Applying Multi-Armed Bandit on Game Development

Kenny Raharjo

Department of Computer Science

University of British Columbia Okanagan

Email: kennyraharjo@alumni.ubc.ca

Abstract: Multi-armed bandit is an increasingly popular statistical learning paradigm which can be used to make optimization decisions. The video game industry is rapidly growing and one of the fastest growing sectors in the U.S economy. The competitive game industry requires complex algorithms and critical choices to ensure that the latest games capture the attention of demanding consumers. This thesis applies multi-armed bandits to game design and evaluation. After describing multi-armed bandits and game analytics, a sample study evaluating multi-armed bandits on a dynamically changing game is presented. The results demonstrate that the multi-armed bandit algorithm can indeed be applied to game algorithm and used to find optimal design.

Keywords and phrases: multi-armed bandits, algorithms, game design, game analytics, game data mining

Contents

1 List of Symbols

The list below provides a reference for commonly used symbols in the thesis.

K	The number of arms available in a multi-armed bandit. It is usually a positive integer
n	The number of iterations that have elapsed
H	The maximum time period. This number may be finite or infinite, and may be unknown ($n < H$).
x_i	The reward obtained by selecting arm i of a multi-armed bandit process at any time
$x_{i,t}$	The reward obtained by selecting arm i of a multi-armed bandit process at time t
v_i	The reward distribution for arm i. The distribution is a random variable and initially unknown.
μ_i	The expected value of the distribution of arm i
μ^*	The expected value of the distribution of the optimal arm
R_t	The measure of regret at time t
ε	The constant set to determine the probability of exploration for epsilon-based bandit algorithm

2 Introduction

A video game refers to any of the various interactive games played using a specialized electronic gaming device, computer or mobile device and a television or other display screen, along with a means to control graphic images [1]. With the advancement in technology, personal computers and other gaming devices are readily available at affordable rates. The game development industry has been a booming sector in the United States. As of 2015, the video game industry generated USD $74 billion worth of sales, which is the third largest industry in the U.S entertainment market.

Video games have evolved into a mass medium. According to the Entertainment Software Association (ESA), more than 150 million Americans play video games and 42 percent play video games at least three hours per week [2]. People rely on games for entertainment, a get-away from reality, and educational games in hope to learn concepts in a fun and creative way, hence the term edutainment (education from entertainment).

Studies show that video games may improve cognitive skills, such as reasoning, memory, spatial and problem solving skills, but the primary benefit is for entertainment. Video games help kids manage stress as well as build teamwork and self-esteem [3]. Games can allow a person to be themselves, without the need to put up a public persona. For educational games, gamers can try the questions without the fear or embarrassment of getting the questions wrong [4]. Researchers found a 57 percent decrease in depressive symptoms among those who played casual video games and 70 percent of teachers said that video games increased students' motivation and engagement levels, according to a national survey of teachers who use games in the classroom [2].

However, due to the lack of time and high demand for quality games, a large number of games nowadays have high abandon rate. If a game does not appeal to a player, or becomes repetitive, they will stop playing it entirely. Different people have their own preference, something developers have no control or knowledge of. If players do not like the game design or concept, the abandon rate will be high. What if there exist a way to dynamically change parameters of a game to suit the users' taste, and improve user experience? A possible option is to apply the multi-armed bandit model to game design, in attempt to optimize player satisfaction.

The multi-armed bandit model is a statistical model which balances exploration and exploitation in order to solve recurring decision problems commonly faced by projects. The model is simple to understand, relatively easy to implement and requires little supervision. Many decision makers have faced a dilemma

and later on regret making the wrong choice. The multi-armed bandit is a solution as its primary objective is to minimize regret, reliably providing the analyst with the best solution to the decision problem faced.

The aim of this study is to apply the multi-armed bandit approach to dynamically changing game design, in which design or difficulty might vary between players, or adjust to the player's preference.

