Dynamic Difficulty Adjustment

Marshall Hoffmann

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

April 20, 2019

Video Games: Demographics

- 60% of American's play video games daily [1]
- Entertainment
- Escape reality or stress

Video Games: Environment

- Fantastical world
  - Imaginary forces
  - Spells: fireball, lightning bolt
  - Weapons: swords, guns, staves

[2]

Video Games: Type of Game

Models conflict (Role Playing Game, First-Person Shooter)

1. Player
   - Attributes: health points (HP), attack rating
2. Non-Player Character (NPC or agent)
   - Attributes: health points (HP), attack rating
   - Rules: spells, weapons
   - Goals: kill opponent

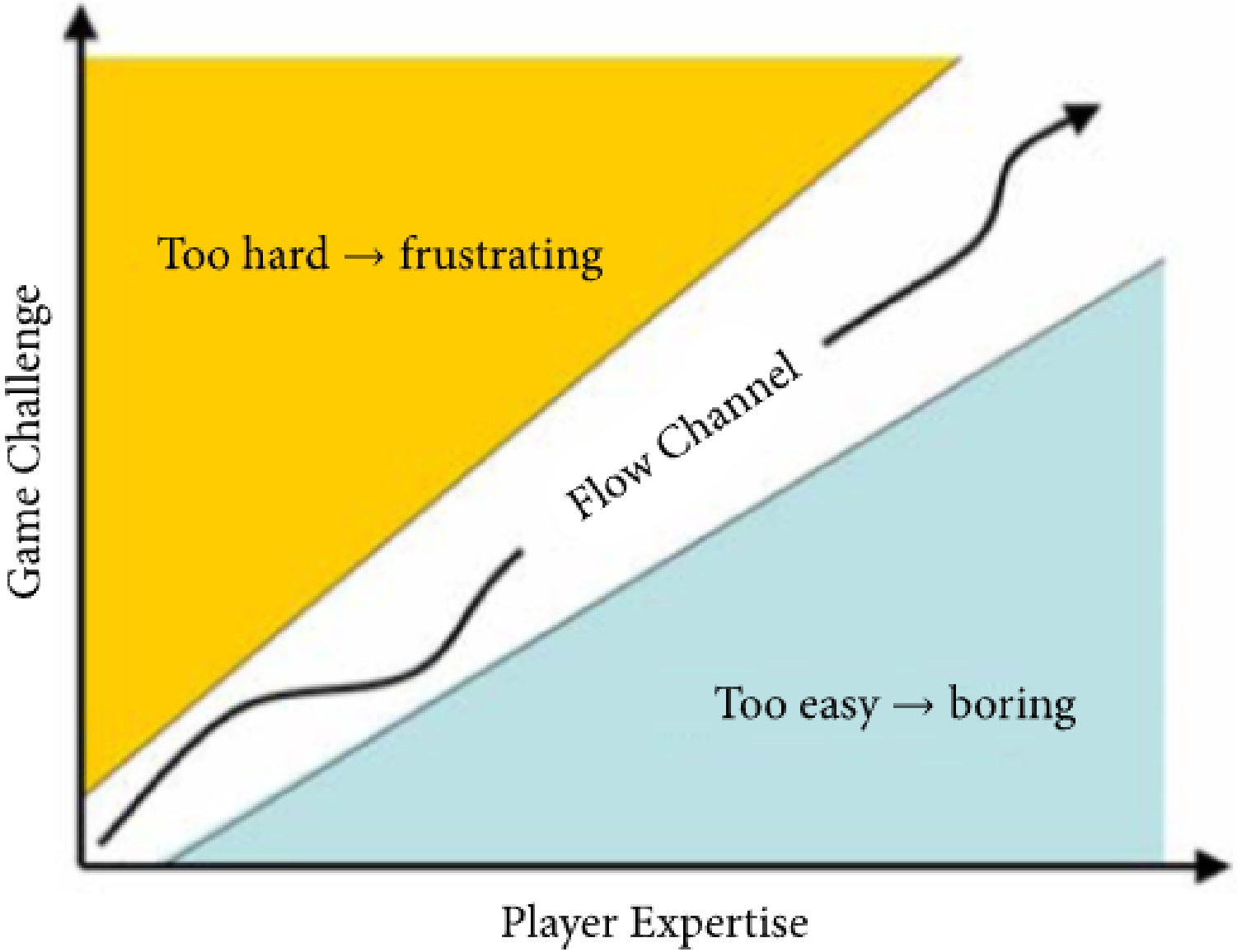
Player Experience

- Opponent responds/adapts to decisions
- Challenged by opponent of equal skill
- Too hard -> frustrated
- Too easy -> bored

