Exploring Game Space of Minimal Action Games via Parameter Tuning and Survival Analysis

Aaron Isaksen, Dan Gopstein, Julian Togelius, and Andy Nealen

***Abstract*—Game designers can use computer-aided game design methods to model how players may experience the perceived difficulty of a game. We present methods to generate and analyze the difficulty of a wide variety of minimal action game variants throughout game space, where each point in this abstract design space represents a unique game variant. Focusing on a parameterized version of *Flappy Bird*, we predict hazard rates and difficulty curves using automatic playtesting, Monte Carlo simulation, a player model based on human motor skills (precision and actions per second), and survival analysis of score histograms. We demonstrate our techniques using simulated game play and actual game data from over 106 million play sessions of a popular online *Flappy Bird* variant, showing quantitative reasons why balancing a game for a wide range of player skill can be difficult. Some applications of our techniques include searching for a specific difficulty, game space visualization, computational creativity to find unique variants, and tuning game balance to adjust the difficulty curve even when game parameters are time varying, score dependent, or changing based on game progress.**

***Index Terms*—Automated playtesting, computer-aided game design, *Flappy Bird*, player modeling, survival analysis.**

I. Introduction

GAME designers define and set game parameters to tune a game to achieve a desirable player experience, typically relying on iterative design, intuition, experience, and user feedback. As shown in Fig. 1, the designer must estimate player skill, set game parameters, play test, evaluate player experience using game play metrics, revise parameters, and iterate until the game reaches an appropriate level of difficulty. We present in this work our novel methodology for tuning game parameters based on the iterative process, but which does not require human input in the feedback loop.

Each unique parameter setting creates a new game variant. We refer to this high-dimensional space of game variants as game space [1].[1] A point in the game space defines the game parameters for the game variant; these game parameters directly affect the player, enemies, or the level generation [2]. For example, Fig. 2 shows a 2-D game space for a *Space Invaders* [3] type game, with four points representing four unique game variants, each with a different difficulty. Exploring the game space to find specific settings for an optimal experience is a considerable challenge, and we demonstrate the use of quantitative methods to better understand the relationship between game parameters and a player's perception of difficulty [4].

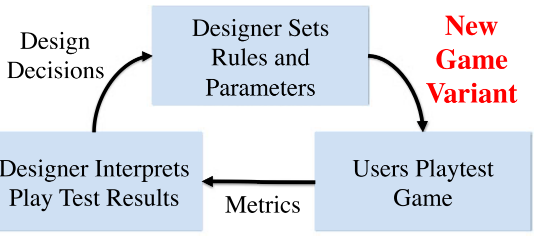

Design Decisions
Designer Sets Rules and Parameters
New Game Variant
Users Playtest Game
Metrics
Designer Interprets Play Test Results

Fig. 1. Iterative game design feedback loop. Each cycle through the loop creates a new game variant.

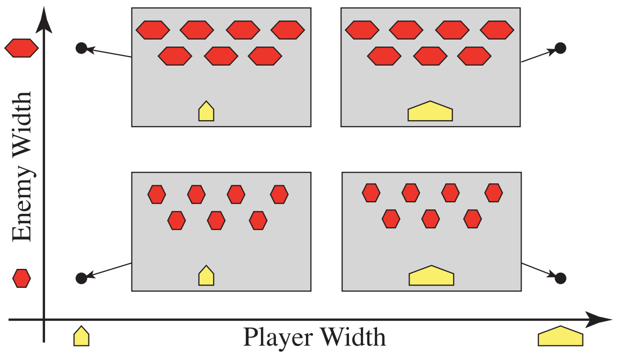


Fig. 2. Two-dimensional game space of Space Invaders-like variants with different player and enemy width. Intuitively, we expect some of these variants to be more difficult than others.

We can expand our general understanding of games by studying a simple example game in detail: in particular, it is useful to compare different game variants, compare players of different skill levels, and compare how difficulty changes as players learn from repeated plays [5]. Here, we focus on how game parameters, without changing the game rules, can affect a game's perceived difficulty. This allows us to better understand how players experience minimal action games [6] and leverage this knowledge to determine how parameters of a particular game affect its difficulty. For example, we expect the variants of *Space Invaders* in Fig. 2 with wider players to be more difficult than those with narrower players (since the player is more likely to be hit by the alien bombs), even if the variants are otherwise identical. The methods in this paper allow us to precisely measure these types of parameter changes algorithmically, without human playtesting or feedback. Inspired by cocreativity and a mixed-initiative design, where human designers work together

Manuscript received December 31, 2016; revised May 3, 2017 and July 14, 2017; accepted August 16, 2017. Date of publication September 8, 2017; date of current version June 14, 2018. *(Corresponding author: Aaron Isaksen.)*

The authors are with the Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: aisaksen@appabove.com; dgopstein@nyu.edu; julian.togelius@nyu.edu; nealen@nyu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2017.2750181

[1]Project website at game.engineering.nyu.edu/exploring-game-space.

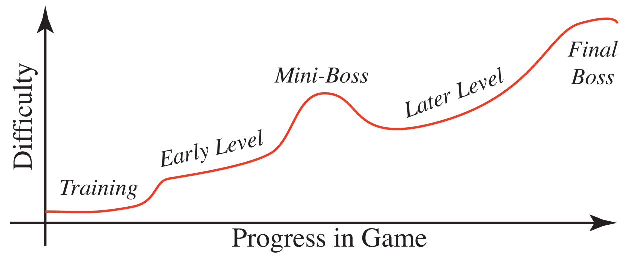


Fig. 3. Difficulty curve allows designers to qualitatively model how difficulty changes in a game. We present quantitative methods to represent this curve.

with computers to cocreate a game [7], our goal is to use quantitative methods to generate content and information for humans to evaluate.

For our research, we performed the first research-oriented quantitative design analysis of *Flappy Bird* [8], a commercially and critically successful mobile game with relatively few adjustable game parameters that impact core game play (see Section III for a description of the rules). *Flappy Bird* is a remarkably simple game, yet its parameter space is also surprisingly rich. While more complicated games will require more detailed analysis, the methods we use to generate, simulate, and analyze an action game still apply to more complex games [9]. The set of all game variants for a specific game can be very large and therefore impossible to search exhaustively—imagine adjusting hundreds of independent control knobs to search for the perfect game. We, therefore, limit our game space exploration by focusing on game variations caused by parameter changes, not rule changes.

Furthermore, we show how score probability distributions are a useful tool for understanding, modeling, and comparing player experiences of game difficulty. Because scores are quantitative by nature, they are particularly useful data for exploring game space and the quantitative analysis of games. To explain how the score probability distributions predictably respond to game parameter changes, we present the first application of survival analysis to the game design. Survival analysis is a branch of statistical modeling that helps predict how long in the future an event may occur [10], [11] and is typically used to study the lifetime of mechanical parts and the effectiveness of medical treatments. We use survival analysis in games to predict the likelihood of a player achieving a specific final score. In many action games, the longer a player survives, the higher their score, so we use score as a replacement for lifetime. Although we focus on probabilities of achieving specific scores, one can use our approach to model any nondecreasing factor, such as time played, distance traveled, etc.

Survival analysis provides a well-defined hazard rate as the rate at which a player will fail given that they have already reached a score x in the game. This provides a quantitative metric analogous to the qualitative difficulty curve, which gives designers a way to think about how difficulty changes throughout a game or level [4]. In Fig. 3, a hypothetical difficulty curve is shown, with the x-axis representing progress and the y-axis representing difficulty. Designers may choose to make parts of the game easier or harder to adjust drama, tension, and challenge [12], [13]. This paper presents our methods for building these difficulty curves using quantitative methods based on artificial intelligence (AI)-enabled player agents.

A. *Organization*

This paper is organized as follows. Section I describes the overall research project, while Section II discusses how our research relates to previous work in game parameter exploration. In Section III, we define the game space for *Flappy Bird*, including new game design parameters for controlling time-varying difficulty. In Section IV, we describe our player model, which simulates error-prone players, modeling inaccuracy in timing precision, and limited actions per second. In Section V, we give a short mathematical overview of survival analysis and define the hazard rate, our mathematical model for difficulty curves. In Section VI, we present our method for estimating hazard rates, using simulation and survival analysis to determine the difficulty curve.

In Section VII, we simulate different games and players, showing how varying different parameters of the game or player model leads to different score probability distributions and hazard rates and, therefore, perceived difficulty. We also use our theory to analyze over 106 million plays of flappybird.io [14], showing the likely presence of learning effects in the score data [15]. We then adjust the hazard rates by changing game parameters to maintain an effectively constant difficulty while players improve.

In Section VIII, we show how our techniques can be used to visualize the design effects of each game parameter. We also use computational creativity and optimization techniques to find games of specific difficulties as well as finding k unique game variants that explore the possibilities in the *Flappy Bird* game space (this process is related to novelty search [16]). Finally, in Section IX, we discuss our existing experiments and present areas for future research.

II. Related Work

In this section, we discuss how our research relates to the existing work in the field of AI-assisted quantitative game design. It is quite common in design to focus on a particular area of the problem (for example, focusing one design iteration on the abilities of enemies and a separate design iteration on the layout of the levels) [12]. This same approach can be used to focus on one set of parameters at a time, lowering the dimensionality of the problem. For games with significantly more parameters, one can use dimensionality reduction techniques to find a lower dimensional space of parameters, addressing correlated parameters [17], [18]. While our techniques presented in this paper can scale, the method here requires the game to be an action game focused on dexterity challenge where score is related to lifetime. For example, in infinite runners like Canabalt [19], one could use this technique to tune collision hit box sizes, gap sizes, and running speeds. Nonetheless, this is a large and popular class of games, especially for web and mobile action games.

