Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment
(AIIDE 2022)

Searching for Balanced 2D Brawler Games: Successes and Failures of Automated Evaluation

Samuel Shields, Ross Mawhorter, Edward Melcer, Michael Mateas

University of California, Santa Cruz
samshiel@ucsc.edu, rmawhort@ucsc.edu, eddie.melcer@ucsc.edu, michaelm@ucsc.edu

Abstract

Automated game design (AGD) research focuses on creating systems that can design entirely new games. This is often done by a genetic algorithm, with a fitness function that is used to find individual games that satisfy certain design criteria. However, it is difficult to tell if generated games actually have the desired emergent properties (such as balance), since the fitness function might not align well with human aesthetic judgments about such properties. This is particularly problematic when trying to automatically design balanced, fair, yet asymmetrical games for multiple players. In this paper we present an early system for automatically designing brawler games, and present findings from a preliminary user study involving the same games. We show that while the system successfully optimizes for our written fitness function during human play, we found that this optimization did not correctly translate to our hypothesized human experience of the game.

Introduction

Procedural content generation has solidified as a commonly-used technique in video games, leading to a wide body of research covering many techniques for creating video game content automatically. More recently, researchers have begun to focus on automated game design (AGD) (e.g. (Browne and Maire 2010; Cook, Colton, and Gow 2016a)). Here, the generation systems are responsible for more than a single static artifact that will be used as part of a wider game. Instead, they generate interactive parts of the game world including narrative, game mechanics, and other game elements like sound effects. Because these systems generate multiple parts of the game, they usually strive to make coherent choices, so that each part makes sense in the context of the other part, a framework called orchestration (Liapis et al. 2018). In addition to designing entirely new games, this framework can also be used to discover new mechanics that integrate well with the fixed design of an existing game.

A primary challenge of AGD is evaluation: many AGD systems are evaluated by defining an objective function that intuitively corresponds to desirable properties of the game, and show that generated examples score well on this metric (Cook, Colton, and Gow 2016a). Because of the complexity of games, the objective is often calculated using statistical data about play traces performed by an AI. However, even if the objective function is well-calibrated towards desirable gameplay, there is no guarantee that this AI behaves in a human-like way, and thus no way to tell whether the outputs of the system are games that achieve their aesthetic goals.

This is problematic when it comes to balance in multiplayer games. While there exists literature on automated balancing in multiplayer games, studies often stop short of performing user studies to determine if proposed methodologies produced artifacts that end-users perceived as balanced. In a study on Real-Time Strategy balancing, for example, human judgments are wisely included in an iterative system that aims to tune a set of parameters to achieve balance. (Preuss et al. 2018) However, evaluations of this system consist of case studies and evaluations using an internal fitness function, leaving the reader with little understanding if a broader or uninformed audience might actually judge produced artifacts as balanced. If we computationally model aesthetically-driven judgments in a game balancing system (this character feels overpowered, this unit is under game tempo, etc.), we should also strive to show in user studies that these aesthetically-driven properties are perceived without priming or bias.

In this paper we describe a system that automatically generates games in the brawler genre, and evaluate its performance with a user study (screenshots from gameplay shown in Figure 1). We show that the system's evolutionary algorithm successfully controls many facets of actual human play, but with a weaker effect on human perception of those facets. These results from real gameplay provide insight into the circumstances under which automated evaluation is correlated with human evaluation.

Background

Brawler Games

Brawler games (a.k.a. Platform Fighting Games) are a subgenre of fighting games where players each control a single character and attack each other with various moves. In contrast to the *Street Fighter* (Capcom Co. Ltd. 1987) style of game where players have a health bar, brawler games require players to knock their opponent off the screen. Knock-

