6-2011

Evolution of Flow in Games

Paul J. Tunison
Union College - Schenectady, NY

Evolution of Flow in Games

By

Paul J. Tunison

****

Submitted in partial fulfillment
of the requirements for
Honors in the Department of Computer Science

UNION COLLEGE
March, 2011

Evolution of Flow in Games

Paul Tunison
Advisor: Prof. John Rieffel

Abstract

Every one wants to play a fun game, but "fun" is a subjective quality. Flow, a psychological theory to define what "fun" is, states that, for an activity to be considered fun, the challenge it presents must correlate with that participant's abilities such that the activity is neither too easy or too difficult. One of the biggest problems for game designers is balancing the difficulty of its content in such a way that it appeals to the largest audience possible. In order to broaden audiences, developers need to invest effort into creating numerous, discrete balances that are aligned to varying difficulty normals. Even then, these discrete categories never exactly match more than a few people's abilities.

Previous research has created systems to adjust online, changing the difficulty the system throws at a player as the he or she plays the game. Creators of these systems often state that more complex evolutionary methods, like genetic algorithms, cannot be viable for such online learning due to lacking efficiency and effectiveness. However, newer techniques like the use of generative grammatical encodings have been shown to break such previous stereotypes of non-efficiency, creating the possibility that they might be now be a viable option.

In my research, I implement a game system that uses an interactive genetic algorithm, further using generative grammatical encodings, as a proof of concept that such a system can noticeably balance a game's difficulty online, to any given player. This effect is backed up with test results from the field as to how players felt it adjusted to them.

1 Introduction and Background

1.1 Developing Fun

Whenever we sit down to play a video game, we ideally want to play one thats fun. As an entertainment medium, just like books or movies, a video game's main goal is to provide its players a fun experience. While "fun" is a simple word, what it means to convey a "fun" experience is nothing but simple.

The hardest part about creating a fun game is that "fun" is a subjective quality. No two people are exactly alike or have the same attitudes towards what they enjoy. One of the major components players must take into consideration when determining if a game is fun or not is the difficulty of the game, and whether that game frustrates them or bores them.

In the 1990s, a psychologist by the name of Mihaly Csikszentmihalyi released his work on *Flow*, or concisely, his theory on how people attain their most happy state (Chen, 2006). The eight elements of *Flow* as described in his work essentially boil down to components that an activity should have to provide a fun experience. In the context of games however, only three are really relevant, and one of them is the requirement that an activity, or game, offers an adequate challenge to match the player's ability (Chen, 2006). This problem of developing an "adequate challenge" is currently a hard problem in the video game industry.

To potentially provide the correct difficulty in relation to player ability for the largest audience possible, the current industry standard is to create multiple discrete difficulty settings. This is why we often see easy, medium, and hard difficulty settings in video games today. Each are often manually tuned, requiring a great deal of effort and time. However, this is often not adequate as the few settings that are created do not match more than a few people in a developer's intended audience, and most times the few people that the settings do match are the developers themselves.

The subjective interpretation of fun means it is on an analogue scale. If we were to follow the current standard, to create an ideal spread of difficulty settings, there would need to be an infinite number of them. Then, the player of the game would need to pick out the correct setting out of the many. This is obviously not feasible or possible. Thus, it would optimal if automatic systems were created to perform online difficulty balancing, or while the player is playing the game, allowing an optimal audience to enjoy the challenge of the game, as it is adjusted specifically for them. If tools existed that could measure the boredom or frustration levels of a player, then it would be a somewhat trivial task to create a difficulty balancing system, however these tools do not exist.

1.2 The Role of Flow

Csikszentmihalyi's work on *Flow* clearly states that an activity's difficulty or challenge must match, or at least be close to, the person's ability level. As shown in figure 1, if an activity or game provides too much challenge in comparison to the player's ability, he or she feels frustrated or anxious. On the other hand, if the activity provides too little challenge compared to the player's ability, then the player feels some degree of boredom. The sweet spot is somewhere in between, where the challenge provided nearly matches with the player's ability level. Thus, since every person, or even the same person over time, has different levels of ability, the challenge a game provides must be dynamic, and change along with the ability of the player (DeKoven, 2010). Computationally, this task of dynamically changing a game's difficulty over time is often called dynamic difficulty adjustment (DDA).