Accurately simulating game play requires an understanding of how players react to game events: this is the process of player modeling [20]. Our player model assumes that much of the difficulty in minimal action games is due to human motor skill, specifically precision and reaction time [21]. This approach is related to biological constraint modeling, where we apply simple models to represent the limitations of human ability [22]. Since we use a static, objective, simulation-based player experience model [23], we do not need to analyze prerecorded human play sessions of the specific game to train our player model and do not rely on active live players to estimate the current level of challenge or fun [24]–[29]. We also focus on single-player games, not the related challenge of quantitatively balancing multiplayer games [30], [31].

Our goal is to make the simulator exhibit human-like play performance (novice, average, or skilled), not play with superhuman ability [32]–[34]. As long as our system properly predicts human perception of difficulty, it fits our purposes. In minimal, well-balanced, and compelling action games like *Flappy Bird* or Canabalt, the player takes a relatively obvious path, but executing that simple path is challenging [6]. In this paper, we examine a game with simple path planning and without enemies, so we can focus on the player's ability to control the game—without analytical evaluation of possible player movement [35], multifactor analysis of player or enemy strategies [36], estimating player abilities [37], dynamic scripting of opponents [38], building machine learning estimators [29], [39], [40], or evaluating design via heuristics [41].

Our work relies on internal game metrics [42], [43]. However, game metrics typically require the game to be publicly available, have a significant number of players, and time to gather enough data, and therefore, they have limited utility for rapid iteration in the early design process. Additionally, when designers become expert at their own game, they can lose perspective on how their game is experienced by new players. It can also be difficult for designers to break out of local optima and creatively explore new regions of game space. Automated playtesting [24], [37], [44], visualization [45], [46], and geometric path modeling [47] can help with this process, guiding designers to create games best suited to individual skill levels and play styles [36].

Probability distributions have been used for modeling session times [48] and total play time [49]. Score probability distributions tell us the likelihood that a player will achieve a specific score, and we can create these distributions by repeatedly playing a game and collecting the resulting score frequencies in a histogram. Automatically creating new rules [30], [50]–[53] is a related problem as rule changes also affect the distribution of scores achieved in game, but parameters alone have a significant impact on game feel [54]: getting Mario's jump to feel right is more about adjusting the parameters of a game physics engine, not coding the engine itself (or trying to emulate realistic physics).

This journal paper builds on a previously published conference paper [1] and two workshop papers [5], [55]. In addition to synthesizing the results of these papers and placing them in relation to each other, we provide a more in-depth survival analysis of *Flappy Bird* hazard rates, using techniques we later developed to handle nonconstant difficulty games and time-varying player skill. This provides a more concrete framework to discuss quantitative difficulty curve analysis than in our previous work. We also present new results on tuning parameters for games with time-varying and score-dependent parameters under conditions of varying player skill, specifically showing the effects on perceived difficulty when exploring dynamic *Flappy Bird* variants.

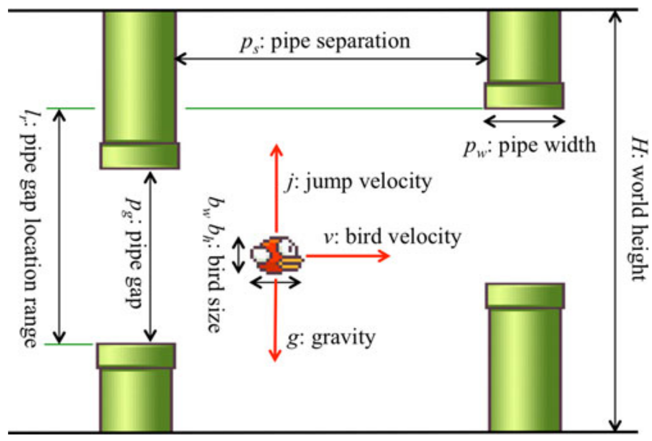


Fig. 4. In *Flappy Bird*, the player must navigate the bird through a series of pipes without crashing. We modify the labeled parameters to generate unique game variants.

III. Flappy Bird Game Space

Action games are defined by rules and parameters, designed to produce a specific player experience through challenges. Predicting the qualitative experience is hard, but we can examine the distribution of final scores to predict a quantitative difficulty of the game. Throughout this paper, we refer to this measured difficulty as $h(x)$, the hazard rate, which defines the rate at which a player is expected to fail given that they have already reached a score of x.

In *Flappy Bird*, a player must fly a constantly moving bird without crashing into a series of pipes placed along the top and bottom of the screen (see Fig. 4). Each time the player taps the screen, the bird flaps its wings, moving upward while gravity constantly pulls downward (resulting in a series of parabolic hops). Each time the bird passes through a pipe gap without crashing, the player scores a point. Part of the appeal for *Flappy Bird* is the comically high difficulty level, especially when typical casual games are easy and forgiving. *Flappy Bird* could have been much easier with a few small adjustments, such as increasing the gap between pipes or decreasing the width of the pipes, but these would have led to different and potentially less rewarding play experiences. In this work, we also examine variants of *Flappy Bird*, which change the pipe gap (the distance between the upper and lower pipes) and the horizontal bird speed as the game progresses. This allows us to examine variants where the game becomes more difficult as the player keeps playing.

We have chosen to use the following parameters for our implementation of *Flappy Bird* (see Fig. 4). The original *Flappy Bird* has a constant value for each parameter during a play session since the game does not change as the player progresses. More generally, game parameters can change as the player gets further into a level so here we also explore variants that get more difficult as the player progresses. *Pipe separation* p_s: More distance between pipes is easier to play, giving more time to react to changing gap locations. *Pipe gap* p_g: The distance between the upper pipe and the lower pipe. Narrower gaps are more difficult as the bird has less room to maneuver, requiring better motor skills. *Pipe width* p_w: Wider pipes increase difficulty as the bird spends more time in the narrow pipe gap. *Pipe gap location range* l_r: Pipe gap locations are uniformly distributed.

Larger ranges are harder because there is more distance to travel between a high gap and a low gap. *Gravitational constant* g: Acceleration of the bird in the y-direction. Higher gravity causes the bird to drop faster. *Jump velocity* j: When the bird jumps, its vertical velocity is set to j. Higher velocity makes higher jumps. Bird speed v: Speed at which the bird travels to the right (alternately, the speed at which pipes travel to the left). *World height* H: Distance between ceiling and floor. *Bird width and height* b_w, b_h: Size of the bird's hit box. Larger birds are harder to jump through gaps. *Change in pipe gap* Δp_g: Rate of change for pipe gap p_g after each pipe. The original game does not have this parameter, but we use it here for score-dependent difficulty. *Change in bird speed* Δv: Change in bird speed after clearing each pipe; also not present in the original game.

By varying these parameters within sensible ranges, we can generate many variants of *Flappy Bird*. Many of these parameters have constraints; for example, the pipe gap must be positive (or the pipes would overlap), the bird height cannot be larger than the pipe gap or the bird cannot fit through the gap, and gravity and jump velocity must have opposite signs.

IV. Player Model

We begin with a model of a player with perfect motor skills: a perfect player with instantaneous reaction who would never lose at the original *Flappy Bird*. Given a version of *Flappy Bird* defined by its game parameters (a single point in game space as defined in Section III), we create an AI that finds a path through the pipes without crashing. Instead of using an A* planner that finds the shortest path, we chose to use a simpler AI, which performs well but is easier to implement and faster to execute. Each time the bird drops below the target path, the AI immediately performs a flap action (which sets vertical bird speed v_y to jump velocity j). Whatever AI is used, it should play with very good performance on solvable levels and should only fail on impossible levels (such as a level with a tiny pipe gap where the bird cannot fit through).

We then extend the AI to perform less well by modeling the main components of human motor skill, which impact difficulty in these types of action games: precision, reaction time, and actions per second. Adjusting these values lets us model different player types, since novices react slower and are less precise than experts.

A. Player Precision

When a player plans to press a button at an exact time, they execute the action with some imprecision. We model this error as a normal distribution with standard deviation σ proportional to a player's imprecision (see Fig. 5). Imprecision is also related to the time a subject has to react to an event, called the speed–accuracy tradeoff: the less time they have to react, the less accurately they will respond [56]. For simplification, our player model assumes that precision is an independent variable and not dependent on bird speed. In a user study [1], we measured standard deviation of precision ranging between $\sigma = 35.9$ ms (when players have plenty of time to react) and $\sigma = 61.1$ ms (when players are required to act very quickly) and use this range for game space exploration.

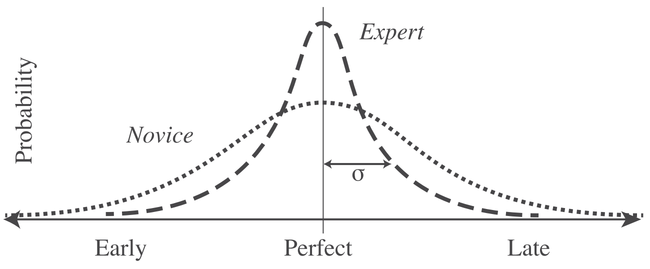


Fig. 5. Modeling skill by adjusting probability of executing moves on time.

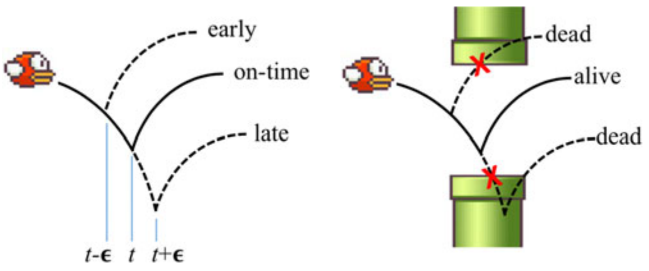


Fig. 6. Modeling precision by randomly adjusting the time the simulated player jumps. Moving the jump earlier or later can cause the bird to crash into the lower pipe or upper pipe.

