Adaptive Single Player Challenge Generation for World War Online

Torstein Lundervold Nesheim
INESC-ID
Instituto Superior Técnico
University of Lisbon, Portugal
torstein.nesheim@tecnico.ulisboa.pt

***Abstract*—In this work we seek to improve the game experience for new players in free-to-play games by using evolutionary computing and procedural content generation. The hypothesis is based on the positive psychology principle of Flow, in that it focuses on generating challenges that are adequate for the player at their experience level within the game in order to improve their player experience. This was implemented and tested in collaboration with Chilltime, through the "Battle Arena" gamemode within their free-to-play browser-based game "World War Online" (WWO). The testing was done outside the actual web-based game, directly in the unity application of Battle Arena, with offline generation of challenges due to long generation times. We use the repeated measures design and the Intrinsic Motivation Inventory (IMI) to evaluate our model, but could unfortunately only do testing with 13 participants due to changes in Chilltime and their servers midway in the testing. The preliminary results we could reach with this sample size indicate that the current model does not increase player enjoyment or improve the experience for new players of WWO, but these results might be a matter of adjusting the difficulty parameter of the evolutionary component of our model. The difference in results between the two versions tested also was not very large, and besides showing that this initial model was close to fulfilling its purpose, it also proves its capabilities of creating interesting challenges, as the main issue found was that the challenges generated were too difficult. We believe this work can be expanded upon and that interesting results can be obtained through parameter tweaking and more rounds of testing**

***Index Terms*—Procedural Content Generation, Genetic Algorithms, Video Games, Flow**

I. Introduction

According to Marc Robinson's 2013 Game Developer Conference talk, "On average, less than 40% of players return to a free-to-play game after just one session." [1] From an economic perspective, this statistic could be simplified to say that Game Publishers of free-to-play games are "losing more than 60% of their potential income" from the very first interaction between a player and their game.

So how do we provide challenges that are adequate for the player at any time in their player lifetime, including at the very beginning - in order to mitigate this huge drop in retention rate at the start of a gaming experience? How do we keep the player progression going long enough for the player to understand the game?

In WWO, a large part of this problem is directly related to the transition from tutorial/campaign to multiplayer battles. The current campaign consists of static, sequential challenges, which presents all players with the same progression curve, even though each player will have a different skill curve. This will, for most players, lead to inconsistent matching between skill and difficulty, which may cause the player to quit playing due to lack of enjoyment. It is this problem of continuously matching player skill with game difficulty in Battle Arena that we seek to solve in this work

We hypothesize, in line with the concept of Flow, that constantly providing the players with challenges that are adapted to their skill level will improve their game experience and thus lead to an improved game experience for new players in the case of WWO. We tested this by attempting to use search-based procedural content generation and a genetic algorithm to help us create meaningful challenges for the players by using choices they make in the tutorial to inform a genetic algorithm of what skill level they are at.

Components of this hypothesis are supported by previous works : when exploring the effects of experience-driven challenge generation in Holiday Knight, Pardal & Martinho [2] concluded that it led to an improved player enjoyment, and similar results where reported by Mestre & Martinho when exploring experience-driven PCG combined with a single player Elo system in "Go Go Hexahedron" [3]. Our search-based PCG is similar to the experience-driven PCG used in these works in that they all aim to adapt challenge difficulty to the player skill, and as a consequence we believe that we will achieve similar results related to the player enjoyment.

We evaluate our hypothesis by having players report on various aspects of their experience after having played through challenges generated for them. For comparison we also ask participants to report on the same aspects after playing through a control set of challenges, static for all participants. We compare the two versions by how appropriate the player rates the challenges in each and how their experience was while playing them. We expected to see that players report meeting a more appropriate difficulty, and having a better experience when playing through the challenges generated for them.

II. Related Work

A. Flow

In Positive Psychology, Flow is the state of being completely and fully immersed in an activity. It is termed as an "optimal experience" by Csikszentmihalyi [4], and one of the key points of flow is that it is a state of inner equilibrium between "anxiety" and "boredom", where your skill is matched by a suitable challenge. This is visualized in Fig. 1, which illustrates the flow state in relation to skill, difficulty (challenge), anxiety and boredom. This figure serves to illustrate that there must be a continuous match between skill and difficulty for someone to remain in the state of flow. This becomes especially apparent in the case of video games.