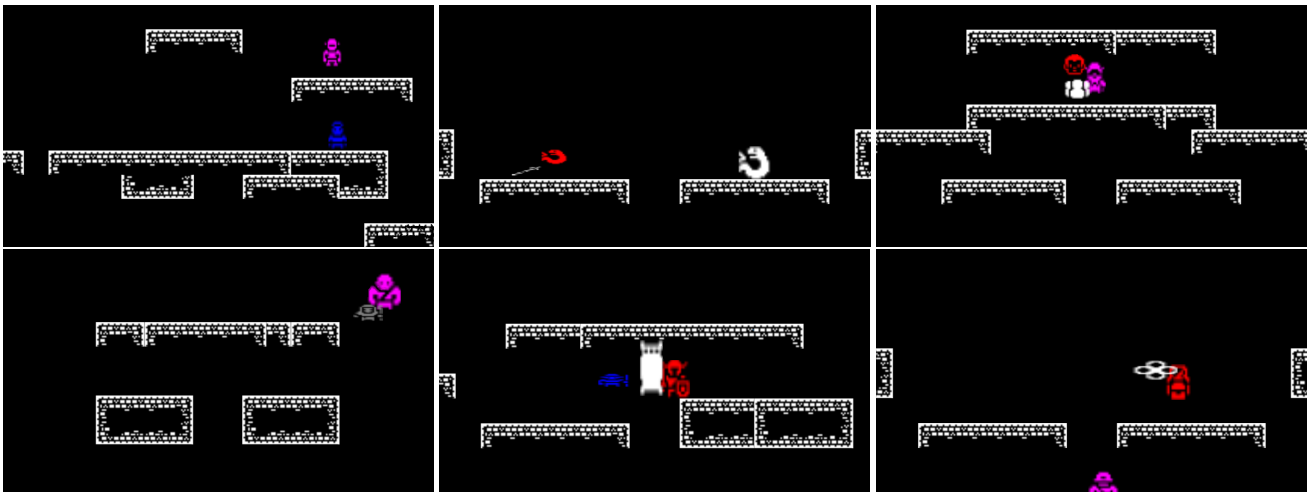

Figure 1: Six screenshots from example games. The games' labels from top left to bottom right respectively are: Game A (Evolved), Game B (Random), Game C (Evolved), Game D (Random), Game E (Evolved), Game F (Random).

back inflicted to a player is scaled with the total damage that that player has - the higher their damage, the farther they fly when hit. Some examples of brawlers are the popular *Super Smash Bros* games, as well as games inspired by these like *Rivals of Aether* and *Nickelodeon All-Star Brawl* (Nintendo Co. Ltd. 1999; Dan Fornance 2017; Ludosity, Fair Play Labs 2021). Brawlers are popular and often host large and competitive eSports communities.

Brawler games are noted for the emergent complexity in their competitive environments in spite of their ease to learn (Codsi and Vetta 2021). Balance and fairness are common concerns of expert players and novice players alike: experts want fairness in tournaments, and novice players want a fighting shot in any given match. Deciding what object properties in a brawling game (damage of a move, layouts of levels, physics characteristics of characters) allows for an interactive, balanced game environment is a complex problem and is the subject of many "balance patches" that publishers apply after initial release.

Our system generates simple brawler games consisting of two characters with one move each and a single level. This simplified game allows us to parameterize the core elements of a brawler game. A full list of system parameters is in Table 1. Simple changes to parameters have dramatic impacts on emergent game feel. An example: The numerical value of a character's jump force has a large effect on how powerful they are overall. Level design also has a great impact on game balance: levels might have platform spacing that is difficult for one character but not the other.

Gameplay differences driven by parameter changes provides an opportunities for a parameter-tuning system to produce asymmetrical yet balanced characters. An ideal system should be able to produce games that feel fair with significantly different characters, not only characters that are mirror copies or are so poorly constructed that they cannot functionally interact. We acknowledge that our system does not cover the full space of brawler games (more players, more moves) - the complexity described above scales dramatically with every new mechanic introduced. Due to the paucity of existing work on brawler optimization, this study serves as a starting point to investigate the generation of emergent properties in this specific gaming environment.

Related Work

Many existing Automated Game Design systems generate high volumes of candidate games and use automated evaluation to choose a subset that are judged to be the most enjoyable for human players (Togelius and Shaker 2016). Among these, it is common to use data from AI play traces to inform the objective function (e.g. (Togelius and Schmidhuber 2008; Liu et al. 2017; Browne and Maire 2010; Cook, Colton, and Gow 2016a)). This approach has also been used extensively for game balancing, where the game is mostly fixed except for optimizable parameters (Beau and Bakkes 2016; Volz, Rudolph, and Naujoks 2016; Preuss et al. 2018). Another approach involves leveraging learning techniques that strive for optimal gameplay strategy as in (Yu and Sturtevant 2019), though these methods are likely too computationally costly to have in an AGD evaluation function.

A wide variety of desired characteristics have been considered and used in fitness functions for automated game design. In addition to simple fairness, the existence of multiple competing strategies (Preuss et al. 2018), learnability (Togelius and Schmidhuber 2008; Liu et al. 2017), and the uncertainty of the outcome (Browne and Maire 2010) are a few examples. Some systems, like ANGELINA, evolve multiple parts of the game at the same time, evaluating each part individually, as well as the interplay between the pieces(Cook, Colton, and Gow 2016a,b). Our work lies somewhere between balancing and game generation: while the base game mechanics are not subject to change, the level geometry is evolved alongside the characters, and the parameter space for the characters is large.