We model imperfect precision in our AI by calculating an ideal time t to flap and, then, adding a small perturbation ϵ, drawn from a normal distribution with 0 mean and standard deviation σ, as shown in Fig. 6. By increasing the standard deviation, the AI plays less well and makes more errors, leading to a higher difficulty estimate. Reducing σ to 0 ms tests if a level is solvable by the AI without any imprecision error.

B. Actions Per Second and Reaction Time

Humans can only perform a limited number of accurate button presses per second. In our user study [1], we measured a maximum rate of 7.7 actions per second on average. We also limit our AI to this same number of actions per second. We simplify our model by keeping this constant, although a more complex model would account for fatigue, since players cannot keep actions per second constant for long periods.

When a player sees a new stimulus, it takes time to react. The speed of the player's reaction is influenced by factors inherent to the system [57], as well as factors affecting the player themselves [58]. We measured this delay as $\tau = 288$ ms, but as found in our experiments, this type of delay has a minor impact on estimating difficulty for *Flappy Bird* [1].

V. Survival Analysis

To quantitatively understand how simulations of games of varying difficulty and players of varying skill give rise to unique score probability distributions, we need a mathematical foundation to model player performance. The simulation data we create are in the form of discrete scores, but the underlying distributions are easier to describe in the continuous domain.

We begin by looking at the probability distribution function $f(x)$ for a game, which tells us the probability that the player will achieve a score of x on the next play. In practice, we can create a discrete probability distribution by recording all of the scores on each play of the game in a histogram.

The survival function $S(x)$ tells us the probability that a player will achieve a score $\geq x$. It is closely related to the cumulative distribution function $F(x)$, which tells us the probability that a score is $< x$, defined as $\int_0^x f(s)ds$ (where s is a free variable that we integrate from a score of 0 to a score of x). The survival function is defined as $S(x) = \int_x^\infty f(s)ds = 1 - F(x) = 1 - \int_0^x f(s)ds$. We have $S(x) = 1$ when $x \leq 0$ because every player will at least achieve a score of 0, and $S(\infty) = 0$ because all players will eventually reach a termination state and receive a final score.[2]

The hazard function $h(x)$ is useful for comparing probability distributions and models the difficulty curve of a game. Also called the hazard rate, it is defined as $h(x) = f(x)/S(x)$. The hazard rate tells us the rate at which we should expect to fail, given we have already reached a specific score x. Low hazards imply that the game is easy; high hazards imply harder games.

Note that $h(x)$ is *not* a probability distribution and is directly related to how one can think about adjusting difficulty curves. We are not as concerned about the entire probability distribution as much as how difficult a game is at a specific section, assuming the player has already reached that point in the game. Therefore, game designers are effectively working on modifying the hazard function $h(x)$ so the difficulty curve feels good to players [54]. Deciding what is "good" is beyond the scope of this work—some designers might want a game that increases in difficulty with an easier on-ramp; others might want a hardcore game that is difficult from the start. We give many examples of hazard functions in Section VII.

Impossible games, where $h(x)$ is undefined, are games that cannot be played successfully by any player (e.g., the bird cannot fit through the pipe gap). Trivial games, where $h(x) = 0$ everywhere, occur when all players will survive assuming a minimal amount of effort (e.g., if the pipe gap is as large as the screen—the player still needs to pay attention, but the game is trivial as long as they are actively participating). Playable games, where $h(x)$ is finite, are games that can be played successfully by some players, but not necessarily all of them.

To evaluate the hazard function empirically, we play the game N times and save each score in a histogram. We define $n(x)$ as the number of scores in the histogram equal to x. We can now calculate these empirical values from our data as

$$f(x) = \frac{n(x)}{N}, \quad S(x) = \sum_{s \geq x} f(s), \quad h(x) = \frac{f(x)}{S(x)}. \tag{1}$$

Because $S(x)$ becomes very small as higher scores become less likely, $h(x)$ is susceptible to noise in practice. For this paper, we simply ignore $h(x)$ for values of x where $S(x) < \epsilon$, where ϵ ranges between 0.03 and 0.01 depending on how many samples were generated. This avoids the noisiest parts of the hazard function, where $S(x)$ is vanishingly small.

[2]Even if the game is trivially easy such that $S(x) = 1$ for most scores of x, a human player will eventually need to sleep or otherwise stop playing, implying that $S(x)$ will eventually approach 0. For even longer time frames, the system running the game and AI agent will eventually no longer function.

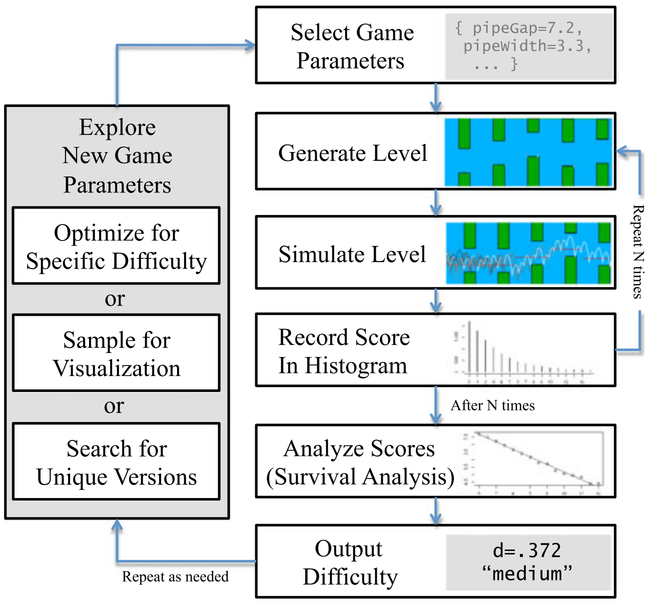

Select Game Parameters
{ pipeGap=7.2, pipeWidth=3.3, ... }
Explore New Game Parameters
Optimize for Specific Difficulty
or
Sample for Visualization
or
Search for Unique Versions
Generate Level
Simulate Level
Record Score In Histogram
Repeat N times
After N times
Analyze Scores (Survival Analysis)
Output Difficulty
d=.372 "medium"
Repeat as needed

Fig. 7. We explore game space by simulating game variants and estimating their difficulty using survival analysis.

VI. Method for Estimating Hazard Rates

Given our game space and player model, we now present our methodology for estimating the hazard rates and thereby the effective difficulty. We explore points in the game space by performing randomized simulations and examining the distribution of scores. Our goal is to describe the perceived difficulty of a game by the hazard rate function $h(x)$.

The process to calculate $h(x)$ is composed of three steps.

1) Generate a new game variant based on the given parameters.
2) Simulate the game with an AI with human-like behavior.
3) Analyze the resulting score histogram using survival analysis to estimate the difficulty of the simulated game.

Generate and Simulate steps are repeated until we have a stable score distribution for measuring in the Analyze step.

Our general approach is shown in Fig. 7. To explore a new game variant, we select parameter values from a valid range. The valid range is determined by the game rules (for example, the Space Invaders player width cannot be ≤ 0 and cannot be larger than the screen size), and we begin our search from the parameter settings that a human designer has seeded. Using these values, we generate a level and simulate playing it using an AI that models human imprecision. We repeat the Generate and Simulate steps until we have a reliable histogram of scores for the game variant. We then analyze the histogram using survival analysis to find the hazard rates (i.e., difficulty curves) of the distribution. We use the hazard rate to determine how we might update the game parameters, depending on the particular design task at hand.

A. Generate

Each simulation begins by using game parameters to generate a new game variant. This involves placing the bird and pipes in their starting positions and randomly uniformly distributing the

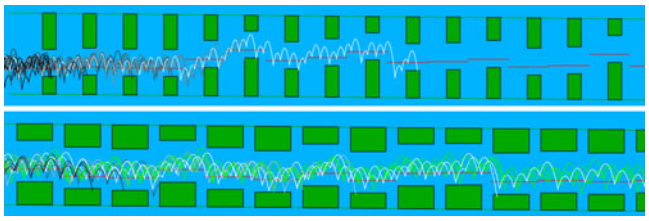


Fig. 8. Two levels created by the Generate step given different parameter sets. More simulations complete the second version, so it has an easier estimated difficulty. The red lines indicate the target location for the AI to flap.

pipe gaps in the pipe gap location range l_r. In Fig. 8, we show two different generated game variants.

Because the levels are generated using a random process, it is important to generate a new level each time the AI runs, even though the parameters do not change. Otherwise, unwanted artifacts can arise in the score distribution if the same layout of pipe gaps is used repeatedly. We also randomize the bird's starting distance from the first pipe to eliminate aliasing between the parabolic jump pattern and pipe locations.

B. Simulate

Given a level created in the Generate step, we use a simple heuristic to find a path through the pipes by flapping when the bird reaches a line in the lower half of the gap (these lines are drawn in red in Fig. 8). At each frame of the simulation, we use a forward model to predict the next time t in the future when the bird is pulled by gravity across this ideal flapping location—the ideal player would flap at exactly this time t.

By reducing or increasing the standard deviation σ of our precision model (see Section IV-A), the AI plays better or worse. We quickly check if a variant is impossible by using perfect precision $\sigma = 0$ ms on a limited number of simulations and only continue testing if a majority of the these simulations score highly.

To keep the AI from flapping faster than a human could tap, t is limited by the number of actions per second. We also tried limiting the AI lookahead to only use information that has been visible on the screen for at least τ (the time it takes for a player to react to a new event), but in our experiments, τ had no effect except in extreme situations where humans would perform poorly anyway (e.g., extremely fast birds).

