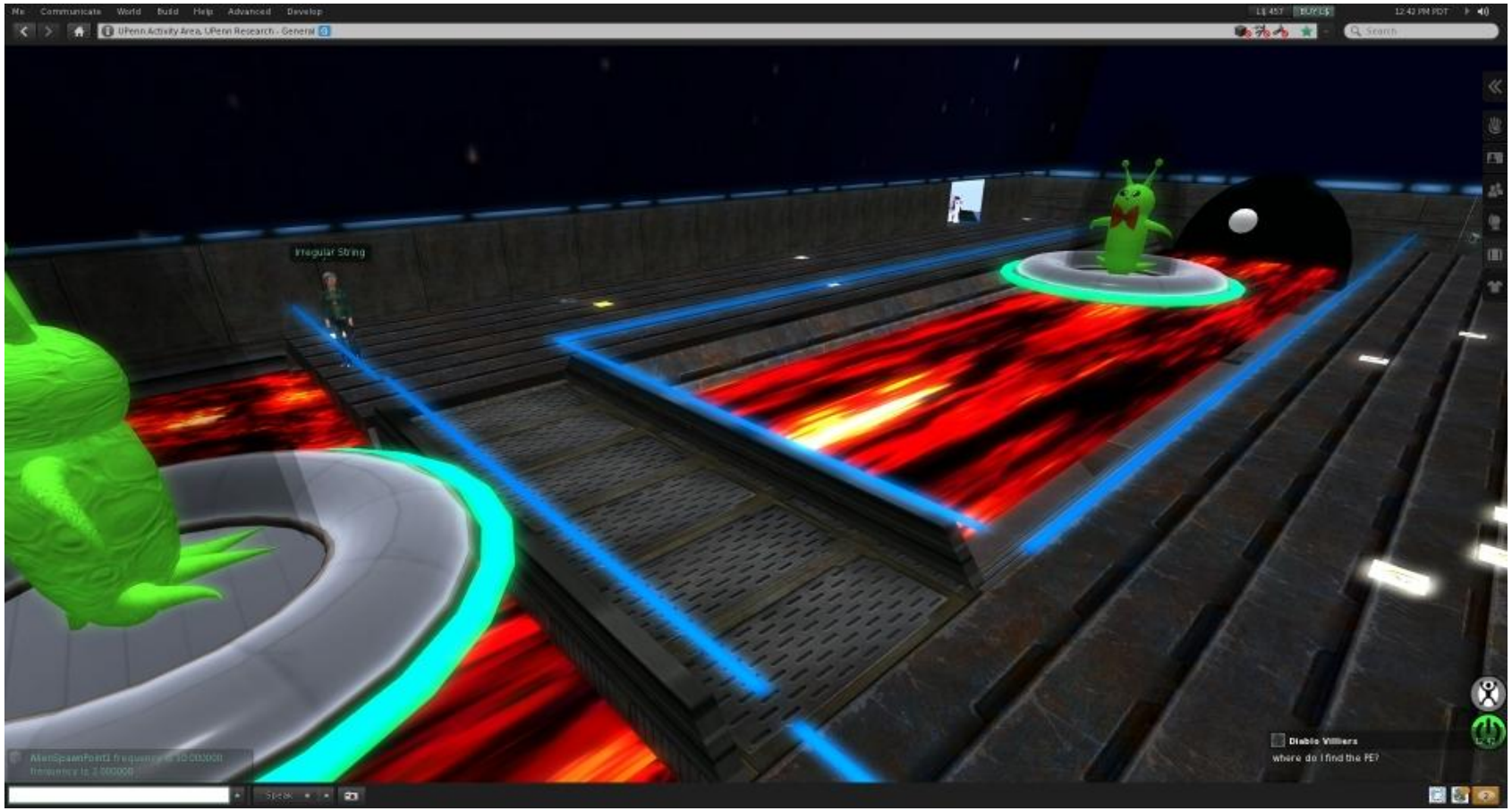
Proof of Concept – Metrics Standing distance (Wide)