2.1 Multi-Armed Bandit

Imagine that you are dispatched by your company to a foreign country for a week. It is your third day and you have had meals at four different diners. While in the shower after a long day of meeting with clients, you begin to ponder where you should go to lunch the next day. You want to make a decision which would bring about the maximum happiness. Would you consider trying a new restaurant across the street which is fairly new, hoping that it would be better than the places you dined at, or just stick to the steakhouse, which is your favourite amongst the 4 restaurants you had? Since the meal expenses can be claimed by the company, price is not a deciding factor. This is an example of making a decision to "explore" new options, or to "exploit" your knowledge of the best restaurant, in order to get the maximum satisfaction and minimal regret.

Another interesting example is a bartender/mixologist trying to find the 'ideal' mix of drink to serve patrons. There are a large number of alcoholic drinks to mix and he cannot afford to try each combination, as he needs to select the best mix before getting drunk. He could "exploit" his knowledge and recommend the best mixture he tasted, or he could "explore" a different mix in hope of finding a world-class brew, or risk of it being a bad mix (and getting drunk).

This exploration vs. exploitation can be formulated as a multi-armed bandit problem. The common real-world metaphor for this problem involves a gambler facing a collection of slot machines at a casino, with each machine being referred to as an "arm", hence collectively known as the "multi-armed" bandit. Each machine holds an unknown distribution and expected payout, and he plans to maximize his cash prize pool, the "reward" through a sequence of lever pulls [5]. After each attempt at the machine, the gambler must decide if he should attempt another slot machine to learn about its reward distribution, or should he use his knowledge of distribution from past experience to acquire large rewards now.

The multi-armed bandit is a popular choice for experimental design since after each action is taken, valuable information is obtained about that action, and this data will be used as feedback to the algorithm to further improve its upcoming actions. Thus, it can be viewed as an evolving algorithm, being fed information and allowing it to make better decisions over time.

2.2 Game Analytics and Game Metrics

In order to assess how good a game is and improve on it, gameplay data will be collected from players, called *gameplay metrics*. Examples of game metrics from a game of *Super Mario Bros*. include the numbers of jumps executed or time taken to complete a stage. For some games, there are a wide variety of metrics on the user's gameplay, but it is not realistic to capture them all due to budget and memory constraints. The goal is to pick the metrics that are the most useful.

Telemetry is used to physically collect data about the player's game playing. In Greek, *Tele* is the English equivalent of "remote" and *metron* meaning "measure" [6]. As long as there is an internet connection, it is possible to continuously collect data from gamers through updates, patches, game multiplayer modes or when players upload their data to a scoreboard or other purpose.

The overall goal is to improve the game design and reduce abandon rate. Player game metrics can be consolidated and statistical analysis performed to obtain additional knowledge of the how the game is received. Hypothesis testing is useful to support or reject a proposed hypothesis, and also find correlation between various game metrics. These results are then used to perform modifications to improve the current game via a game patch which the users will be able to use to update their current game version, or to implement them for upcoming games of the same genre. The latter is commonly practiced for seasonal games which have new versions each year, such as the *Madden NFL* series.

2.3 Motivation Topics / Example Applications of Multi-Armed Bandits

2.3.1 **Game Theory**

Game Theory studies the behavioural relations between two parties who are cooperating or in conflict. One example is finding Nash equilibrium, which is a set of decisions each player makes such that the opponent has no incentive to change their decision. Multi-armed bandits can be used to find solutions for adaptive, sequential, and repeated games.

A classic example in Game Theory is the Prisoner's Dilemma, which describes two criminals who have been arrested in separate cells and have no means of communication between each other. During interrogation in each of their respective cells, each criminal can either confess that he did the crime or deny the crime. If they both confess, they both will get convicted 1 year in prison. If they both deny, they will get 2 years in prison. If one confesses and the other denies, the person who confesses will get 3 years and the one denying will be set free without any prison time.