For each simulation, the AI achieves a score equal to the number of pipes passed before crashing, and we record each score in a histogram. If a player crashes into pipe x or the ground immediately before it, they will achieve a score of x (the first pipe is $x = 0$). If the AI reaches a goal score $x_{\max}$, we terminate the run so we do not get stuck simulating easy games where the AI will never crash. Although *Flappy Bird* can theoretically go on forever, human players will eventually make a terminal mistake, but the AI can truly play forever unless we enforce a maximum score. Using $x_{\max}$ between 20 and 100 for *Flappy Bird* gave us a sufficient tradeoff between repeatability and total simulation time.

The Generate and Simulate steps are run repeatedly until we have enough samples to adequately analyze the histogram. Too few samples cause an unstable estimate of difficulty, while too many samples require longer simulation time. We use statistical methods to calculate a minimum safe number of samples [1] by bounding the acceptable expected standard error. For our different experiments, we used 20 000–100 000 samples, between 200 ms and 4 s on a single-core 2.8-GHz Intel Core i7 per experiment. More difficult games take less time to simulate as the number of jump calculations performed is relative to the final score.

C. Analyze

After running the Generate and Simulate steps, we examine the score distribution using survival analysis as described in Section V and in (1). We use a histogram to calculate the probability $f(x)$ that a player crashes into pipe x or the ground before it. We then calculate the survival function $S(x)$ by doing a cumulative sum on the histogram and, finally, calculate the hazard rate $h(x) = f(x)/S(x)$. We must take special care when examining $h(x_{\max})$ because it contains scores for every simulation that passes successfully through $x_{\max}$ pipes and terminates the simulation sample.

In the next section, we provide examples of various derived hazard rates created by repeating the above methods using various game parameters and player skill parameters. This provides a detailed analysis of the types of hazard rates that might be found in various games.

VII. Hazard Rate Analysis of Simulated Games

Different variants of *Flappy Bird* give rise to different types of hazards. We try different versions of the game, varying some of the game parameters or player parameters and examine the impact on the distributions of scores that result from the simulation. While in *Flappy Bird* the game parameters are constant, here we also test variants where the parameters change (e.g., each consecutive pipe gap is slightly smaller). We also vary the modeled skill level of the AI, allowing us to 1) simulate novice and expert players and 2) simulate players that learn and improve each time they play. Each of these simulations results in data that shows a different score probability distribution and hazard rate distribution. We also use our techniques to examine actual game data collected from over 106 million game plays of flappybird.io.

A. Constant Difficulty: Exponential $f(x)$; Constant Hazard

We start by examining variants that do not modify their parameters as the game progresses and, therefore, can have a constant difficulty if one ignores learning effects. We show some settings of *Flappy Bird* game parameters, which lead to a constant hazard $h(x) = \lambda$. Using the score results from our simulations, the data shown in Fig. 9 show empirical evidence of an exponential distribution and constant hazard rate when using a constant pipe gap, constant skill level, and ignoring learning effects.

In Fig. 9(a), we show the probabilities for four versions of the game, each with a different pipe gap (decreasing in size from the black line to the red line). Because we cannot describe every possible variation in the scope of this paper, we focus here on the pipe gap variable to illustrate the effectiveness of the method. Four representative values were chosen to demonstrate the effect and make the figure easy to understand. Exponential distributions become linear in log plots, so we can tell from Fig. 9(b)

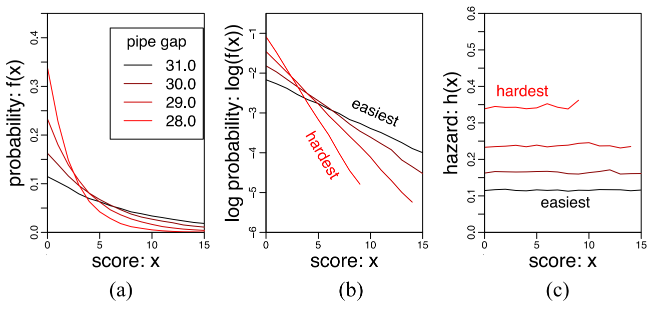


Fig. 9. Data collected from simulated games with nonvarying game and player parameters (i.e., no learning); pipe gap is changed for each of the curves. (a) Harder games (smaller pipe gap, in red) have a higher probability of getting a lower score. (b) Log probabilities are linear, indicating exponential distributions. (c) Constant hazard rates indicate constant difficulty.

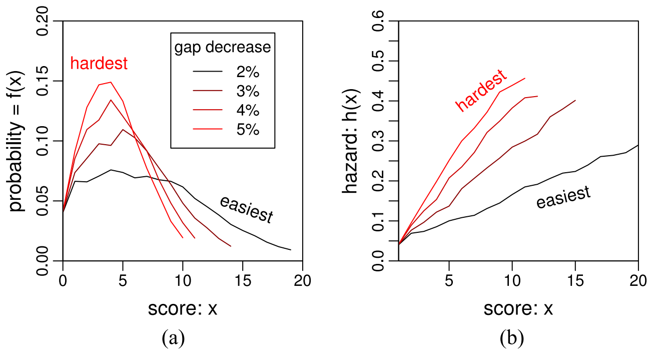


Fig. 10. Data collected from simulated games with pipe gap decreasing by the given percentage, constant skill, and no learning. (a) Smaller decreases are easier games and more likely to exhibit higher scores. (b) Steeper hazard lines indicate more rapid difficulty increases.

that the data indeed come from the exponential distribution. The derived hazard rates, shown in Fig. 9(c), also increase, indicating as expected narrower gaps lead to a more difficult game. There is some noise in the hazard function as we simulate using a stochastic process. This noise can be reduced with more samples. The harder games have shorter lines, because it is unlikely a player will achieve higher scores in them.

The constant hazard function $h(x) = \lambda$ gives rise to exponential probability distribution $f(x) = \lambda e^{-\lambda x}$ [5], [11]. A constant hazard means that the player is equally likely to die at every moment in the game (i.e., constant difficulty). A more difficult game has a higher λ. The mean $1/\lambda$ indicates that more difficult games will have a lower average score.

Due to complicated interactions between the parameters (e.g., bird speeds that hop through multiple pipes on a single parabolic arc), not all constant-parameter variants of *Flappy Bird* create constant hazard rates. For example, at high speeds, the hazard for the first pipe can be significantly easier, with a slight decreasing trend in the hazard as scores increase. We discuss nonconstant hazards in the next subsections.

B. Increasing Difficulty: Rayleigh Distribution; Linear Hazard

Most of the games do not have constant game parameters, but instead increase in difficulty as the player makes progress into the game. Fig. 10 shows four simulated games where we start with a common pipe gap and multiply it by $(1 - \Delta p_g)$ after each consecutive pipe. The black line indicates the smallest decrease

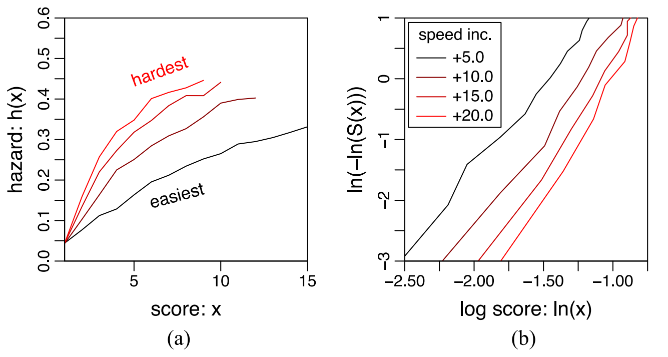


Fig. 11. Data collected from simulated games with bird speed increasing by the given *additional amount* after each point scored, assuming constant skill and no learning. (a) Hazard is more curved than in Fig. 10(b). (b) Lines in the Weibull plot indicates that the data matches the three-parameter Weibull distribution.

in pipe gap and the red line is the largest decrease. We include a lower bound on the pipe gap so that it does not get so small that the bird cannot pass through it.

Fig. 10(a) shows the resulting empirical probability distribution for each variant. As expected, games with the slower pipe gap decreases show a higher probability of a higher scores. The bump in the probability plots indicates that the player is more likely to achieve a score of around 3–5 than a score of 0, which means the player will likely experience some success at the start (unlike the constant hazard example which is equally difficult everywhere).

In Fig. 10(b), we show the calculated hazard rate for each variant. The hazards are approximately linear, although there is a slight curve visible in the harder variants. Each line comes to the same point because each variant starts out with the same pipe gap value. Decreasing the pipe gap at a larger rate means the game gets more difficult more quickly, indicated by a steeper slope in the hazard plot.

We can theoretically model this with a linear hazard function $h(x) = a + bx$, where a defines the game's base difficulty and $b > 0$ defines the rate at which difficulty increases. The theoretical probability distribution is $f(x) = (a + bx)e^{-ax-b/2x^2}$ [5], [11]. This is a two-parameter Rayleigh distribution with location parameter (due to the $a > 0$ constant term).

C. Weibull Distribution; Power-Law Hazard

Instead of multiplying pipe gap by a scale factor as in the previous section, here, we increase bird speed by adding a fixed constant. We can see a curving hazard function in Fig. 11(a). In Fig. 11(b), we show a Weibull plot of the data, which shows the relationship between $\ln x$ and $\ln(-\ln S(x))$. In such a plot, curves will be linear if the data follow a Weibull distribution, which arises in survival analysis when the hazard rate follows a power law [11], [59]. Since we observe lines in the Weibull plot, we can be confident our data fit the Weibull distribution.