Flow

"Extreme state experienced by the player when the task is so rewarding, they are willing to perform the task for the sake of undergoing that experience as if the task was an excuse to do so" [3]

- Optimal enjoyment
- Full immersion


Too hard → frustrating
Flow Channel
Too easy → boring
Game Challenge
Player Expertise

[3]

Artificial Intelligence (AI)

- Intelligence displayed by agents
  - How the agent determines its actions
    - Goal: kill opponent
    - Occasional block $>$ always attacking
    - When to block
  - Learning from past experiences
    - Blocks repeated attacks

Difficulty

- Degree of challenge
  - Attributes: health or damage
  - Opponent strategy, tactics (AI)
  - Inventory (weapons, spells)

Outline

- Static Difficulty
- Dynamic Difficulty Adjustment (DDA)
- Dynamic Scripting
  - Overview
  - Four Components
  - First-Person Shooter Implementation
  - Optimizing
- Conclusion

Static Difficulty

- Levels of difficulty
  - Easy, medium, hard
  - Varying player skill
- Gradually increasing difficulty
  - Improving player

[4]

Static Difficulty: Problems

1. Self rate skill level
2. Impossible to predict player growth
   - Player improves too quickly
   - Player falls behind
3. Predictable AI interaction that can be exploited
   - Agent always lunge attacks after a block
   - After you attack into a block, dodge to the side and attack from behind

[4]

Dynamic Difficulty Adjustment

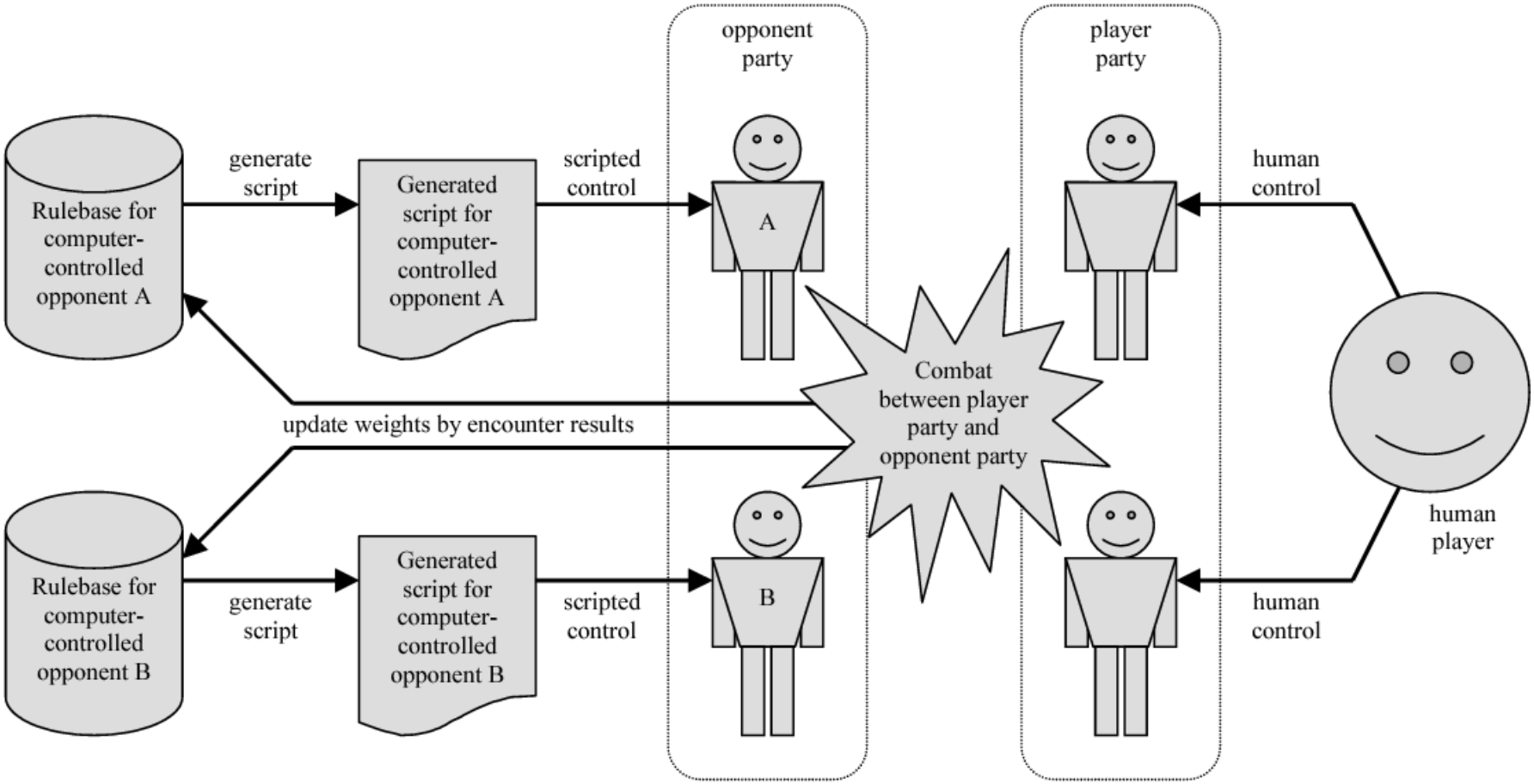
- Modifying difficulty in real-time
  - Benefits
    - Game remains fun and challenging
    - Maintain balance between player and game
  - Requirements
    1. Track and adapt to player ability
    2. Maintain balance between player skill and difficulty
    3. Seamless adaptation

