Lecture 13

Procedural Content Generation
In the Beginning, There Was Rogue

---

Procedural Content
In the Beginning, There Was Rogue

**Roguelike Genre**

- Classic RPG style
- Procedural dungeons
- Permadeath
### A Brief History of Roguelikes

<table>
<thead>
<tr>
<th>Period</th>
<th>Games</th>
</tr>
</thead>
</table>
| Precursors (1978) | - *Beneath Apple Manor*  
                  |  - *Dungeon* (unfamous one)                                       |
| Rogue (1980)    | - Like *Rogue*, but less famous                                      |
|                 | - Limited content generation                                        |
| Immediate Copycats | - *Hack* (‘82), *NetHack* (‘87)                           |
|                 | - *Moria* (‘83), *Angband* (‘90)                                    |
| Modern Revival  | - Multiplatform launch                                               |
|                 | - All very close in playstyle                                        |
|                 | - Open source development                                            |
|                 | - Middle Earth themed                                                |
|                 | - Massively (~80) multiplayer                                        |
|                 | - But content less procedural                                        |
|                 | - Relaxing RPG requirement                                           |
# Changing Perspectives on Permadeath

## Advantages
- Greater challenge
  - Used as a badge of honor
- Higher emotional stakes
  - Easy to instill fear & horror

## Disadvantages
- Greater discouragement
- Seen as a personal failure
- Missed game content
- Cannot progress in story

---

*Procedural Content*
Changing Perspectives on Permadeath

**Advantages**
- Greater challenge
- Used as a badge of honor
- Higher emotional stakes
- Easy to instill fear & horror

**Disadvantages**
- Greater discouragement
- Seen as a personal failure
- Missed game content
- Cannot progress in story

Make dying expected & inevitable
Make each session a complete experience

Permanent Death
You have but one life, eager hero. If you should die, though your deeds will be remembered, you shall not return again.
Changing Perspectives on Permadeath

**Advantages**
- Greater challenge
- Used as a badge of honor
- Higher emotional stakes
- Easy to instill fear & horror

**Disadvantages**
- Greater discouragement
- Seen as a personal failure
- Missed game content
- Cannot progress in story

**Content Generation**
- Make dying expected & inevitable
- Make each session a complete experience

**Procedural Content**
• Design if often **horizontal**
  • Many verbs, game elements
  • Little coupled behavior

• Each play is a **slice**
  • Access to limited elements
  • Work with what you get

• “Expensive” to create
  • Requires a lot of content
  • But historically just text

• Difficult to balance

**Issues with Roguelikes**

---

<table>
<thead>
<tr>
<th>WEAPON (Table 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COST</strong></td>
</tr>
<tr>
<td>Dagger</td>
</tr>
<tr>
<td>orcish dagger</td>
</tr>
<tr>
<td>dagger</td>
</tr>
<tr>
<td>silver dagger</td>
</tr>
<tr>
<td>athame</td>
</tr>
<tr>
<td>elven dagger</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Knife</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COST</strong></td>
</tr>
<tr>
<td>worm tooth</td>
</tr>
<tr>
<td>knife (shito)</td>
</tr>
<tr>
<td>stiletto</td>
</tr>
<tr>
<td>scalpel</td>
</tr>
<tr>
<td>crysknife</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Axe</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COST</strong></td>
</tr>
<tr>
<td>axe</td>
</tr>
<tr>
<td>battle-axe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pick-axe</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COST</strong></td>
</tr>
<tr>
<td>pick-axe</td>
</tr>
<tr>
<td>dwarvish mattock</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Short sword</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COST</strong></td>
</tr>
<tr>
<td>orcish short sword</td>
</tr>
</tbody>
</table>
Issues with Roguelikes

- Design if often **horizontal**
  - Many verbs, game elements
  - Little coupled behavior
- Each play is a **slice**
  - Against the same world
  - With limited elements
- “Expensive” to create
  - Requires a lot of content
  - But historically just text
- Difficult to balance