While automatically evolving games is common, it is

Entity	Parameter	Description	Range
Character	Ground Acceleration	Acceleration of players on the ground	0-1
	Air Acceleration	Acceleration of players in the air	0-1
	Max Ground Speed	Maximum self-applied speed on the ground	2-10
	Max Air Speed	Maximum self-applied speed in the air	2-10
	Ground Jump Force	Force applied from jump on the ground	1-15
	Air Jump Force	Force applied from jump in the air	1-15
	Mass	Impacts momentum and knockback	0.5-2.5
	Drag	Impacts air control	1-6
	Width	Scales width of player hitbox	0.7-1.5
	Height	Scales height of player hitbox	0.5-2.5
	Gravity	Scales how gravity impacts player	0.3-1.3
	Hitstun	Scales how a character responds to hitstun	0.1-0.3
Move	Distance	Distance between move and character center	0.8-1.5
	Angle	Angle Made between character and move	0-2*PI
	Width	Scales width of move hitbox	0.5-1.5
	Height	Scales height of move hitbox	0.5-1.5
	Warm-Up	Duration of move warm-up	0.1-0.6
	Execution	Duration of move execution	0.1-0.4
	Cool-Down	Duration of move cool-down	0.1-0.4
	Damage	Scalar for damage applied on hit	0-10
	Knockback	Scalar for knockback applied on hit	1-16
	Knockback Direction	Direction knockback applies on hit	(0-1, -1-1)
	Hitstun Duration	Determines base hitstun applied by move on hit	0-1
Platform	Location	Position of the platform (top left)	-
	Size	Size of platform in width and height	-

Table 1: A List of parameters used for game generation including ranges for parameter generation. Platform width and location are dependent on the game's bounds.

less common to empirically evaluate the fitness function. The success of the generated games does not necessarily mean that the fitness function actually optimized for the desired qualities. Some prior work solves this problem directly by using player preferences in the fitness function (Colton and Browne 2009; Hastings, Guha, and Stanley 2009). This solves the problem that the fitness function may be misaligned with player judgment, but it requires users to manually play and evaluate many games. Some notable prior work does take an extra step to validate their evolutionary strategy. In (Browne and Maire 2010), Browne and Maire validate their fitness function with a user study that ranked the games, showing how well each criterion in the fitness function correlated with human enjoyment of the games. In (Isaksen, Gopstein, and Nealen 2015), Isaksen et. al. optimize directly for a target difficulty of a one-button single-player game. They performed a user study to validate their results, showing that human judgment broadly aligns with AI judgment in terms of the difficulty of the games. However, the AI in question was a player model that makes human-like mistakes and it might be infeasible to create an accurate model of human play for more complex games. Without such a model, it is still possible to use more rudimentary AI techniques to playtest a candidate game. However these playtests may be very different from actual human play in crucial ways, and understanding these differences is necessary to designing effective AI playtesters.

Brawler Generation

We used the Unity game engine to implement a brawler game generator BrawlerAGD [1]. We used free assets from Kenney licensed under CC0 1.0 [2]. All games are playable with either a keyboard or Microsoft XBox controllers.

Base Game Mechanics

Each generated game consists of two characters, a move that each character uses, and a stage that the characters play on. During gameplay, each character has 3 stocks (or lives), and there is a static "blast zone" rectangle which causes characters to lose a life when they leave it. The loss condition is static: a player that loses all of their stocks loses the game.

The characters are parameterized by statistics used by other brawler games. Each character can jump twice, but the height of each jump is controlled by the generator, along with other physical attributes like the character's size, movement speed, falling speed, and weight. The sprite is chosen randomly and scaled according to the generator. To create nuanced attacks, the generator controls the timings of the

[1]Code for this project is available at https://github.com/smshields/BrawlerAGD. All trial games used are playable in the repository, and the full evolutionary process is available for interested readers to generate their own new games. An example screenshot is shown in Figure 1.

[2]https://www.kenney.nl/assets/bit-pack

attacks, the placement of the attack relative to the character, and the knockback direction and damage of the attack. Finally, the stage is a list of platforms that each character can stand on, and block the characters from moving through them. The space of characters is large, with each character-move pair consisting of 24 independent parameters (Table 1) each of which has a predefined range of valid values. Stage designs can consist of up to 10 separate platforms.

Game Fitness

We perform evolutionary search over the space of candidate games. Stages are initially designed with an algorithm that creates a symmetrical stage where platforms are placed in accordance with player jump height. Players are generated initially by generating each parameter according to a uniform distribution on the valid set of parameter values.

We evaluate each game using the results of an automated playtest. The playtest is performed by a decision-tree AI. The AI continually tries to move towards a position that would put its opponent within the hitbox of its attack. Randomness is added to character movement (e.g. changing directions unexpectedly) to ensure that separate trials of the same game are not identical and so that characters do not get "stuck" when separated by platforms. If its character is off-stage, the AI switches to a recovery behavior tree, where it tries to get back onto the stage. This AI is capable of playing interesting matches, but the games it plays are visually distinct from human play. For this implementation, we determined that this simple agent would suffice in eliminating most extreme edge cases of generated characters: overpowered characters show imbalances in damage distribution between players, and underpowered characters lose all lives or inflict no damage to their opponent.