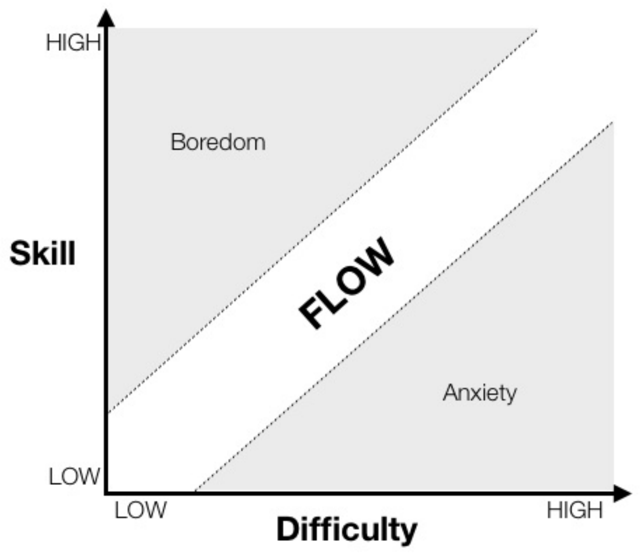

Fig. 1. The mental state of Flow visualized - Adapted from Csikszentmihalyi, 1990.

B. Procedural Content Generation

Procedural Content Generation (PCG) refers to the algorithmic, automatic generation of game content as opposed to the traditional method of manual creation [5]. In this paper, game content will refer to all aspects within a game that impact gameplay, with the exceptions of Non-Player Character behavior and the game engine itself. Within the context of the Battle Arena in WWO, this content generation translates to generating challenges for the players. By using procedural content generation, we can generate large amounts of challenges without having to spend time designing and testing them individually. This also allows for each challenge to be tailored for a much more specific skill level than if we had to design all the challenges manually, which will make it easier to create an optimal state of play for the players according to the principle of Flow.

In our approach we employ the use of a specific case of PCG called Search-based PCG in order to generate challenges for Battle Arena. We will therefore briefly go over what search-based PCG entails, in this subsection. Search-based PCG is a special case of the **generate-and-test** approach to PCG for which Togelius et al. [6] defined the following qualifications:

- The test function does not simply accept or reject the candidate content, but grades it using one *or a vector of* real numbers. Such a test function is often referred to as a fitness function or utility function. In this work we will refer to this as the *fitness function*, and we will refer to its result as the *fitness* of the content.
- The generation of new content depends on the fitness of the previously generated content. In this way, we will gradually produce new content that has a higher value than the previous ones.

C. Evolutionary Algorithms

In many of the cases of search-based PCG, the search algorithm is of an evolutionary nature in line with the criteria listed above. This is because evolution through natural selection of "randomly" chosen individuals can be thought of as a search through the space of possible chromosome values [7] and this will also be the case for our work. Another aspect of having an evolutionary search algorithm is that because of its inherent randomness combined with its converging properties, a very large portion of the search-space is covered naturally, and interesting, unforeseen solutions will often be found.

Evolutionary algorithms is a branch of algorithms within the field of studies called "Artificial Life" in which scientists explore natural systems to draw inspiration for artificial "copies" of natural phenomena to further our technological advancement. These algorithms are based on the theory of evolution by Charles Darwin, in which species adapt to their surrounding slowly through mutations and the famously coined term "survival of the fittest". This coincides well with many of the goals of training AI, in that this process, although "random" in its nature, converges globally towards a solution to a problem over time.

Evolutionary algorithms use a *population* of individuals, where an individual is referred to as a *chromosome*. A chromosome defines the parameters/characteristics of individuals in the population. Each characteristic is referred to as a *gene*. The value of each gene is called an *allele*. In each generation, each individual competes against each other to reproduce themselves. The individuals with the best survival capabilities have the best chance to reproduce and pass on their genes to the next generation. The next generation is generated by combining parts of the parents; this process is referred to as a *crossover*. Each individual in the population can also undergo *mutation* which slightly changes some of the allele in the chromosome. The survival strength of an individual is measured using a *fitness function* which reflects the objectives and constraints of the problem to be solved. After each generation, the strongest part of the population will reproduce and survive to the next generation [7].