---

**Procedural Content for Modern Games?**

---

**WEAPON (Table 1)**

<table>
<thead>
<tr>
<th>Weapon</th>
<th>COST</th>
<th>WGT</th>
<th>PROB</th>
<th>MATL</th>
<th>APPEARANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>orcish dagger</td>
<td>54</td>
<td>10</td>
<td>12</td>
<td>IRON</td>
<td>crude dagger</td>
</tr>
<tr>
<td>dagger</td>
<td>4</td>
<td>10</td>
<td>30</td>
<td>IRON</td>
<td>--</td>
</tr>
<tr>
<td>silver dagger</td>
<td>40</td>
<td>12</td>
<td>3</td>
<td>SILV</td>
<td>--</td>
</tr>
<tr>
<td>athame</td>
<td>4</td>
<td>10</td>
<td></td>
<td>IRON</td>
<td>--</td>
</tr>
<tr>
<td>dhaun dagger</td>
<td>4</td>
<td>10</td>
<td></td>
<td>IRON</td>
<td>runed dagger</td>
</tr>
<tr>
<td>lantern</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>METL</td>
<td>--</td>
</tr>
<tr>
<td>crysknife</td>
<td>100</td>
<td>20</td>
<td>0</td>
<td>MINL</td>
<td>--</td>
</tr>
<tr>
<td>Axe</td>
<td>COST</td>
<td>WGT</td>
<td>PROB</td>
<td>MATL</td>
<td>APPEARANCE</td>
</tr>
<tr>
<td>axe</td>
<td>8</td>
<td>60</td>
<td>40</td>
<td>IRON</td>
<td>--</td>
</tr>
<tr>
<td>battle-axe</td>
<td>40</td>
<td>120*</td>
<td>10</td>
<td>IRON</td>
<td>double-headed axe</td>
</tr>
<tr>
<td>Pick-axe</td>
<td>COST</td>
<td>WGT</td>
<td>PROB</td>
<td>MATL</td>
<td>APPEARANCE</td>
</tr>
<tr>
<td>pick-axe</td>
<td>50</td>
<td>100</td>
<td>tool</td>
<td>IRON</td>
<td>--</td>
</tr>
<tr>
<td>dwarfish mattock</td>
<td>50</td>
<td>120*</td>
<td>13</td>
<td>IRON</td>
<td>broad pick</td>
</tr>
<tr>
<td>Short sword</td>
<td>COST</td>
<td>WGT</td>
<td>PROB</td>
<td>MATL</td>
<td>APPEARANCE</td>
</tr>
<tr>
<td>orcish short sword</td>
<td>10</td>
<td>30</td>
<td>3</td>
<td>IRON</td>
<td>crude short sword</td>
</tr>
</tbody>
</table>
Main Types of Procedural Content

- Simulation
- World Generation
- Puzzle Generation
- Story Generation
- Dynamic Challenges
- Adaptive Difficulty

Procedural Content Wiki: http://pcg.wikidot.com
Simulation

- Complexity appears random
- Often a physical process
  - Fires, Fluids, Weather
  - Terrain generation
  - Artificial life
- **Teleological**
  - Run the full simulation
  - Accurate; hard to control
- **Ontological**
  - Create reasonable output
  - Inaccurate; easy to control
Simulation

- Complexity appears random
- Often a physical process
  - Fires, Fluids, Weather
  - Terrain generation
  - Artificial life
- Teleological
  - Planning
- Ontological
  - Design

Scientific Computing
Ad Hoc Algorithms
Simulation

- Complexity appears random
- Often a physical process
  - Fires, Fluids, Weather
  - Terrain generation
  - Artificial life
- Teleological
  - Run the full simulation
    - Accurate; hard to control
  - Ontological
    - Create reasonable output
    - Inaccurate; easy to control
- Minimal effect on gameplay
  - Often largely aesthetic
  - Hard to control difficulty
- Lot of work for little payoff

Scientific Computing
Ad Hoc Algorithms

Procedural Content
World Generation

- Often thought of as map generation
  - But really generation of game *geography*
  - Particularly broad category of PCG

- **Basic Format**
  - Start with basic geography building blocks
  - Include combination rules for blocks
  - Build until reach a stopping point

- Algorithms vary widely
Example: NetHack

Izchak the Curator      St:18/11 Dx:16 Co:17 In:18 Wi:18 Ch:17  Lawful
Dlvl:8   $:94041 HP:217(234) Pw:190(195) AC:7   Exp:30

Procedural Content
Example: NetHack

Procedural Content
Example: NetHack

Room

Hallway

Izchak the Curator  St:18/11 Dx:16 Co:17 In:18 Wi:18 Ch:17  Lawful
Dlvl:8  $:94041 HP:217(234) Pw:190(195) AC:7  Exp:30
Example: NetHack
Example: Vertical Drop Heroes

- **Movement**
  - Can move left-right
  - Down arrow to stomp/fall
  - Cannot jump at all!