Dynamic Scripting: Static Agents vs Dynamic Agents

Dynamic Scripting: technique for how agents behave

	Static	Dynamic
Goal	Numerically quantifiable	Numerically quantifiable
Behavior	Rules: e.g. blizzard	Rules: e.g. blizzard
Generation	Identical	Unique (Weights)

Dynamic Scripting: Overview



[4]

Dynamic Scripting: Four Components

1. Set of Rules
2. Script Selection
3. Rule Policy
4. Rule Value Updating

Dynamic Scripting: Set of Rules (1/2)

Rulebase: set of rules an agent can be generated with

Name	Blizzard
Condition	Character is in arctic environment
Effect	Surrounding area has heavy snow and wind damage enemies in an area

Name	Crush
Condition	Character is bigger than opponent
Effect	Pick up opponent and throw them to the ground

Name	Shoot Gun
Condition	Enemy is within range
Effect	Shoot bullet at enemy with gun/bow

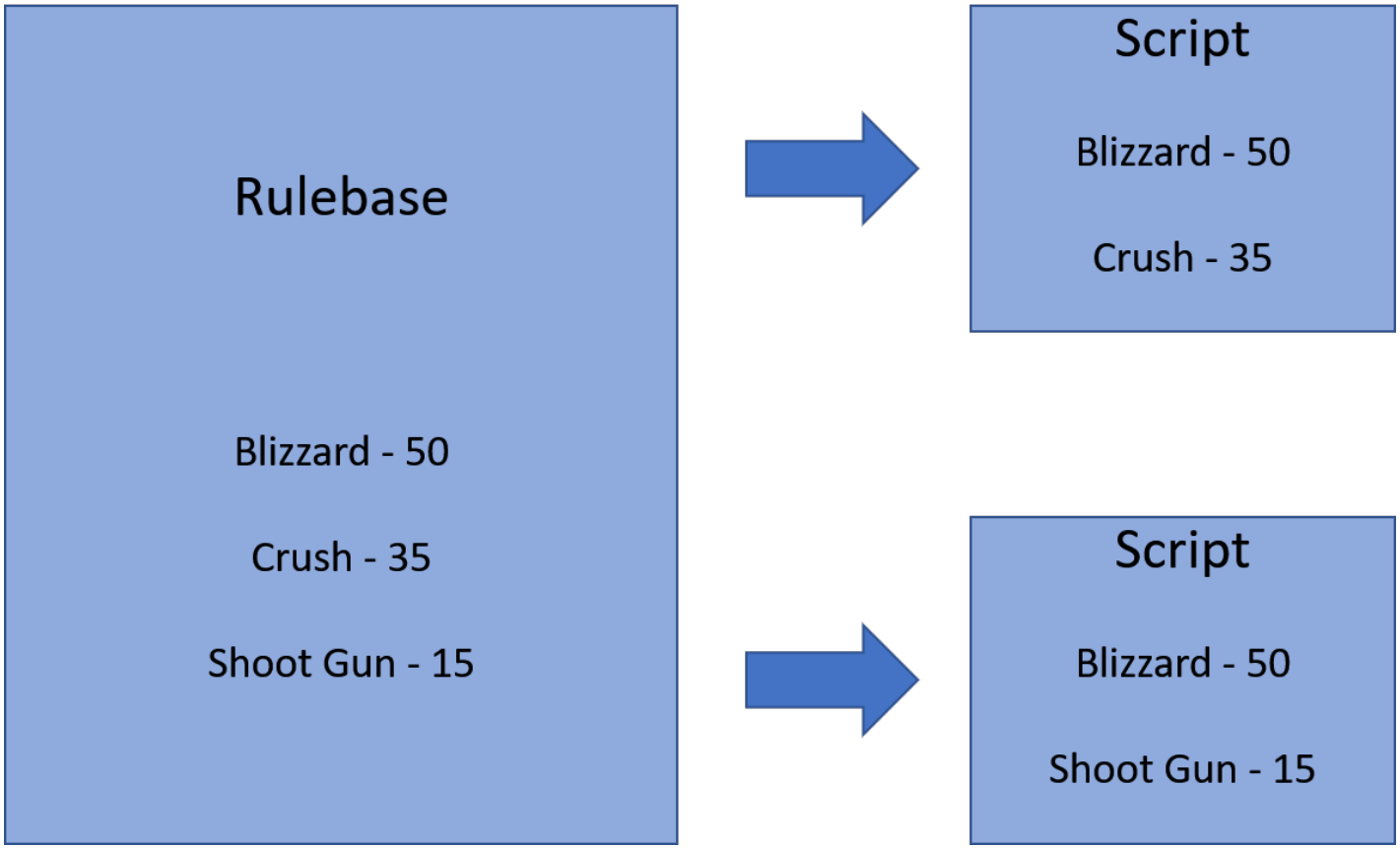
Dynamic Scripting: Set of Rules (2/2)

