Balance and Difficulty
(Or, How I Learnt to Stop Punishing Players and Love Flow)
In This Lecture

- Balancing Mechanics
- Difficulty
- Exercise to Rebalance a Game
In This Lecture

- Balancing Mechanics
- Difficulty
- Exercise to Rebalance a Game

Nothing replaces good old fashioned play testing.

Watch someone else play!
What is Balance?

PvP
Player vs Player

PvE
Player vs Environment
What is Balance?

PvP
Player vs Player

PvE
Player vs Environment

Game Provides Meaningful Choices

Players perceive the game to be fair

Chance does not outweigh player skill
What is Balance?

**PvP**  
Player vs Player

- Game Provides Meaningful Choices
- Players perceive the game to be fair
- Chance does not outweigh player skill
- Players can catch up before game end

**PvE**  
Player vs Environment

- Seldom results in stalemate
What is Balance?

**PvP**
Player vs Player

- Game Provides Meaningful Choices
- Players perceive the game to be fair
- Chance does not outweigh player skill
- Players can catch up before game end

**PvE**
Player vs Environment

- Level of difficulty should be consistent*
- Seldom results in stalemate

*does not mean it never changes
PvP fairness

- Equal chance of winning (excepting skill)
- Symmetry
  - starting player
- Asymmetry
  - harder to balance
  - test permutations

- Point assignment - for asymmetry
  - let player choose
  - orthogonally related
PvE fairness

- No sudden spikes in difficulty
- Avoid learn-by-dying (TrapAdventure 2!)
- Enough (in-game) information to make decisions
- Challenges should be genre-relevant
  - Avoid challenges that are non-sequiturs for this type of game
  - E.g. Critical path Racing Game in RPG
Depth vs Complexity

- **Depth**: amount of experiences possible from mechanics

- **Complexity**: mental burden put on player
  - data to remember
  - rules to process
  - calculations to make
  - (function of mechanics and interface)

- Complexity enables depth, but...
- Too much complexity limits depth

Dwarf Fortress
Depth vs Complexity

- Aim for **maximum** depth, **minimum** complexity
  - Can avoid by re-using/combining mechanics
  - Or slowing down the pace of play

- Avoid **Irreducible Complexity**
  - Otherwise is impossible to learn
  - Can the player compartmentalize the system?

- James Portnow: “**Elegance is a high depth to complexity ratio** – e.g. Go”
Part I - Balancing Mechanics
A clearly superior strategy that will win
- E.g. Tank Rush!!!

Removes meaningful choice -> boring
Can completely break asymmetric PvP games

How to avoid?
- Test to avoid unstoppable exploits
- In live games -> buffs and nerfs

In design, depends on whether you have
- Transitive or Intransitive relationships
Transitive Relationship

Problem:
- If $A > B$ and $B > C$
- Then $A > C$
- Strategies: why ever use $C$!? 

Solution:
- Impose *direct cost* ($A$ costs more than $C$)
- Impose *shadow cost* (a hidden cost, or consequence, of that decision)
Intransitive Relationship

A

C

B
Intransitive Relationship

Rock - Paper - Scissors (-Lizard - Spock)

Lizard poisons Spock, Spock smashes Scissors, Scissors decapitates Lizard.
PAPER disproves Lizard eats Paper.
Spock vaporizes Rock, Rock crushes Scissors.

http://www.samkass.com/theories/RPSSL.html
Balancing the Environment

- Orthogonal Unit Differentiation
- (Same for weapons/equipment)
Feedback is when the state of the system is an input into the system.

Positive Feedback
- Advantages to the winning player
- E.g. FPS: weapons, equipment

Negative feedback
- Advantages to the losing player
- E.g. FPS: random respawn, full health

Need to tune feedback carefully!
Imbalanced and Ideal Feedback Loops

1. Sprint foot race (no feedback)
2. Unbalanced rules in B’s favor
3. Stalemate (insufficient feedback to produce victory)
4. Balanced rules, but feedback operates too fast
5. Wild swings in the lead (powerful negative feedback)
6. Ideal game progression (lead changes hands, better player wins eventually)
Controlling Positive Feedback

- Tune the reward of power
  - Artificially limit the player’s power
  - Associate costs
  - Allow players to collaborate against leader

- Raise difficulty of challenges

- Use chance!

