

Lecture 9

Gameplay Modeling

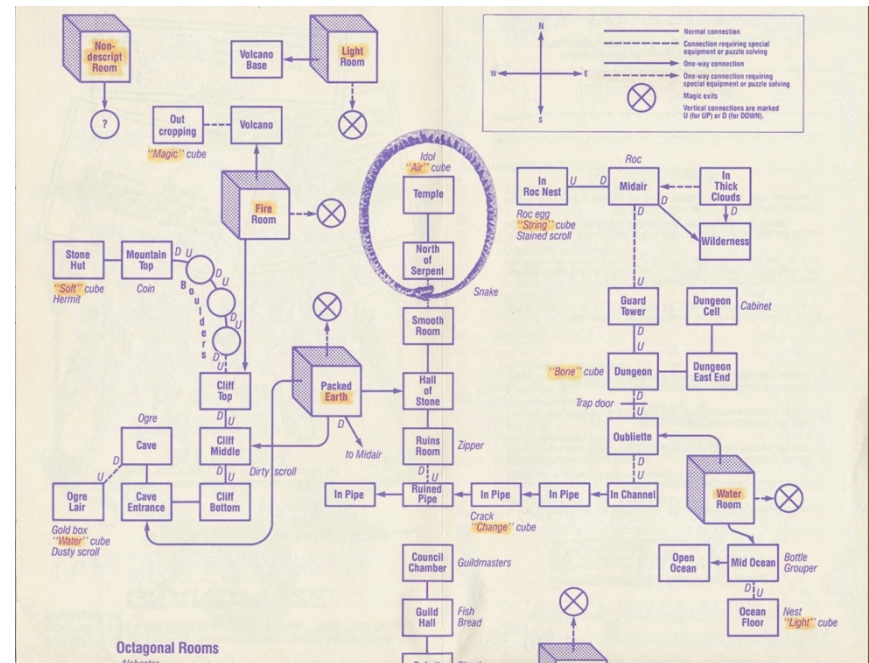
Next Week: Nondigital Prototype

- No software involved at all
 - Board game
 - Card game
 - Something different?
- Goal is to **model gameplay**
 - How? Nondigital/digital is very different
 - Model will be far removed from final result
 - What can we hope to learn from this?



Discrete Progression

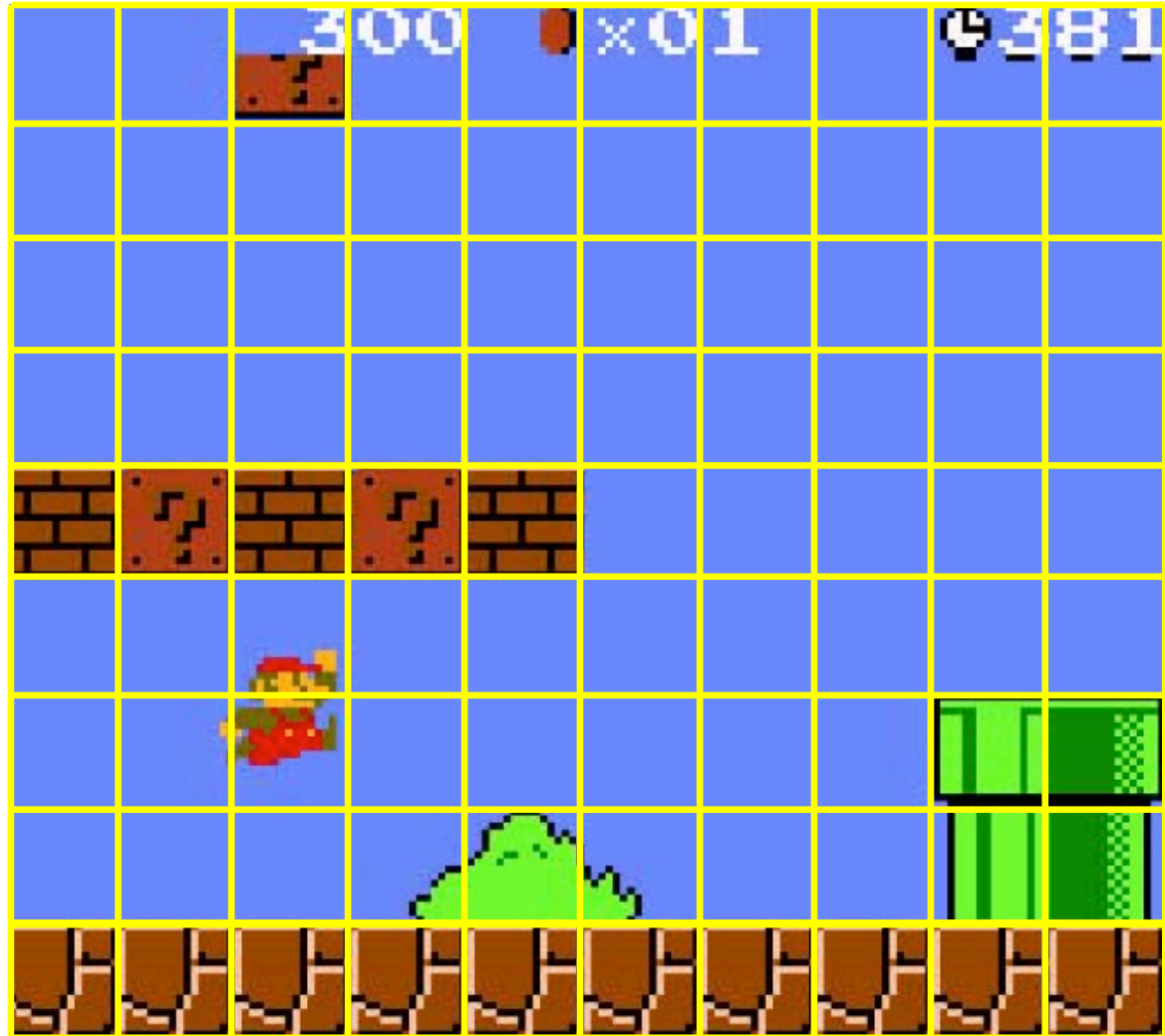
- Design is **discretization**
 - Impose flow chart on state
 - Each box is an **equivalence class** of game states
- **Spatial Discretization**
 - Contiguous zones
 - **Example:** past a doorway
- **Resource Discretization**
 - Range of resource values
 - **Example:** build threshold