The Weibull distribution hazard is $h(x) = c/b$ $(x - a/b)^{c-1}$ leading to a probability $f(x) = c/b$ $(x - a/b)^{c-1}\exp[-(x - a/b)^c]$ [11]. a is a location parameter, b is a scale parameter, and c is the shape parameter of the Weibull distribution. When $c > 1$, we have an increasing

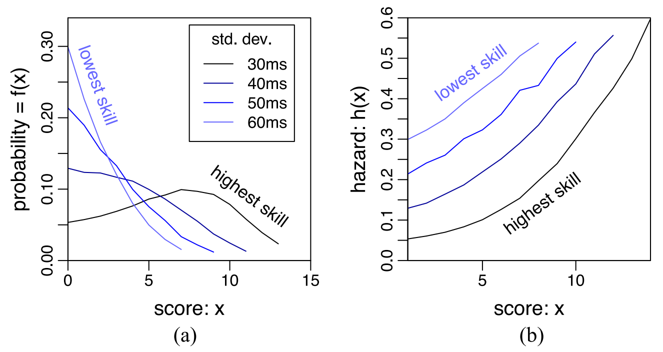


Fig. 12. Data collected from the same game with AIs of different skill. Smaller standard deviation models higher skill. (a) Skill influences the shape of the empirical probability distribution. (b) Skill affects the y-intercept and linearity for the hazard, causing the probability distribution to shift and change shape.

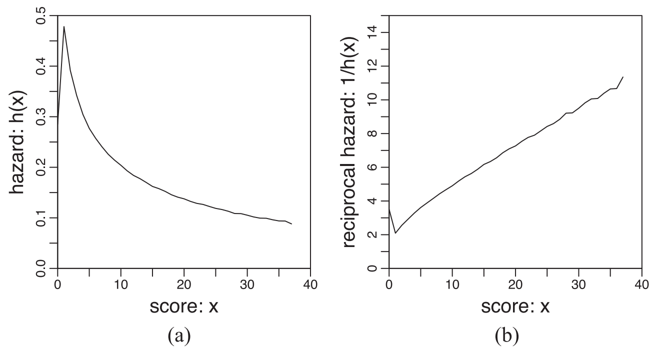


Fig. 13. Actual, not simulated, player data from over 106 million plays of flappybird.io matches a Generalized Pareto hazard. The hazard spike occurs because, in *Flappy Bird*, the first pipe is easier to pass than the rest. (a) Hazard rate is clearly not constant. (b) Reciprocal hazard is approximately linear.

hazard rate; $c < 1$ gives a decreasing hazard rate; $c = 1$ reduces to a constant hazard rate.[3]

D. *Different Skill Levels With Linear Hazards*

We now explore how a single game variant with decreasing pipe gap can be experienced by players of different skill levels. We evaluate this for linear hazards as the effect of skill is more interesting to analyze for this case than for constant difficulty games, where the constant hazard simply scales. In Fig. 12, we simulate the same variant, but use a different player skill for each line. The black line is the highest skilled player and the lightest blue line is the lowest skilled player. Recall that we increase simulated player skill by decreasing the standard deviation of the time adjustment.

The resulting empirical probability distributions from the experiment, shown in Fig. 12(a), are especially interesting. We can see that players of low skill may experience a very different game from high-skilled players. The low-skilled players find that a score of 0 is most likely; they do not experience early successes to encourage them. The high-skilled players, however, have some early notion that they are doing well as their probabilities increase at the beginning.

In Fig. 12(b), we see that the hazard rates for this variant are no longer linear for the higher skilled players, as in Fig. 10. This occurs because the hazard rate cannot be negative, so the curve trends toward horizontal at lower scores as visible in the data from higher skilled players [5].

It is important to reiterate that the shape of the score probability distribution and hazard curves are dependent on the player's skill—the low-skilled player and high-skilled player do not experience the game in the same way. This quantitatively impacts the designer's ability to make a game that can please all players without making some sacrifices on game balance.

E. *Evidence of Varying Skill in Actual Game Distributions*

We now apply survival analysis to examine the distribution of scores from real-life (i.e., not simulated) game data. We examine data obtained from flappybird.io [14], a popular web-based version of the original *Flappy Bird*. As explained in Section VII-A, *Flappy Bird* has a constant difficulty, so without learning, effects would exhibit a constant hazard rate and an exponential probability distribution.

Fig. 13 shows the calculated hazard rates for March 2014, when flappybird.io first launched, covering over 106 million plays. The spikes at the left edge occur since the first pipe is easier to score due to the original game scoring at the pipe center and additional setup time for the first pipe. In our system, we eliminated this spike by randomizing the starting location and shifting the scoring position to the end of each pipe.

By deriving the hazard from the data, we see in Fig. 13(a) that the hazard rate decreases rapidly, indicating that learning and/or past experience may be a factor at making the game less difficult for higher scoring players. The dataset does not provide unique player ID to determine how each particular player is improving over repeated plays; however, our analysis of Canabalt and Drop7 scores with player ID does show power-law learning is present [15]. Plotting score versus reciprocal hazard $1/h(x)$ in Fig. 13(b) shows a linear relationship, indicative of the generalized Pareto distribution [60]. This distribution does not look like the others we have examined, so we will now discuss an explanation for the empirical reciprocal hazard.

F. *Varying Skill Model: Generalized Pareto; Hyperbolic Hazard*

People generally improve over time as they repeatedly perform the same task [61] or game [15]. Learning can be modeled with a power law function $T = A + B(n + E)^{-R}$, where the time T to perform a task decreases as the number of repetitions n increases, A defines the best possible time to achieve the task, B defines the performance on the first trial, $R > 0$ is the learning rate, and E represents prior experience [61]. Power laws model improvement that goes quickly at the beginning, but then slows down as the player learns the easiest ways to improve, but then takes more time to develop the ability to improve at higher level skills [62].

Instead of modeling a decrease in time to complete a task, we model a decrease in player error, which has similarly been shown to follow power laws [61]. In our system, we model this improvement by decreasing the standard deviation of the time adjustment after each play. To generate the data, we simulate 5000 AI players, each repeating the game ten times. After each

[3]The Weibull distribution commonly occurs when a manufactured part, such as a light bulb, becomes more likely to fail the longer it is used ($c > 1$).

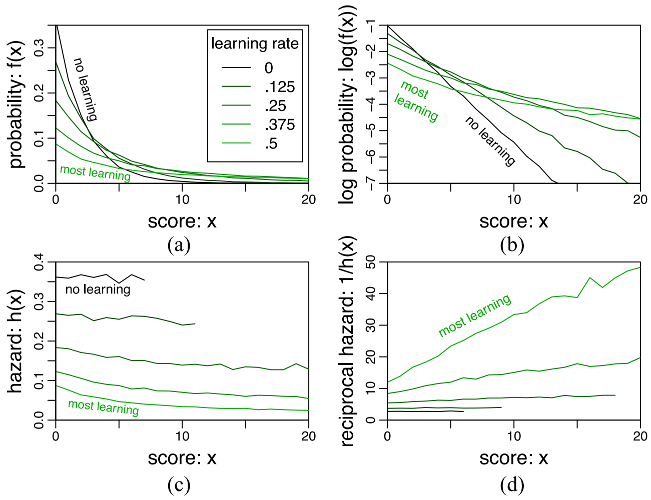


Fig. 14. Data collected from simulated games with constant game parameters when modeling learning effects. (a) Empirical pdf follows the generalized Pareto distribution (GPD). (b) Log plots show learning increasingly diverges from exponential. (c) Hazard rates shows a decreasing trend. (d) Reciprocal hazards $1/h(x)$ derived from the data are linear, which fit the GPD.

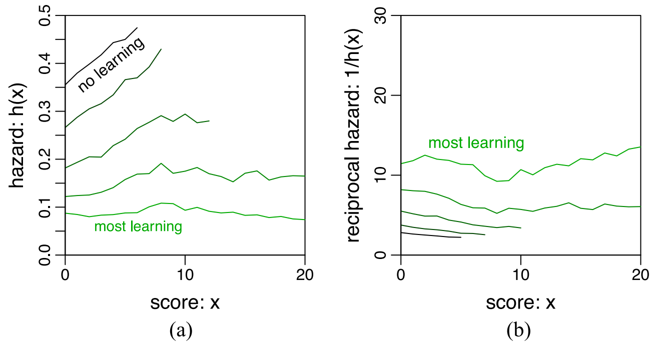


Fig. 15. Hazard curves tuned to appear more constant under conditions of player improvement and increasing parameters. (a) For high learning, hazards are relatively flat compared to Fig. 14(c). (b) Reciprocal hazard is also flattened.

of the ten times, the standard deviation is reduced to follow the power law learning equation. We vary the learning rate R for each test to explore the learning effect.

We can see the empirical results of modeling learning in Fig. 14. The black curve has no learning, and the lightest green curve has the fastest learning. From the empirical distributions in Fig. 14(a), we cannot tell exactly what the distribution may be, but the log plot in Fig. 14(b) shows that faster learning rates cause a larger departure from exponential (as exponential curves are lines in a log plot, and we can see an upward bend in the lines with most learning).

By deriving the hazard from our data, as shown in Fig. 14(c), we observe a decreasing hazard rate with increasing score x. The decreasing behavior arises because with repeated plays, the player is learning and improving, which makes higher scores easier to obtain. The hazard decreases faster with a higher learning rate and reduces to a constant hazard for a zero learning rate. Additionally, with higher learning rates, we will see more "expert" plays in the data; this helps model the behavior that good players may play more games than poor players due to satisfaction and positive feedback. We can interpret the decreasing hazards as the game getting easier when you have players increasing their skill, leading to a higher probability of higher scores.

