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?
Understanding Game Progression

- Level design about **progress**
  - Sense of closeness to goal
  - Choice of “paths” to goal (**dilemma challenge**)
  - Path choice can relate to play style and/or difficult
- Easier to design if **discrete**
  - Flow-chart out progression
  - Edges are mechanic(s)
- But game state values are **continuous** (sort of)
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 Spacial Locality

Prototyping
Discretizing Spacial Locality

Prototyping
Discretizing Spacial 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?)

Gameplay Modeling
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 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

**Gameplay Modeling**

**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

**Simulates (real-time) player reaction time**

**Gameplay Modeling**
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
Reaction Time: **Runaway Rails**

- “Free runner” with coaster
  - Coaster can go faster/slower
  - Speed tests reaction time
- Model with hidden info
  - Cannot “process” all at once
  - Faster go, less screen to see
Reaction Time: *Runaway Rails*

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
Interactions: 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
Cost-Benefit Analysis: *Bounce*

Jetpack expends oxygen (=health)
Tracking Oxygen as a Resource
Player Skill as a Resource

*Game Grammar for PacMan*

**Resource Flow Legend:**
- **Pool**
- **Drain**
- **Gate**
- **Source**

Diagram details include:
- **Ghost House** 1/5 to **Maze** +2
- **Threat** to **Evade** 50%
- **Game Over** = 0
- **Lives**
- **Reset**
Player Skill as a Resource

**Game Grammar for PacMan**

Resource Flow Legend:

- Pool
- Drain
- Gate
- Source
- Game Over
- Lives
- Reset

Diagram:

- Ghost House
- Maze
- Threat
- Evasion
- Drains Threat
- Evade
- 50%
- Ghosts
- Add Threat
- >100
- all
- 9
- 0

Diagram details:

- 1/5
- all
- ==0

Game Grammars
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 Testing: Angry Bunny

Early Design:
Bunny movement controlled by battery “attraction”
Modeling Movement Controls

Strings attached at board corners

Control piece by pulling strings
Usability Testing: *Reflexio*

Touch zippers to open mouth

Can reflect world across various axes
Creating Puzzle Levels

Gameplay Modeling
Experiential Prototypes

- Some prototypes do not test gameplay
  - They test an experience or feeling
  - You determine if the feeling is enjoyable
  - Then go back and design gameplay for that

- Discouraged in this course
  - A very advanced design technique
  - Can easily end up with worthless prototype
  - Have only seen a few successes at this
Experiential Prototype: Aeronautical
The Experience of Threat
Most Important Thing: *Progression*

- Do not want a **one-level** game
  - Major problem with “flick” games in this course
  - Endless runners 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
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**)