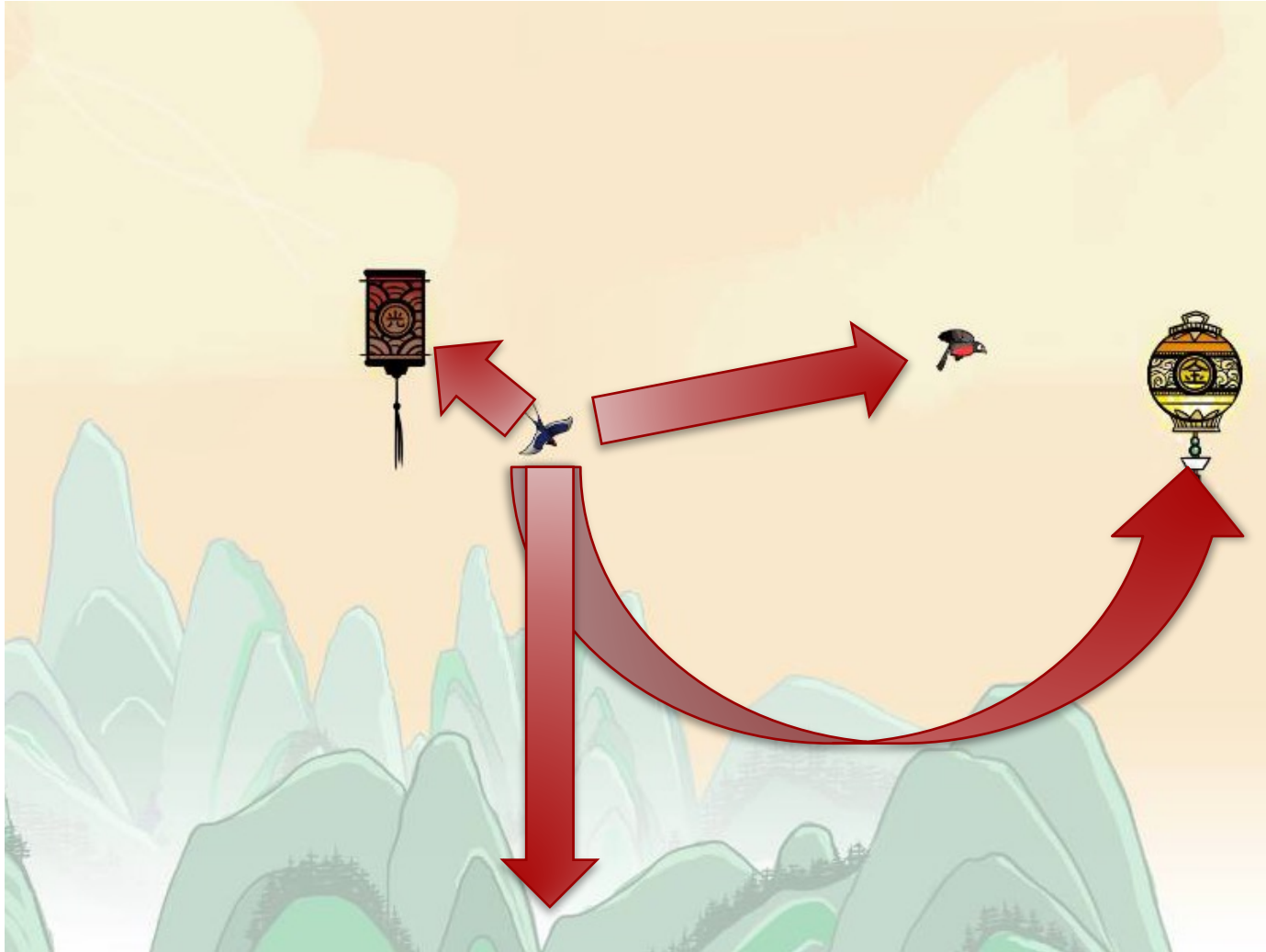
Discretizing Spatial Locality



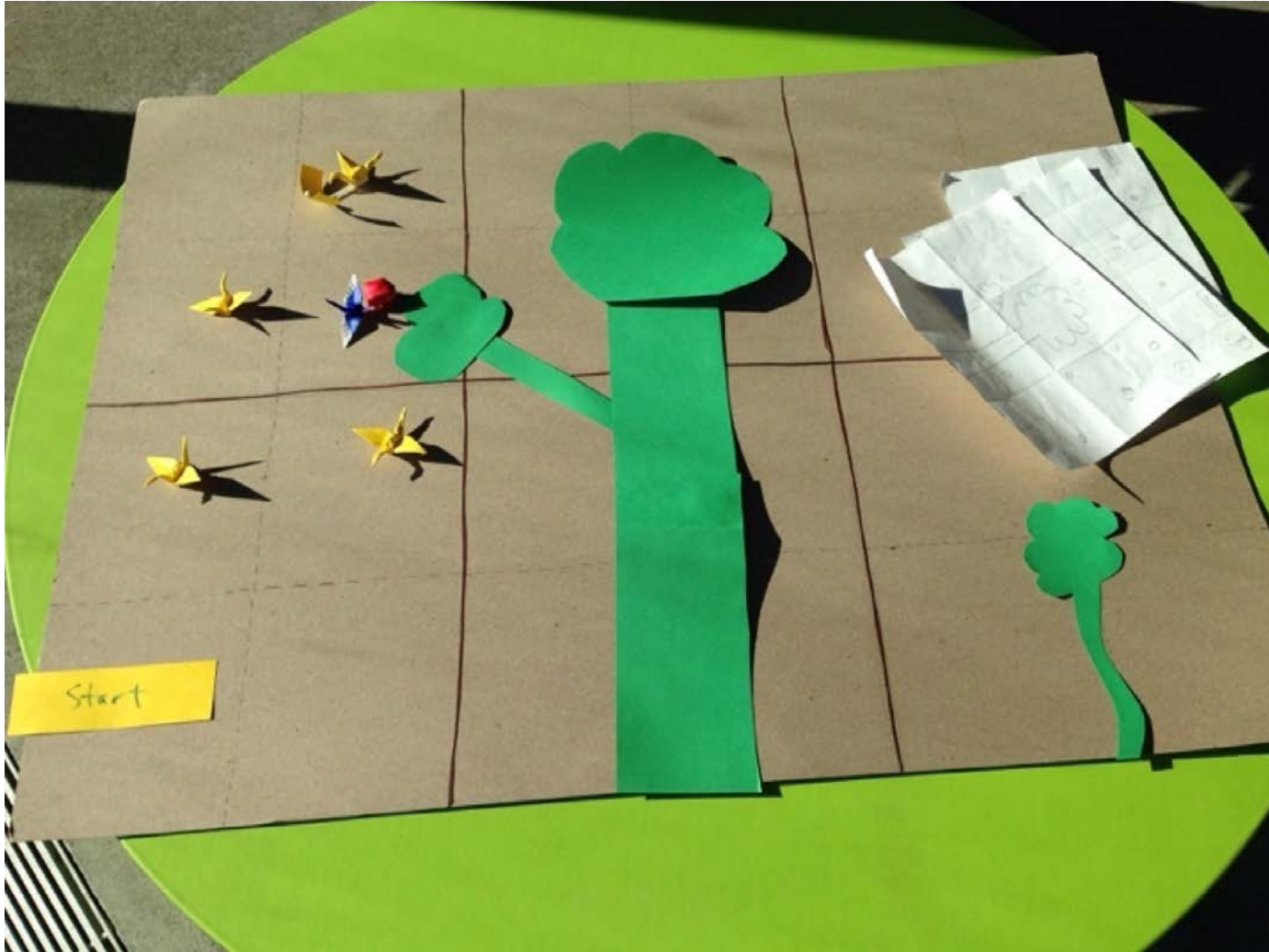
Discretizing Spatial Locality



Discretizing Spatial Locality



Discretizing Spatial Locality



Nature of Discretization

- State must be **unambiguous**
 - Must be an accurate, precise way to determine state
 - **Example:** string to measure distance in a wargame
- Actions must be **significant**
 - May correspond to several animation frames
 - **Example:** movement and attack in single turn
- Mechanics must have **compact interactions**
 - Avoid mechanics that depend on iterated interactions
 - **Example:** physics is *iterative* and hard to discretize

Discretization and Turns

- Discretization requires *turns*
 - Represent a unit of action
 - When done, game “at rest”
- Turns can be **multistep**
 - Multiple actions in a turn
 - Environmental interactions
- Turns can **alternate**
 - between other players
 - with a gamemaster
 - not at all (one player?)



Game Turn Record Track							
Turn 1 12-13 May S: 8x CH A: 4x CH VP: -2 to 16	Turn 2 14-15 May S: 8x CH A: 6x CH VP: -9 to 17	Turn 3 16-17 May S: 7x CH A: 8x CH VP: -8 to 12	Turn 4 18-19 May S: 8x CH A: 7x CH VP: -10 to 8	Turn 5 20-21 May S: 8x CH A: 5x CH VP: -13 to 4	Turn 6 22-23 May S: 4x CH A: 7x CH VP: -17 to -3	Turn 7 24-25 May S: 6x CH A: 8x CH VP: -14 to 0	Turn 8 26-27 May S: 4x CH A: 6x CH VP: -19 to -10

Game Turn Sequence Track					
Administrative Segment	1st Soviet Player Segment	2nd Axis Player Segment	3rd Soviet Player Segment	3rd Axis Player Segment	Victory Check Segment
Move First	Fight First				
Fight Second	Move Second				

General Records Track								
0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17