Examining the payoff matrix below, observe that if they are both able to communicate and agree to confess, they will get the optimal social outcome which is both getting 1 year of prison time. However, since they are not able to cooperate, they have to think independently and pick the most rational decision for their own benefit. The prisoner considers making two (pure) strategies, confess or deny. His payoff (which is defined as negative jail time, since they want to minimize jail time) is set to bold in the payoff matrix. If he chooses to confess, he would have to endure 1 year of prison time if the other prisoner confesses, or else he would have to face 3 years of prison time. If he chose to deny, he will get either 0 or 2 years of imprisonment. Thus, he would be better off denying if he is not certain of the other criminal's decision, since he will get a shorter jail time despite his accomplice's decision. With this reasoning, both players will deny and get 2 years of jail time, which is the Nash equilibrium, but not the optimal outcome of both confessing and getting only 1 year jail time.

A\B	Confess	Deny
Confess	**1**,1	**3**,0
Deny	**0**,3	**2**,2

Prisoner's Dilemma payoff matrix

The example above shows the most basic case of a single stage simultaneous move game. Now consider a general case where there were more players and strategies for each player, and with the game having multiple stages. The prisoners will have additional knowledge of the accomplice's behaviour and can implement a variety of strategies on the second game onwards. This repeated game can be generalized into the multi-armed bandit problem with the strategies as arms, and payoff as a function of jail time [7].

2.3.2 **Artificial Intelligence: Machine Learning**

Another popular example application of the multi-armed bandit in the Artificial Intelligence (AI) field is *reinforced learning*. Consider a scenario which you are tasked to develop a game of standard chess and also develop an AI for the player to play against. Even though chess has a finite number of possible states/moves, mathematician *Claude Shannon* estimated that the computer will need to store at least 10^120 (also known as the *Shannon-number*) [Shannon, 1950] states in order to obtain a perfect play on a chess game, which at worst result in a draw. *Shannon* estimated that it will take 10^90 years to fully solve chess on a 1 MHz processor. However, with the advancement of technology and research, some AI can play chess almost as well most expert players. A modern chess engine, Giraffe, can achieve a top score rivaling top chess engines and chess experts by being exposed to game logs and self-play within 72 hours [22].

Most chess engines rely on a brute force search strategy which searches all possible valid moves within the next few steps and selects the most optimal move. There are a number of open source chess engines which improvises machine learning by analyzing sample inputs and uses algorithms to learn, develop models and make predictions.

The adversarial multi-armed bandit allows algorithms to learn new knowledge for each action chosen, with the opponent (referred to as the adversary) providing the payoff and regret. The multi-armed bandit algorithm then uses prior information to optimize its future choices, after it has explored more options [13].

2.3.3 **Clinical Trials**

Another common application of the multi-armed bandit is on healthcare and drug testing [10]. To test various treatment methods or drugs, a clinical trial is used to determine how it compares to previously known or tested methodologies. Given prior treatment methods, analyst have knowledge on their effectiveness and are able to rank them according to how efficient they are as remedies to patient illness (which would be quantified as the "payoff"). Suppose that researchers propose a new type of drug, clinics might consider "exploring" this drug and its effectiveness. Since the drug is new, analyst has no history of its actual effectiveness, and will want to observe the result. Alternatively, they could decide to forgo testing this new drug and "exploit" the best treatment they have tested and have knowledge on, based on the budget constrain (how much the patient can afford)

The Gittins Index [9] has also been commonly used to minimize the number of clinical trials required to access effectiveness of new drugs. It is also worth mentioning that there have been ongoing ethical issues about administrating experimental drugs to patients by placing their health at risk, but at the meantime possibly find a better treatment than what the healthcare industry already has [10].