Within evolutionary algorithms, Genetic Algorithms (GA's) are some of the most commonly used in relation to PCG and the training of Artificial Intelligence (AI) in games [8]. The process of a genetic algorithm is similar to a generic evolutionary algorithm's, and can be listed as the following steps:

1 Initialize the population

2 Measure the fitness of each individual by the fitness function
3 **Selection**
4 **Crossover**
5 **Mutation**

Steps 2-5 are then repeated until a satisfactory condition is met, typically that the highest fitness of the population is above a certain threshold, a maximum iteration count has been reached or until the population is uniform [9].

By representing the unit setups of WWO as a chromosome where each gene is a unit on the battlefield with an associated unit type, position, strategy and amount, we could evolve the unit setup and test its performance against other unit setups to measure its fitness. This is an important step in generating unit setups/challenges with a difficulty adapted to a given players skill level.

D. *Evaluation Methods*

The Intrinsic Motivation Inventory (IMI) is a set of questions related to a scale of 1-7, designed to assess the subjective experiences of participants in experiments. The participants are asked to state how much they agree with each of a series of statements that can be divided in different "dimensions" of experience, four of which are notable in this work; "Interest/Enjoyment", "Perceived Competence", "Effort/Importance" and "Pressure/Tension". The results in each of the dimensions of the IMI are assembled by adding the scores from each statement in a dimension, inverting it in the case of reversed statements, and then averaging the final sum.

A/B testing is a user experience research methodology where the users of a certain service are subjected to two different versions of the same service. This provides an easy way to compare design and implementation decisions related to user experience simply by comparing the results from the two groups A and B, according to pre-defined metrics [10].

Repeated measures design is an experimental design where each participant takes part in both groups A and B, as opposed to A/B testing, where each participant is only part of one group. Repeated measures design is advantageous over A/B testing in that it naturally requires fewer participants, but in return it comes with some disadvantages. There is a larger risk of fatiguing the participants in repeated measures, and there is also a bias introduced when a participant takes part in both groups. This bias is often mitigating by alternating the order in which the participants take part in the groups.

III. World War Online

World War Online is a browser-based, free-to-play, real-time strategy game with players from over 50 countries, in which players build armies and bases to conquer and defend areas for their in-game country. The players army can consist of a variety of different unit types, which are unlocked as the player increases their *military rank*. They can then use this army by assigning units to defend their bases, by attacking other players bases in the multiplayer mode of WWO, or by going through the pre-determined challenges in a classic level-based progression system in the campaign, which doubles as the tutorial. Our work introduces a new game mode to WWO, *Battle Arena*.

In this work we have assisted the creation of a new game mode in WWO: the *Battle Arena*. WWO is largely built in webGL, and it is a 2D game. Battle Arena therefore distinguishes itself graphically from the rest of WWO by being 3D and built in Unity. Battle Arena is a single player game mode, and in the current version, the players play against challenges that are randomly generated based on their current rating in Battle Arena. As players defeat challenges, their rating increases or decreases based on the result of the match. Within the context of "Battle Arena", both **Challenge** and **Level** refers to the **Enemy Unit Setup**.

When a player encounters a challenge in Battle Arena, they have to create an attacking unit setup, consisting of a maximum of 5 different units, to try to defeat the enemy. In this process they have to choose the units, position them on the battlefield, and determine the amount of each unit type they wish to send into battle. The amount of units determines strength and health of that unit type, but for each challenge there is a total budget that the player has to distribute between the units, and each unit lost in battle is also permanent lost to the player. The amount of units to invest must therefore be carefully chosen.