Victory Points Track									
-9	-8	-7	-6	-5	-4	-3	-2	-1	0
1	2	3	4	5	6	7	8	9	

Soviet Substitute Unit Display					
21 TC	22 TC	23 TC	3 GTC	2 CC	5 CC
				6 CC	

A Single Turn in *Squad Leader*

1. Rally Phase

- Damaged units heal/repair

2. Prep Fire Phase

- Choose units to attack/fire
- Cannot act in later phases

3. Movement Phase

- Move units about the board

4. Defensive Fire Phase

- Opponent (not you) acts
- Fires on units that moved

5. Advancing Fire Phase

- Moved units may now fire
- Combat strength is reduced

6. Rout Phase

- Damage units go for cover

7. Advance Phase

- Move every unit one hex

8. Close Combat phase

- Find enemies on your hexes
- Units engage in combat

A Single Turn in *Squad Leader*

1. Rally Phase

- Damaged units heal/repair

2. Prep Fire Phase

- Choose units to attack/fire
- Cannot act in later

3. Movement Phase

- Move units about t

4. Defensive Fire Phase

- Opponent (not you) acts
- Fires on units that moved

5. Advancing Fire Phase

- Moved units may now fire
- Combat strength is reduced

6. Rout Phase

units go for cover

Simulates (real-time)
player *reaction time*

7. Phase

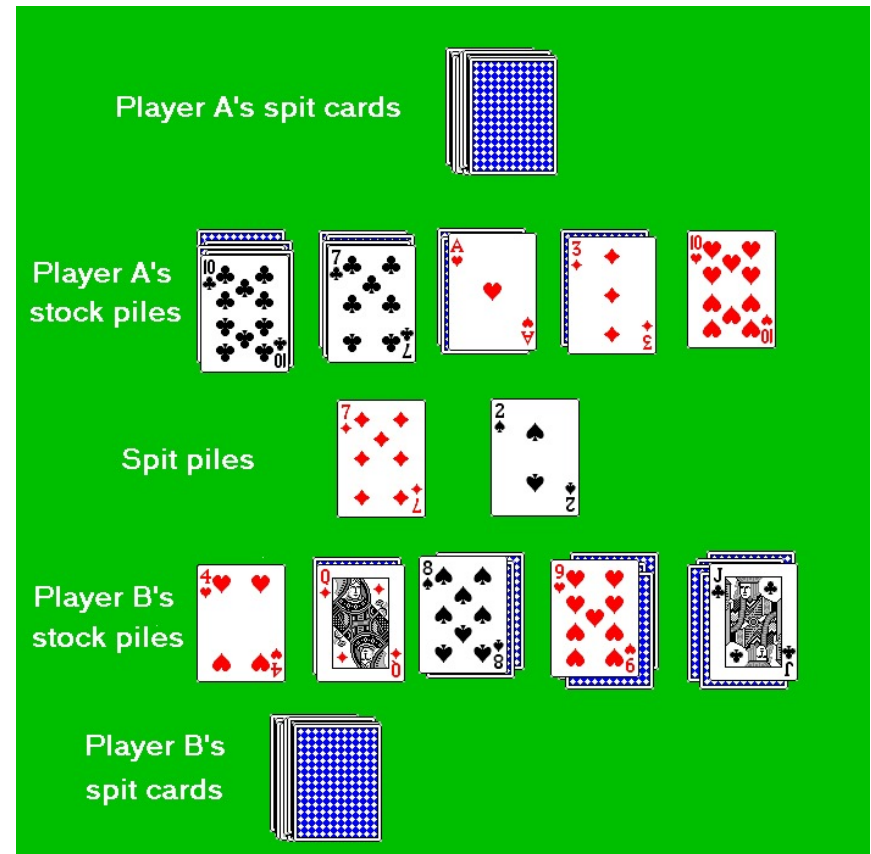
every unit one hex

8. Close Combat phase

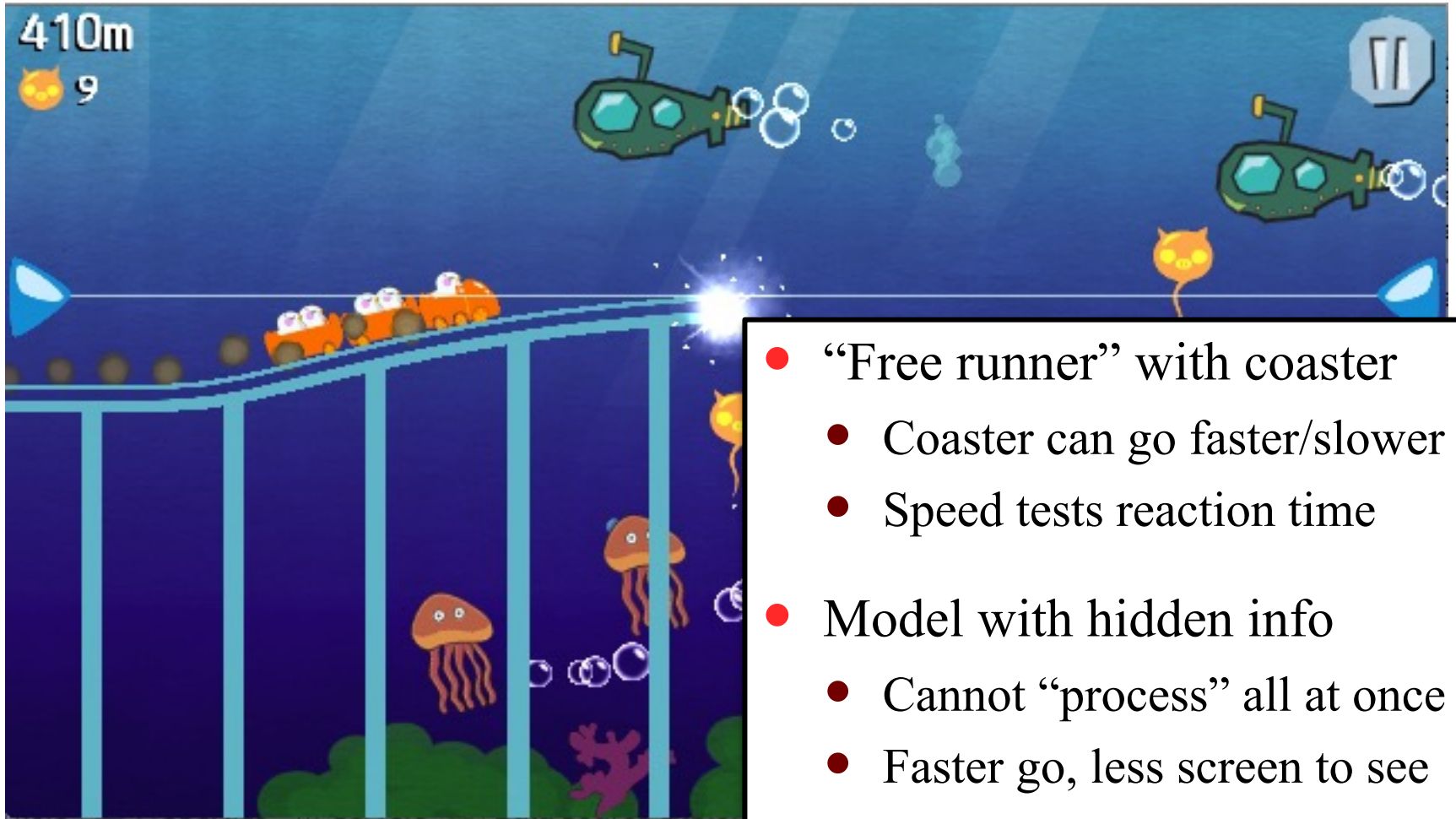
- Find enemies on your hexes
- Units engage in combat

Discretization and Reaction Time

- Allow opponent to **interrupt**
 - Action that reacts to yours
 - Played after you act, but before action takes an effect
 - Core mechanic in *Magic: TG*
- Make play **asynchronous**
 - Players still have turns
 - But take turns as fast as can
 - Conflicts resolved via speed
 - Often need a referee for aid

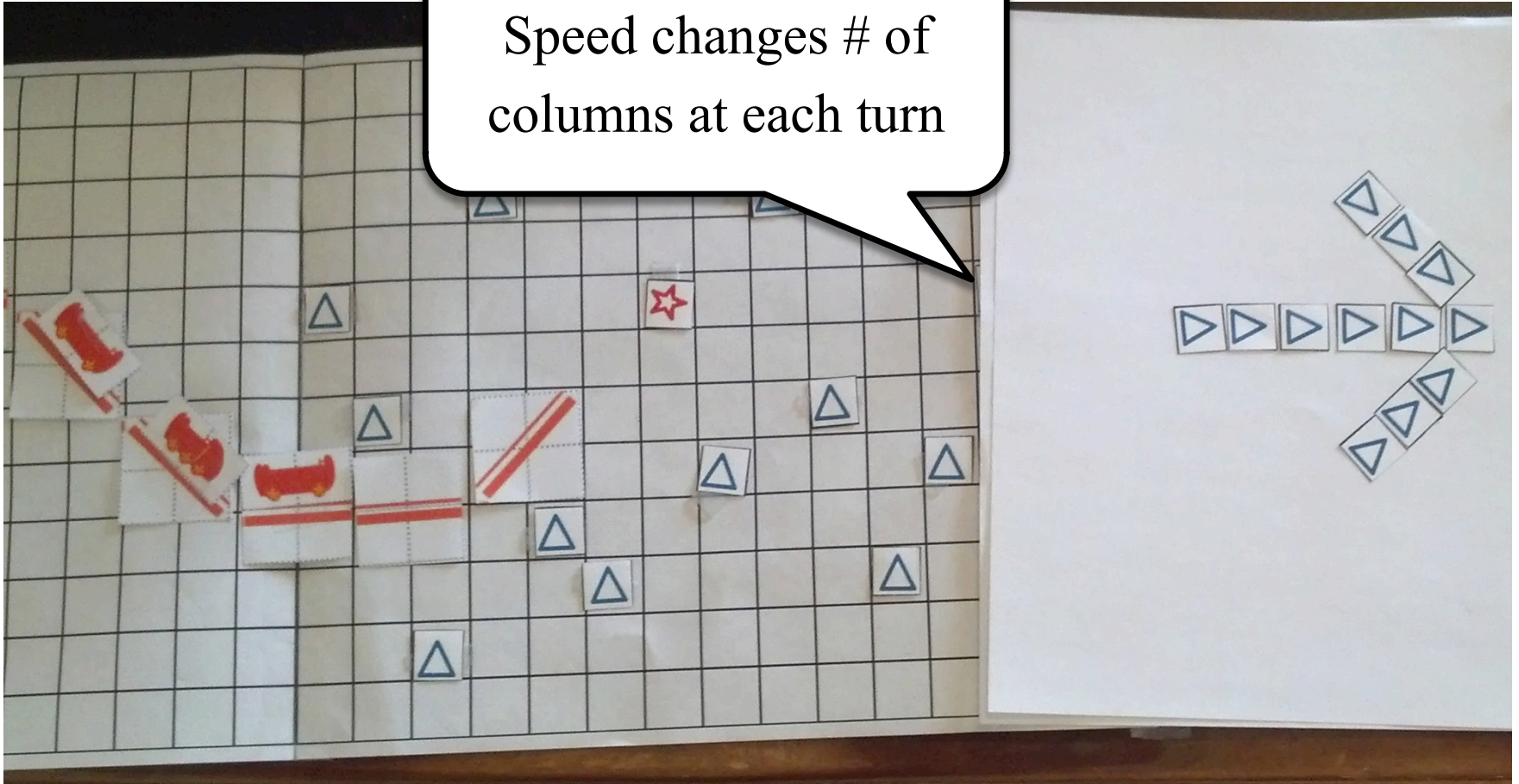


Case Study: *Runaway Rails*



Reaction Time as Hidden Information

Speed changes # of columns at each turn

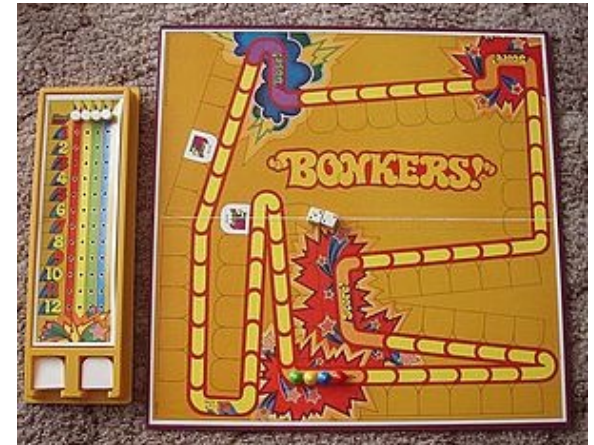


What Can We Do Discretely?