3 Measures and Constraints of Multi-Armed Bandits

There are many variants of algorithms for the multi-armed bandit. However, they share the same key idea of finding the optimal parameter and minimizing regret. For demonstration purpose, consider a video game designer who just moved to a new city and is tasked to design the box art of a new video game. The video game designer currently has 5 different designs in mind but is unsure which one he should pick to attract the most customers and maximize game sales. Assume that he does not have any prior knowledge or data of the gamers in the city, and he is unable to afford spending a few weeks to research further, because game competitors might release a better game anytime. The store policy is that game case design

can only be changed at the start of a new day. The designer decides to implement the multi-armed bandit model to determine the optimal variant.

A dilemma game designers face when selecting eye-catching box art for video games (Source: IGN.com)

The horizon, H, is defined as the amount of rounds or time periods that can be played in the multi-armed bandit. This value might be finite or infinite, a known integer or unknown variable. The number of rounds or iterations of the process that have been completed so far will be denoted as n, where $n < H$.

3.1 Arm

Similar to the slot machine example, the arms refer to the array of choices or parameters, and the goal is to find the optimal. Denote K as the number of arms to choose from with a reward probability distribution $(v_1, v_2, \dots, v_k)$ associated with each arm. These distributions are initially unknown to the player. During time t $(t = 1, 2, \dots n)$, the player picks arm i and obtains a reward following the distribution v_i. Using the game designer example above, a possible definition of arms is the 5 different box art designs $(K = 5)$.

Arm	Background Color	Theme
1	Red	Volcano
2	Blue	Waterfall
3	Green	Nature
4	Blue	Ocean
5	Rainbow	Mixture of the above themes

Figure 1. Example of arms when selecting a game theme.

3.2 Reward

The reward refers to the utility obtained after observing an unknown outcome, either a quantifiable amount, or an evaluated value of satisfaction obtained after selecting an arm, performing the action and

reaping the result. There are many ways to calculate reward. The reward might be quantifiable, such as revenue obtained or game conversion rate after selecting arm i. However, if the reward is not quantifiable, such as the measure of happiness obtained or contribution to the society, there might be a need to compare it to other quantifiable values and define a consistent mapping function to transform it to a numerical value for computation.

Denote x_i as the payoff associated with selecting arm i of a stochastic multi-armed bandit process at any time, and $x_{i,t}$ as the payoff associated with selecting arm i of a stochastic multi-armed bandit process at time *t*. Since the reward distributions are random, consider evaluating the expected reward obtained from selecting arm i as μ_i. Thus expected reward of arm i: $\mu_i = E[v_i]$.

Arm	Expected value of associated arm
1	28%
2	25%
3	55%
4	52%
5	20%

Figure 2. Example of the supposed expected value of arms, these value are unknown by the player.

Following the example above on selecting a character design, the reward can be quantified as the percentage of customers who shows interest in the game and purchase it. Note that if the horizon is relatively large, and lack initial data, it is common practice to allow the first few iterations to be evenly distributed amongst the arms, to get the initial data (seed) up front. This is illustrated in the first 5 rows of the table below, which allows each arm to be exposed once before running the algorithm.

Day	Arm chosen	Reward (% Interested Customers)
1	1	30%
2	2	24%
3	3	60%
4	4	55%
5	5	11%
6	3	58%
7	3	45%
8	4	54%

Figure 3. Example of reward. With epsilon greedy variant of multi-armed bandit, the algorithm picks the average best reward with a small chance of re-evaluating other arms.

3.3 Regret

In economics, opportunity cost is defined as the loss of potential gain by selecting an alternative. The regret, R, follows a similar concept. At a time t, the total expected regret is defined as the difference between expected total reward obtained by the optimal arm and the accumulated reward obtained using the MAB algorithm, over the time period $(t = 1,2,..,T)$.

$$\text{Regret at time T: } R_T = T\mu^* - \sum_{t=1}^{T} \mu_{i,t}$$

Where $\mu^* = \max(\mu_1, \mu_2, \dots, \mu_k)$, the expected reward from the optimal arm, and $\mu_{i,t} = E[v_{i,t}]$ is the expected reward from selecting arm i at time t.