Fortnite
Use of Chance

- Randomness
  - Can moderate positive feedback

- Use sparingly, and rules-of-thumb
  - Small risk & reward
  - Let player play the odds
  - Let player choose their stake

Settlers of Catan
Part II - Difficulty
Challenge Hierarchy

- PvE is a series of predefined challenges
- Balancing is selecting the level and order of these!
PvE is a series of predefined challenges.
Balancing is selecting the level and order of these!
Overlap levels
Give breathing space
Get back in the zone after a break from game
Flow

Mihaly Csikszentmihalyi

Diagram showing the flow state on a graph with axes for Skill and Difficulty, illustrating the zone where flow occurs between high skill and high difficulty, avoiding states of boredom and anxiety.
Flow

Mihaly Csikszentmihalyi

- A challenging activity requiring skill;
- A merging of action and awareness;
- Clear goals;
- Direct, immediate feedback;
- Concentration on the task at hand;
- A sense of control;
- A loss of self-consciousness; and
- An altered sense of time.
1. Know your audience
2. Underestimate the player's learning curve.
3. Don't reward skilled players by making the game easier!
4. Allow players to change the game's difficulty.

“No matter how well you balance the game yourself, unless you are the sole audience of the game, you will need to know what it's like for others”
Estimating Difficulty

- **Absolute Difficulty**
  - versus trivial baseline challenge

- **Power**
  - player or avatar’s strength and stats

- **Relative Difficulty**
  - absolute difficulty adjusted for power

- **In-game experience**
  - amount of practice at this type of challenge

Dark Souls (Remastered)
We want to tune **Perceived Difficulty**

= *relative difficulty* - *in-game experience*

= (*absolute difficulty* - *power*) - *in-game experience*

= *absolute difficulty* - (*power* + *in-game experience*)
Static Relative Difficulty means Perceived Difficulty must drop
Rising Relative Difficulty means Perceived Difficulty can rise
Sawtooth levels

Overlap levels
Give breathing space
Get back in the zone after a break from game
Punishing Games
- Make players repeatedly do the same thing with small chance of success
- Or worse: Contain uninformed choices
- Even worse: Have inconsistent rules (play tricks)

Be less punishing
- Usable controls, logical rules
- Lower iteration times help (respawn!)
  - (TrapAdventure 2: 12 attempts in 90 sec)

Good Designers – design to challenge
Bad Designers – design to win!
Tuning for Median Skill

- Test with hardcore
- Test with casual
- Aim in between
Practicalities of Tuning

- Design your code to make tuning easy
  - load rules parameters from an external source (*public properties in Unity*)
  - fix the random seed

- Change 1 parameter at a time
- Binary search
- Keep records
Questions
Break

Don’t just sit there, get up, stretch, wander off (then come back)
Part III - Exercise

Balancing an Asymmetric Game
Fox and 13 Geese

- In 2-3s
- English solitaire board
- 1 fox, 13 geese
- Fox and geese can move to any empty adjacent space
- Geese cannot move backwards
- Fox can also jump over and capture
  - Chain jumps
  - Doesn’t have to jump
- Game is unbalanced in favour of fox. **Fix it!**
Fox and 13 Geese

- In 2-3s
- English solitaire board
- 1 fox, 13 geese
- Fox and geese can move to any empty adjacent space
- Geese cannot move backwards
- Fox can also jump over and capture
  - Chain jumps
  - Doesn’t have to jump
- Game is unbalanced in favour of fox. **Fix it!**
This is a two-player game. One person plays the red circle (the “Fox”) and the other player plays the thirteen white circles (the “Geese”). The two players have different objectives for winning.

Cut out the white and red pieces. Place the Geese on the black circles along the points of the board. Place the Fox in the red circle in the middle of the board (or anywhere else on the board, for more variation).

The Geese move first. Players alternate turns after that.

During their respective turns, the Fox and the Geese may move along the lines in any direction, but only to the next available point.

The Fox captures the Geese by jumping over them to a vacant spot beyond them. The Fox can jump over multiple Geese if possible.

Geese cannot jump over each other or the Fox.

The Geese win if they hem in the Fox and make it unable to move. The Fox wins if it captures enough Geese so that they cannot do this.
Discussion

- What did you change first?
- What did you hope to achieve?

- Which changes:
  - made play time longer/shorter?
  - made fox win more?
  - made geese win more?
Strongly Solving Fox-and-Geese on Multi-core CPU

Stefan Edelkamp and Hartmut Messerschmidt
TZI, Universität Bremen, Germany

Abstract. In this paper, we apply an efficient method of solving two-player combinatorial games by mapping each state to a unique bit in memory. In order to avoid collisions, such perfect hash functions serve as a compressed representation of the search space and support the execution of an exhaustive retrograde analysis on limited space. To overcome time limitations in solving the previously unsolved game Fox-and-Geese, we additionally utilize parallel computing power and obtain a linear speed-up in the number of CPU cores.