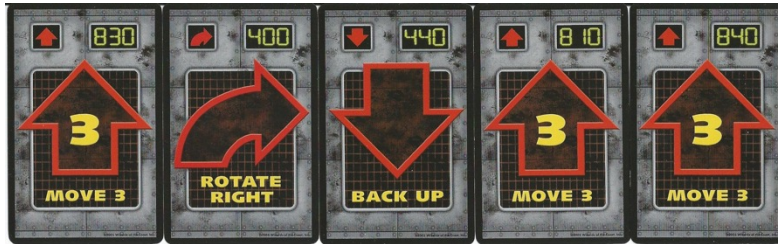
- **Evaluate emergent behavior**
 - Allow player to commit simultaneous actions
 - Model interactions as “board elements”
- **Model player cost-benefit analyses**
 - Model all resources with sources and sinks
 - Focus on economic dilemma challenges
- **Test player difficulty/usability**
 - Ideal for puzzle games (or puzzle elements)
 - Can also evaluate unusual interfaces

Evaluating Emergent Behavior

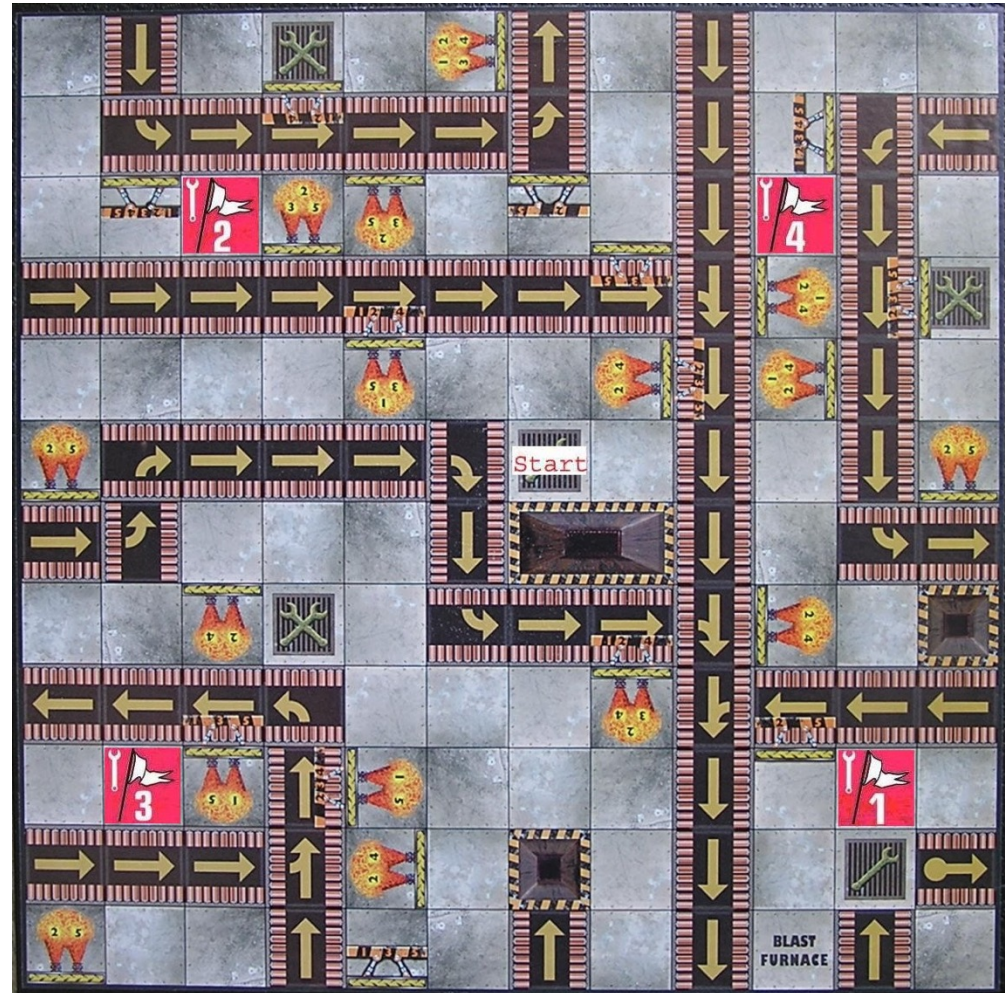
- **Recall:** coupled, context-dependent interactions
 - Requires an action and interaction
 - Or (alternatively) multiple actions
- Model interactions as “board elements”
 - Rules to follow after your action
 - May follow several in succession
 - **Examples:** *Chutes & Ladders*, *Bonkers*, *RoboRally*



Case Study: *RoboRally*



- Player “programs” robot
 - Picks 5 movement cards
 - Committed to that choice
- After each card
 - Obey board elements in order
 - Check robot collisions
- Move = board elements + cards + collisions



Multiple Actions

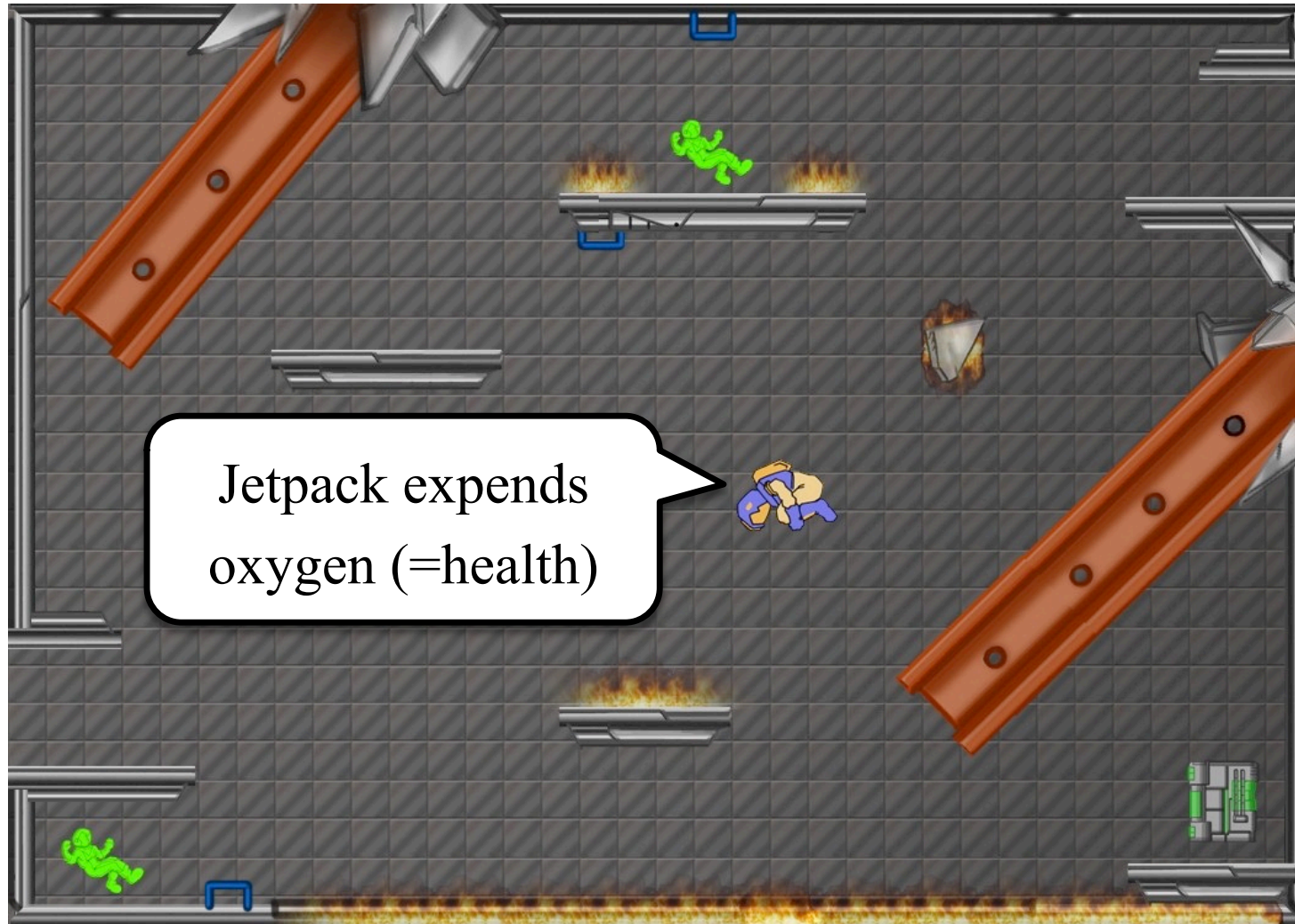
- Necessary if have no interactions
 - Allow multiple actions in a turn
 - Typically needs complex turns
- Standard method: *action points*
 - Player has so many AP per turn
 - Actions cost AP to perform
 - Turn done when AP are all spent
- Might want other restrictions
 - Groups actions into types
 - Require types in certain order
 - **Example:** no attack after move