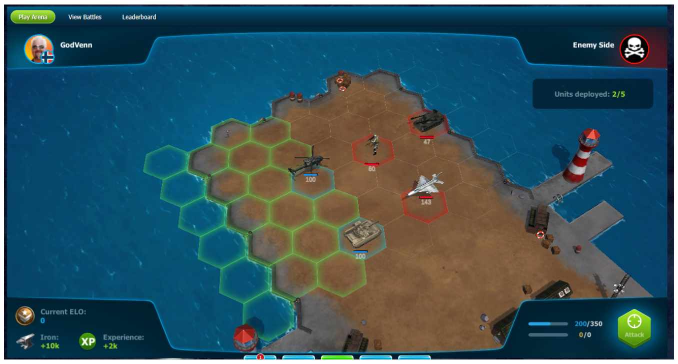

Fig. 2. Battle Arena - The new game mode in WWO. Here the human player "GodVenn" is in the process of choosing his attacking unit setup against the defending computer opponent, whose setup will always be determined beforehand.

IV. Challenge Generation Algorithm

The goal of the challenge generator is to be able to provide any given player with a challenge that matches their skill level. In order to achieve this, we chose the player's recent match history as the representation of their skill. This was also done to encourage players to explore different play styles, as the challenge generator would naturally create some sort of counter to their match history, hopefully leading the player to adapt a new play style to defeat the new challenge. An architectural overview of our final implementation can be seen in Fig. 3.

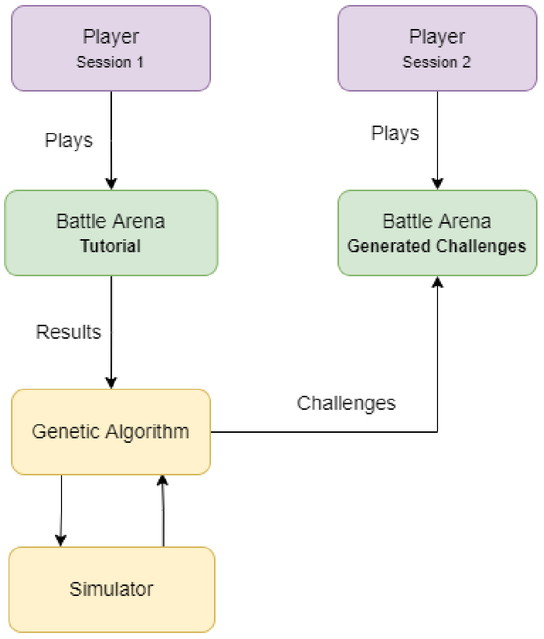

Fig. 3.

For this generation we used offline, search-based PCG, where the search-algorithm is a genetic algorithm, and as a consequence the PCG can be classified as stochastic, generating different outputs each time if run several times for a single input. To create the genetic algorithm, we used C# with the *Genetic Sharp* [11] library for genetic algorithms, because it is open source, has been used successfully in several other academic works already, and is also available for commercial use, an important point in our case.

The GeneticSharp library provides us with a skeleton for our algorithm, alleviating us from setting up the cycling and the transfer between steps in the cycle, and allowing us instead to focus on creating the operators themselves and defining WWO in terms of variables and components in the genetic algorithm. It also provided us with some basic operators to use, although we ended up only making use of the elitist reinsertion operator provided by the library.

The genetic algorithm is responsible for producing a defending unit setup that is of an appropriate difficulty based on the player's skill and experience in WWO. In order to achieve this, two important factors are the initial population and the fitness function. We create an initial population which includes all the different unit types available, guaranteeing that each unit is represented at least once in the collection of chromosomes. This is done in order to guarantee covering the solution-space as efficiently as possible in the dimension of unit variety.

The chromosome in our genetic algorithm is meant to represent a unit setup, and as such it needs to contain information on a set of 3-5 different units and their positions. We used a chromosome which is simply an array with size equal to the maximum unique units, set to five in this work. Each slot in the array can either be empty, or filled with a unit. By allowing for empty cells in the chromosome, we simplify the crossover operator, since we know that all chromosomes will have the same length, despite varying in their unit count.

A unit is described by 4 parameters:

- **Unit ID**: The ID which represents the unit type itself.
- **Amount**: The amount of the given unit type. This determines its strength, and is limited by the total budget.
- **Strategy**: A unit can be set to follow one of three different strategies; Default, Ambush or Guard.
- **Position**: The position of the unit on the battlefield, given as an x and y coordinate in the battlefield grid.