1 Introduction

Strong computer players for combinatorial games like Chess [2] have shown the impact of advanced AI search engines. For many games they play on expert and world championship level, sometimes even better. Some games like Checkers [7] have been decided, in the sense that the solvability status of the initial state has been computed.

In this paper we strongly solve Fox-and-Geese (Fuchs-und-Gänse), a challenging two-player zero-sum game. To the authors knowledge, Fox-and-Geese has not been solved yet.

Fox-and-Geese belongs to the set of asymmetric strategy games played on a cross shaped board. The lone fox attempts to capture the geese, while the geese try to block the fox, so that it cannot move.

The first probable reference to an ancestor of the game is that of Hala-Taff, which is mentioned in an Icelandic saga and is believed to have been written in the 14th century. According to various Internet sources, the chances for 13 geese are assumed to be an advantage for the fox, while for 17 geese the chances are assumed to be roughly equal.

The game requires a strategic plan and tactical skills in certain battle situations.

The portions of tactic and strategy are not equal for both players, such that a novice often plays better with the fox than with the geese. A good fox detects weaknesses in the set of geese (unprotected ones, empty vertices, which are central to the area around) and moves actively towards them. Potential decoys, which try to lure the fox out of its burrow have to be captured early enough. The geese have to work together in form of a swarm and find a compromise between risk and safety. In the beginning it is recommended to choose safe moves, while to the end of the game it is recommended to challenge the fox to move out in order to fill blocked vertices.