Cost-Benefit Analysis

- Where nondigital prototypes really shine
 - Resources are very easy to discretize
 - Economic choices easily map to turns
 - Understanding dilemma challenges is important
- Some believe this is *all* of game design
 - Claim everything can be reduced to a resource
 - Common in board game adaptations of other media
 - **Example:** balance game with instability resource

Case Study: *Bounce*



Tracking Oxygen as a Resource



Case Study: *Trino*



Measuring Shapeshifting Resources



Usability Analysis

- **Unusual user-interfaces**

- Recall that actions correspond to inputs
- Some inputs are not simple buttons
- Example: touch gestures, motion controls

- **Puzzle-style games**

- Create a game with module elements (e.g. cards)
- Laying out levels creates a new game level
- Allows you to quickly change and test levels

Usability Analysis

- **Unusual user-interfaces**

- Recall the
- Some in
- Example: touch gestures, motion controls

Mainly in mobile games

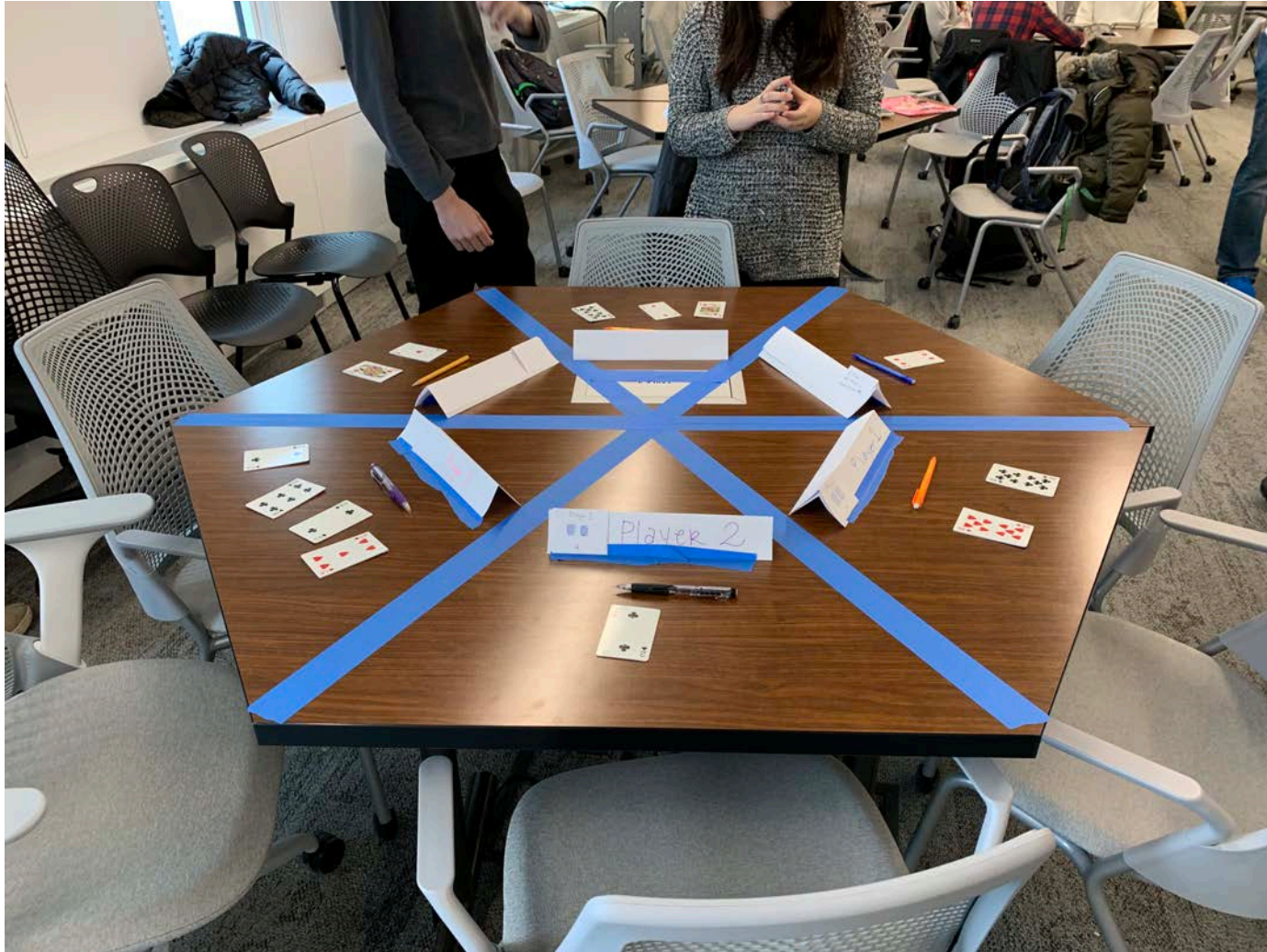
- **Puzzle-style games**

- Create a game with module elements (e.g. cards)
- Laying out levels creates a new game level
- Allows you to quickly change and test levels