- **Combat**
  - Space to fire weapon
  - Weapon depends on class
  - Free cage to switch class

- **Goal**
  - Collect treasure
  - Reach (a possible) exit
Example: Vertical Drop Heroes

![Game Screenshot]
Example: Vertical Drop Heroes

What if a platform were here?
The Reachability Problem

- Levels are effectively graphs
  - Edges are player choices
  - Choices are discretized
  - Fully connected (why?)

- PCG might a graph
  - with a lot of dead ends
  - with a lot of backtracking
  - that is unconnected

- Need to remember goal
  - Should always be reachable
  - Else, reset must be painless

Reachability is not just a spatial issue.
Example: Spelunky

PRESS C TO SELECT BOMBS AND PRESS X TO USE IT.
Ensuring Reachability

Two Options:

- Limit generation to reachable game states
- Verify goal is reachable or regenerate
Ensuring Reachability

Two Options:

- Limit generation to **possibly** reachable states
- Verify goal is reachable or regenerate
Grammars: A Formal Approach

- **Notation**
  - Set $\mathcal{N}$ of nonterminals
  - Set $\Sigma$ of terminal symbols
  - Set $\mathcal{P}$ of production rules
    - Have the form $A \Rightarrow B$
    - $A, B$ are *words* of symbols
  - To generate a value
    - Start with word $XAY$
    - Pick any rule $A \Rightarrow B$
    - Replace with $XBY$
    - Repeat until only terminals

- **Example**
  - $\mathcal{N} = \{ S, B \}$
  - $\Sigma = \{a, b, c\}$
  - $\mathcal{P}$ is the list of rules
    - $S \Rightarrow aBSc$
    - $S \Rightarrow abc$
    - $Ba \Rightarrow aB$
    - $Bb \Rightarrow bb$
  - Possible *outputs*
    - $abc, aabbcc, aaabbbcccc, \ldots$
Grammars on Graphs

- Symbols are colored nodes
  - Either terminal or not
  - Edges replace word order
- Words are now graphs
  - Productions on subgraphs
  - LHS is node+boundary
  - RHS alters the node
- Output built as before
  - But rule matching harder
  - Graph equivalency
Grammars on Graphs

- Symbols are colored nodes
  - Either terminal or not
  - Edges replace word order

- Words are now graphs
  - Productions on subgraphs
  - LHS is node+boundary
  - RHS alters the node

- Output built as before
  - But rule matching harder
  - Graph equivalency
Grammars on Graphs

- Symbols are colored nodes
  - Either terminal or not
  - Edges replace word order

- Words are now graphs
  - Productions on subgraphs

    Game Geography is a graph

- Productions on subgraphs
  - RHS alters the node

- Output built as before
  - But rule matching harder
  - Graph equivalency
Puzzle Generation

- Basic puzzle structure
  - Discrete actions/moves
  - Moves applied in sequence
  - **Goal**: get correct sequence

- Identify move sequences
  - Could be a loose category
  - Represent specific strategies

- Build up from sequences
  - Start from solved state
  - Invert moves (scrambling)

- Will require verification
Example: Lyne
Example: Lyne

Backtrack Pattern
Story Generation

- **Narrative** is tightly crafted
  - Must have emotional arc
  - Very hard to generate

- But **backstory** is looser
  - Collection of tales/subplots
  - Combine to form a story
  - Often displayed in a codex
  - Much easier to generate

- **Idea**: Create list of subplots
  - Pick some subset at a time
  - Mix with NLG techniques
Example: Dwarf Fortress
Natural Language Generation

- Function that outputs language
  - **Given**: complex set of data
  - **Outcome**: comment on data
  - Major area of CS research

- Comment requirements
  - Must be **simpler** than data
  - Should also be **natural**