### Table 1. Retrograde Analysis Results for Fox-and-Geese

<table>
<thead>
<tr>
<th>Geese</th>
<th>States</th>
<th>Space</th>
<th>Iterations</th>
<th>Won</th>
<th>Time Real</th>
<th>Time User</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,112</td>
<td>264 B</td>
<td>1</td>
<td>0</td>
<td>0.05s</td>
<td>0.08s</td>
</tr>
<tr>
<td>2</td>
<td>32,736</td>
<td>3.99 KB</td>
<td>6</td>
<td>0</td>
<td>0.55s</td>
<td>1.16s</td>
</tr>
<tr>
<td>3</td>
<td>327,360</td>
<td>39 KB</td>
<td>8</td>
<td>0</td>
<td>0.75s</td>
<td>2.99s</td>
</tr>
<tr>
<td>4</td>
<td>2,373,360</td>
<td>289 KB</td>
<td>11</td>
<td>40</td>
<td>6.73s</td>
<td>40.40s</td>
</tr>
<tr>
<td>5</td>
<td>13,290,816</td>
<td>1.58 MB</td>
<td>15</td>
<td>1,280</td>
<td>52.20s</td>
<td>6m24s</td>
</tr>
<tr>
<td>6</td>
<td>59,808,675</td>
<td>7.12 MB</td>
<td>17</td>
<td>21,380</td>
<td>4m37s</td>
<td>34m40s</td>
</tr>
<tr>
<td>7</td>
<td>222,146,996</td>
<td>26 MB</td>
<td>31</td>
<td>918,195</td>
<td>27m43s</td>
<td>208m19s</td>
</tr>
<tr>
<td>8</td>
<td>694,207,800</td>
<td>82 MB</td>
<td>32</td>
<td>6,381,436</td>
<td>99m45s</td>
<td>757m05s</td>
</tr>
<tr>
<td>9</td>
<td>1,851,200,800</td>
<td>220 MB</td>
<td>31</td>
<td>32,298,253</td>
<td>273m56s</td>
<td>2,083m20s</td>
</tr>
<tr>
<td>10</td>
<td>4,257,807,840</td>
<td>507 MB</td>
<td>46</td>
<td>130,237,402</td>
<td>1,006m52s</td>
<td>7,766m19s</td>
</tr>
<tr>
<td>11</td>
<td>8,515,615,680</td>
<td>1015 MB</td>
<td>137</td>
<td>633,387,266</td>
<td>5,933m13s</td>
<td>46,759m33s</td>
</tr>
<tr>
<td>12</td>
<td>14,902,327,440</td>
<td>1.73 GB</td>
<td>102</td>
<td>6,828,165,879</td>
<td>4,996m36s</td>
<td>36,375m09s</td>
</tr>
<tr>
<td>13</td>
<td>22,926,657,600</td>
<td>2.66 GB</td>
<td>89</td>
<td>10,069,015,679</td>
<td>5,400m13s</td>
<td>41,803m44s</td>
</tr>
<tr>
<td>14</td>
<td>31,114,749,600</td>
<td>3.62 GB</td>
<td>78</td>
<td>18,483,934,148</td>
<td>5,899m14s</td>
<td>45,426m42s</td>
</tr>
<tr>
<td>15</td>
<td>37,337,699,520</td>
<td>4.24 GB</td>
<td>73</td>
<td>18,301,131,418</td>
<td>5,749m65s</td>
<td>44,038m48s</td>
</tr>
<tr>
<td>16</td>
<td>36,671,305,740</td>
<td>4.61 GB</td>
<td>64</td>
<td>20,022,660,514</td>
<td>4,903m31s</td>
<td>37,394m1s</td>
</tr>
<tr>
<td>17</td>
<td>37,337,699,520</td>
<td>4.24 GB</td>
<td>57</td>
<td>19,475,378,171</td>
<td>3,833m26s</td>
<td>29,101m2s</td>
</tr>
<tr>
<td>18</td>
<td>31,114,749,600</td>
<td>3.62 GB</td>
<td>50</td>
<td>16,808,655,989</td>
<td>2,661m51s</td>
<td>20,098m3s</td>
</tr>
<tr>
<td>19</td>
<td>22,926,657,600</td>
<td>2.66 GB</td>
<td>45</td>
<td>12,885,372,114</td>
<td>1,621m41s</td>
<td>12,134m4s</td>
</tr>
<tr>
<td>20</td>
<td>14,902,327,440</td>
<td>1.73 GB</td>
<td>41</td>
<td>8,693,422,489</td>
<td>858m28s</td>
<td>6,342m50s</td>
</tr>
<tr>
<td>21</td>
<td>8,515,615,680</td>
<td>1015 MB</td>
<td>36</td>
<td>5,169,727,685</td>
<td>395m30s</td>
<td>2,889m45s</td>
</tr>
<tr>
<td>22</td>
<td>4,257,807,840</td>
<td>507 MB</td>
<td>26</td>
<td>2,695,418,693</td>
<td>158m41s</td>
<td>1,140m33s</td>
</tr>
<tr>
<td>23</td>
<td>1,851,200,800</td>
<td>220 MB</td>
<td>26</td>
<td>1,222,085,051</td>
<td>54m57</td>
<td>385m32s</td>
</tr>
<tr>
<td>24</td>
<td>694,207,800</td>
<td>82 MB</td>
<td>23</td>
<td>477,731,423</td>
<td>16m29s</td>
<td>112m35s</td>
</tr>
<tr>
<td>25</td>
<td>222,146,996</td>
<td>26 MB</td>
<td>20</td>
<td>159,025,879</td>
<td>4m18s</td>
<td>28m42s</td>
</tr>
<tr>
<td>26</td>
<td>59,808,675</td>
<td>7.12 MB</td>
<td>17</td>
<td>44,865,396</td>
<td>55s</td>
<td>5m49s</td>
</tr>
<tr>
<td>27</td>
<td>13,290,816</td>
<td>1.58 MB</td>
<td>15</td>
<td>10,426,148</td>
<td>9.81s</td>
<td>56.15s</td>
</tr>
<tr>
<td>28</td>
<td>2,373,360</td>
<td>289 KB</td>
<td>12</td>
<td>1,948,134</td>
<td>1.59s</td>
<td>6.98s</td>
</tr>
<tr>
<td>29</td>
<td>327,360</td>
<td>39 KB</td>
<td>9</td>
<td>281,800</td>
<td>0.30s</td>
<td>0.55s</td>
</tr>
<tr>
<td>30</td>
<td>32,736</td>
<td>3.99 KB</td>
<td>6</td>
<td>28,347</td>
<td>0.02s</td>
<td>0.08s</td>
</tr>
<tr>
<td>31</td>
<td>2,112</td>
<td>264 B</td>
<td>5</td>
<td>2001</td>
<td>0.00s</td>
<td>0.06s</td>
</tr>
</tbody>
</table>
Chapter 15 and 13 of “Fundamentals of Game Design”
Chapter 10 of “Game Design Workshop”
Play (or Let’s Play) a hard game:
Anything by FromSoftware
  - Demon’s Souls
  - Dark Souls
  - Bloodborne
  - Super Meat Boy
  - Hotline Miami

Consider:
  - How does it avoid being punishing?
  - How does it remain feeling fair?
Depth is not Complexity
Avoid Dominant Strategies
Do you have Transitive or Intransitive relationships
Are your Feedback Loops working?
Use Chance sparingly and carefully
Create a Challenge Hierarchy
Aim for Flow (balance of difficulty and skill)
Aim for rising Perceived Difficulty
Aim for Difficulty not Punishment

Playtest and Tune (a lot!)
COMP3218 Website: https://secure.ecs.soton.ac.uk/module/COMP3218/

David Millard

@hoosfoos | davidmillard.org | dem@soton.ac.uk