- Important: a good rulebase is an enabler
  - Advanced tactics
  - High variability
  - Adaptability

Dynamic Scripting: Script Selection

- Script: agent's collection of rules
  - Size: determined by developer
- Rules are chosen probabalistically
  - Higher weight $>$ Lower weight

Dynamic Scripting: Rules Overview



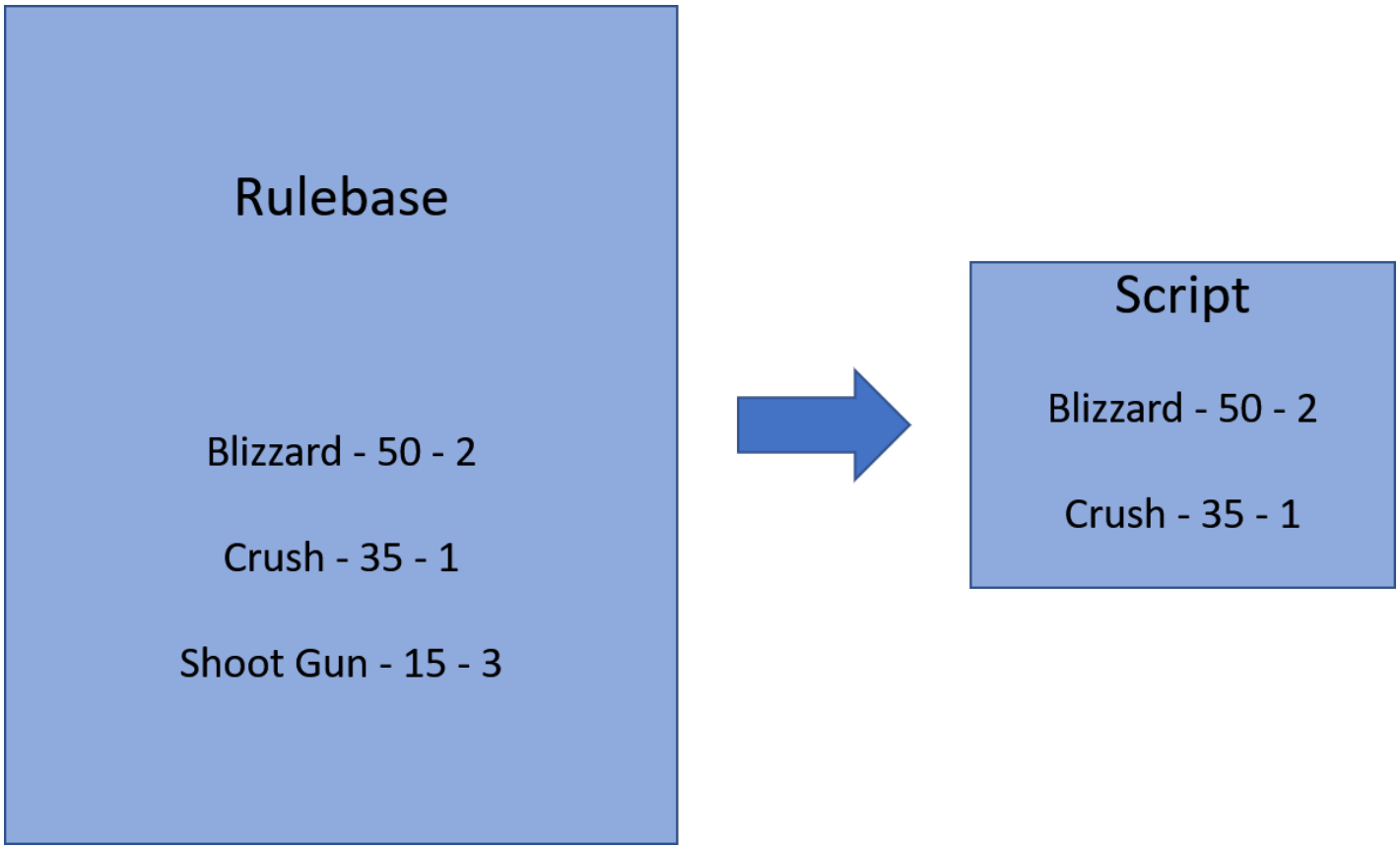
Dynamic Scripting: Rule Policy

- How the script is run
  1. Based on weights
  2. Priorities

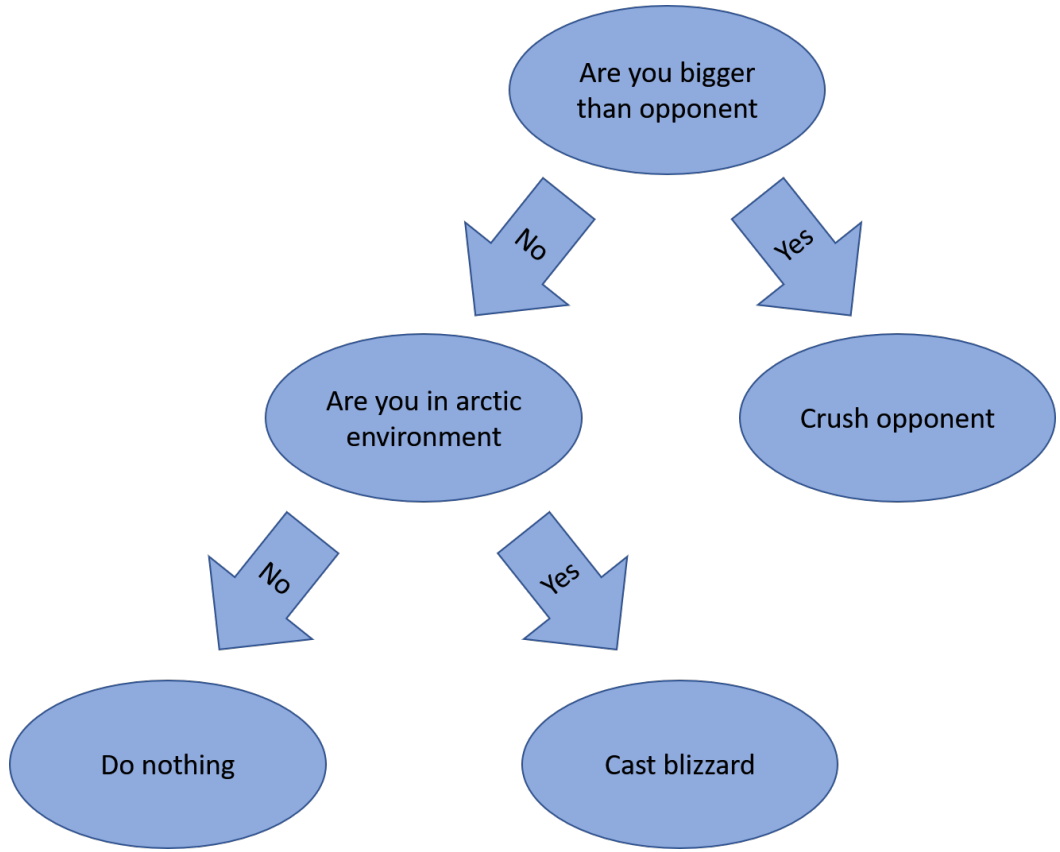
Rule Policy: Weight Based

- Condition: arctic environment and bigger than opponent
- Chance to use rule determined by weight
  - $50/85 = 59\%$ blizzard spell
  - $35/85 = 41\%$ crush attack

Rule Policy: Priorities (1/2)



Rule Policy: Priorities (2/2)



Dynamic Scripting: Rule Value Updating (1/2)

- Fitness function
  - How well agent performed (Scale: 0 - 1)
  - Variables to measure performance (after a conflict)
    - H(a): remaining health of agent (0 - 1)
    - D(o): damage dealt to opponent (0 - 1)
  - Goal: kill opponent first
  - $F(a, o) = .5^* H(a) + .5^* D(\mathrm{o})$

Dynamic Scripting: Rule Value Updating (1/2)

- Fitness function
  - How well agent performed (Scale: 0 - 1)
  - Variables to measure performance (after a conflict)
    - H(a): remaining health (0 - 1)
    - D(o): damage dealt (0 - 1)
  - Goal: kill opponent first
  - $F(a, o) = .5^{*}H(a) + .5^{*}D(\mathrm{o})$
  - $F(a, o) = .5^{*}.50 + .5^{*}1 = .75$