A. *Main Fitness Function - Simulation*

In order to evaluate the adequacy of each proposed solution (unit setup) in the GA, we searched to estimate how each setup would perform against the player for whom the challenge is being generated. To achieve this, we use the processing part of Battle Arena, "Battle Engine", to simulate the battles between the proposed solution and the fitness group. To determine the fitness value of a unit setup, we compare the average result of the simulations to the desired win ratio, which was set to 50% in this work.

Because our fitness function involves an external simulation, it is relatively time consuming: the average duration of a full simulation is 3.5 seconds, implying that the duration of a fitness evaluation is equal to this duration times the amount of setups in the fitness group. We can estimate this to be 17.5 seconds, but it does vary based on the complexity of the battles. Naturally, this implies that evaluating the fitness of each setup sequentially would be excessively time consuming. To avoid this, we use C#'s parallelisation library, which is already integrated with the *Genetic Sharp* library, to evaluate all individuals in a generation simultaneously on different threads of the CPU.

B. *Selection*

The selection operator in our genetic algorithm is a standard Tournament Selection Operator. We compared tournament selection to roulette wheel selection and found that tournament selection yielded an improved conversion rate for our search, as seen in Fig. 4.

C. *Crossover*

In the case of WWO, we envisioned that our crossover operator would switch units between two parents in order to create two children. Originally, we attempted to use a standard 1-point crossover in the array of units that represent the chromosome. This, however, caused a lot of inconsistency in how many individuals in a generation that were actually valid according to the constraints determined by the game.

Our crossover operator "GeneSwitching" aims to randomly choose a random number of units from each parent and switch them, but verifies in the process that each unit switch is "valid" according to the game's constraints, before making the eventual switch. The budget is not verified in the switching process, as it is not a game breaking constraint, and it is

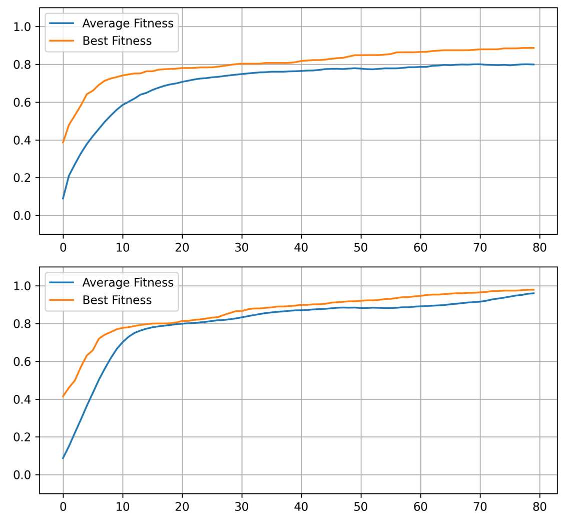

Fig. 4. A comparison between the fitness evolution per generation of **Roulette Wheel Selection (Top)** and **Tournament Selection (Bottom)**. This data is the average result from 20 runs of the genetic algorithm, each ran for 80 generations, with a population size of 50.

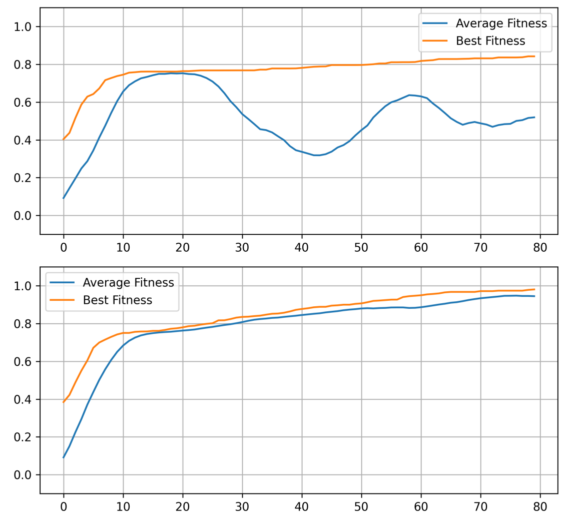

Fig. 5. A comparison between the fitness evolution per generation of **Multi-Dimensional Mutation (Top)** and **Single Dimensional Mutation (Bottom)**. This data is the average result from 20 runs of the genetic algorithm, each ran for 80 generations, with a population size of 50.

instead corrected for each unit setup at the end of the crossover process. The probability of a crossover taking place for two selected chromosomes is set to 80%.