Proof of Concept – Metrics
Standing distance (Narrow)

Proof of Concept – Metrics
Crossing timing (slow traffic)

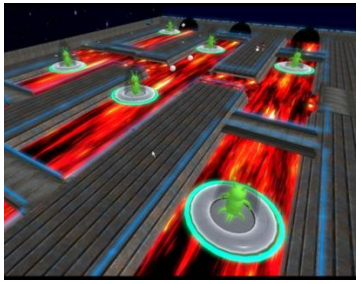


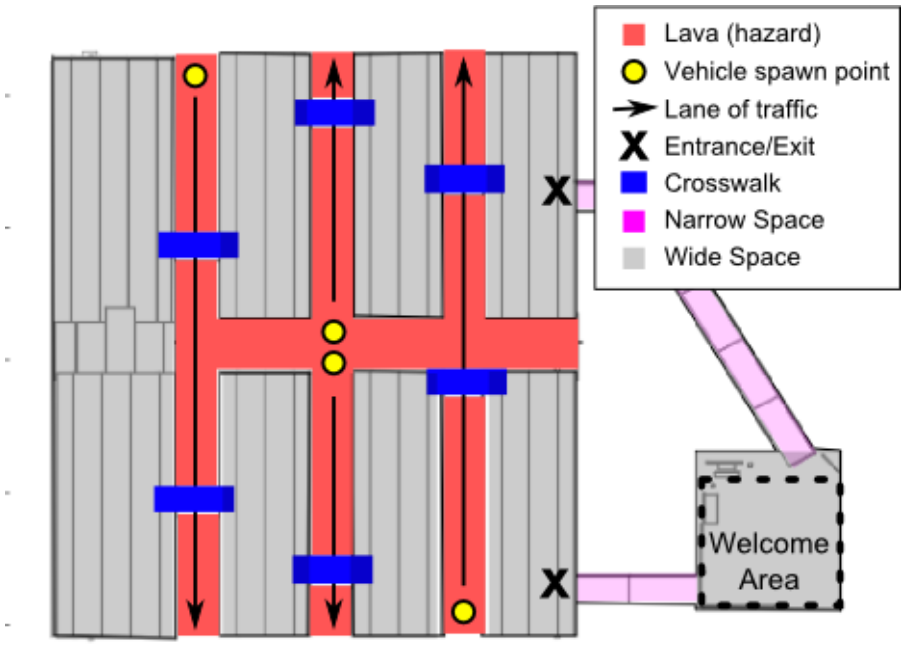
Proof of Concept – Metrics
Crossing timing (fast traffic)