We can see in Fig. 14(d) further evidence that these curves are hyperbolic (i.e., reciprocal linear) hazard rates, when free parameters for inverting the hazard are set appropriately. We theoretically model this using hyperbolic hazards $h(x) = a + b/x + c$, which gives a probability distribution $f(x) = (a + b/x + c)\, e^{-ax}(1 + x/c)^{-b}$, where a is related to initial difficulty at the start of the game, b determines the learning rate, and c allows us to have scores $x = 0$ and helps model previous experience. These equations match the generalized Pareto distribution [60], [63].

It is important to note that while we are calling this effect learning, it is also possible that there is additionally a wide variety of skill in the player population. Having α players improving over β games is effectively the same as having $\alpha\beta$ players of various skill each playing one game. To determine the difference and the effect of each component would require a controlled user study beyond what is available in the flappybird.io user analytics data.

G. Tuning Difficulty/Hazard Curves

To counteract the decreasing hazard that arises in learning, we can use a score-dependent game parameter to flatten out the hazard rate curves. In Fig. 15, we see the effect of decreasing the pipe gap, up to a lower bound, but now under conditions of varying skill. This makes the game more difficult as scores increase, offsetting the learning effect that reduces difficulty. The chosen parameter shows relatively flat hazards and relatively flat reciprocal hazards for high learning. By allowing the parameters to vary with a more complicated function of score, the hazard curve can be tuned as flat as the designer wishes. However, as in Section VII-D, the low-skilled player finds the game increasingly difficult; we cannot effectively use one set of parameters for all learning levels to get effectively constant hazards (i.e., constant perceived difficulty).

This method can be used to adjust the difficulty of a game so that it appears to have constant difficulty, even though players have varying amounts of skill. The technique here is significantly different than dynamic difficulty adjustment (DDA) [26] because we are analyzing the game statically. Our method does not rely on runtime feedback from the player, but instead uses simulated players to determine how a game should increase in difficulty. One significant advantage to our approach is that one can meaningfully compare scores between players on a leaderboard, because all players get the same game parameters, and therefore, score can compare relative skill. In DDA frameworks, comparing scores is difficult because the players are not playing the same game variant due to the parameters fluctuating with individual player performance.

VIII. Exploring Game Space

Given the analysis techniques described in the previous section, we would now like to efficiently explore the high-dimensional space of possible games—an exhaustive search over all parameters will not work. Through intelligent sampling

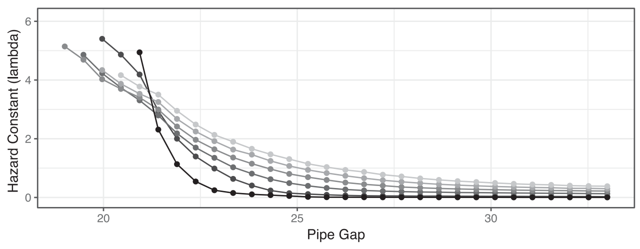


Fig. 16. Increasing pipe gap p_g decreases difficulty. Each line shows a different setting for player precision σ. Lighter lines plot lower precision, modeling poorer performance. Missing data points indicate where no players scored.

and visualization, we can gain insight about how each game parameter affects the difficulty curves (see Section VIII-A), adjust parameters to target a specific difficulty (see Section VIII-B), generate new versions (see Section VIII-C), or find k unique variants as far apart from each other in game space as possible (see Section VIII-D).

A. Visualization Methods

We use the following techniques to sample and visualize difficulty changes as we explore different points in game space. Visualization is a powerful data analysis technique that can provide insights; we are trying to understand the structure of game space, which is not revealed through black box exploration alone. We start the search from the game parameters that define the original *Flappy Bird*. We estimated the original parameters by examining video capture of the game, carefully measuring frame to frame differences.

1) Single-Dimensional Sampling: Beginning with the original *Flappy Bird*, we keep each parameter fixed and vary one at a time, exploring along each dimension, and sampling at fixed intervals. Fig. 16 shows a plot of pipe gap p_g versus constant hazard λ. Each line uses a different value for player precision σ. Lighter lines in the figure have a higher standard deviation, so the AI makes more errors, and the game is more difficult for the modeled player. As one expects, the model predicts that players with less precision will find the same game more difficult to play, and narrower gaps are harder for everyone.

2) Two-Dimensional Sampling: Varying in two dimensions shows dependent parameters and can help designers find interesting relationships between dimensions of game space. We visualize these results using dot plots, displaying varying difficulty by the radius and color saturation of each point.

For example, we see in Fig. 17 that jump velocity j and gravity g are dependent. When gravity is too high or low relative to jump velocity, the bird crashes into the floor or ceiling. In the middle, gravity and jump velocity are balanced, and we see as they increase together, the game gets more difficult—faster reaction times are required as the bird is moving rapidly up and down. Lower values of gravity and jump velocity give the player more time to react and are easier to play. Holes and islands are due to stochastic simulation and can be reduced with a larger number of simulations n_s.

Fig. 18 shows a hyperbolic-like relationship between bird velocity v_x versus pipe gap location range l_r. As the bird moves faster, there is less time to aim for the next gap. As we increase the location range, the bird must on average travel further to clear the pipes, so requiring more time to adjust.

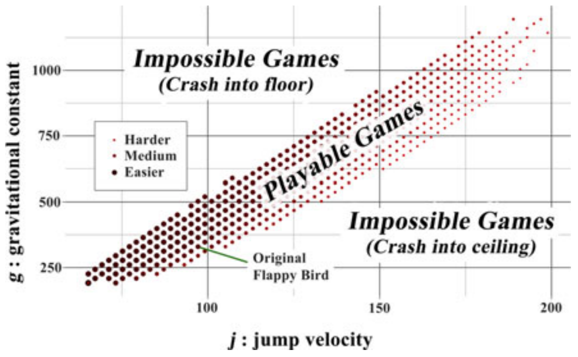


Fig. 17. Sampling game space in two dimensions, jump velocity j versus gravity g, shows a narrow band of playable games.

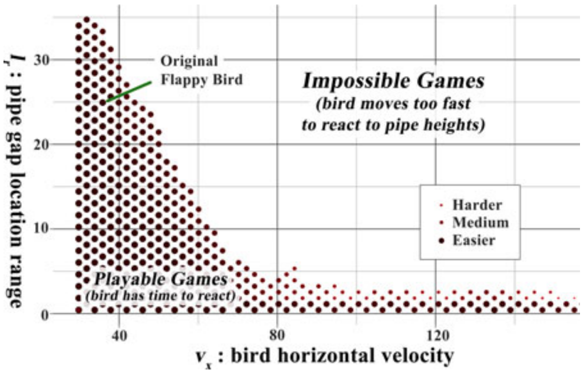


Fig. 18. Sampling in two dimensions, l_r pipe gap location range versus v_x horizontal bird velocity. High speeds require a lower pipe range, so the player has enough time to react to the varying gap locations.

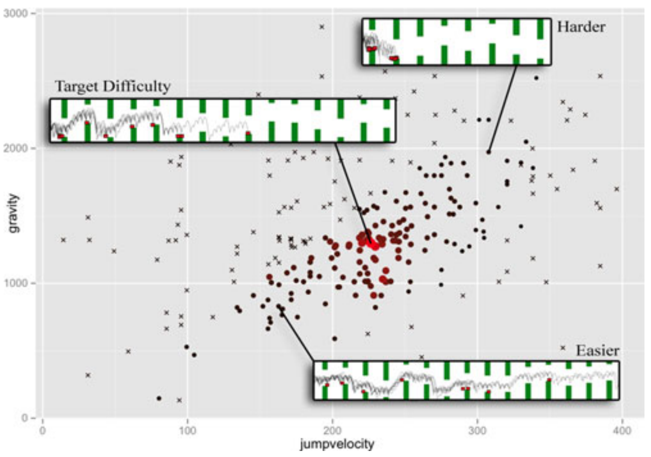


Fig. 19. Differential evolution optimization helps us search for a target difficulty. Each point indicates a variant tested to find the target; X indicates impossible games; Dot size and color indicates closeness to the target.

B. Exploration via Optimization

Global optimization algorithms are designed to efficiently search parameter space to find the optima of a function. We use optimization to find the parameters that will give a specific hazard $h(x) = \lambda_{\text{target}}$ by searching the parameter space to minimize $(\lambda - \lambda_{\text{target}})^2$. Because we evaluate λ stochastically, we are optimizing over a noisy function and cannot use strategies that rely on differentiable functions and gradients [64]. Differential evolution [65] is a genetic search technique designed for stochastic functions, and the *DEoptim* optimizer [66] quickly

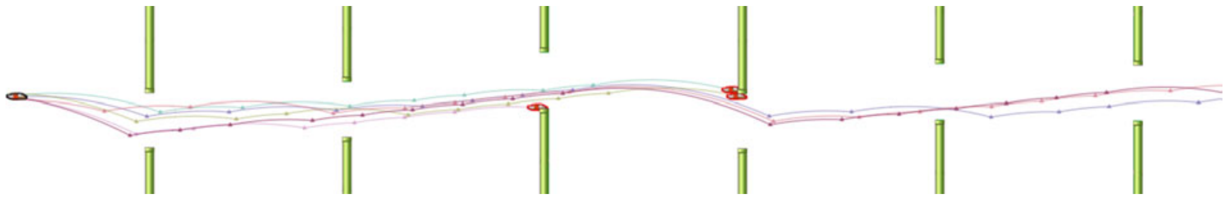


Fig. 20. “Frisbee Bird” was unexpectedly created by our system, an example of computational creativity. We show six simulated paths with jump points. The player is wide and thin, with weak jump velocity and gravity that exhibits floating game feel. We manually adjusted the parameters shown here for visualization.

explores game space find our target with 0.1% accuracy in approximately 10–20 iterations using the default options for the library. Fig. 19 shows that 280 points were explored to find a hazard within 0.1% of $\lambda = 0.155$.

