Hybrid Systems and Game Design

Woody McCoy, LFCPS Grand Prix 2021

Goals of the project

- Games have already been used in conjunction with Hybrid Systems, mostly in showing the efficacy of a proven system
- Hybrid systems model very well to games, though, and could solve a variety of existing design problems
- This project aims to investigate the applicability of Hybrid Systems to games by
  - Modelling 2 probable use cases
  - Proving those models to be safe and effective
  - Translating those models into a game environment

Hybrids Systems for Games (Physics)

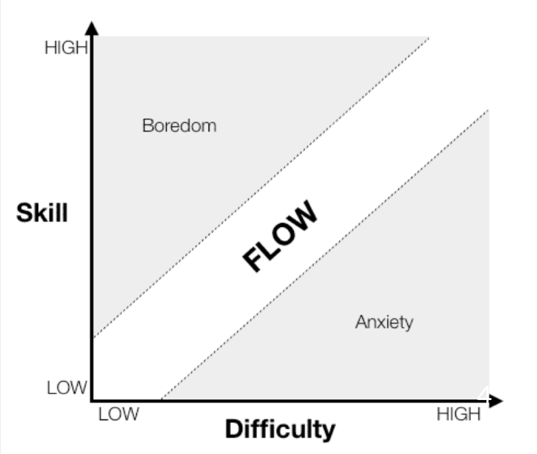
- Modern games heavily rely on physics for interactibility
- Timestep differences create a disconnect between controls and simulation for most objects
- In other words, a time-triggered hybrid system!
  - Discrete control phase
  - Continuous evolution phase (for a nondeterministic amount of time)

Standard Timestep

Fixed Timestep

Hybrids Systems for Games (Balance)

- Game balance is a difficult task, the general solution to which is trial and error
- Very challenging to know if something with a given set of parameters will disrupt the balance, and then the player's flow
    - Great fit for a hybrid system, which can try all possibilities at once
- Dynamic difficulty balancing is a currently unsolved problem, which maps well to a hybrid system



Model 1 - AI survival

Models an AI that is attempting to keep itself alive

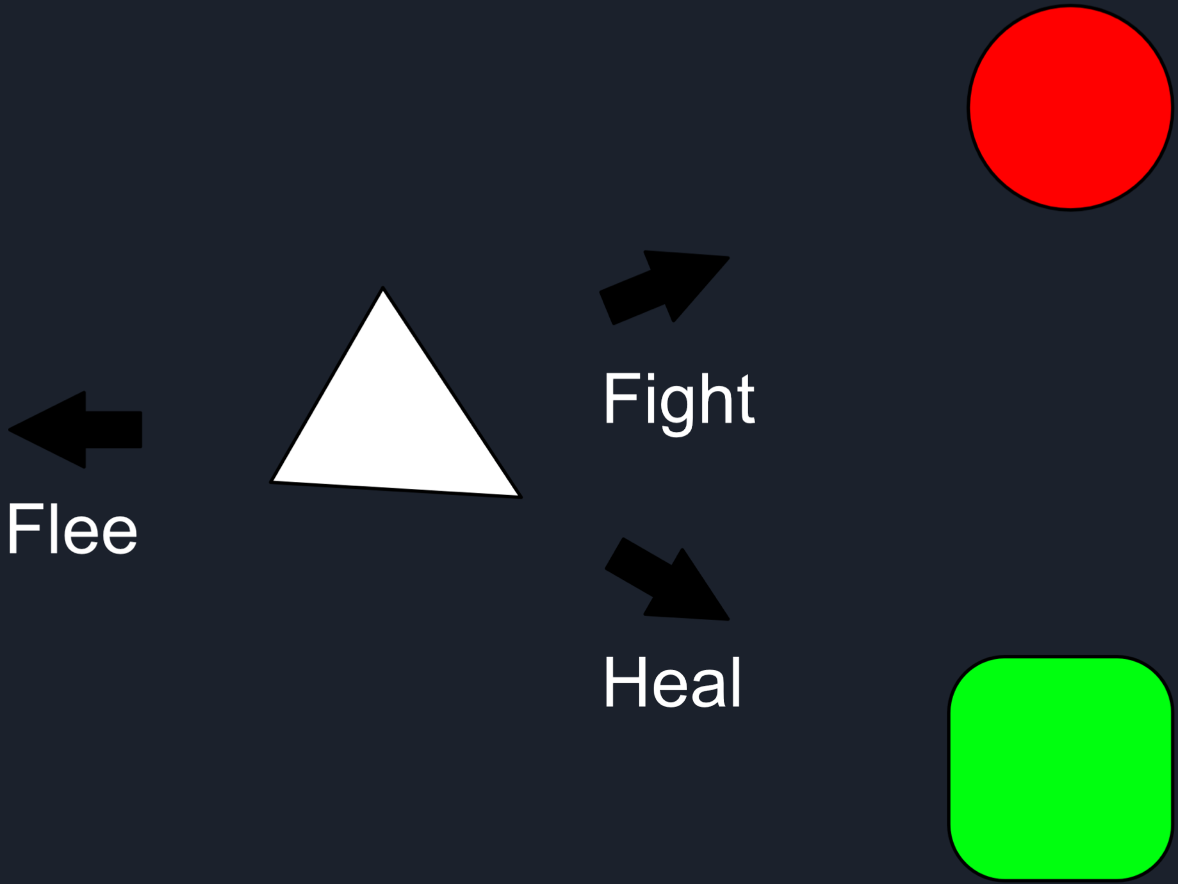
3 entities in the world

- Player AI, which must keep it’s health above a certain threshold
- An enemy, which chases the player and attempts to attack them
- Health packs, which return some amount of health to the player