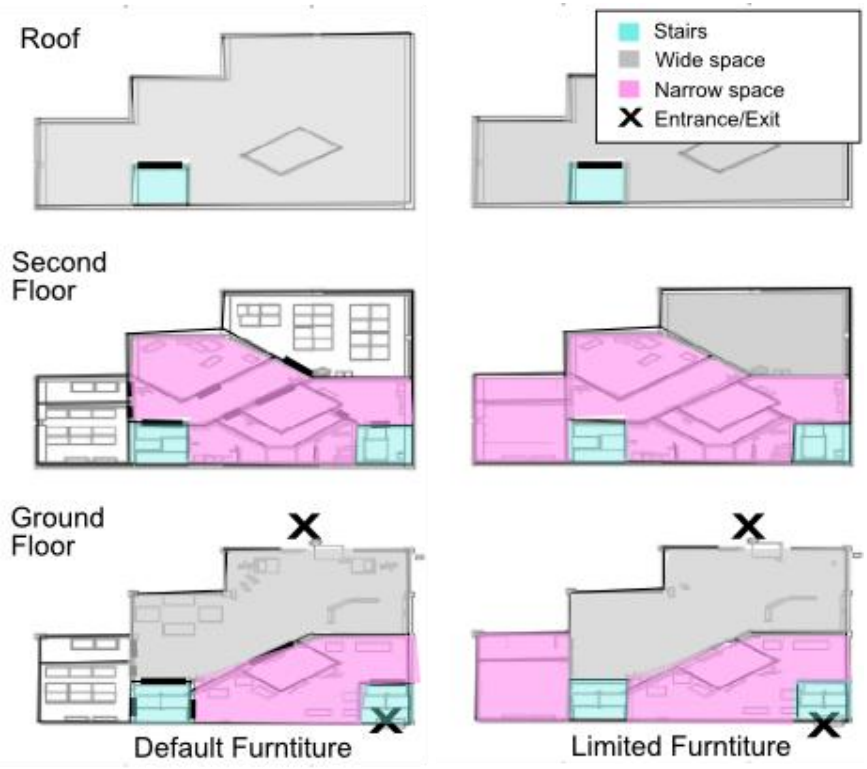


Proof of Concept – Metrics Health kit usage

Proof of Concept - Environments







Alien Teleportation Facility

Office Space

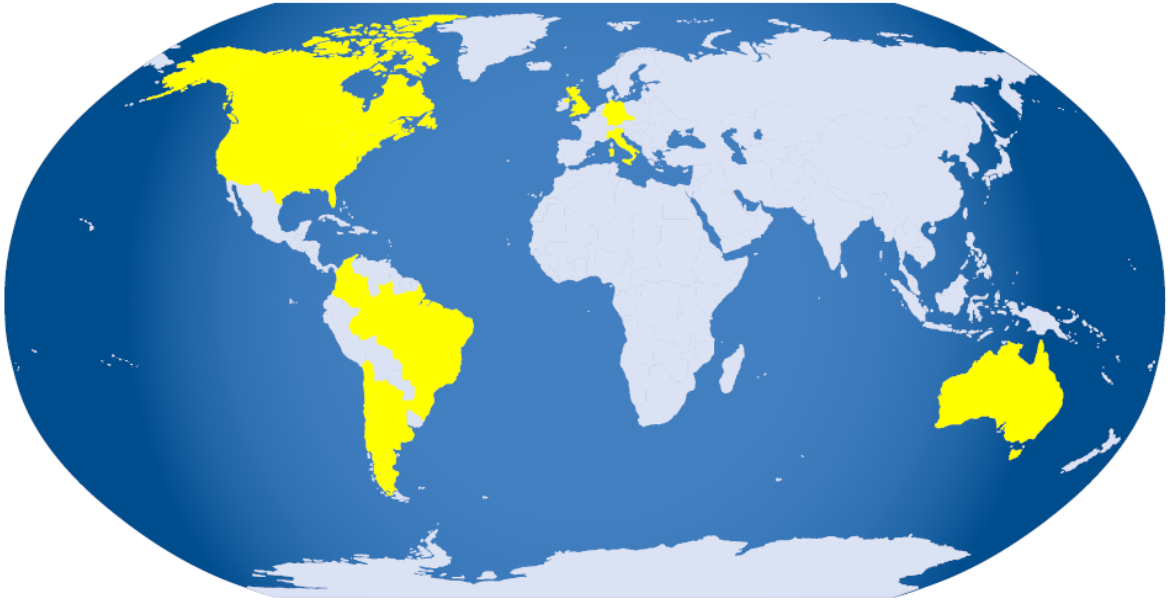
Game configurations

Experiments – Data Collection

- Game activities were advertised through the Second Life event calendar
- 5 weeks
- 70 games
- ~23 hours online recording
- 179 participants
  - 80 unique based on Avatar ID