C. Computational Creativity

We interpret exploratory computational creativity [67], [68] as the process of finding playable variants that are as different as possible from the existing versions that have already been explored. By allowing the system to vary every design parameter, unique combinations can quickly be explored by the AI and optimizer. Unplayable levels are invalidated, and the system only returns the games that can actually be played by humans. The designer then can examine the unique playable games to find inspiration and new ideas. While a game is composed of many parts, including visuals, audio, narrative, systems of rules, level architecture, game play, and interactions between those facets [69], here we only focus on the creative aspect of designing novel game play.

Using the optimization algorithm described in Section VIII-B, we generated interesting and surprising variants [1], such as “Frisbee Bird,” which has a very different game feel from the original *Flappy Bird*. This variant, shown in Fig. 20, was created by allowing the optimizer to vary every static design parameter, including speed, player width, player height, jump velocity, gravity, pipe distance, pipe width, and pipe randomness. The optimizer returned a game with a wide flat bird, which moves horizontally very fast but slow vertically and requires short bursts of rapid presses followed by a long pause while the bird floats in the air. This unexpected variant, discovered by our system while using the optimization algorithm, still relies on the mechanics and rules of the original but is a significantly different play experience.

D. Finding k Unique Games

To help designers find new design possibilities, we look for k unique variants in game space that are as far apart from each other as possible [55]. The approach consists of generating a bounded search space and running an optimization algorithm to find the k games furthest apart in game space.

The search domain can be arbitrarily large, but as we are focused on games that are playable for humans, we begin by creating reasonable bounds on the parameter space to explore. Although unbounded creativity could possibly lead to entirely new games, the extra effort required to search the larger space means many computational resources is wasted. We balance this by setting boundary conditions on the game space. Many of our parameters have natural bounds, as described in Section III.

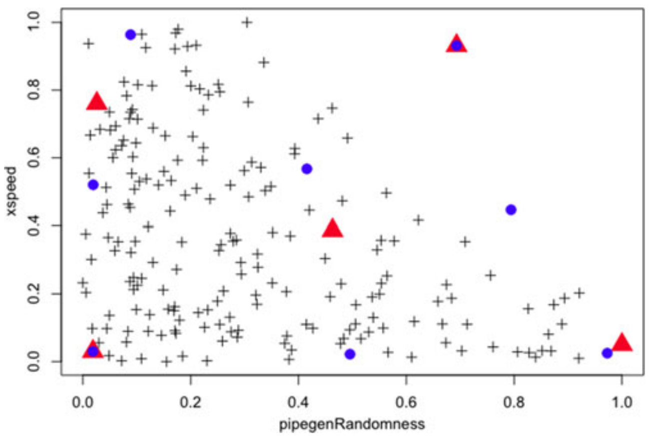


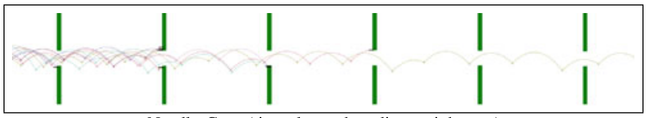
Fig. 21. Using genetic optimization, we find games that are far apart from each other in the 2-D game space. + indicates each generated playable game, blue dots indicate the set for $k = 8$, and red triangles are for $k = 5$.

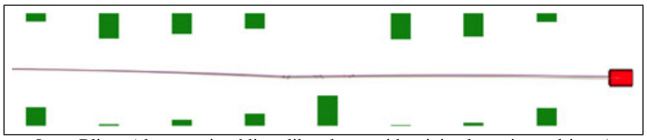
We begin by seeding reasonable bounds and then generating samples using a stratified sampling scheme that covers the search space without clumping that may occur with uniform random sampling. We can simulate each point and calculate its difficulty and average play time. We mark as playable all the games that have a difficulty, where between 1% and 60% of players are expected to crash on the first pipe and the average human play time is between 1 and 30 s. We eliminate games with hazard close to zero, as they are too easy and we expect them to be uninteresting and boring to players.

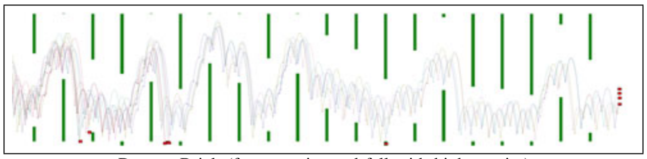
Next, we calculate the hypervolume bounding box that tightly contains the playable games. For each dimension, we compare the effective upper and lower bounds that lead to playable games to the original search range. If the effective bounds are near the guessed bounds, we could, if desired, expand the search area and try again because it is possible that we have missed sampling some playable games.

Now that we have a point cloud of thousands of playable games variants, we would like to reduce this to a small number of unique games. Clustering algorithms tend to find interior representative points [55]. Instead, we want to find games further apart, showing us what is possible on the frontiers of game space—pushing the parameters to their extremes.

To find games as far apart from each other as possible, we find a subset of k games that maximizes the minimum Euclidean distance (in normalized parameter space) between any pair of points in the set. We use a genetic search to evolve the k elements to be included in the set. By running for more generations, the examples approach optimally unique. Fig. 21 shows how this technique finds unique game space points ($k = 5$ for the red triangles and $k = 8$ for the blue circles).


Needle Gnat (tiny player threading a tight gap)


Lazy Blimp (slow moving blimp-like player with minimal gravity and jump)


Droppy Brick (frequent rise and fall with high gravity)

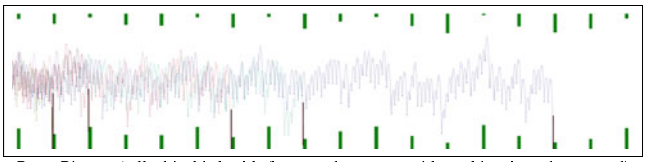

Pogo Pigeon (tall, thin bird with frequent hops to avoid crashing into the ground)

Fig. 22. Game variants discovered by finding $k = 4$ most of the unique games.

We present several novel *Flappy Bird* variants discovered by our methods in Fig. 22. Varying all nine static game parameters, we created four unique games. These appear to us to approximate the creativity a human designer might have used if tasked with designing four unique variants.

IX. Discussion and Conclusion

One long-term goal of this area of research is to eventually provide game designers with tools and methods that can help them design better games and to expand their creativity via computer-aided game design tools. We envision designers and computers working together to craft a better player experience.

In this paper, we argue that using player modeling and survival analysis helps us better understand the relationship between game parameters and player experience of difficulty in minimal action games. The algorithms presented here can be used by game designers to tune games and explore parameter variations. We show the utility of using simple AIs, which make human-like errors for automated playtesting to explore game space. The hazard rate, which is analogous to a difficulty curve, allows the designer to estimate how players will experience difficulty as the game progresses. This allows designers to explore the game space of playable games to find unique variants, to better understand the effect of game parameters, to adjust parameters to tune difficulty as a player progresses, and to evaluate how players with different levels of skill will experience a game. Future experiments to validate that these techniques actually help designers is a necessary next step.

The framework presented here can be used for many types of minimal action games, not just *Flappy Bird*. However, the player model presented primarily models dexterity, player accuracy, and timing errors: therefore, it is most suited for action games where difficulty is determined by motor skill, not by path planning or strategic decisions. There are various kinds of difficulty, such as strategic difficulty, requiring deep look-ahead and search, or representational difficulty, which makes it challenging to visually determine how pieces fit together to solve a puzzle. Various types of difficulty require new models to accurately simulate and measure their effects.

As the difficulty and playability of the game is determined by the AI, it will only find games that it can play effectively. However, there may be games interesting to human players which are passed over because of the AI's assumptions and deficiencies. The method of search can influence which points in game space are discovered [68]. Additionally, the methods to find unique variants are focused on the distance in the game parameter space, which is one aspect that makes games interesting to players and designers, but "interesting" has a much broader meaning and is significantly more complex to quantify.

Our survival analysis only looks at a single dimension, d, which describes the score distribution of our simulation. Future work should explore additional output variables such as time played, optimizing for multiple dimensions at once. In theory, we expect a high-dimensional game space to have dependencies, which can be squeezed into a lower dimensional space using model reduction techniques, finding the intrinsic dimensionality of a game space [17], [18]. This would reduce the number of knobs a designer needs to adjust. In addition, lower dimensional spaces are likely faster and easier to search.

All *Flappy Bird* variants can be seen as a point in a much larger design space. This research and the developed methods are a step toward exploring this space of computationally assisted designs, where we can modify rules and parameters, explore games of strategy and dexterity, and combine difficulty with human player preference and aesthetic judgment.

Acknowledgment

The authors would like to thank the reviewers for their suggestions and comments. They would also like to thank K. C. Fox for extracting the original *Flappy Bird* parameters, M. McDonnell for providing score distribution data, T. Hiebert for helping with demonstration software, and D. Nguyen for inspiring this research with *Flappy Bird* and for permission to use his images.