After the automated playtest, the objective function is calculated from a variety of information. From the playtest, we obtain ℓ the game length in seconds, d_1, d_2, the total amount of damage dealt by player 1 and player 2, s_1, s_2, the number of remaining stocks for each player after the game has ended, and h_1, h_2, the total number of hits taken by each player. From this, we calculate many fitness variables, each intended to influence a specific aspect of the candidate games. These variable are normalized to ensure that we do not overtune for any given property. The overall fitness function is the sum of the fitness variables.

We want games to last an appropriate amount of time and remove games that are unplayable. To keep the game "fast" paced and to keep our genetic algorithm efficient, we set a desired time and punished games that were too short or too long. We terminated overtime games, adding a constant score punishment (-35) if games lasted longer than a minute. The time fitness for a game is

$$f_{time} = ||\ell - \ell_{target}|| - 35 * g$$

Where ℓ_{target} is the target game length: 45s, and g is an indicator variable for whether the game ended in a draw due to running out of time after 1 minute of gameplay.

We encourage games where more damage is dealt, and more total player interactions take place:

$$f_{damage} = (d_1 + d_2)/10$$
$$f_{hits} = h_1 + h_2$$

We want each life for each player to end at an appropriate time and ensure that the system cannot blindly optimize for f_d by creating a game where players are stuck in a corner hitting each other. We set the desired damage per stock to 100, and if the total damage exceeds that, we add a linearly scaling penalty variable $f_{stocklength}$.

We also want each game to be fair. We apply a fitness penalty to games where the total damage dealt by each player is different, or where the stocks remaining are different:

$$f_{damagefair} = ||p_1 - p_2|| * 10$$
$$f_{stockfair} = ||s_1 - s_2||$$

With the overall fitness function being:

$$f = f_{time} + f_{damage} + f_{hits} + f_{stocklength} + f_{damagefair} + f_{stockfair}$$

Evolutionary Search

We evolve candidate games with a population size of 100, a mutation rate of 0.4, and a dropout rate of 0.5. Characters and moves are crossed-over by treating them as a list of parameters and applying single-point crossover. We leverage crossover as our character and level parameter lists are not organized according to emergent gameplay output, meaning we effectively randomize the design space between games by selecting a random point in the list and combining. Stage designs are crossed-over by treating them as a list of platforms with single-point crossover. To mutate a player or move, we re-generate 5 randomly chosen parameters using the ranges of parameters used in initial generation. To mutate a list of platforms, the entire stage is re-generated. Table 2 lists pilot study games' specific fitness scores as well as the average fitness scores of the generations that those games were selected from.

User Study

We conducted a user study to understand how the genetic search of BrawlerAGD influences emergent properties of the games when played by human players. In particular, we focused on game fairness, i.e., the equal chance for each player to achieve a win given equal skill. We had users play both (1) randomly generated games and (2) games with high fitness chosen after genetic search. This study examines how players rate games from each of these groups across the following traits: ease to learn, balance, enjoyability, understandability, immersion, quality of design, and ease of control. We also examine differences in gameplay data by recording the total number of hits, the amount of damage, winners, and the game length. Players were asked to describe every

Game	Number of Generations	Agent Fitness	Human Fitness	Generation Top Fitness	Generation Average Fitness	Generation Average Holdover Fitness
A	315	109.32	88.26	128.27	50.12	83.14
C	98	129.47	146.29	138.47	63.56	96.93
E	224	122.23	64.53	109.52	48.67	84.44
B	0	-12.08	11.84	41.16	-10.44	2.12
D	0	-9.98	21.37	41.16	-10.44	2.12
F	0	-10.00	13.44	41.16	-10.44	2.12

Table 2: A table highlighting data from the generations of game samples used in the study. Note that randomly generated came from the same batch of 100 games. Average fitness denotes the average fitness of all games in a generation, and average holdover fitness shows the average fitness of all games that were kept for the subsequent generation.

game in 3 words and identify their favorite game. If we can show that evolved games have perceivably higher qualities than our randomly generated games, we can say our system is successful in designing "balanced" brawler games.

We ran 12 total trials with 24 participants. Two instances of participant data (from trials 11 and 12) were excluded due to an incomplete survey. On average, our players were moderately familiar with brawlers, with one player indicating they were not at all familiar with brawlers and five indicating that they were extremely familiar with them.

Game Selection

To compare the random and evolved games, we used 6 sample games. Games were tested by having players play a survival match against each other, with each player having four stocks per game. To choose the random games, we generated and evaluated a set of 100 games, with a mean fitness score of -10.22. We randomly selected 3 games with fitness -10 ± 5 to use in the study. These randomly generated games were named B, D, and F.

Each run of the evolutionary search plateaued with a mean fitness score of 80-100 after 12 hours of computation. Fitness runs were taken consecutively. To select evolved games, we ran the evolutionary search to completion three times (average fitness above 80 for 20 generations). Within the final generation of each run, we grouped all games with fitness 100 or higher and randomly selected a game for use in the trial. Evolved games A, C, and E were generated from this process. The shortest evolved game was generated in generation 98, while the longest was created in generation 315.