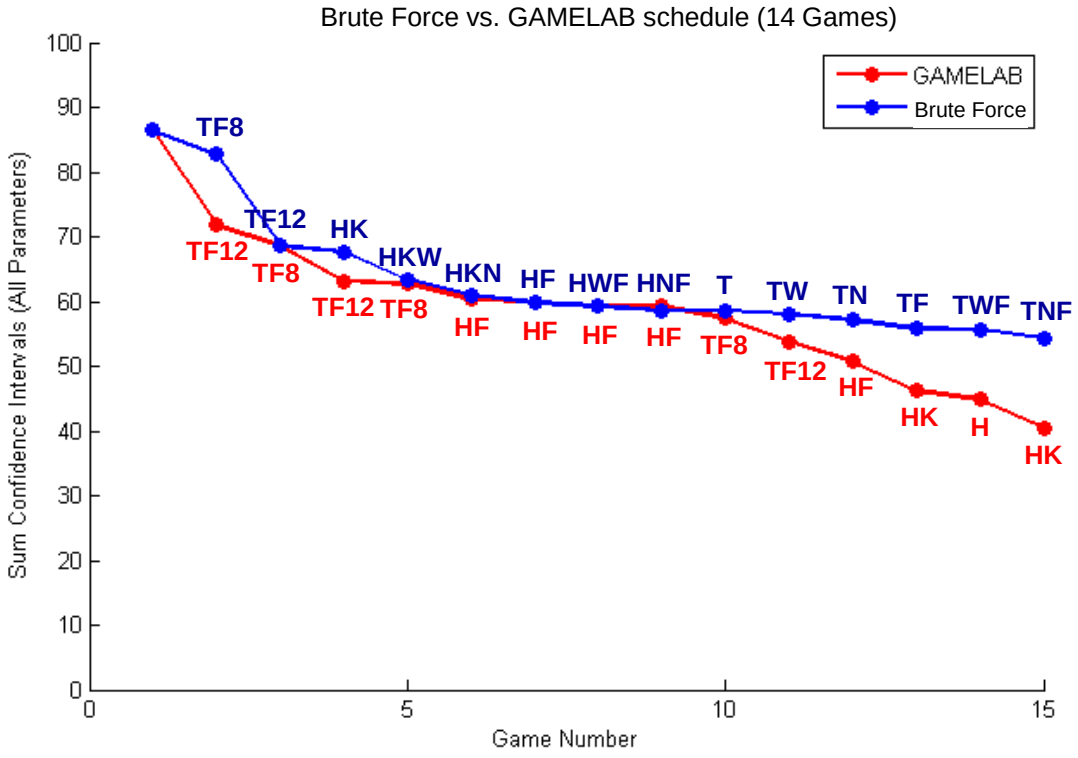
Experiments – Participants

- 39% male
- 61% female
- Ages
  - 18-25: 53%
  - 25-45: 29%
  - 45-65: 16%
  - 65+: 2%
- Residence
  - 75% US