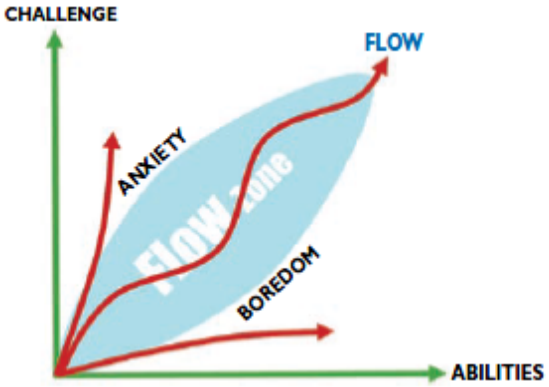

Figure 1: Graph showing the Flow zone and the psychic entropies adjascent to it (Chen, 2007).

1.3 Motivation

The concept of a system that dynamically balances difficulty is not a new one. Research as well as a select few published games have attempted to implement DDA systems, like *Half-Life* from Valve Software (Hunicke and Chapman, 2005). While research has produced valid difficulty balancing systems, researchers often express the opinion that more complex learning systems, and specifically evolutionary systems, are not feasible enough to make an adequate online learning DDA system.

One claim is that the speed at which an evolutionary system learns is too slow and random to be considered for online learning (Pieter Spronck and Postma, 2003). It is true that evolutionary systems, such as genetic systems, are inherently random in their growth, however, the speed at which they converge on higher fitnesses is a function of the method that the system implements. This has been shown through the use of generative grammatical encodings by Hornby et. al., where more optimal table designed were evolved considerably faster than a traditional, basic genetic evolution system using direct encodings (Hornby and Pollack, 2001). More importantly, the in-feasibility of evolutionary systems has not been universally proven in any way, leaving room for the possibility that a feasible, and perhaps highly successful evolutionary system, can be made for online DDA.

Secondly, claims exist that passive DDA systems cannot correctly model and balance player enjoyment (in relation to challenge) and that an active DDA system is a better choice (Chen, 2006). An active DDA would be one that acts upon minute choices performed by the player, adjusting difficulty based on those responses. This would in turn let the player make the balancing choices instead of relying on a passive system in the background to make the balancing decisions. With his game, *flOw*, being a prime example of an active system, not all game mechanics or systems may be able to be driven by player choices alone, or at all. It is also possible that presenting such choices to the player would break a player's sense of immersion. Even if cases were to exist where a purely active DDA system could not fully provide sufficient difficulty balancing, it would be more advantageous if some kind of passive system underneath could pick up the slack.

2 Method

As described above, there is no concrete evidence that evolutionary methods truly cannot be the basis behind a successful DDA system. To test whether or not an evolutionary system is feasible, a proof of concept game which uses an evolutionary back end, called Genetic Tetris, was created. This system comprises of a genetic algorithm, further taking advantage of generative grammatical encodings to represent and vary difficulty. Subsection 2.1 will discuss the basis of the game system and how difficulty is represented. The genetic evolution method used in this system is described in subsection 2.2. Subsection 2.3 then describes the use of generative grammatical encodings to represent shapes throughout the game. Finally, the function used to evaluate difficulty will be discussed in subsection 2.4.

2.1 Genetic Tetris

The Genetic Tetris system is a game created as a proof of concept to show that dynamic difficulty adjustment may be performed online by an evolutionary system. This game functions close to that of its namesake: the original Tetris game (Tetris Holdings, 1985-2011). With a DDA system in mind, however, an exact duplicate of Tetris is not very feasible for difficulty balancing. The main reason for this is that the only change in difficulty from stage to stage in the original Tetris is the rate at which shapes drop from the top of the screen. This is a very limiting parameter to balance over. Thus, difficulty must be observed in a different manner.