Measurements

Users started the study by playing a tutorial level where they learned the basic mechanics of games generated by BrawlerAGD using human-defined characters. Users were asked to verbally confirm that they understand game controls and goals before leaving the tutorial. Users then played each game in a random order, answering questions after each game is completed.

During trials, we collected the same gameplay data used to evaluate the fitness function. This included the total amount of hits and damage to each player, the game length, and the number of stocks remaining for each player. We leverage this data to get an understanding of how human gameplay corresponded to our written heuristic. We dropped f_{time} and $f_{stocklength}$ terms because humans play slower than our AI and occasionally took time to joke, ask questions, or experiment with characters.

Results

We outline the general trends across the various games, both as observed by human players, and as seen through the statistical data about their matches.

User Perception

To understand if the evolved games had better perceived characteristics, we first had to understand if the evolved games scores were statistically different from the random game user survey results. We used a t-test between the two groups (evolved and random, 3 games each) to examine the perceived differences between evolved and random games. Only one factor had a statistically significant ($p < 0.017$) difference: ease of control. Evolved games were on average slightly easier to control as perceived by users. To understand which of the games were specifically impacting that result, we performed a one-way analysis of variance (ANOVA) on all six games.

In our ANOVA comparison between all six individual games, we found that both balance and ($p < 0.001$) ease of control ($p < 0.013$) had significant differences. In particular, game C (an evolved game) was perceived to be more balanced and easy to control than the other games. Game F (a random game) was also perceived to be balanced, but was extremely difficult to control.

One simple observation comes from our post-trial survey: 63% of users listed an evolved game as their favorite game. None of the randomly generated games had a higher share of votes than the lowest evolved game.

Game Statistics

We also examined how human gameplay in the evolved games differed from gameplay in the random games, using t-tests on the quantitative data gathered from the user study. We recorded damage dealt to each player, the number of hits received by each player, the game length, and the game

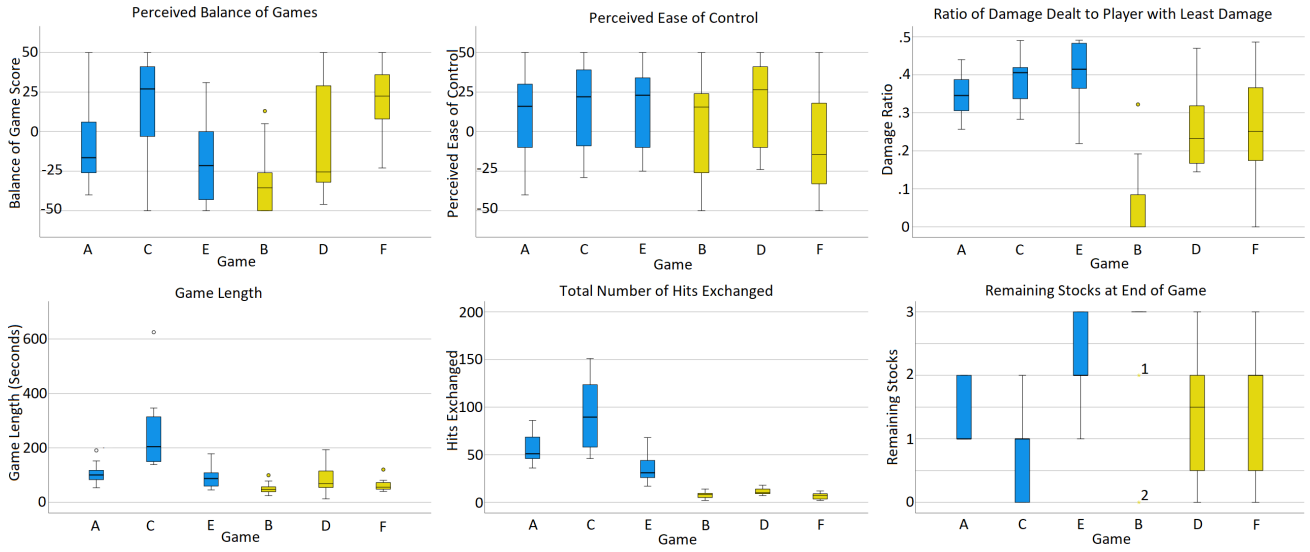

Figure 2: Distribution of user scores across a variety of measurement for evolved games (blue) and random games (yellow).

loser. Every one of these metrics, aside from game loser, showed statistically significant differences when comparing evolved games to random games ($p < 0.001$). Some examples of these trends: Evolved games tended to have higher amounts of damage and hits to each player. Each evolved game lasted, on average, more than twice as long as the randomly-generated games. These differences indicate that players had more interactions for a greater amount of time when playing evolved games compared to random games. Figure 2 summarizes these trends.

