ELIMINATION
from Design to Analysis

Ahmed Khalifa, Dan Gopstein, Julian Togelius
Game Innovation Lab, New York University, NY, USA
ahmed@akhalifa.com, dgopstein@nyu.edu, julian@togelius.com

***Abstract*—Elimination is a word puzzle game for browsers and mobile devices, where all levels are generated by a constrained evolutionary algorithm with no human intervention. This paper describes the design of the game and its level generation methods, and analysis of playtraces from almost a thousand users who played the game since its release. The analysis corroborates that the level generator creates a sawtooth-shaped difficulty curve, as intended. The analysis also offers insights into player behavior in this game.**

***Index Terms*—word games, game design, procedural content generation, puzzles, difficulty measurement, postmortem**

I. Introduction

Using procedural content generation (PCG) in games, especially for core game elements such as levels, is a bit of an art. When it is executed well, it enhances the player experience and keeps the player wanting to play again [1] but any mistake in the system and the game can feel boring and repetitive, and may lead to disappointment [2]. Just like game design, PCG systems are designed using designers' intuition and a trial and error process. Designers usually test the system on a group of players and adjust the generator based on the feedback until it works as intended [3].

The designers of successful games often write a postmortem about what went right and what went wrong during the development process to summarize the lessons learned and help themselves and others better understand the underlying systems [3]–[5]. Few postmortems, however, are supplemented by player data to validate the design decisions and understand player behavior; conversely, game analytics papers are rarely written by the designers of the analyzed game [6], [7]. In this paper, we will discuss the design decisions for the word puzzle game *Elimination* and its level generator. Our postmortem is complemented by an analysis of player data to validate our generation system and better understand the parameters influencing player decisions.

II. The Design of Elimination

Elimination is a word puzzle game that was designed during GameZanga 8[1], a game jam, with a theme of "Minimalism". The game follows the theme literally by showing the player a bunch of letters and the player has to remove some of these letters so that the remaining letters form a recognized English word. In this paper, we are analyzing the final version of the game which was released on January 23, 2019 for browsers, iOS, and Android devices. The game supports three different gameplay modes: Infinite, Daily, and Levels. In this work, we will only focus on the most popular mode, Levels, which presents players with a fixed sequence of challenges. The Levels mode includes 30 levels with 10 challenges each, all of which were procedurally generated initially, and reused in each playthrough.

[1]https://itch.io/jam/gamezanga8

A. Challenges

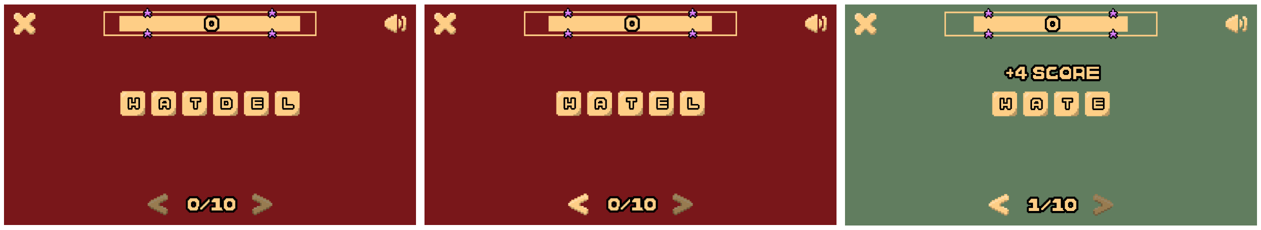
An Elimination challenge is a sequence of letters which doesn't form a word by itself but through the removal of letters will result in a recognized English word. A challenge should have more than one word that can be discovered after removing various groups of letters. Figure 1 shows an example challenge in the game. The player is shown 6 letters "HATDEL" (see figure 1a). The player can eliminate letters by clicking them. A solution for that challenge can be seen in figure 1b and 1c respectively where the player removes the letter "D" then "L" to find "HATE". That is not the only solution, as the player can remove "H", "D", and "L" to find "ATE". One of the incentives to lead the player to find certain words is that the score depends on the length of the found word. Looking back on the example in figure 1, "HATE" will have score of 4 while "ATE" will have score of 3.