D. Mutation

We started off with a mutation operator which could change all the 4 dimensions of a gene (Unit ID, position, amount & strategy) simultaneously. This operator had a different probability for mutating each of the 4 dimensions, and could potentially modify all 4 in a single operation. This caused the search for solutions to become too wide and random, leading us to search for an operator which could do single dimensional mutation, but without impeding its reachability.

One way to achieve such an operator, is to never mutate the *Unit ID* of a gene, as this is a key cause for imposing constraints on the other dimensions. Instead we would guarantee an initial population which contains at least one gene of each unit ID, spread across the chromosomes, in order to assure that they were all given a chance at being selected for the solution. By doing this, we achieved a single dimensional mutation operator without having to worry about dependencies, while also ensuring full reachability. The difference in performance between the original multi-dimensional operator and the single-dimensional operator with "static" unit ID's can be seen in Fig. 5, where we can clearly see the much improved conversion rate and more stable search of the single-dimensional mutation. The probability of mutation occurring in a chromosome is set to 15% in our project.

V. Evaluation

In order to evaluate our hypothesis, we asked players for feedback regarding their experience while playing through the challenges generated by our genetic algorithm. For this purpose we used a repeated measures design, where players played through two sets of challenges, one generated by us, *Version B*, and another "random" set, *Version A*. We used repeated measures design over A/B testing to reduce the amount of participants needed for the experiment. The challenges in version A are the same for each participant, and were generated through the current solution in Battle Arena for an entry-level rating. The feedback was given through a questionnaire which contains an IMI section to give us data on different aspects of their experience.

A. Procedure

The experimental procedure can be divided in two: the playtesting of the game, and the questionnaire to provide feedback about their experience. In the playtesting part, the participants went through a tutorial of 10 levels, before being presented with the two different sets of 5 challenges each. Because our challenge generation is dependent on already having the player's recent match history available, we used the last 5 tutorial levels as this input - the fitness group in our genetic algorithm. A consequence of not having this information before the user testing process, was that we had to split the process into two sessions, scheduled at different times due to the long generation time of our algorithm in between.

In the first session, the participants fill out the first part of the questionnaire, which contains an introduction to the game for those that do not already know it, and a demographic section. They then proceed to the first playtesting session, in which they play through 10 tutorial levels of an increasing level of difficulty. The 5 first tutorial levels are copied from the campaign in WWO while the last 5 are actually generated for low-rated players by the current solution in battle arena. These last 5 had to be more complex than the first because we needed to gather as much information as possible for the generation of challenges described in the previous section.

Between the two sessions, our genetic algorithm runs five times to create five different challenges for each participant. In each of the five executions, the participant's last 5 unit setups from the tutorial are used as the fitness group. Our algorithm is able to generate a different challenge each time due to the stochastic nature of genetic algorithms, and the result is a set of challenges in which each one is as individually different as they are in the set of challenges that are randomly generated (version A).

In the second session, the participants play through the two sets of challenges and provide feedback on each of them. The participants are not informed about how the challenges for each version were created, only that challenges have been generated for them since the last session. To mitigate the bias introduced by a participant playing one version before the other, each participant has a 50% chance of starting with either version. After the participant has played through one version, they are presented with the IMI questions to report their experience during that playthrough. They then repeat the same process for the second version, and finally they answer a couple of comparative summary questions in the end.

To investigate the aspects of enjoyment, we used 4 dimensions of the IMI: **Interest/Enjoyment, Perceived Competence, Effort/Importance** and **Pressure/Tension**. From each dimension, we used all the items, for a total of 23 statements that the participant had to evaluate in regards to their experience, rating them on a scale from "1 - not at all true" to "7 - very true".