To evaluate fairness, we looked at the ratio of damage dealt to the player who was dealt the least damage (see Figure 2). This ranges from 0 to 0.5, where 0.5 indicates that player 1 and player 2 each dealt the same amount of damage to each other, whereas 0 indicates that one player took all damage. We use the ratio to be able to compare games where different amounts of total damage was dealt. We found that evolved games had on average 0.19 ($p < 0.001$) increased damage ratio, indicating that evolved games encouraged more trading of blows than random games. We found the same trend when looking at number of hits. Despite having more activity from both players, and a more equivalent distribution of attacks and damage, we did not observe the evolved games had a more equal distribution of wins (player 1 wins as often as player 2).

Discussion

Here we discuss qualitative observations collected throughout trials and game evolution. These trends help provide explanations for our survey and gameplay data.

Game Quality

Two of the games stood out both qualitatively and statistically - Game C, an evolved game, and Game F, a randomly generated game. C consisted of one short and one tall character, both with moves that pointed downward. In order for characters to land hits in C, they had to jump a single time, position themselves over the opponent, and trigger their attack. In F, both characters are roughly equal size but jump very high and move very quickly, making them challenging to control and land attacks on opponents. Attacks in F were high-powered and often killed in one or two hits.

Both of these games were evaluated as "balanced" by our players by a similar margin (and these results are also borne out by the empirical gameplay data). Game C (Evolved) scored the highest on most survey questions, while also producing games with the greatest length and damage/number of hits. Game F (Random) scored high on balance but scored poorly in control. In descriptions of the game, 3 players called the game "slippery" while others commented that the controls felt "touchy" and characters felt "floaty". That the generator produced multiple examples of balance with human-perceived differences in game-feel indicates that the system is capable of producing balanced games that are not homogeneous or trivial in their gameplay.

C and F are just one example of a general trend in the differences between random and evolved games. While the system can randomly create games with some desirable characteristics, it is unlikely to find games that combine multiple desirable characteristics without some kind of search. This explains the lack of statistical significance in the individual categories, despite the preference for evolved games in aggregate.

Automated Playtester Alignment

Players often developed gameplay patterns similar to behaviors performed by the AI during evolution. Game C scored highly during evolution as AI characters would jump over one another inside of a corridor, attacking one another with their downward-facing attacks (as shown in Figure 1). This caused the automated testers to keep trading hits until one was flung outside of the corridor. During trials, most partici-

pants followed an extremely similar behavior pattern, which in turn led to similar damage and hit outcomes.

This effect backfired when users did not follow or understand the evolved game's key quirk. When users diverged from the most common AI behaviors, evolved gameplay started to become very unpredictable and unbalanced. I.e., the automated playtesting did not cover a broad enough set of gameplay styles to stand in for human play. Some stages forced or enticed users to play in a similar way as the automated tester, and these scored numerically higher on fitness when played by humans. The design of each game influenced how well the automated playtester was aligned with human play. While the system is far from perfect in covering all gameplay possibilities, the ability for the system to even infrequently create such intricate behaviors in players by tuning parameters indicates that this style of search can create distinct, meaningful, emergent playstyles while designing games.

Trends in Evolved Games

There were some general trends in game design that we observed in the evolved games. Most of the evolved games tended to have characters with move hitboxes that would directly put the character in range of their opponent's hitbox. Evolved games also often had characters with attacks that could only be performed in some specific character state: for example, the downward attack in game C required players to jump first. These patterns meant that every attack would put player at risk, which encouraged interaction between the characters. If a player missed an attack, it can be immediately punished by their opponent. Another quirk of high-fitness games is that many attacks had knockback directions that pointed diagonally backwards towards the attacking character (for example, game E). Most human players found this knockback confusing or disorienting. Players would line up what they thought was a fatal hit, only to have the resulting knockback push themselves off of the stage. This is an example of how a fitness function designed to increase the number of player interactions can create game elements that are incomprehensible to humans.

Limitations

While we had modestly positive results towards evolved games in user perception, we did not show that users consistently perceived evolved games as more balanced than randomly generated ones, even if their created heuristics would have indicated to us that they should have. We attribute this in part to the complexity of the term "balance" and making judgments about it. While balance may be intuitive to most people, that intuitiveness does not translate to agreement on what is balance between users or games. Narrowing definitions of the emergent properties we desire in these systems or grouping users into persona categories could help in future evaluations of emergent properties in AGDs.

Players also played the games against each other. When playing random games with clear mechanical imbalances (see Game B, where one player had their move behind and below their sprite), the winner would often feel very positive about their win. They might gloat, and on one occasion, one said "that was the most balanced game we've played yet". Furthermore, a player with a long history of brawler fighting games matched against a newcomer might make both feel that every game was not balanced. It should be noted that while this was qualitatively observed, the data does not clearly bear this out - winners and losers did not have statistically significant differences in balance judgments in games.