We also use bonus letters which are called the 2X letter to influence the player choices. The 2X letter multiplies the score of the found word by 2. For example: if the player found the word "TRUE" where the "U" is a 2X letter the resulting score would be 8 instead of 4. The 2X letter is a good way to influence the player choices toward certain words which could be used in creating easier levels for the player.

B. Levels

An Elimination level consists of 10 consecutive challenges where each challenge has to be solved before a timer runs out. A level ends either when the player find the 10th word or if the timer runs out. The timer is added to make sure that the players don't solve the challenges using brute force by listing all the different possible words and then choosing the highest score. The timer also adds an element of pressure to a generally casual and relaxed game. Equation 1 shows the time in seconds allocated for each challenge in the level where $challengeNumber$ ranges between 1 and 10.

$$challengeTime = \frac{30}{1 + (challengeNumber - 1)/5} \quad (1)$$



(a) Current Challenge. (b) Eliminate "D" letter. (c) Eliminate "L" letter.

Fig. 1: Example of a challenge where the player eliminated "D" and "L" letters to find the word "HATE" for 4 points of score.

When the level ends (regardless of whether all 10 words are found), the game automatically unlocks the next level for the player to be played. We decided to allow players progress to higher levels without completing previous levels to make sure the game is more relaxing and casual by not getting players stuck on particular level. This will make sure that players who play the same level are doing so willingly and not because the game forces them to.

III. Puzzle generation

Elimination levels are generated by running an evolutionary algorithm 10 times with same parameters to generate a single challenge per run. No human curation or editing was performed when generating the included levels.

A. Generative System

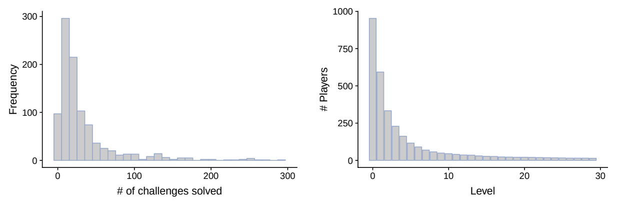
For the generation we used Feasible Infeasible 2-Population (FI-2Pop) Algorithm [8] to generate the game challenges. Each challenge is generated by mixing a group of English words called the source words (usually two or three words) together to form the challenge word. The challenge here is that the mixture has to be efficient and this is tested by satisfying two basic constraints: the final word has to be less than a target length and the order of the letters in the challenge has to obey the same order in the source words. The first constraint is added to make sure the system does an intelligent mixing of the source words so there is no repeated letters. For example: "CAR" and "COOL" could be mixed as "CARCOOL" or it can be mixed as "CAROOL" where the second one is a better mixing than the first as it has fewer repeated letters. The second constraint is added to make sure that the player can solve the level and find any of the source words. For example: if "CDATOG" is a mixture of the source words "CAT" and "DOG", the 'G' can't come before the 'O' as it will be impossible to construct "DOG" from that mixture by just eliminating letters.

Based on that, we decided that the chromosome consists of a list of integer values. The first three values point to the source words in the corpus (We don't use the full corpus during generation. We define a fraction of the corpus to work with for each level so earlier levels uses more frequent source words than later levels.), while the remaining values are used for the mixing procedure. The mixing procedure uses the integer values to determine which mixing word to grab the next letter from. After the mixing is done (before testing for constraints or fitness), the challenge word is shortened by using a greedy algorithm that removes a single letter at a time, each time verifying that the source words can be still extracted from the new challenge word. The reduction algorithm continues removing letters until either no more letters can be removed or the new final word satisfies the length constraints. Since the problem of letter ordering solved by the representation, the only constraint being satisfied is the length of the final word. Target word length is calculated by the formula in equation 2 where w is the challenge word and tl is the target length.