B. Results

Due to company changes in Chilltime along with a server change for WWO, our user testing process was unfortunately interrupted unexpectedly and prematurely, as the webflow and API's used in the playtesting stopped working in this period of time. As we had based our user testing process around the existing architecture of WWO, it would have been too time consuming to work around this, and we ended up with only 13 participants completing the user testing. With such a low number of data sources, the results we can extract from this experiment are limited and not presentable as strong conclusions. Nevertheless, we discuss our preliminary findings and indications that can be extracted from this small sample size.

The demographics section of the questionnaire shows that our participants were largely young adults with 92% being between 22 and 27 years old, with an even division between casual or dedicated gamers, and with all, except for one, never having any experience with WWO. This last information is especially important, as it implies that the results we collected mainly concerns new players of WWO. The average times spent playing during user testing was 27 minutes for the tutorial and 37 min 12 seconds for the combined set of challenges (Version A + Version B), for a total play time of 64 minutes and 12 seconds on average. This means that these results mostly represent the participants' first impressions, which should be taken into account when interpreting them.

When performing the Wilcoxon signed-rank test to compare version A and version B in each of the IMI dimensions, there was only a meaningful statistical difference in one of them, the *Perceived Competence*, with $Z = -2.202$ and $p = 0.028$. The mean values of this dimension showed players felt more competent in version A ($mean_A = 4.705$ vs $mean_B = 3.589$) which could indicate that the challenges generated by our challenge generator were more difficult than those in version A, or it could mean that players felt a stronger sense of accomplishment when completing challenges in version A because they proved more challenging.

The fact that there was no significant statistical difference in the *Interest/Enjoyment* could indicate that the challenges generated did not provide the players with an improved experience compared to those of version A, which would counter our hypothesis, but this would need further exploration with a larger sample size.

The Wilcoxon signed-rank test also showed significant differences in results for two of the custom questions that focus on a more direct rating of the versions, these being "How would you rate the difficulty of these challenges, compared to your skill level?", hereafter referred to as the **Adaptiveness To Skill**, and "How would you rate the overall experience you had playing through these challenges?", referred to as the **Overall Experience**.

The adaptiveness to skill showed a difference with $Z = 2.295$ and $p = 0.028$, with mean values of $mean_A = 3.770$ and $mean_B = 5.230$. The scale of this question goes from **1 - Too Easy** to **7 - Too Hard**, with 4 being the ideal score of a perfect match between skill and difficulty. These results therefore indicate that the challenges in version A were slightly easier than adequate and that the challenges in version B were noticeably harder than what would be an adequate difficulty, with the challenges in version B being further from the best score on this scale.

The overall experience differed between version A and B with $Z = -2.511$ and $p = 0.012$ and mean values ($mean_A = 6.000$ vs $mean_B = 5.000$) showing that players rated version A as a slightly better experience overall, which counters our hypothesis in which we expected version B to yield a better experience overall.

C. Discussion

When combining the results of the perceived competence showing that participants felt more competent in version A, with the results from the adaptiveness to skill showing that version A seems to better match the skill of each participant, they seem to indicate that version B was slightly too difficult for the average participant in this test process. This could be due to the parameter that controls the desired difficulty of the challenges generated, the "DesiredWinRatio" of our Genetic Algorithm's fitness function. For the challenge generation in our experiments, we had this parameter set to 0.50, which means that the challenges generated would ideally break even with the player's match history. Based on these results, we can say that this parameter might be too high, leading to harder challenges, and it could be interesting to experiment further with different values for this parameter.

This difference in difficulty seems to have lead to participants rating their experience in version A as superior. It is worth noting that people tend to rate games as better than others when they perform better at them, so their self-reported experience is not an absolute measurement.

In terms of our challenge generator, these results are not all bad. They confirm that our challenge generator is capable of creating interesting challenges, which is, generally speaking, harder than creating challenges that are too easy. The fact that nearly all of our participants were new to WWO might mean that the challenge generator would have had better results with more experienced WWO players, as these will generally be better players and might appreciate complex challenges in a different way than new players might so that the increased difficulty of version B would be a positive contributor to their experience. The results might also indicate that the player's 5 recent matches do not provide enough information on a player for the algorithm to be able to produce an adequate challenge for them.