Conclusion

We created an automatic game designer that generates brawler-fighting games. We optimized these games for simple metrics that measure desirable properties about the games like balance, ease of control, and interactions between players. By evaluating the generated games with a user study, we learned about the ways in which our fitness function influenced human gameplay. Broadly, evolved games exhibited higher fitness when played by humans, indicating that automated playtesting (even with a very simple AI) can help optimize human play for given gameplay properties. The qualitative and quantitative matching of simple AI performances and human play indicate that our designer can also discover interaction patterns and mechanics that humans might enjoy.

These statistical differences did not necessarily translate well to human perception of the games, a problem also described in (Isaksen, Gopstein, and Nealen 2015). Ultimately, the goal of evolving games towards a fitness function is to improve and shape the user experience of the person who eventually plays those generated games. Towards that end, it is important to consider how the fitness function might be misaligned with how users will experience the game. A "high-fitness" game might not actually have the properties it was optimized for when played by humans. It is also important to tune the evaluation agent to better match human behavior. A future study might look to evaluate what factors might be critical in creating a balanced game for a future heuristic, or what types of agents create the most accurate representation of human play.

This means that data from human players should be more central when it comes to the design of automated evaluation. One path forward is more advanced automated playtesting that incorporates user data to play the game in a human-like way. This would extend research on player modeling in PCG such as (Shaker, Yannakakis, and Togelius 2010). Another method would be to change the generative techniques themselves to incorporate user data (for example, (Sorochan et al. 2021) does this for a single-player game). A novel approach might be to use techniques from Human-Computer Interaction, such as "Design Personas" - user profiles that guide design of an artifact (Salminen et al. 2022). Finally, by correlating design variables with human perception of the resulting game, a system could guess the perception of a novel game, allowing it to automatically adjust the fitness function in response to user feedback.

Appendices

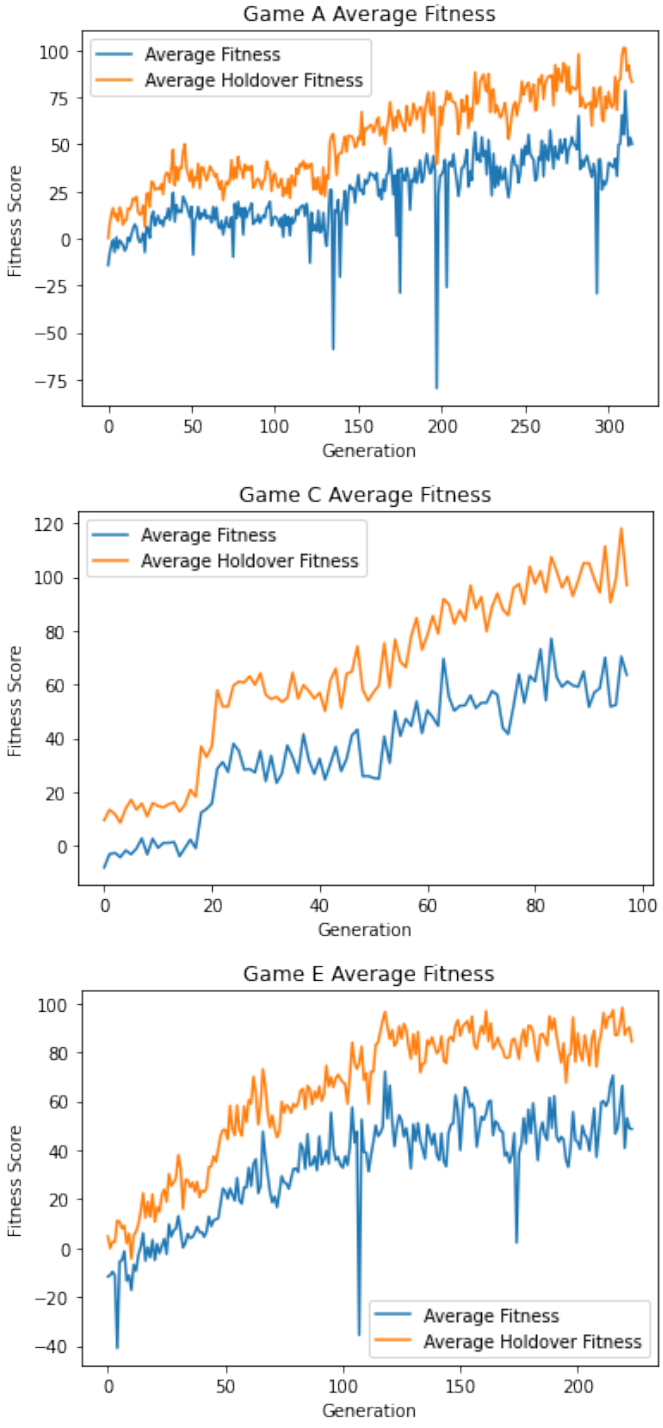

Figure 3: Fitness graphs for each of the evolved games from the user study. Each graph shows the fitness of the population that games A, C, and E were selected from, respectively.