$$c = \begin{cases} 1 & |w| \leq tl \\ 1 - \log(|w| - tl + 1) & |w| > tl \end{cases} \tag{2}$$

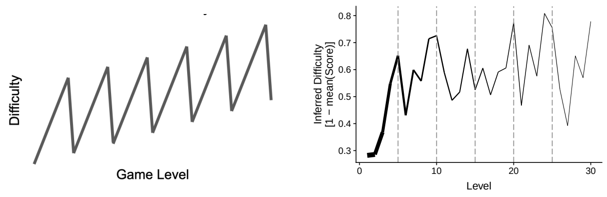
As soon as the chromosome satisfies the constraint function shown in equation 2, it gets tested for its fitness which measures the degree to which recognized English words in the challenge word are obvious to spot or not. For example: "CADOGT" and "CDAOGT" are two different challenge words for "CAT" and "DOG" where the word "DOG" is easier to be noticed in "CADOGT" than in "CDAOGT". The fitness function first generates all the possible recognized English words in challenge word. For example: "HATDEL" is a mixture of "HATE" and "DEL" but these are not the only words in that challenge word as you can get "HADE", "ATE", "TEL", etc. The possible words set is split based on each word length using a certain value called the maximum sequence parameter to form two sets of words: long words and short words. The fitness function tries to minimize the number of words from the long words set that appears as a full words in the challenge word and maximize the number of words from the short words set that appear as a full words in the challenge word. Equation 3 describes the fitness function f, where $maxSeq$ is a the maximum sequence parameter, $chWord$ is the challenge word, and $words$ is a set of all possible words in the $chWord$.

$$\begin{aligned} Long &= \{w | w \in words \land |w| > maxSeq\} \\ Short &= \{w | w \in words \land |w| \leq maxSeq\} \\ v &= \{w | w \in Long \land w \in chWord\} \\ e &= \{w | w \in Short \land w \in chWord\} \\ f &= (1.1 - \frac{|e|}{|Short|}) * (1.1 - \frac{|v|}{|Long|})/1.21 \end{aligned} \tag{3}$$



(a) Histogram of challenges solved. (b) The number of players that reached each level.

Fig. 2: General gameplay statistics.



(a) (b)

Fig. 3: Comparison between the ideal difficulty curve on the left, and the observed difficulty curve on the right.

It is important to note that the fitness function works as a soft constraint compared to the target length in equation 2. We consider length as a hard constraint since a too-long word might not fit on the screen. Because of this we chose FI-2Pop over a standard genetic algorithm, because it makes sure that the hard constraint is satisfied before checking the fitness.

After all 10 challenges are generated for each level, the system picks a fixed number of these challenges to which to apply the 2X letter bonus. The 2X letter is picked by finding the least frequent letter in each source word then picking a random letter out of that list.

B. Parameter Choices

All 30 levels in Elimination were generated automatically by the system with minimum interference from the designers. We only controlled the difficulty of the generated level by controlling the system parameters for each level. We used the following five parameters to control the difficulty of the generated level, $corpusFreq$: the subset of the corpus used during generation (this subset is defined by a minimum word frequency and a maximum word frequency), $sourceWords$: a list of the length of the source words being selected (usually two or three source words), $targetLength$: the length of the challenge word after mixture that is being used to calculate the constraints, $maxSeq$: the split parameter used in the fitness function to calculate the fitness, and $num2X$: the number of challenges in the level that has 2X.

The level generator of Elimination was designed to guide the user through ups and downs in difficulty of play. The idea was to alternate between challenging the user and rewarding them with successes. We selected the parameter values for each level such that the difficulty of the level increases every 5 levels then drops at the 6th level to form a saw-toothed difficulty curve [9]–[11] which is discussed in details in section IV-B. The reason for the difficulty drop is to allow players to have some breathing room after a hard sequence of levels, also the sequence of gradual increase followed by sharp decline is designed to make sure that the generated levels will help the user to be in state of flow [12].