Simplifications

- Distances are considered instead of coordinates
- Only 1 of each entity is considered

Model 1 - AI survival



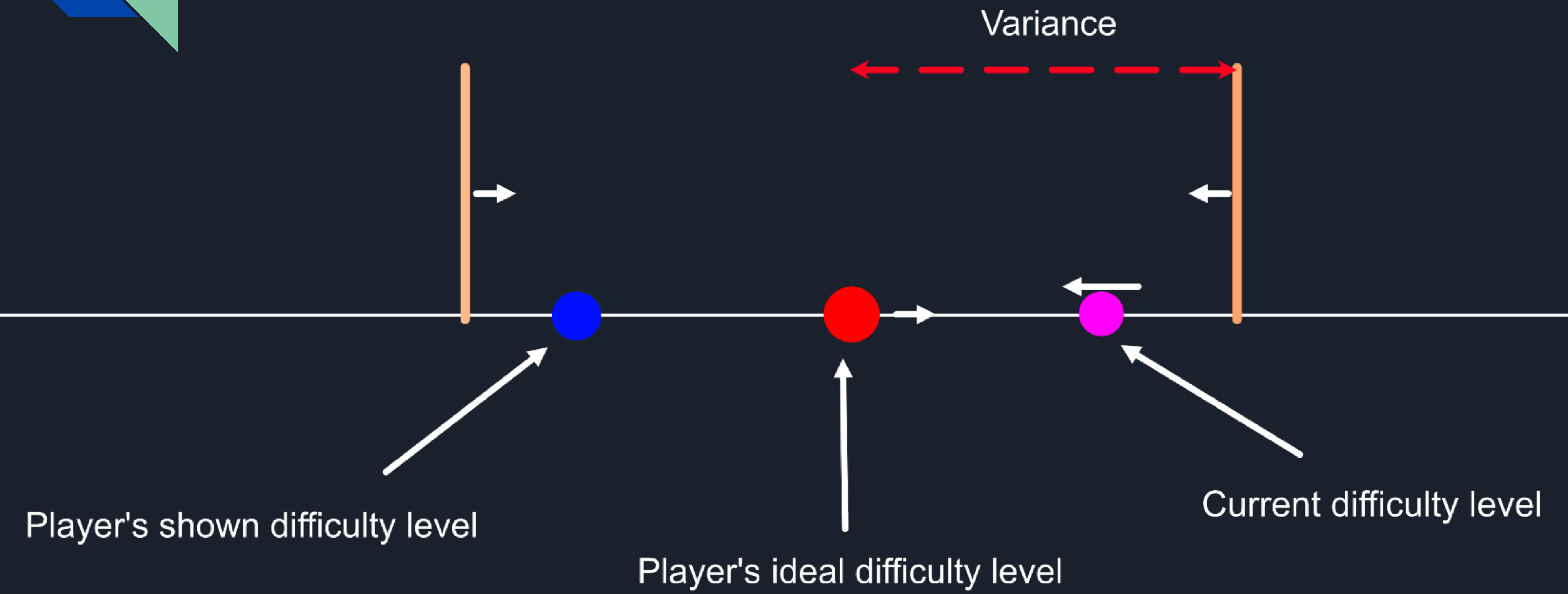
Model 2 - Dynamic Difficulty Balancing

Model of a player being judged on their performance using a heuristic, with a goal of setting a difficulty value to match the player's ideal difficulty value

- The model assumes that the player's current scoring is made with a heuristic - it will likely be inaccurate, but bounded within some variance
- The player's actual, ideal score can also be moving as the player improves at the game
- Lastly, the model has a maximum speed with which it can update the difficulty

Given these parameters, the model makes sure that the game's difficulty also lies within the variance of the ideal value.

Model 2 - Dynamic Difficulty Balancing



Implementation

The full translation of the models into code!

- Roughly 500 lines of model code, with 250 lines of supporting code (UI, Animation, Setup, etc.)
- Agents in the game choose and run actions non-deterministically, just like their models
- The AI survival agent attempts to play the game that the model describes, while the dynamic difficulty balancer attempts to balance the game for the AI

Demo!

Results and Conclusion

- Successfully model, proved, and implemented the designs I set out to
- Final implementation works - the AI keeps itself alive, and the balance eventually reaches a stable area around 40-50
- Applicability
  - Implementation was fast, but modelling was slow
  - As of right now, most likely not possible for a real production pipeline
- Design-wise, this modelling would have paid dividends in a complicated project
  - The design safety is a very appealing argument
  - Hybrid systems for balancing (especially dynamic) are likely worth the upfront modelling investment
  - With a more dedicated, general purpose tooling, this could easily find industry use