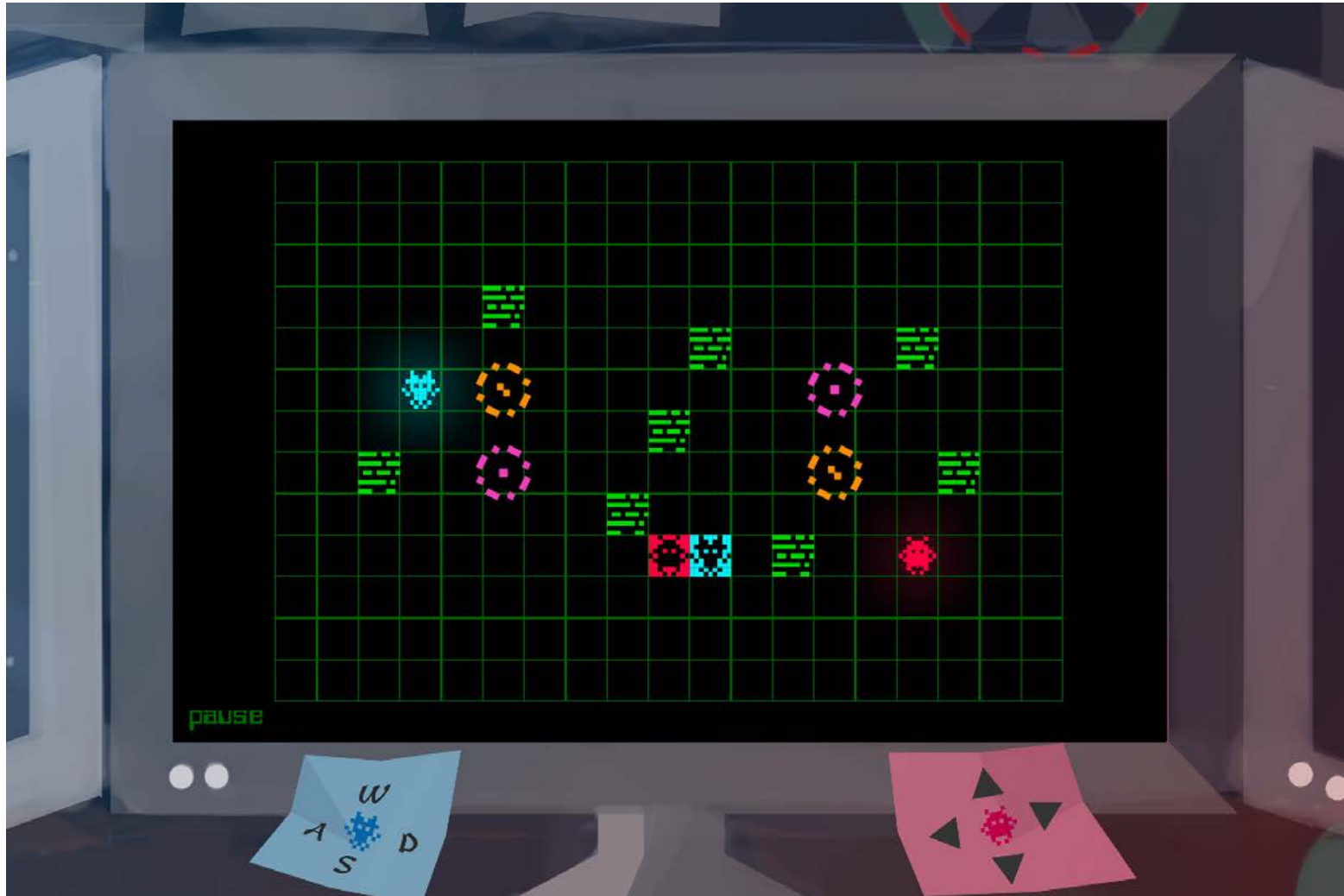
Case Study: *Family Style*



Modeling Multiplayer Restrictions



Case Study: *Operation Bitwise*



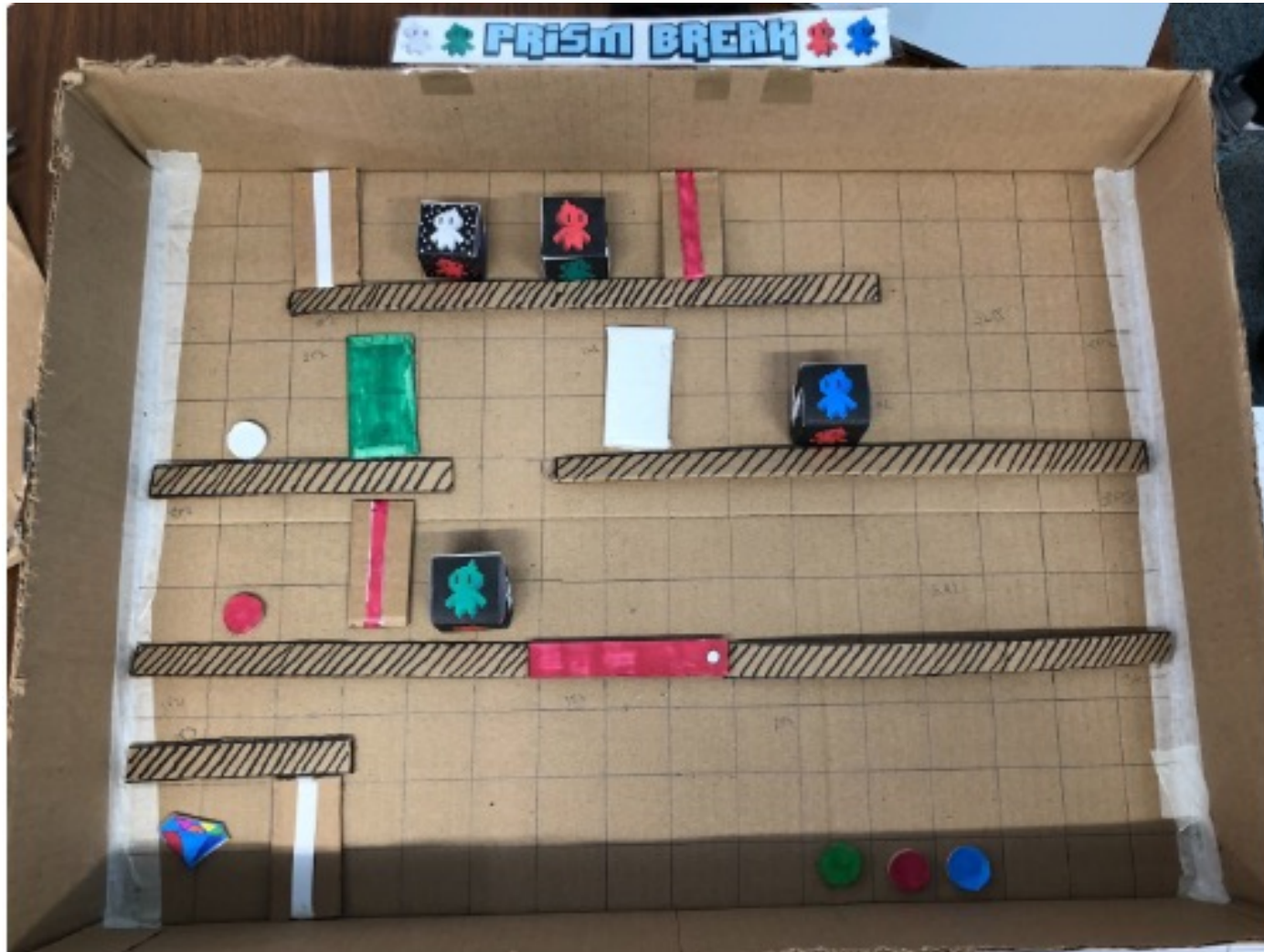
Configurable Prototype from Elements



Case Study: *Prism Break*



Exploring Puzzle Difficulty



Most Important Thing: *Progression*

- Do not want a **one-level** game
 - Major problem with endless runners
 - Survival games also have this problem
- We want some evidence of a **progression**
 - What is an easy level?
 - What is a medium level?
 - What is a hard level?
- Your prototype should be *reconfigurable*