IV. Player Analysis

Without a large sample of play-testers, the design goals of the game, and the accuracy of the level generation parameters was not fully tested before the game was released. After the public launch sufficient player data was gathered to measure the relationship between the intended design of the game and the actual reception by players.

A. General Gameplay Statistics

In total 975 users played Elimination over the course of 98 days. The dedication of these players varied from casual, people who stumbled across the game and quickly realized it wasn't for them, to the hardcore, playing levels over and over again trying to achieve perfect results. The median player solved 20 challenges, equivalent to solving all the challenge of the first 2 levels, and the most diligent player solved exactly 2000 challenges, repeating several levels over dozens of times perfecting them. Figure 2 illustrates the distribution of player dedication through the number of completed challenges and the number of players that played each level.

B. Validation of Difficulty Measure

The shape of the difficulty curve was intended to be a saw-tooth graph (as discussed in section III-B) similar to the one showed in Figure 3a. In practice it's impossible to measure the absolute difficulty of a level (as players get better playing the game), but we can estimate the relative difficulty by comparing players' score from one level to another. We can assume that levels in which players score higher are easier, and levels in which players score lower are harder. Under this interpretation we had hoped to see that player scores would follow the inverse saw-toothed pattern the game's difficulty was designed to produce. The observed outcome is displayed in Figure 3b.

To validate the efficacy of our level-generation parameters on the resultant game, we employ a linear regression to measure the predictive power of the generation parameters on the observed player scores. Our linear regression over the generation parameters took the form of $normalizedScore \sim min(corpusFreq) + maxSeq + targetLength + num2X + min(sourceWord)$, predicting the mean normalized score of each level based on the parameters described in section III-B where $min(corpusFreq)$ is the popularity of constituent words in the English language and $min(sourceWord)$ is the size of the smallest constituent word. Together these features explain almost 20% of the variance ($R^2 = 19.43$) of player scores. What this model indicates is that, of the generation parameters used, the ones most correlated with user score were the ones controlling word popularity, and the length of the challenge word.

C. Understanding Word Choices

Since a level in Elimination consists of 10 independent challenges, we can get a more detailed understanding of the players' responses by focusing on the outcomes of each individual challenge. Similar to our analysis of level generation parameters, we can use a linear regression model to explain a player's selections on a word-by-word basis. Rather than evaluating the efficacy of our AI algorithm, this analysis is more suited to teaching us about the way players think while searching for words in a challenge. We analyzed the following regression $selectionRate \sim wordLength + maxSequence + has2X + splitDistance + firstOccurrence + dirtyWord$ which measures the correlation between the frequency a particular word is selected in a challenge ($selectionRate$) and the following features: the length of the selected word ($wordLength$), the largest number of letters from the selected word appearing consecutively in the original challenge ($maxSequence$), whether the selected word has a 2x bonus on one of its letters ($has2X$), the total number of letter from other words that are interspersed with letters of the selected word ($splitDistance$), the position in the challenge word of the first letter of the selected word ($firstOccurrence$), and whether or not the selected word is profane ($dirtyWord$). Together these features account for 21% of the variance in word selection rates ($R^2 = 21.0$).

The biggest contributing features are $wordLength$ and $maxSequence$. It is fairly intuitive that words with more sequential letters directly in the challenge word are more likely to be selected, as they can be read without any visual interpretation. However, it is less obvious why longer words are more likely to be selected. There are several reasons why players might tend to select longer words. Firstly, longer words are more highly rewarded by the scoring system, selecting longer words rewards the player with a bigger score. Counter-intuitively, though, longer words may also be easier to select than shorter words. For example, longer words necessarily require fewer eliminations (clicks) to be selected than shorter words. Some elimination paths (series of clicks required to select a short word) will accidentally select larger words in the process, immediately solving the challenge. In fact, in the (somewhat rare) context of specific challenge words, there are some desired words that exist, but cannot be selected. For example: the challenge word "AFART", if the player tries to select the word "FAR" they will quickly find out it is impossible. To transform "AFART" into "FAR" requires removing the first letter 'A' and the last letter 'T', however neither can be removed without inadvertently creating another word ("FART" or "AFAR", respectively). This property was completely unknown to us before finding the prevalence of longer word selections.