In short, it is the quantified value of the additional utility you would have if you selected the best possible arm from the start instead.

Day	Arm chosen	Actual Reward	Expected Reward	Regret
1	1	30%	28%	0.27
2	2	24%	25%	0.57
3	3	60%	55%	0.57
4	4	55%	52%	0.60
5	5	11%	20%	0.95
6	3	58%	55%	0.95
7	3	45%	55%	0.95
8	4	54%	52%	0.98

Figure 4. Assuming that the optimal arm is arm 3, the regret can be calculated using data from expected value table and observed reward table.

Relating to the game designer example, possible measures of regret are the amount of players who would have been interested or purchased the game if the algorithm has selected the optimal game box art from the start throughout the horizon. The regret for the 8th day is computed as such:

$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{i,t}$

= current iteration count * E [distribution of best variant] - $\sum_{t=1}^{T}$ (expected value of arm chosen at time t)

= 8 * 0.55 – [0.28+ 0.25+ 0.55 +… + 0.55 + 0.52]

= 0.98

3.4 Types of Bandit

3.4.1 Stationary Bandit

The strongest assumption of many statistical models, including a large number of multi-armed bandit models, is that the distribution and parameters are static; they do not change over time. This might not be a reasonable assumption for a number of problems in reality as many factors evolve over time. New trends are discovered each day and die down without warning. Consumers' taste and preferences are hard to model and predict and might change without any indication. However, on the other hand, complex, rapidly-changing models and distributions have unsatisfactory performance on fixed policies.

Algorithms for stationary bandits are simpler to understand. Even though there is still randomness contained within the reward distribution, with enough sample and exposure, there will be sufficient

information to confidently estimate the expected value of the distribution. This makes it easy for the algorithm to adapt its strategy and ultimately converge to the desired optimal solution.

3.4.2 **Restless Bandit**

In a restless multi-armed bandit problem, the reward distribution function changes over time. An arm which is believed to perform well for a few iterations might change its behaviour and perform poorly after that, relative to the other arms. On the other hand, a poorly performing arm might have drastic improvement over time. Algorithms used for stationary bandits will not perform well as they assume that the expected value of the arms remain unchanged.

This makes problems involving dynamic bandits more complex as it has to constantly re-evaluate its explored and exploit phases. In fact, the exploration versus exploitation trade-off might not be optimal for dynamic environments [11]. In restless bandit problems, the problem has to be evaluated as a set of stochastic process, considering previous reward and determine the next arm to select. It is also critical to apply the appropriate statistical tests for detecting environment changes in abruptly changing environments, such as the Page-Hinkley test (PHT) [12] which is a sequentially used analysis technique for monitoring change detection.

3.4.3 **Adversarial Bandit**

Some multi-armed bandit algorithms such as the UCB1 provide near-optimal solutions to decision problems by "optimism". Realistically, these payoff structures are naïve especially for problems which are competitive, such as the stock market [13]. Instead of the rewards being selected from the fixed distribution, the result obtained is the worst possible payoff selected from the adversary.

In adversarial bandit problems, the adversary first picks the reward distribution before the player gets to choose an arm and obtain the reward. Adversarial multi-armed bandit algorithms such as the exponential-weight algorithm for exploration and exploitation (Exp3) are gaining popularity. The Exp3 algorithm maintains a vector of weights for each action taken, and uses these weights to randomly determine the next action. It then increases or decreases the weights depending on its performance [14].

3.5 Advantages and Disadvantages of Bandit Applications

Consider a case which the optimal variant performs below expectation and given less weight, or if an inferior arm happens to perform above average during the initial stage of the trial, all arms will still have a chance to be re-evaluated and re-considered. If the inferior arm has been selected a few times, its

observation count will increase, with its observed reward slowly decreasing. This causes the arms with less weight to be re-evaluated and given more weight.