Experiments – Effect of Scheduling

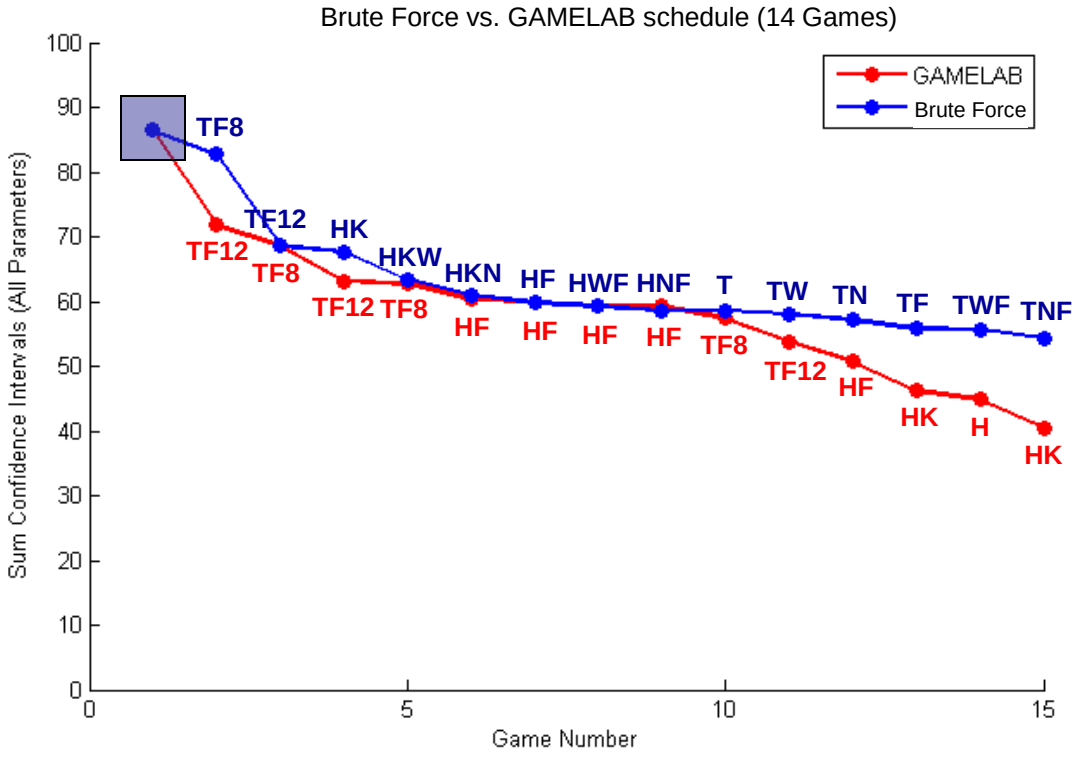
Comparison of a policy against a brute force schedule



TF8 - Slow Traffic
TF12 – Fast Traffic
HK – Kit
HKW – Health Kit (Wide Only)
HKN – Health Kit (Narrow Only)
HF – Health Kit (Limited Furniture)
HWF – Health Kit (Limited Furniture, Wide)
HNF – Health Kit (Limited Furniture, Narrow)
T – Tokens
TW – Tokens (Wide Only)
TN – Tokens (Narrow Only)
TF – Tokens (Limited Furniture)
TWF – Tokens (Limited Furniture, Wide)
TNF – Tokens (Limited Furniture, Narrow)

Experiments – Effect of Scheduling

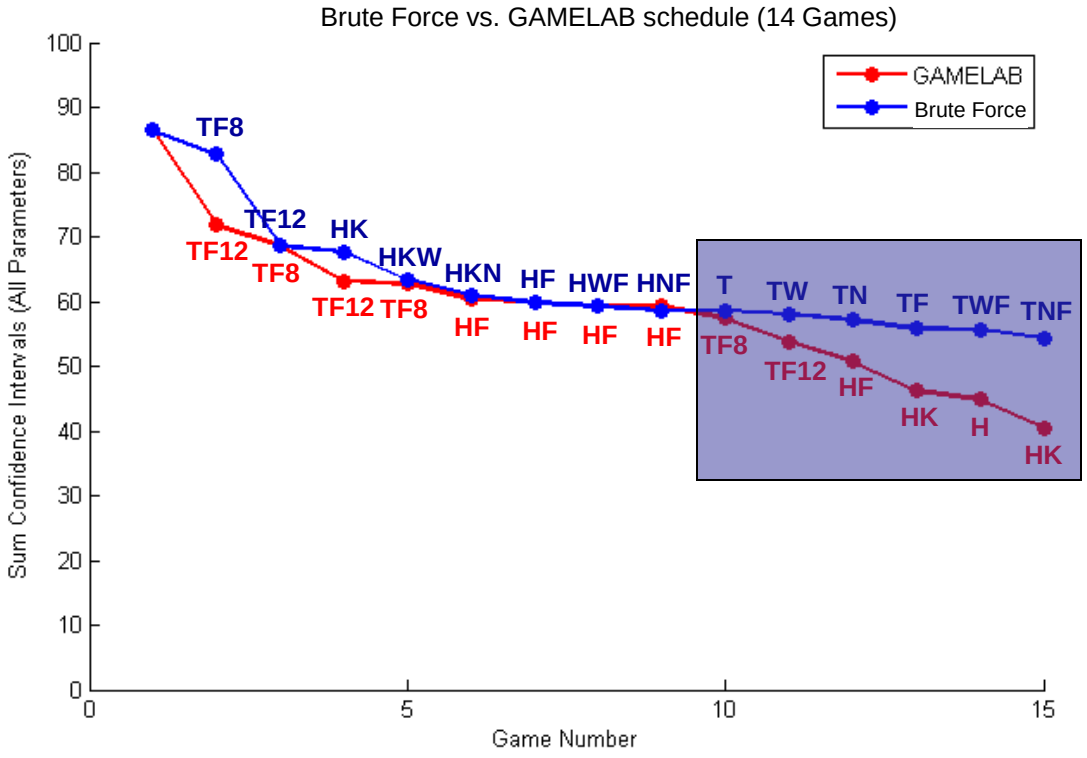
Comparison of a policy against a brute force schedule