- **Examples**
  - Sports commentary
  - Party combat chatter
  - Intelligent townsfolk
NLG and Story Dialogue

- Often a set of “canned” text
  - React to specific events
  - NPC picks text as appropriate
- Text is *parameterized*
  - “What do we do, <name>?”
  - “Someone killed <monster>!”
  - “That was <numb> days ago.”
- Choosing text to say
  - Favor important events?
  - Favor recent events?
  - Random (pull-toy)?
Skyrim’s Radiant Quest System

- Geography includes NPCs
  - Mobile, removable location
  - Dialogue is also a space
- System “randomly” chooses
  - Quest giver
  - Quest location
  - Location’s challenges
  - Quest redeemer
- Randomness is limited
  - Lists appropriate to quest
  - Depends on earlier actions

- Goals:
  - Send to unexplored areas
  - Adjust challenges to level
  - Can never be missed
- Largely a success
Skyrim’s Radiant Quest System

- Geography includes NPCs
  - Mobile, removable location
  - Dialogue is also a space

- System “randomly” chooses
  - Quest giver
  - Quest location
  - Location’s challenges
  - Quest redeemer

- Randomness is limited
  - Lists appropriate to quest
  - Depends on earlier actions

Guarantees
- reachability
- unexplored areas
- Adjust challenges to level
- Can never be missed
- Largely a success
Skyrim’s Radiant Quest System

- Geography includes NPCs
  - Mobile, removable location
  - Dialogue is also a space
- System “randomly” chooses
  - Quest giver
  - Quest location
  - Location’s challenges
  - Quest redeemer
- Randomness is limited
  - Lists appropriate to quest
  - Depends on earlier actions

- Goals:
  - Send to unexplored areas
  - Adjust challenges to level
  - Can never be missed
- Largely a success
Dynamic Challenges

- Challenges that can change
  - Become easier or harder
  - Just be different
- **Example**: Autoleveling
  - NPCs have statistics
  - Adjust to character level
  - Difficulty always reasonable
  - Allows true “open” world
- Not always popular
  - Can lead to design recycling
  - Sense of risk is lost

**Rat: Level 1**
- ATK: 1
- DFN: 0
- HP: 5

**Rat: Level 50**
- ATK: 30
- DFN: 10
- HP: 90
Other Types of Dynamic Challenges

• **Composite Challenges**
  • Encounter is a collection of NPCs, obstacles
  • Add or remove individuals from encounter

• **Dynamic NPC AI**
  • NPCs have a choice of AI scripts
  • Choose one that matches the player

• **Player Boosting**
  • Change result of player actions, interactions
  • Modifications make challenges easier/harder
Dynamic Challenges and Overlays

“Shooting Gallery”

Cover Busters
Assigning Dynamic Challenges

Player

Challenge

Extract feature vector from play history

Match the challenge to the play style

Parameterize challenge difficulty

(a_1, a_2, a_3, \ldots, a_n)

(b_1, b_2, b_3, \ldots, b_k)

Procedural Content
Assigning Dynamic Challenges

Player

Challenge

Matching Function is hardest to balance

Extract feature vector from play history

Match the challenge to the play style

Parameterize challenge difficulty

\[(a_1, a_2, a_3, \ldots, a_n)\]  \rightarrow  \[(b_1, b_2, b_3, \ldots, b_k)\]
Adaptive Difficulty

Player

Extract feature vector from play history

(a_1, a_2, a_3, …, a_n)

Match via machine learning

Challenge

Parameterize challenge difficulty

(b_1, b_2, b_3, …, b_k)

Procedural Content
Adaptive Difficulty

- Manually define the **gameplay model**
  - Metrics that identify player behavior
  - Parameters that define challenge behavior
  - Also metrics to evaluate player success or failure

- **Goal**: Use learning to find player-challenge match-up
  - Use playtesting/beta to get a large training set
  - Create an initial model from these results
  - Adjust in the game according to current player

- Still largely an academic exercise
Summary

- Procedural content started with Rogue(likes)
  - Tightly coupled with permadeath, horizontal design
  - Becoming fashionable once again

- Many applications to modern game design
  - World Generation
  - Puzzle Generation
  - Story Generation
  - Dynamic Challenges