Furthermore, unlike the popular A/B testing, which requires constant supervision and modifications, the MAB algorithm is adaptive and performs continuous optimization, and could be run continuously without further input or modification. It will eventually move traffic towards winning variations by the extra data collected on high performing variants.

Additionally, there is no "testing phase" in MAB algorithms. As long as there is some initial seed data, it will proceed to find the optimal arm, with effectiveness depending on the number of trials it is allowed to iterate through. This minimizes opportunity cost of data collection which is incurred for testing phases of some other algorithms. It also minimizes regret, which is the ultimate objective of MAB algorithm.

However, the multi-armed bandit is not without its flaws. For problems which the reward is dependent on various external variables, it might be wise to consider if implementing multi-armed bandit is appropriate. An example is the study of optimal learning environment to improve the grades of students. Suppose the MAB algorithm currently favors "group How learning" instead of "self-study". Do that imply that "group learning" is indeed the optimal choice? It might be attributed to the fact that the students have matured over the period of time and the expected grades for both study methods have improved something which was not factored into the algorithm.

4 Multi-Armed Bandits Algorithms

4.1 Epsilon-First

The epsilon-first (ε -first) algorithm explores opportunity for a set number of runs before repeatedly selecting the best arm to exploit.

First set a value for epsilon (ε). Given N trials, the explore phase is executed by trying (ε*N) arms, during which the lever is selected randomly from the arms using a uniform distribution. Each arm and its associated payoff are recorded. The remaining (1- ε)*N trials will be the greedy, or pure-exploitation phase, during which the algorithm always pick the lever with the best payoff.

The ε-first algorithm is preferred when the horizon is fixed and when the arms are guaranteed to be stationary, since exploration/learning component is performed at the start of trial. As such, it might fail to explore potential arms which give a significantly higher payoff. If the algorithm did not get to pull the

lever for an unexplored arm during the explore phase, or if the selected arms does not perform as well as expected, it might eventually result in large regret.

4.2 Epsilon-Greedy

The epsilon-greedy (ε-greedy) is equivalent to weighted average with ε% chance to explore and (1- ε) % chance of being "greedy" and exploiting the best arm.

Starting the experiment without any data, allocate a few trials to obtain seeds (initial values) for the arms. Given N trials, the algorithm will explore arms with probability ε (with uniform distribution). With probability (1- ε), the algorithm will analyze prior data and pick the lever which has provided the largest expected payoff.

The ε-greedy algorithm is a significant improvement from the ε-first algorithm as arms which might have poor performance due to bad luck, will still be given a chance to be examined again. Thus, if sample size is large enough, it will be possible to explore all arms. The algorithm discussed above treats epsilon as a constant, thus it is guaranteed to behave randomly ε% of the time throughout the entire horizon. This is preferred for smaller samples, or when there is limited time to perform the experiment.

Alternatively, many algorithms also uses ε-decreasing algorithm. These strategies follow the exact same logic by exploring with probability ε and exploiting with probability (1- ε). However, the value of epsilon will decay over time, eventually converging to zero (ε = 0). This can be implemented by variance weighted strategies or using a monotonic decreasing function (possibly logarithmic) based on some observation or over time. An example of an epsilon-decreasing algorithm is the GreedyMix algorithm [14].

4.3 UCB1

The reward obtained after selecting an arm is prone to randomness due to variance. This causes the result to differ each time the arm is pulled. As such, there is a probability of a "bad draw", where the reward is much lower than the expected value μ_i. Instead of only considering exact calculated values, the UCB1 (Upper Confidence Bound 1) evaluates upper bounds on the rewards observed. It is known as an "optimistic" algorithm as it treats arms which have been played less as more uncertain compared to commonly played arms. The UCB1 algorithm initializes by playing each of the K arms once, to obtain an initial seed to work with. For each of the remaining trials, the algorithm uses the following equation to evaluate $UCB1_i$ for each arm and pick the arm with the largest associated value.

$$UCB1_i = \bar{x}_i + \sqrt{2 * log \frac{t}{n_i}}$$

where n_i represents the number of times arm i has been selected so far, and $\bar{x}_i$ represents the average observed reward so far for arm i.