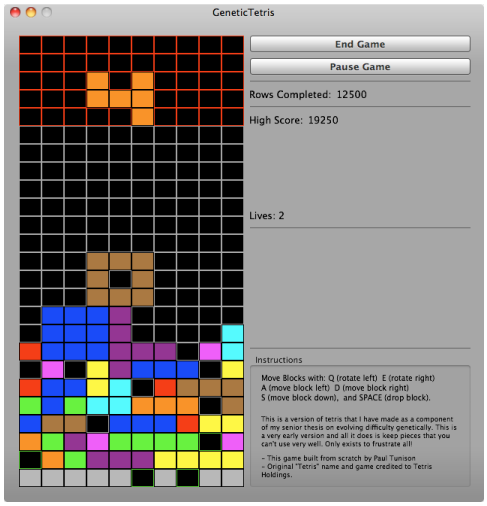

Figure 2: A screen-shot of the interface of the Genetic Tetris game.

In the Genetic Tetris game, difficulty is represented in the shape of pieces that the game has players manipulate, instead of how fast they drop, as can be seen in Figure 2. It can be argued that, in the original Tetris game, increasing the drop rate of shapes tested a player's spacial reasoning, or ability to manipulate the shapes in order to place them such that rows were completed. In a similar way, changing the shape of the pieces altogether also challenges the player's spacial reasoning, but to an expanded degree. In terms of balancing difficulty, different shapes of pieces can represent a much wider spectrum of challenges and provides a better opportunity to balance correctly to a player.

2.2 Evolutionary Adjustment

The evolutionary method implemented to adjust difficulty over time is an interactive genetic algorithm. For review, a genetic algorithm is a type of random search technique that resembles a digital model of Darwin's theory of survival of the fittest. In Genetic Tetris, the population to evolve over time is comprised of genotypes which represent shapes.

One of the key differences in this implementation is the use of a per-loci mutation policy. At a probability of 20%, new genotypes produced each generation have a chance of being noticeably different than a seed genotype. Smaller probabilities were tested, however, after a number generations, smaller values often led to a majority of the genotypes in the population becoming too similar. This then led to an uninteresting play experience. The 20% probability seemed to provide sufficient diversity in mutations while maintaining noticeable lineage.

2.3 Generative Grammatical Encoding

The genotypes in the population were encoded using generative grammatical encodings. This uses a Lindenmayer system, which applies a set of rules to describe the structure of the resulting phenotype, or shape to be thrown at the player. These rules work to construct command strings to be passed into a simple turtle drawing system. The advantages of using a generative grammatical encoding can be seen through work performed by Hornby et. al. with generative grammatical encodings and the building of table structures (Hornby and Pollack, 2001). They're results showed that the use of generative grammatical encodings produced more optimal results, and faster than traditional direct encodings. As the task of building shape structures in this game is similar structural task, the use of generative grammatical encodings should provide better and faster results than from using direct encodings.

Another advantageous aspect of using generative grammatical encodings, and transitively the L-systems that underly them, is that they can scale and create "families", or sets of results with similar traits (figure 3). The set of rules that make up the encodings can be applied on top of a base structure any number of times, creating phenotypes, and in this case, structures, of ever increasing complexity. This ability to scale is conveniently used as a means of increasing difficulty over successive stages. Thus, a single genotype which uses generative grammatical encodings not only represents a family of shapes with similar traits, but also the rate at which complexity, and thus difficulty, scales throughout the game.

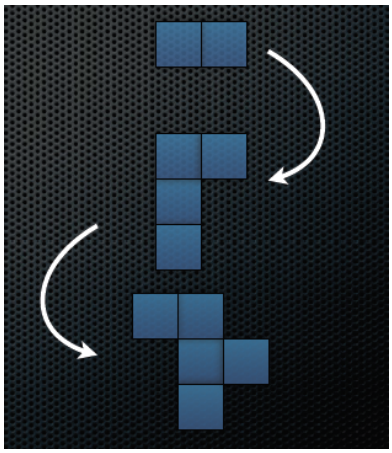

Figure 3: An example progression of an generative grammatical encoding. The similarity between successive reapplications of the rule set can easily be seen, as well as the increase in complexity.