Easy



Medium



Hard



The Difficulty Curve



Easy

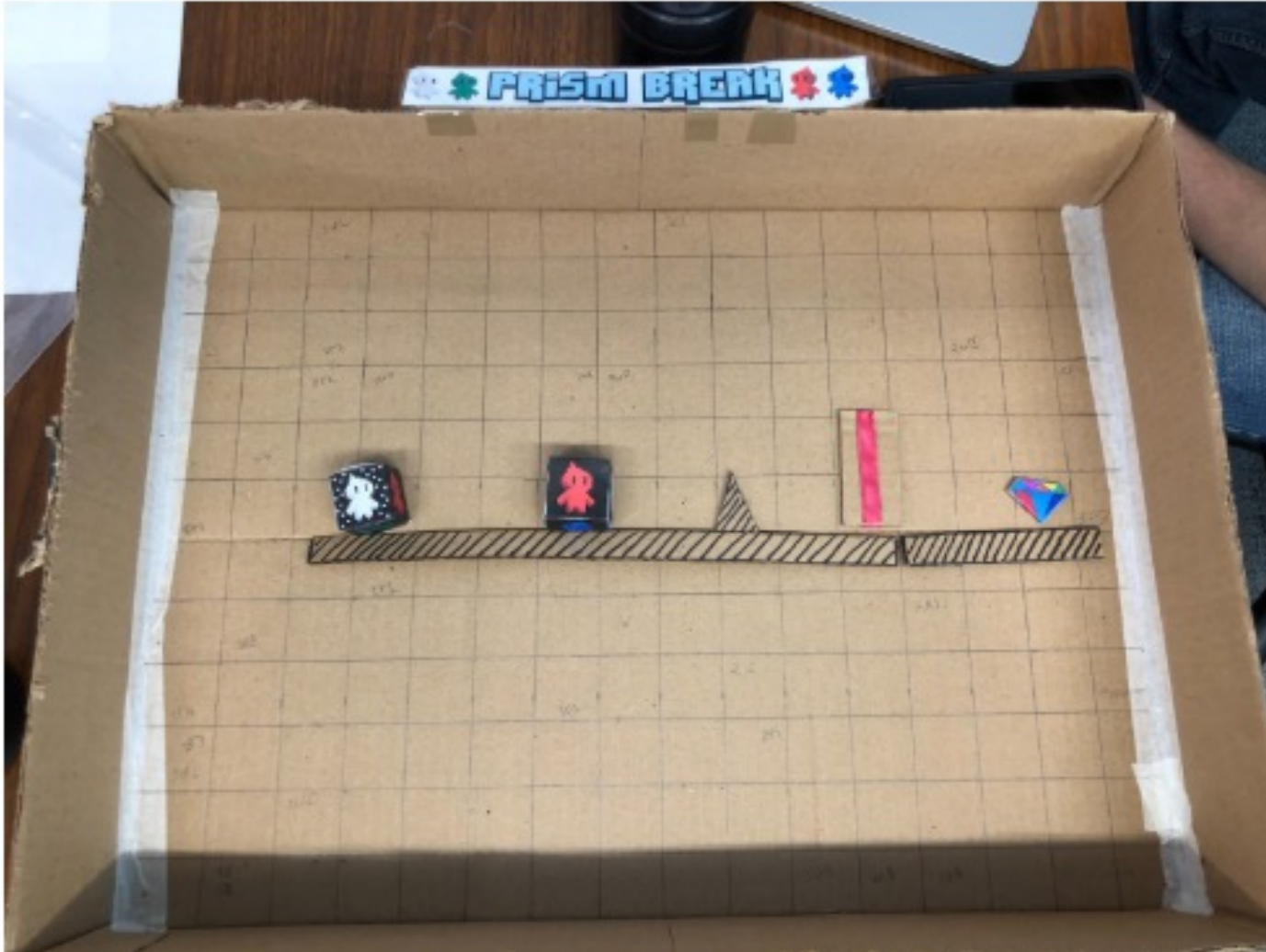


Medium

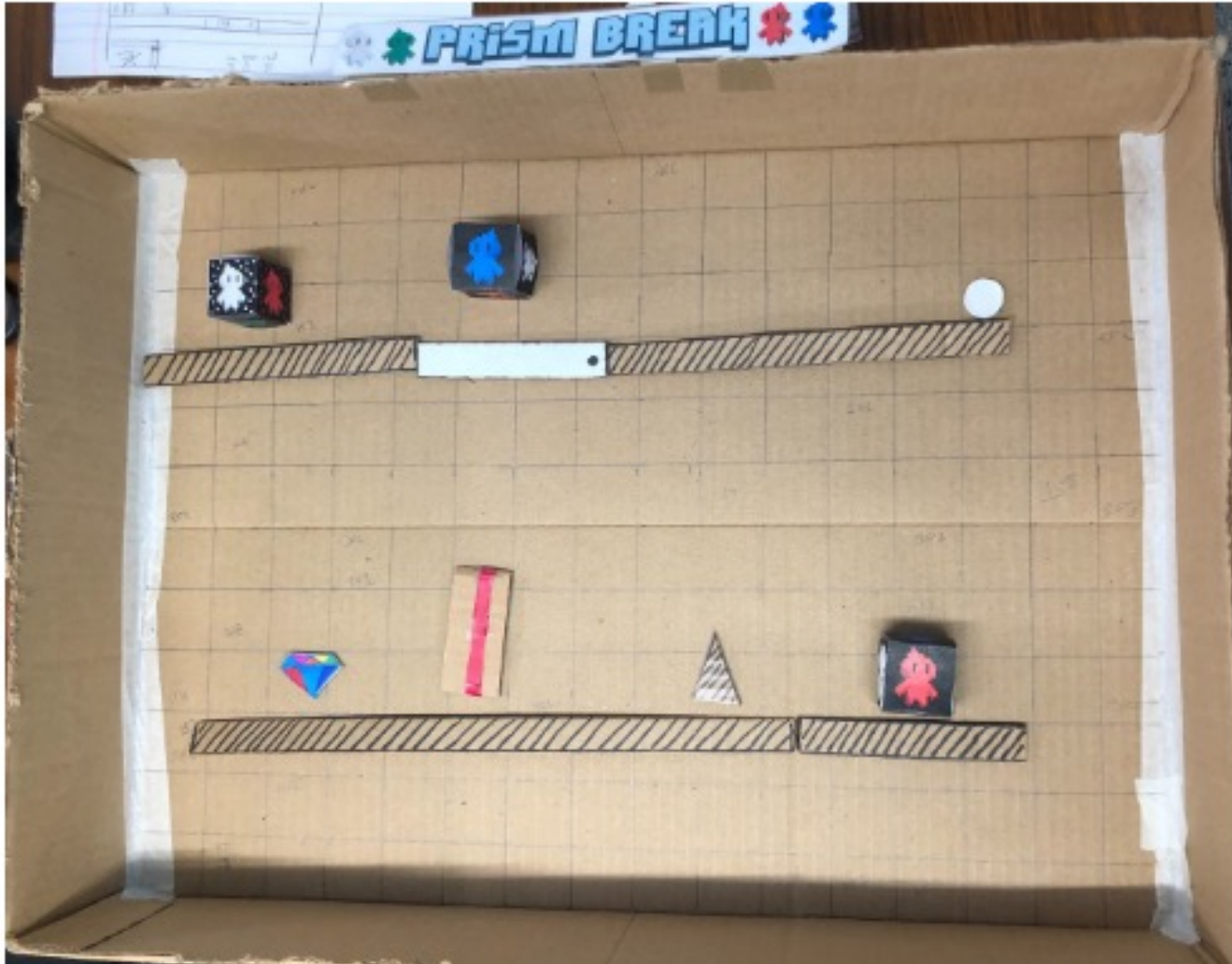


Hard

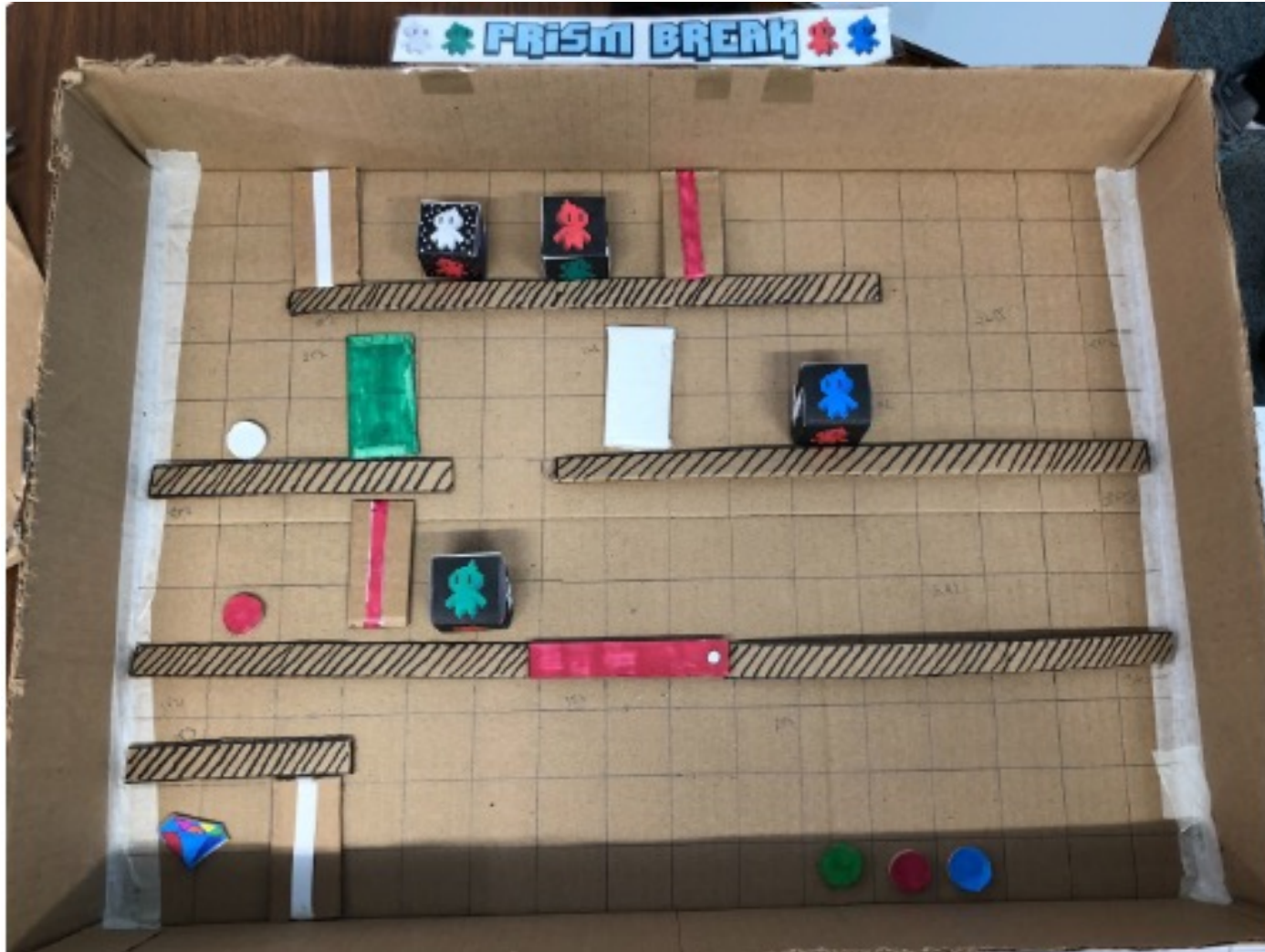
Easy: *Prism Break*



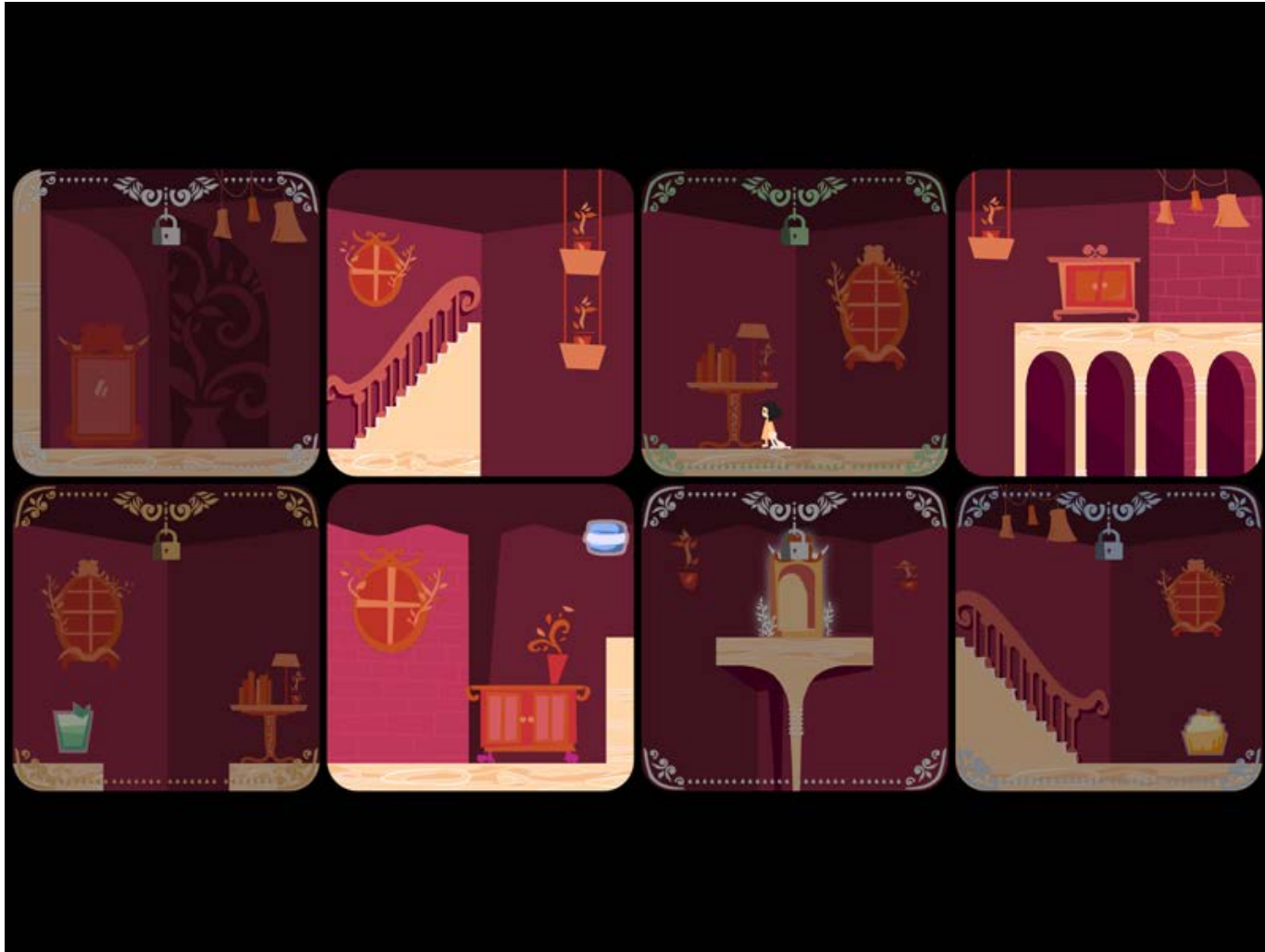
Medium: *Prism Break*



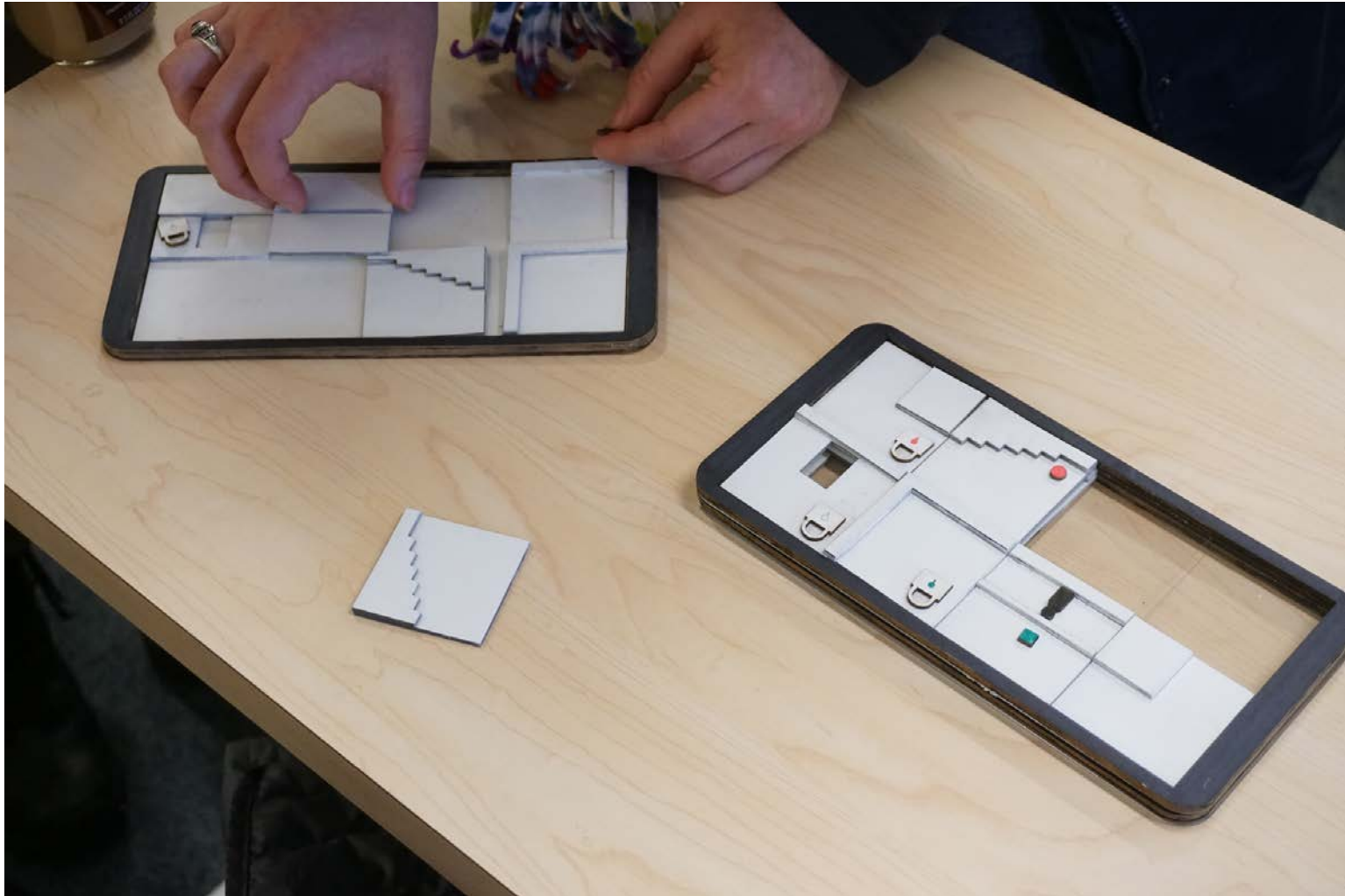
Hard: *Prism Break*



Case Study: *Magic Moving Mansion*



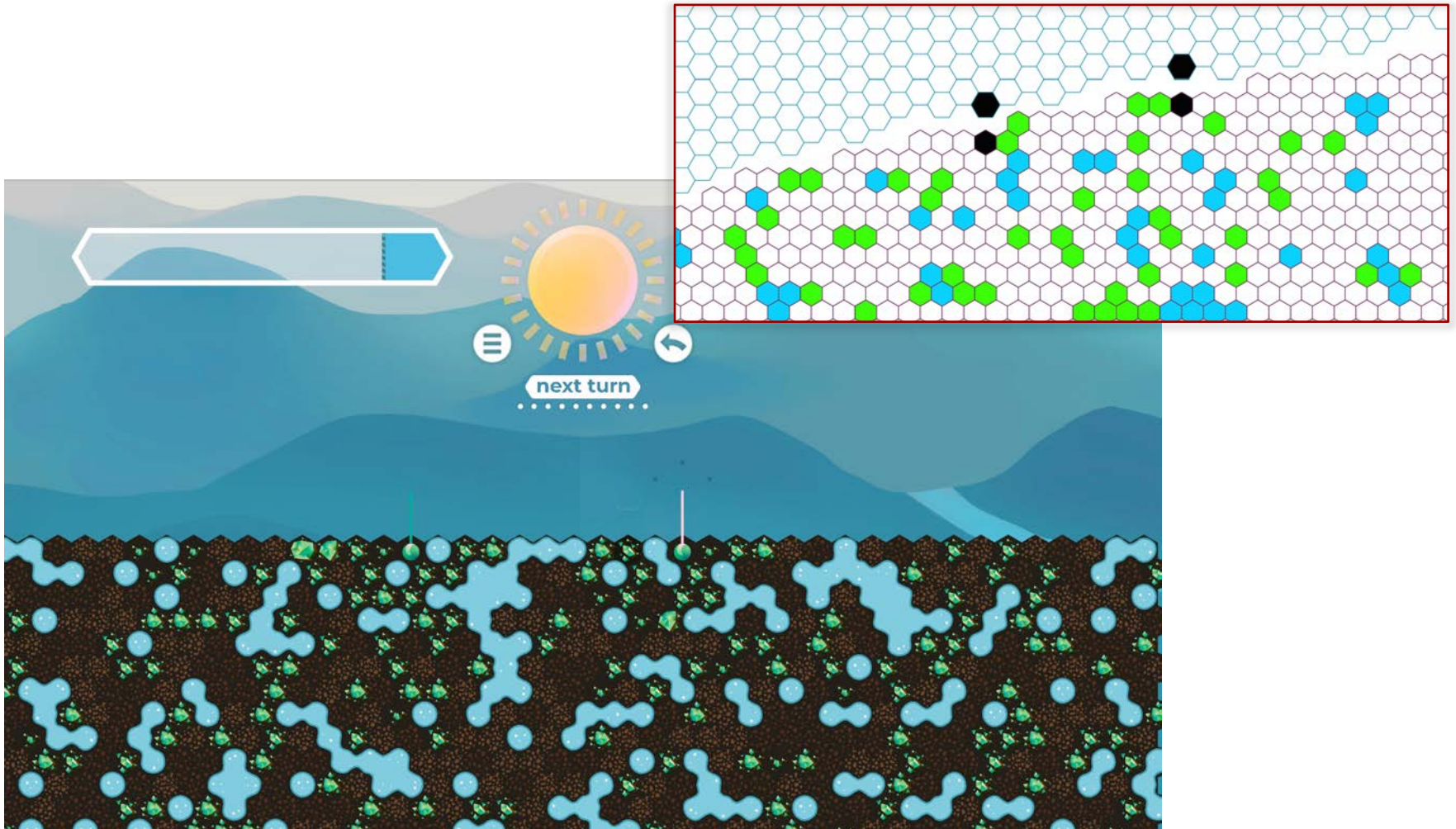
Configurable Puzzles at Scale



Reflecting on What You Have Learned

- Your prototype should teach you *something*
 - About one of the things covered today
 - Even if it is “this design will not work”
- You will be asked about this at **presentation**
 - Must be prepared to answer
 - Write-up as part of submission
- Lesson matters more than **physical artifact**
 - You are not going to sell this prototype

Case Study: Flourish



Case Study: Flourish

Our game seemed unclear at the beginning for some players because [they had to conceptually] balance growth above ground and below ground.

...

In general, we learned about the **specificity we need for different rules that we had thought needed less explanation.**

Summary

- Nondigital prototypes are about **discretization**
 - Group continuous state into course groups
 - Simplify mechanics into discrete turns
 - Sometimes requires mechanics substitution
- They are ideal for **early gameplay testing**
 - Evaluate emergent behavior
 - Model player cost-benefit analyses
 - Test player difficulty or usability
 - Capture player experiences (**advanced**)