References

[1] A. Isaksen, D. Gopstein, and A. Nealen, "Exploring game space using survival analysis," in *Proc. Int. Conf. Found. Digit. Games*, 2015.
[2] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 172–186, Sep. 2011.
[3] Taito Corporation, "Space Invaders," Arcade Game, 1978.
[4] E. Adams, *Fundamentals of Game Design*, 3rd ed. London, U.K.: Pearson, 2014.
[5] A. Isaksen and A. Nealen, "Comparing player skill, game variants, and learning rates using survival analysis," in *Proc. 1st Player Model. Workshop AIIDE*, 2015.
[6] A. Nealen, A. Saltsman, and E. Boxerman, "Towards minimalist game design," in *Proc. 6th Int. Conf. Found. Digit. Games*, 2011, pp. 38–45.
[7] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," in *Proc. Int. Conf. Found. Digit. Games*, 2014.
[8] D. Nguyen, "Flappy Bird," Apple App Store, 2013.
[9] A. Isaksen, "Score distribution analysis, artificial intelligence, and player modeling for quantitative game design," Ph.D. dissertation, Tandon School Eng., New York Univ., Brooklyn, NY, USA, 2017.
[10] E. T. Lee and J. W. Wang, *Statistical Methods for Survival Data Analysis*. New York, NY, USA: Wiley, 2013.
[11] H. Rinne, *The Hazard Rate: Theory and inference (with Supplementary MATLAB-Programs)*. Giessen, Germany: Justus Liebig Univ. Giessen,

2014. [Online]. Available: http://geb.uni-giessen.de/geb/volltexte/2014/10793

[12] J. Schell, *The Art of Game Design: A Book of Lenses*. Boca Raton, FL, USA: CRC Press, 2014.

[13] R. Koster, *Theory of Fun for Game Design*. Sebastopol, CA, USA: O'Reilly Media, 2013.

[14] M. McDonnell, "flappybird.io," 2014. [Online]. Available: http://flappybird.io

[15] A. Isaksen and A. Nealen, "A statistical analysis of player improvement and single-player high scores," in *Proc. 1st Int. Joint Conf. DiGRA/FDG*, 2016.

[16] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proc. 11th Int. Conf. Artif. Life*, 2008, pp. 329–336.

[17] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[18] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Wiley, 2002.

[19] A. Saltsman, "Canabalt," Apple App Store, 2009.

[20] A. M. Smith, C. Lewis, K. Hullet, and A. Sullivan, "An inclusive view of player modeling," in *Proc. 6th Int. Conf. Found. Digit. Games*, 2011, pp. 301–303.

[21] R. A. Magill and D. Anderson, *Motor Learning and Control: Concepts and Applications*, vol. 11. New York, NY, USA: McGraw-Hill, 2007.

[22] N. Fujii, Y. Sato, H. Wakama, K. Kazai, and H. Katayose, "Evaluating human-like behaviors of video-game agents autonomously acquired with biological constraints," in *Advances in Computer Entertainment*. New York, NY, USA: Springer-Verlag, 2013, pp. 61–76.

[23] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Trans. Affective Comput.*, vol. 2, no. 3, pp. 147–161, Jul.–Sep. 2011.

[24] A. Zook, E. Fruchter, and M. O. Riedl, "Automatic playtesting for game parameter tuning via active learning," in *Proc. Int. Conf. Found. Digit. Games*, 2014.

[25] A. Drachen, A. Canossa, and G. N. Yannakakis, "Player modeling using self-organization in Tomb Raider: Underworld," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 1–8.

[26] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games," in *Proc. Challenges Game Artif. Intell. AAAI Workshop*, 2004, pp. 91–96.

[27] O. Missura and T. Gärtner, "Player modeling for intelligent difficulty adjustment," in *Discovery Science*. New York, NY, USA: Springer-Verlag, 2009, pp. 197–211.

[28] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 1, pp. 54–67, Mar. 2010.

[29] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," in *Proc. IEEE Symp. Comput. Intell. Games*, 2007, pp. 252–259.

[30] C. Browne and F. Maire, "Evolutionary game design," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 1, pp. 1–16, Mar. 2010.

[31] A. B. Jaffe, "Understanding game balance with quantitative methods," Ph.D. dissertation, Univ. Washington, Seattle, WA, USA, 2013.

[32] M. Campbell, A. J. Hoane, and F.-H. Hsu, "Deep blue," *Artif. Intell.*, vol. 134, no. 1, pp. 57–83, 2002.

[33] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[34] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[35] K. Compton and M. Mateas, "Procedural level design for platform games," in *Proc. 2nd AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, 2006, pp. 109–111.

[36] J. Fraser, M. Katchabaw, and R. E. Mercer, "A methodological approach to identifying and quantifying video game difficulty factors," *Entertainment Comput.*, vol. 5, pp. 441–449, 2014.

[37] M.-V. Aponte, G. Levieux, and S. Natkin, "Measuring the level of difficulty in single player video games," *Entertainment Comput.*, vol. 2, no. 4, pp. 205–213, 2011.

[38] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Mach. Learn.*, vol. 63, no. 3, pp. 217–248, 2006.

[39] N. Shaker, G. N. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proc. 6th AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, 2010, pp. 63–68.

[40] G. N. Yannakakis and J. Hallam, "Real-time game adaptation for optimizing player satisfaction," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 2, pp. 121–133, Jun. 2009.

[41] H. Desurvire, M. Caplan, and J. A. Toth, "Using heuristics to evaluate the playability of games," in *Proc. Human Factors Comput. Syst.*, 2004, pp. 1509–1512.

[42] A. Drachen and A. Canossa, "Towards gameplay analysis via gameplay metrics," in *Proc. 13th Int. MindTrek Conf.: Everyday Life Ubiquitous Era.*, 2009, pp. 202–209.

[43] M. S. El-Nasr, A. Drachen, and A. Canossa, *Game Analytics: Maximizing the Value of Player Data*. New York, NY, USA: Springer-Verlag, 2013.

[44] M. J. Nelson, "Game metrics without players: Strategies for understanding game artifacts," in *Proc. AIIDE Workshop Artif. Intell. Game Des. Process*, 2011, pp. 14–18.

[45] E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Titusville, FL, USA: Graphics Press, 2001.

[46] B. Victor, "Learnable programming," *Viitattu*, vol. 2, p. 2012, 2012.

[47] F. Mourato, F. Birra, and M. P. dos Santos, "Difficulty in action based challenges: Success prediction, players' strategies and profiling," in *Proc. 11th Conf. Adv. Comput. Entertainment Technol.*, 2014, Art. no. 9.

[48] W.-C. Feng, F. Chang, W.-C. Feng, and J. Walpole, "A traffic characterization of popular on-line games," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 488–500, Jun. 2005.

[49] C. Bauckhage, K. Kersting, R. Sifa, C. Thurau, A. Drachen, and A. Canossa, "How players lose interest in playing a game: An empirical study based on distributions of total playing times," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 139–146.

[50] M. Cook and S. Colton, "Multi-faceted evolution of simple arcade games," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 289–296.

[51] V. Hom and J. Marks, "Automatic design of balanced board games," in *Proc. 3rd AAAI Conf. Artif. Intell. Interactive Digit. Entertain.*, 2007, pp. 25–30.

[52] A. M. Smith and M. Mateas, "Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games," in *Proc. IEEE Conf. Comput. Intell. Games*, 2010, pp. 273–280.

[53] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proc. IEEE Conf. Comput. Intell. Games*, 2008, pp. 111–118.

[54] S. Swink, *Game Feel*. San Mateo, CA, USA: Morgan Kaufmann, 2009.

[55] A. Isaksen, D. Gopstein, J. Togelius, and A. Nealen, "Discovering unique game variants," in *Proc. Comput. Creativity Games Workshop*, 2015.

[56] W. A. Wickelgren, "Speed-accuracy tradeoff and information processing dynamics," *Acta Psychologica*, vol. 41, no. 1, pp. 67–85, 1977.

[57] R. Hyman, "Stimulus information as a determinant of reaction time," *J. Exp. Psychol.*, vol. 45, no. 3, pp. 188–196, 1953.

[58] W. H. Teichner, "Recent studies of simple reaction time," *Psychol. Bull.*, vol. 51, no. 2, pp. 128–149, 1954.

[59] D. G. Kleinbaum and M. Klein, *Survival Analysis*. New York, NY, USA: Springer-Verlag, 1996.

[60] L. M. Leemis and J. T. McQueston, "Univariate distribution relationships," *Amer. Statist.*, vol. 62, no. 1, pp. 45–53, 2008.

[61] N. E. Lane, *Skill Acquisition Rates and Patterns*. New York, NY, USA: Springer-Verlag, 1987.

[62] F. Gobet, J. Retschitzki, and A. de Voogt, *Moves in Mind: The Psychology of Board Games*. Hove, U.K.: Psychology Press, 2004.

[63] L. M. Leemis, D. J. Luckett, A. G. Powell, and P. E. Vermeer, "Univariate probability relationships," *J. Statist. Educ.*, vol. 20, no. 3, pp. 1–11, 2012.

[64] A. Redd, "Optimising a noisy objective function," 2013. [Online]. Available: http://www.r-bloggers.com/optimising-a-noisy-objective-function/

[65] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization* (ser. Natural Computing). New York, NY, USA: Springer-Verlag, Jan. 2006.

[66] K. Mullen, D. Ardia, D. Gil, D. Windover, and J. Cline, "DEoptim: An R package for global optimization by differential evolution," *J. Statist. Softw.*, vol. 40, no. 6, pp. 1–26, 2011. [Online]. Available: http://www.jstatsoft.org/v40/i06/

[67] M. A. Boden, *The Creative Mind: Myths and Mechanisms*. Hove, U.K.: Psychology Press, 2004.

[68] G. A. Wiggins, "A preliminary framework for description, analysis and comparison of creative systems," *Knowl.-Based Syst.*, vol. 19, no. 7, pp. 449–458, 2006.

[69] A. Liapis, G. N. Yannakakis, and J. Togelius, "Computational game creativity," in *Proc. 5th Int. Conf. Comput. Creativity*, 2014, pp. 285–292.

Authors' photographs and biographies not available at the time of publication.