2.4 Evaluation of Difficulty

To evaluate whether the each genotype in the population is providing an adequate challenge for the player, the fitness function for the genetic algorithm relies on a couple basic assumptions to balance difficulty around as well as measured

metrics for each genotype.

Assumptions The goal of this system is to create set of genotypes which define shapes that will provide an amount of difficulty to match a player's ability at the game. This however cannot be done without some kind of basis to measure a player's performance in order to determine which genotypes are not adequately challenging. It is assumed that an ideally balanced set of genotypes will produce shapes over the course of the game such that, through placing them on the board towards the goal of completing rows, the average height of the board over the course of the game will be the same as when the game started. Thus, if the game started with blocks on the board such that the height was $H/2$ where H is the height, in blocks, of the playable board space. This assumed height might actually be different based on human psychology, however for now $H/2$ is assume ideal.

Measurements Two measurements are taken into account when determining the fitness of a genotype. The first is the ratio between the actual drop time and the maximum drop time ($\frac{ActualDropTime}{MaximumDropTime}$). The "actual drop time" is defined as the time between when the shape is created at the top of the board and when that shape's blocks are integrated into the board (collides with the bottom of the board or other blocks). The "maximum drop time" is defined as the time that the shape would maximally need to take to integrate into the board by its own devices. Thus, this value is a function of the average board height at the time of the shape's spawn and the system defined constant for the timing between automatic shape drops. As there is no preview of the next usable shape, like in the classic Tetris, players should have no reason to be distracted and should focus solely on the shape at hand and in trying to manipulate it. This measurement is based on the idea that if a player finds a particular piece easy to drop into a useful location, that player will most often place it quicker than trying to drop into place a more difficult piece, where they must think about it longer.

There, however, exist cases where a player does not perform a "soft drop" or "hard drop" during a game instance, where a piece is forced to fall faster than the automatic drop rate. In this case, our second measurement of how many gaps were introduced into the board, is needed to correctly determine the relative difficulty of a genotype.

It can be argued that pieces, when integrated into the board, who produce more gaps, are more difficult than those who produce less or no gaps. The major sign of when a player is having a difficult time placing pieces is when the board builds up to the point that it exceeds the height limit of the board, triggering a height and difficulty reduction, or a game over. This can only happen if, as the player placed pieces onto the board, gaps were introduced and rows could not be completed. Thus, pieces that produce more of these gaps are hindering the player more than those that do not.

Collective Evaluation When evolving the population over time, we obviously must determine the fitnesses for each genotype. To do that, we must use a judgment of the player's performance since the last evolution by observing the state of the board, as well as the measurements taken for each genotype (drop time ratios and gaps introduces, explained above). In order to determine which grammars to eliminate each generation, we must determine which ones are hardest and easiest compared to each other genotype in the population. Difficulty must be measured comparatively because of the fact that difficulty is a subjective quality, meaning that there is no one absolute way of measuring how difficult something is, and in this case, how difficult a shape or piece is to manipulate successfully.

Each genotype in the population is assumed to have measurements collected for both metrics, as explained above, when evolution occurs (this is enforced by the game system). For each genotype G_i in a population of size n, let the set of drop time ratio values and gaps introduced values be $S_d(i)$ and $S_g(i)$ respectively, where i is the unique index of the genotype.