Question Timing	Question	Response Format	Required
Pre-Trial	Please indicate your familiarity with "brawler" fighting games (examples include: Super Smash Bros, Playstation All Stars Battle Royale, Nickelodeon All Star Brawl)	Familiarity, 1-5	Yes
	Full Name	String	Yes
	Player Number	1 or 2	Yes
Post-Game	I found the game easy to learn	Slider, -50 to 50	Yes
	I found the game to be balanced between both characters.	Slider, -50 to 50	Yes
	I found the game to be enjoyable	Slider, -50 to 50	Yes
	I understood what was happening throughout the game.	Slider, -50 to 50	Yes
	I felt mentally immersed in the experience.	Slider, -50 to 50	Yes
	I found the game to be well designed.	Slider, -50 to 50	Yes
	I found the game easy to control.	Slider, -50 to 50	Yes
	Did you find your character substantially stronger or weaker than your opponent's character?	Stronger, Neither, Weaker	Yes
	Please indicate how you think this game was created.	Random, AI, Human	Yes
	In less than three words, describe the game.	Text	Yes
Post-Trial	Provide a description of allowed player actions (e.g. move right) and their corresponding control mapping.	Text	Yes
	Did you observe any characters that felt under- or over-powered? If so, please describe.	Text	No
	Were there any distinguishing designs that separated some games from others? For example, did a set of characters or level stand out to you? If so, please describe.	Text	No
	Indicate your favorite game played.	1, 2, 3, 4, 5, 6	Yes
	Please provide any other feedback you'd like about the game and/or survey.	Text	No
	Any additional Comments?	Text	No

Table 3: Full list of questions used in the user study survey. Specific slider values were derived from a slider that the users would move between Disagree/Agree.

References

Beau, P.; and Bakkes, S. 2016. Automated game balancing of asymmetric video games. In *2016 IEEE conference on computational intelligence and games (CIG)*, 1–8. IEEE.

Browne, C.; and Maire, F. 2010. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1): 1–16.

Capcom Co. Ltd. 1987. Street Fighter.

Codsi, J.; and Vetta, A. 2021. A Case Study in Learning in Metagames: Super Smash Bros. Melee. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 17(1): 2–9.

Colton, S.; and Browne, C. 2009. Evolving simple art-based games. In *Workshops on Applications of Evolutionary Computation*, 283–292. Springer.

Cook, M.; Colton, S.; and Gow, J. 2016a. The angelina videogame design system—part i. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(2): 192–203.

Cook, M.; Colton, S.; and Gow, J. 2016b. The angelina videogame design system—part ii. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3): 254–266.

Dan Fornance. 2017. Rivals of Aether.

Hastings, E. J.; Guha, R. K.; and Stanley, K. O. 2009. Evolving content in the galactic arms race video game. In *2009 IEEE Symposium on Computational Intelligence and Games*, 241–248. IEEE.

Isaksen, A.; Gopstein, D.; and Nealen, A. 2015. Exploring Game Space Using Survival Analysis. In *FDG*.

Liapis, A.; Yannakakis, G. N.; Nelson, M. J.; Preuss, M.; and Bidarra, R. 2018. Orchestrating game generation. *IEEE Transactions on Games*, 11(1): 48–68.

Liu, J.; Togelius, J.; Pérez-Liébana, D.; and Lucas, S. M. 2017. Evolving game skill-depth using general video game ai agents. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2299–2307. IEEE.

Ludosity, Fair Play Labs. 2021. Nickelodeon All-Star Brawl.

Nintendo Co. Ltd. 1999. Super Smash Bros.

Preuss, M.; Pfeiffer, T.; Volz, V.; and Pflanzl, N. 2018. Integrated balancing of an rts game: Case study and toolbox refinement. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Salminen, J.; Guan, K. W.; Jung, S.-G.; and Jansen, B. 2022. Use Cases for Design Personas: A Systematic Review and New Frontiers. 1–21. ACM. ISBN 9781450391573.

Shaker, N.; Yannakakis, G.; and Togelius, J. 2010. Towards automatic personalized content generation for platform games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 6, 63–68.

Sorochan, K.; Chen, J.; Yu, Y.; and Guzdial, M. 2021. Generating Lode Runner Levels by Learning Player Paths with LSTMs. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, 1–7.

Togelius, J.; and Schmidhuber, J. 2008. An experiment in automatic game design. In *2008 IEEE Symposium On Computational Intelligence and Games*, 111–118. Citeseer.

Togelius, J.; and Shaker, N. 2016. The search-based approach. In *Procedural Content Generation in Games*, 17–30. Springer.

Volz, V.; Rudolph, G.; and Naujoks, B. 2016. Demonstrating the feasibility of automatic game balancing. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 269–276.

Yu, K.; and Sturtevant, N. R. 2019. Application of Retrograde Analysis on Fighting Games. In *2019 IEEE Conference on Games (CoG)*, 1–8.