V. Conclusion

This paper described the design of Elimination, a word puzzle game where all the levels were generated using the FI-2Pop algorithm. The generator was controlled using five different parameters that were adjusted to reflect a saw-toothed difficulty curve. We analyzed the collected user data to validate our difficulty curve by using a linear regression model to predict the players' scores based on the generation parameters. The model was able to explain 20% of the data with the highest correlation for the frequency of the source words and the challenge word length. We further analyzed the data to understand more about the player word choices. We found that the length of the selected word and number of consecutive letters from the selected word in the challenge word have the most effect on the player's choice.

There is much room for improvement to the game, for example we can use automated tuning to adjust the parameters to fit the saw-toothed difficulty curve similar to what was done by Isaksen et al [13]. Another direction can be achieved by using Constrained Map-Elites [14] for offline generation instead of FI-2Pop to generate challenges that explores the full space of the game behaviors.

Acknowledgment

Ahmed Khalifa acknowledges the financial support from NSF grant (Award number 1717324 - "RI: Small: General Intelligence through Algorithm Invention and Selection.").

References

[1] P. Glagowski, "Life after death: Edmund mcmillen on the success of the binding of isaac and his future," https://www.destructoid.com/life-after-death-edmund-mcmillen-on-the-success-of-the-binding-of-isaac-and-his-future-523209.phtml, 2018, last Accessed: May 14, 2019.

[2] E. Maiberg, "'No Mans Sky' is like 18 quintillion bowls of oatmeal," https://www.vice.com/en_us/article/nz7d8q/no-mans-sky-review, 2016, last Accessed: May 14, 2019.

[3] D. Yu, *Spelunky: Boss Fight Books# 11*. Boss Fight Books, 2016, vol. 11.

[4] E. McMillen, "Postmortem: Mcmillen and himsl's the binding of isaac," https://www.gamasutra.com/view/feature/182380/postmortem_mcmillen_and_himsls_.php?print=1, 2012, last Accessed: May 14, 2019.

[5] Lee, L. Kenny, and Teddy, "Rogue legacy design postmortem: Budget development," https://www.gdcvault.com/play/1020541/Rogue-Legacy-Design-Postmortem-Budget, 2014, last Accessed: May 14, 2019.

[6] B. G. Weber, M. John, M. Mateas, and A. Jhala, "Modeling player retention in madden nfl 11," in *Twenty-Third IAAI Conference*, 2011.

[7] A. Drachen, M. S. El-Nasr, and A. Canossa, *Game Analytics: Maximizing the Value of Player Data*. Springer, 2013.

[8] S. O. Kimbrough, G. J. Koehler, M. Lu, and D. H. Wood, "On a feasible–infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch," *European Journal of Operational Research*, vol. 190, no. 2, 2008.

[9] P. Holleman, *Reverse Design: Super Mario World*. CRC Press, 2018.

[10] J. Brown, "Difficulty curves start at their peak," https://www.gamasutra.com/blogs/JonBrown/20100922/88111/Difficulty_Curves_Start_At_Their_Peak.php?print=1, 2010, last Accessed: May 14, 2019.

[11] D. Strachan, "The idea of a difficulty curve is all wrong," http://www.davetech.co.uk/difficultycurves, 2018, last Accessed: May 14, 2019.

[12] J. Nakamura and M. Csikszentmihalyi, "The concept of flow," in *Flow and the foundations of positive psychology*. Springer, 2014.

[13] A. Isaksen, D. Gopstein, and A. Nealen, "Exploring game space using survival analysis." in *FDG*, 2015.

[14] A. Khalifa, S. Lee, A. Nealen, and J. Togelius, "Talakat: Bullet hell generation through constrained map-elites," in *Proceedings of The Genetic and Evolutionary Computation Conference*. ACM, 2018.