An average value for each metric is derived for the population, determined by the averages of the data for each genotype. Collectively, these two population-wide averages represent the average "difficulty" of the population. It is admittedly true that these values do not directly relate to the *absolute* difficulty of the population, but it does allow us to compare within the population to judge which genotypes might be harder or easier than others. Let P_d and P_g be the population average values for drop time ratios and gaps introduced, respectively, such that:

$$P_d \leftarrow \frac{\sum_{x=1}^{n} \text{avg}(S_d(x))}{n}$$
$$P_g \leftarrow \frac{\sum_{x=1}^{n} \text{avg}(S_g(x))}{n}$$

Once we are able to determine where a genotype stands in the population, we need to know how the player performed over all. If the player performed poorly over all, then we know that the "easier" pieces from the population are closer to providing the correct challenge for that player, and vise versa. To do this, we must observe the average height of the board, where height is in number of blocks, since the last evolution.

The idea here is that if the average build up of blocks on the board is too high in comparison to an ideal medium height (assumed to be $H/2$, as mentioned in Assumptions), we want new genotypes that are "easier" than the currently observed average difficulty of the population, and vise versa. The percent difference between the actual height and the ideal height could be considered the degree to which the population was too easy or difficult. Knowing this percent delta, we can adjust the average "difficulty" rating of the population by that same percentage. This is useful in that it

influences genotypes that were easier or harder, depending on whether the board was seen as hard or easy respectively, to have higher fitnesses. If this value is negative $(-)$, then the actual average is above the ideal height, and if the value is positive $(+)$, the actual average is below the ideal height. Let this percent deviation be $b\Delta$.

After knowing the percent deviation from the ideal height, we adjust the population averages in a manner that reflects this deviation. To do this, we introduce a function $m(S_v, v)$. Given a set of values S_v and a value v, such that $v \geq \min(S_v)$ and $v \leq \max(S_v)$:

$$m(S_v, v) = \min(v - \min(S_v), \max(S_v) - v)$$

This method, when multiplied against $b\Delta$, prevents the population average from being adjusted too drastically, irregardless of the range or grouping pattern of values.

As mentioned, in conjunction with $b\Delta$, we adjust the population average measurement values to reflect the impact of the player's performance. Thus, aP_d and aP_g may be defined as:

$$aP_d = P_d + (b\Delta \times m(\{s|s = S_d(i), 0 < i \leq n\}, P_d)$$
$$aP_g = P_g + (b\Delta \times m(\{s|s = S_g(i), 0 < i \leq n\}, P_g)$$

Now that we have performance adjusted population averages, we can determine the fitnesses for the individual genotypes. A mean squared error is calculated for each genotype, squaring the difference between the genotype's average measurement values and the population's adjusted average measurement values. This method is used to penalize those that are farther away from the adjusted population values. Fitness values are then calculated by normalizing the reverse values of the results from the mean squared error calculation (see stage2 calculations in Algorithm 1). This causes the genotypes whose average measurements were closest to the populations adjusted averages to have the highest fitness.

Using the variable names defined thus far, this process proceeds closely to what is shown in Algorithm 1, calculating a fitness for each genotype in the population.

3 Experiment and Results

As the concept of difficulty is subjective, testing and response from actual users is necessary if we are to determine the success or failure of the system. We will go over the experimental setup and the method of pilot testing this proof of concept system in subsection 3.1. Subsection 3.2 will present the results of the pilot study.

3.1 Experimental Setup

Each participant is required to play two versions of the game: one that uses the evolutionary learning system, and a second version that uses a randomized evolution system. Participants were not told which game was which before playing. Each game was played for at least 15 minutes, but participants are allowed to player a little over time if wanted. After the play session, a survey was filled out.

The random version plays and functions the relatively same as the learning system does, except that the evaluation function assigns fitnesses to the genotypes at random. Such a random system is used as a base of comparison because a number of games, like the original Tetris, rely on randomness to convey part of or all if its intended challenge. Thus, if such a system could provide a better or more consistent challenge for a person than a random system, we know some measure of effective learning is being performed.

The purpose of the survey after the play session was to collect the subjective data of how each participant felt each of the games balanced to their abilities, how frustrating or boring the game they felt learned better was, and how much "fun" in general they felt while playing the game they felt "learned" best. As the participants were not told which game was the learning game and which were random, this primarily tests to see if the learning and balancing was noticeable to the player in comparison to a random system.