Both schedules were initialized with the same set of 14 experiments

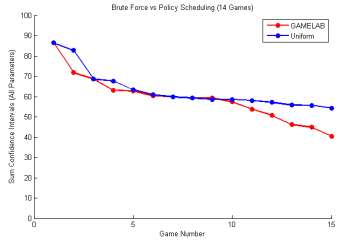
Experiments – Effect of Scheduling

Comparison of a policy against a brute force schedule



The policy outperforms the schedule

Discussion



- Potential for saving time and money
  - Brute force schedule would need to run more games for same results
- In this setup, many games did not contribute useful samples
  - Removing these -> brute force is just as good

Discussion

- Caveats
  - Scheduling design not worth it when it’s simpler to just run more games
  - Scheduling not worth it when all game configurations provide samples for all metrics equally

- Prototype shows potential for
  - reducing cost and collection time through scheduling
  - manipulating game configurations to target data needs

Discussion

Scheduler automatically reasons about differences between game configurations

Final Thoughts

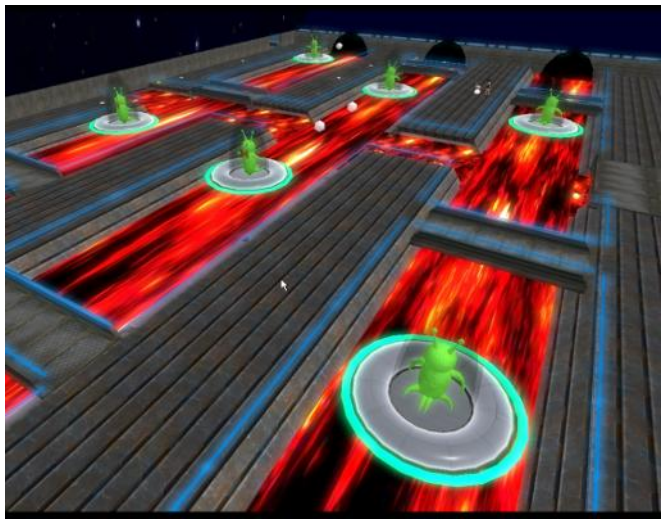
- Modeling relationship between player metrics and game configurations can by itself yield insights into player behaviors

- Using Second Life
  - Interesting mix of people
  - Varied behaviors
    - open ended
    - noisy data
  - Not a game, expectations of participants differ

Future Work

- More sophisticated examples
- More analysis
- Apply method outside of playtesting contexts
  - Decide where/when to record

Questions?



Game-based Data Capture for Player Metrics

Aline Normoyle, John Drake
University of Pennsylvania

Maxim Likhachev
Carnegie Mellon University

Alla Safonova
Disney Research, Pittsburgh

Special thanks to Benedict Brown, Ben Sunshine-Hill, Alex Shoulson, the anonymous reviewers, and NSF Grant IIS-1018486.

Proof of Concept – Blocking rooms example

Experiments – Data Collection