VI. Conclusions

In this work, we explored how a genetic algorithm could be used as the search-algorithm in a search-driven procedural content generation of challenges in order to improve player experience in the commercial game "World War Online", in the single-player game mode "Battle Arena". This work was done in collaboration with Chilltime.

We hypothesized that we could improve a player's experience in Battle Arena by using their recent match history to create a challenge of a difficulty which matched their skill. We implemented this by creating a genetic algorithm to act as challenge generator which measures the fitness of each challenge candidate by simulating battles between the candidate and the player's match history using the processing unit of the actual game, called "Battle Engine", aiming for a 50% win ratio.

To test the hypothesis, we had 13 participants play through two sets of challenges, one static set that was equal for all players - Version A, and one set that was generated specifically for that player based on their match history in the tutorial - Version B. We asked each participant to respond to a questionnaire, in which they compared the two versions indirectly by responding to an Intrinsic Motivation Inventory about each playthrough, and directly by responding to a couple of comparative questions. The low amount of participants was due to changes in Chilltime and to their game servers, which lead to a premature ending of the user testing process.

The final results showed no statistically significant difference in the IMI dimensions of Interest/Enjoyment, Effort/Importance or Pressure/Tension, but showed a difference in Perceived Competence, which showed that participants felt more competent while playing through version A. There was also a significant difference in the Overall Experience and the Skill/Difficulty ratio reported by the participants, where they rated version A as a slightly better overall experience as well as being slightly better adjusted to their skill levels.

Due to the low sample size, we can not reach any strong conclusions from these results, but we have some ideas as to what they could indicate. Taking into account that these results are from players who are new to the game, we can say that it seems the challenges generated by our genetic algorithm were too difficult for these players, which led to a worsened experience when compared to the static challenges. This implies that our current model did not succeed in creating challenges that were adequate for newer players, and did not provide them with an improved experience compared to the static challenges.

We believe that repeating this experiment with players who are experienced in WWO might yield different results, as the challenge generator shows potential to create interesting challenges which newer players might not appreciate the same way. We also believe that different parameterization of the genetic algorithm could yield different results, especially concerning the aforementioned "DesiredWinRatio" which basically sets the desired difficulty for the generated challenge.

Acknowledgement

This work was a collaboration with Chilltime.

References

[1] M. Robinson, "Why players are leaving your game," Game Developers Conference (GDC). Available at: https://www.gdcvault.com/play/1020698/Why-Players-are-Leaving-Your (Accessed: 05 January 2021).

[2] J. F. L. Pardal, "Holiday knight: a videogame with skill-based challenge generation," Master's thesis, Instituto Superior Técnico, May 2019.

[3] F. S. Mestre, "Multi-dimensional elo-based challenge progression for single-player games," Master'sthesis, Instituto Superior Técnico, October 2020.

[4] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience. Harper & Row, 1990.

[5] M. Hendrikx, S. Meijer, J. Van Der Velden and A. Iosup, "Procedural content generation for games: Asurvey", ACM Transactions on Multimedia Computing, Communications, and Applications(TOMM) 9(1) (2013) p. 1.

[6] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," Computational Intelligence and AI in Games, IEEE Transactions on, vol. 3, pp. 172 – 186, October 2011.

[7] A. P. Engelbrecht, Computational Intelligence: An Introduction, 2nd ed. John Wiley & Sons, Ltd., 2007.

[8] S. Mascarenhas and C. Martinho, "Darwin's adventure," in Proceedings of the 7th InternationalConference of Videogame Sciences and Art (VJ'15), 2015.

[9] O. Kramer, "Genetic algorithm essentials," ser. Studies in Computational Intelligence. Springer, 2017, vol. 679.

[10] S. W. H. Young, "Improving library user experience with a/b testing: Principles and process," Weave- Journal of Library User Experience, vol. 1, 2014.

[11] D. Giacomelli, "Geneticsharp," https://github.com/giacomelli/GeneticSharp, 2019.