Note that the UCB1 algorithm is only appropriate for Bernoulli arms, which means that the quantified reward measure has to fall between the range [0, 1]. Additionally, unlike the algorithms described earlier, the UCB1 algorithm is not a semi-uniform strategy, which means it does not differentiate explore and exploit phases.

4.4 UCB2

In an attempt to improve the UCB1 algorithm, the UCB2 (Upper Confidence Bound 2) algorithm manages to further reduce the overall regret, with the trade-off for a more complicated algorithm. Similar to UCB1, for each trial, the algorithm uses the formula to compute $UCB2_i$ for each arm and pick the arm with the largest associated value.

$$UCB2_i = \bar{x}_i + \sqrt{\frac{(1+\alpha) * \ln(\frac{en}{(1+\alpha)^{r_i}})}{2 * (1+\alpha)^{r_i}}}$$

Where *r* is the number of epoch arms *i* that has been selected and $0 < \alpha < 1$ is an adjustable parameter which influences the learning rate [15].

Furthermore, unlike UCB1, which only conducts one "pull" with the optimal arm, the UCB2 algorithm plays the optimal arm exactly $\lceil (1+\alpha)^{r_i+1} - (1+\alpha)^{r_i} \rceil$ of the time, before completing the iteration.

5 Multi-Armed Bandit Conclusion

The multi-armed bandit is an "earn while learn" strategy which improves over time. In a nutshell, the algorithms can be summarized in 3 steps. The first step is using the selected algorithm to decide on an arm (pick a lever). The action is then executed and result is obtained (get reward). After obtaining the reward, the third step is to use the result to provide feedback to the algorithm. There are different variants to select from depending on the problem scope and constraints imposed. Each algorithm has its own benefits and limitations.

Theoretically, the multi-armed bandit will converge to the optimal arm if the number of trials is large enough. For small sample sets, no statistical algorithm will be able to provide convergence with 100% certainty. However, the multi-armed bandit aims to minimize regret, and have been tested to work well.

6 Game Analytics and Playability Optimization

Game analytics is the process of discovering and communicating patterns in game data obtained from players, in hope of using this information to improve the current game version or enhancement for future game software. For the video game industry, it is crucial to find ways to develop games which appeal to the target audience, minimize frustration and keep their interest for as long as possible. Over the years, video games have improved not only in design but also added new concepts and a more complex Artificial Intelligence (AI).

The role-playing-game Legend of Zelda which was released in 1987 (left) (Source: www.dailymotion.com).
A remake of the same game with additional features and enhanced graphics is anticipated to be released in 2016 (right) (Source: www.gamersnet.nl)

6.1 Data Collection Methods

Gameplay data are mainly collected through telemetry. This section will discuss how various game data are collected for games which do or do not require an internet collection.

6.1.1 **Offline-Games**

It might be tricky to collect user data from video games which do not require an internet connection. One way to evaluate user experience for these games is to collect feedback from the gamers in the form of a questionnaire or interview. Incentives might be given to these gamers who provide feedback in the form of various in-game items which the player can redeem by entering a serial-code or password in-game. However, with the popularity of online game discussion boards, the code to redeem the virtual reward is

easily shared online. Other players can easily obtain them without going through the survey. A solution for this is to allow these passwords to be unique one-time use codes, such that the code cannot be redeemed by more than one player. However, this would require an internet connection to verify the code and to flag it as "used", preventing it from being used by another player.

Some offline games have online leaderboards for players to upload their top scores. The player has an option to share their high scores with others and compare it with other players. Players can aim to get the highest overall score, or aim to get the highest score within a certain time period in order to obtain exclusive in game items, as well as intrinsic reward.