Dynamic Scripting: Rule Value Updating (2/2)

- Reward function
  - Updates weights based on fitness score
    - Rewards winning behavior
    - Punishes losing behavior
    - Every rule in active script is altered evenly
    - Remaining (unused) rules are adjusted accordingly to maintain an equal sum of weights

Dynamic Scripting: Rule Value Updating (2/2)

- Reward function
  - Updates weights based on fitness score
  - P_{max} = max weight penalty
  - R_{max} = max weight reward
  - F = fitness value; W = weight

$$\Delta(W) = \begin{cases} -P_{max} \cdot (1 - F) & loss \\ \\ R_{max} \cdot F & win \end{cases}$$

Dynamic Scripting: First-Person Shooter (Implementation)

- Two characters: static agent and the dynamic agent
- Goal: kill opponent
- In an arena with a few items
  - Health pack: increases health to max (200HP)
  - Ammo box: refills ammo to max
  - Explosive barrel: explodes when shot dealing damage in an area
- Each agent has two weapons
  - Rocket launcher: slow firing and deals 100 damage at center (decreasing)
  - Machine gun: fast firing and deals 5 damage
- Each agent has equal access to guns and items to ensure behavior is only variable

First-Person Shooter: Set of Rules

- There are 14 rules in the ruleset
  - Attacks
  - Item pickups
  - Searching for opponent

First-Person Shooter: Script Selection

- Script: comprised of 4 rules
  - Static agent same 4 rules throughout
- Rules are chosen probabalistically
- Default rule: idle
  - Most games engines require an action to be selected

Name	Idle
Effect	Remain stationary

First-Person Shooter: Rule Policy

- Weight based system

First-Person Shooter: Rule Value Updating

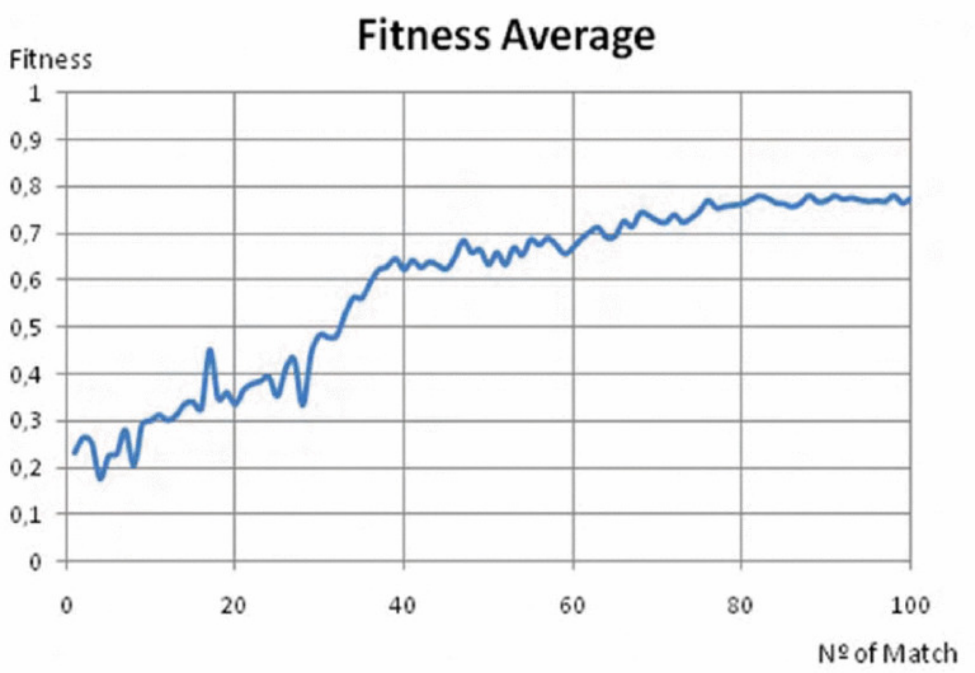
- Fitness function
  - Variables
    - H(a) = health remaining of dynamic agent
    - D(o) = damage dealt to opponent
    - T(g) = time in victory or defeat
    - t_t = duration of the match
    - t_m = maximum game time allowed

$$F(a, o, \mathrm{g}) = \frac{4^* H(a) + 4^* D(\mathrm{o}) + 2^* T(\mathrm{g})}{10} [6]$$

$$T(g) = \begin{cases} \frac{t_t}{t_m} & loss \\ \\ \frac{t_m - t_t}{t_m} & win \end{cases} [6]$$