3.2 Results

As can be seen in Figure 4, out of 10 participants, 6 (60%) discerned correctly the system that implemented the learning algorithm. Out of those, 100% rated the experience a 4 or higher on a 5 point scale.

The survey also asked each participant to rate the amount of "fun" they had playing the game that they guessed was the better learning system. This means that if they participant thought the random system was the learning system, their "fun" rating is with respect to the random game system, and not the learning game, and vise versa (Figure 5).

Algorithm 1 Fitness evaluation process for genotypes $G[1...n]$ in a population of size n.

for $i \leftarrow 1...n \wedge i \in \mathbb{Z}$ **do**
  //record the difference between a genotype's average
  //measurement and the adjusted population average
  $\text{stage1}_d[i] \leftarrow (aP_d - \text{avg}(S_d(i)))^2$
  $\text{stage1}_g[i] \leftarrow (aP_g - \text{avg}(S_g(i)))^2$
end for
for $i \leftarrow 1...n \wedge i \in \mathbb{Z}$ **do**
  //normalize reciprocal values
  $\text{stage2}_d[i] \leftarrow \left(\frac{\max(\{x|x \in \text{stage1}_d\}) - \text{stage1}_d[i] + 1}{\max(\{x|x \in \text{stage1}_d\})}\right)$
  $\text{stage2}_g[i] \leftarrow \left(\frac{\max(\{x|x \in \text{stage1}_g\}) - \text{stage1}_g[i] + 1}{\max(\{x|x \in \text{stage1}_g\})}\right)$
end for
for $i \leftarrow 1...n \wedge i \in \mathbb{Z}$ **do**
  //sum stage 2 values for end fitness value
  $G[i].fitness \leftarrow stage1[i] + stage2[i]$
end for

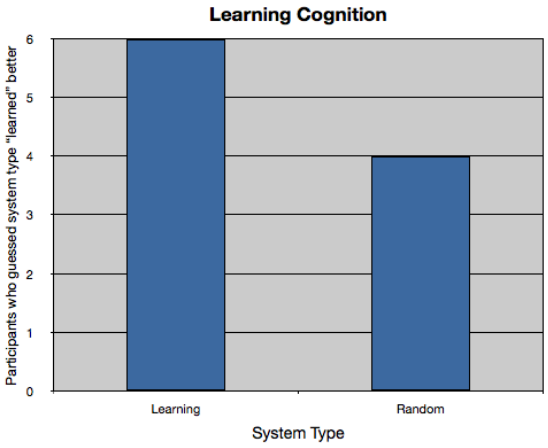

Figure 4: Results of which system participants guessed learned better.

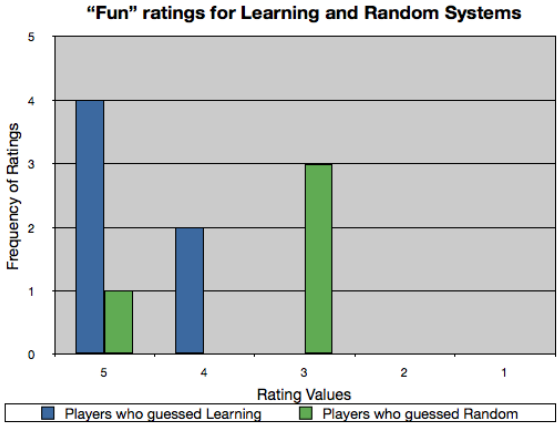

Figure 5: The ratings at which participants believed their chosen systems to be "fun".

Below are a few comments from participants who recognized the learning system:

> "The first game [learning] was more of a challenge for me, but I think that's only because I didn't realize what I was getting into before I started. ... The second game was easier only because I knew what to expect from the game. Also with the second game I feel like it took longer for the game to pick up on my level of difficulty, than in the first game. Likewise the first game picked up my progression faster, but the shapes weren't as challenging."