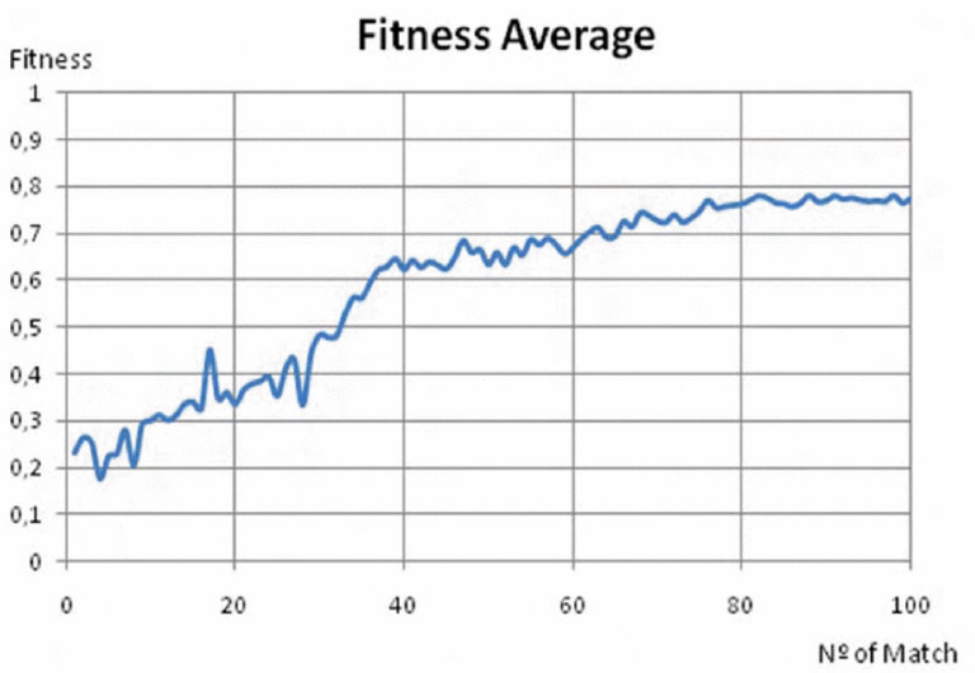
First-Person Shooter: Results (1/3)

- Results recorded in 5 batches of 100 matches
- After each batch of 100 matches, the weights were reset
- 5 batches averaged



First-Person Shooter: Results (2/3)

- First 30 matches: struggled
- After 40 matches: consistent victories
- Able to reach fitness values above 0.7



First-Person Shooter: Results (3/3)

- Good
  - Dynamic AI was able to reliably determine which tactics were effective against static opponents tactics
    - Shows dynamic scripting can be an effective way of adjusting AI in real-time
- Bad
  - Took some trials to reach optimized tactics

Dynamic Scripting: Introducing Sub-optimal Tactics

- In games optimal behavior is not always desired
- 3 modifications
  1. High-fitness penalizing
  2. Weight clipping
  3. Top culling

Introducing Sub-optimal Tactics: High-Fitness Penalizing

- Penalizes optimal behavior
  - Instead of weights being rewarded for positive behavior, they are diminished
  - When player loses, this forces weaker traits to be present

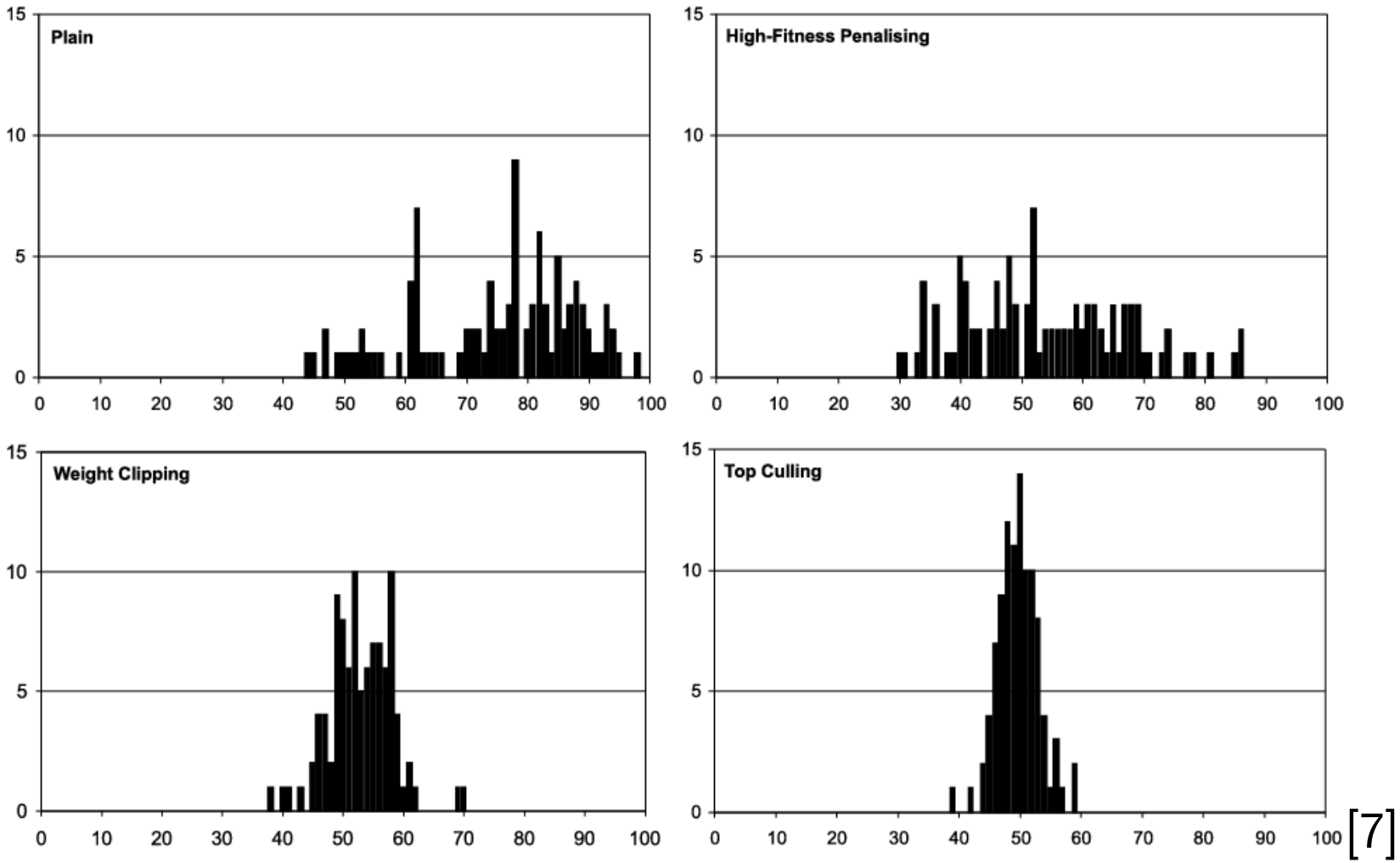
Introducing Sub-optimal Tactics: Weight Clipping

- Puts a cap on how much weight a rule can hold
  - When an optimized rule develops instead of continually increasing weight it is capped
  - Allows for weaker rules to be chosen more frequently

Introducing Sub-optimal Tactics: Top culling

- Similar to weight clipping
- Weight clipping: puts a cap on how much weight a rule can hold
- Top culling: allows weight to go over a set maximum value
  - At this point it is no longer selectable from the rulebase until it drops below maximum
  - Other weights being rewarded passively decrease its weight
  - Allows for weaker rules to be chosen more frequently

Introducing Sub-optimal Tactics: Results



[7]

x-axis: fitness value (%); y-axis: frequency

Conclusion

- Dynamic scripting can be an effective form of dynamically adjusting difficulty in video games
  - Adaptability
    - Creates weights unique to the players tactics
  - Increased variety
    - Each agent class has its own rules
    - Each agent generated has its own script
  - Fast
    - Requires extraction of rules to generate script
    - Update weights once per encounter

Acknowledgments

- Thank you to Peter Dolan, Elena Machkasova, and KK for their helpful feedback, insight, and expertise.
- Thank you as well to my friends, and classmates for their support, and feedback

Acknowledgments

- Thank you to Peter Dolan, Elena Machkasova, and K.K. Lamberty for their helpful feedback, insight, and expertise.
- Thank you as well to my friends, and classmates for their support, and feedback.

Questions?

References

[1]
D. Ang. Difficulty in Video Games: Understanding the Effects of Dynamic Difficulty Adjustment in Video Games on Player Experience, in C&C 17 Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition, pages 544-550.
[2]
https://a.wattpad.com/cover/159037574-288-k151276.jpg
[3]
M. Zohaib. Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review, in Advances in Human-Computer Interaction Volume 2018, Article ID 5681652

References

[4]
http://www.3ice.hu/blog/wp-content/u/diffdiag.jpg
[5]
P. Spronck, I. G. Sprinkhuizen-Kuyper, E. O. Postma. On-line Adaptation of Game Opponent AI with Dynamic Scripting, in Int. J. Intell. Games & Simulation 2004.
[6]
D. Policarpo, P. Urbano, T. Loureiro. Dynamic scripting applied to a First-Person Shooter, in 5th Iberian Conference on Information Systems and Technologies.

References

[7]
P. Spronck, M. J. V. Ponsen, I. G. Sprinkhuizen-Kuyper, E. O. Postma. Adaptive game AI with dynamic scripting, in Machine Learning 2006.