> "Game B [random] was just frustrating, but game A [learning] kept me engaged."

> "I also found GeneticTetris_A [learning] more 'addicting' than the other version (I assume it's because of the better game-play / experience). Also version B [random] frustrated me more because I felt that it didn't adjust back down enough in difficulty for me."

4 Conclusions

This work has attempted to create an online learning system within a video game to dynamically balance the challenge presented to the player based on their observed ability to tackle that challenge. It can be easily seen through these results that this system is far from perfect or completely viable for mass production in commercial games yet. However, this study did result in more than 50% of participants able to notice which version of the game was able to "learn" better than the other in only 15 minutes. If we look into how long the average span of time a video gamer sits down and plays games, it is often longer than 15 minutes, so a longer play time might result in the learning system making itself more noticeable in terms of its ability to balance to the player's ability in comparison to the random system.

Another curious result, shown in figure 5, is that all of the participants that correctly determined the learning system rated their fun at about 4.67 our of 5. The others rated their fun with the random system at an average of 3.5 out of 5. This is positive for the learning system as it seems to show, for those that were able to recognize it, the learning system correctly balancing pieces towards peoples' ideal challenge levels. The comments above from participants also seem to support this. However, this should also be taken with a grain of salt at this time because the game, due to its crazy looking shapes and new mechanics, might have bewildered player's to some degree, creating a novelty, or gimmick effect. Again, longer testing periods over a larger participant pools may work to answer this question.

4.1 Future Work

In the future, this work first and foremost needs to be tested with more people as well as in more game environments. Further testing beyond a mere pilot test would bring in more data that can provide more meaningful and reliable data in the results. Also, additional testing methods would need to be employed. For example, a longer single-session set of participant tests could yield better results as to whether the learning system is able to balance quickly and correctly, and then maintain that balance over time even as the player learns to player the game with a higher ability. Yet another test should be a long-term test where the same participants play the system over a multi-session span of time. This would test to the same concepts, but on a longer time scale, maybe for possible use in MMORPGs.

Even if it is shown that this learning system works to an acceptable degree in this Genetic Tetris mechanic, determining whether such an evolutionary system can generally perform online DDA will also need further test implementations over other video game genres (FPS, RPG, etc.). This

would work to ensure that this an evolutionary DDA system doesn't just only work for games where physical structure is the direct determinant of difficulty. The use of generative grammatical encodings have been shown to be useful in systems that have some kind of structure to grow, but certain components in today's game AI already have structure based components. In the CRPGs that Spronck et. al. worked with, the AI non-playing characters were controlled by scripts of predicate-action pairs. This script is in its basest form a type of structure that could potentially be evolved over time (which Dynamic Scripting does in a simple manner).

Yet another manner in which this work can be extended is by testing the performance of other evolutionary techniques and by optimizing the fitness function. For this research, it was assumed that the use of generative grammatical encodings would provide better performance than the use of simple direct encodings based on other research. However, other methods of encodings exist that may be tests. As for the fitness function, simple logically deduced math was applied, but more complex and accurate mathematical concepts may be able to better and more efficiently determine player performance to a concrete normal..

References

Chen, J. (2006). Flow in games. MFA Thesis.

Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM*, 50:31–34.

DeKoven, B. (2010). Of fun and flow.

Hornby, G. S. and Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *In Congress on Evolutionary Computation*, pages 600–607. IEEE Press.

Hunicke, R. and Chapman, V. (2005). Ai for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM New York, NU, USA.

Pieter Spronck, I. S.-K. and Postma, E. (2003). Online adaptation of game opponent ai in simulation and in practice. In Mehdi, Q. and Goughi, N., editors, *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, volume 3, pages 93–100.

Acknowledgments

I would like to thank my friends here at Union College for keeping me sane. I also would like to thank my thesis advisor Professor John Rieffel for keeping me on track and encouraging me along. I would also like to thank NextStep and Apple, and this system is build upon the cocoa framework, for creating an easy to use, elegant and fun to work with environment.