In PlayStation-Vita game Disgaea 3, players are encouraged to share their top scores to the leaderboards, and will be rewarded with in-game items.

Downloadable contents (DLC), limited time in-game items and patches are developed to fix known bugs and improve gameplay experience. These contents require the player to have an internet connection to get the updates. When these data are being downloaded from the server, some game metrics might also be collected.

The above methods encourage the user to connect to the internet to claim their reward. Meanwhile, if the user plans to enable internet connection with the device, it is possible to get more gameplay data through telemetry. Gamers enjoy getting rewarded for their action. Many offline games, especially console games, have implemented a game trophy feature which allows the gamer to acquire various shiny trophies to their collection if they have completed a certain designated objective. These objectives include easy to obtain trophies, such as "Complete the Tutorial stage", or difficult to achieve trophies, such as "winning 30 ranked matches online." These trophies act as a form of achievement and can be viewed at their pleasure or to share them with friends. Rankings and leaderboards were also implemented to compare their trophy count with other players. Gamers who are competitive or perfectionist will aim to collect as many trophies as they can. Unknown to many gamers, gameplay data is collected through telemetry when the

user syncs their trophies with the server to update their online profile. While the trophy history data is being sent to the server, gameplay data is also being uploaded to the server.

Sony PlayStation gamers have the option to customize and share their personal gaming profile, showcasing their medals and achievements (Source: psnprofiles.com)

6.1.2 **Online-Games:**

Most modern games require an internet connection. Massively multiplayer online games (MMO) are games which require the player to interact with other gamers in a virtual environment. MMOs and mobile-based social games are examples of games where the user needs to connect to a server and have the ability to interact with other players. Since game data are being sent to and from the server, it is easier to collect game data from users. However, these games are more complex as it has to factor in player social behavior and the detection of hackers who might negatively affect the community. As such, the challenge with online-games is the choice of data to be collected and analyzed, instead of how they are being collected.

These games may collect more data on the player by providing an in-game incentive for them to link their account to their *Facebook* or other social media profile. Not only does linking their *Facebook* account allow them to easily retrieve their game account in case they are unable to log-in, they might get exclusive virtual rewards from the game software by doing so. In exchange, the game company might be able to obtain personal information from the player such as their age and location listed on their *Facebook* profile. With this information, they are able to build player profiles and conduct study on game behaviour, preferences and spending habits.

In the mobile social game Pokémon Shuffle, players have the option to link their game account with Facebook.

Additionally, most MMORPG have game masters (GMs), who are game moderators acting as law-enforcement personnel in the game community. These GMs are readily available to obtain feedback on bugs and hackers from the players, and are responsible to verify and contact the team to work on a solution to the problem.

6.2 Game Metrics

Gameplay metrics are the measure of player behavior, or "breadcrumbs" of player actions, such as the amount of time spent on a stage, cause of death, or number of times a player replays a stage. It is the quantitative measure of at least one attribute that operate in the game. There is a broad range of game metrics to collect, each with different degree of importance depending on analysis goal and game type. This section will briefly discuss examples of game genre and commonly used metrics for each of these games types.

6.2.1 **Platformer**

A platformer is a type of video game in which the player gets to control an avatar and traverse through a 2D or 3D space, avoiding obstacles and reaching a destination. These games usually challenge the player's reflexes as well as familiarity of the stage, with difficulty level scaling after each stage. An example of a well-known platformer is *Super Mario Bros*. Some game metrics which the analyst could

extract from the players include the number of jumps made by the character, time taken to complete each stage, and death locations.

Super Mario Bros is an example of a platformer which requires the player to control an avatar and reach a specific destination. (Source: game4v.com)

6.2.2 **Role-Playing**

A role-playing game (RPG) is a genre of game which allows the player to customize and control the protagonist. RPGs are generally story heavy, and the player will be able to role-play as the main character following narrated instructions and objectives. These games are targeted towards gamers who like customization and freedom to explore, as well as problem solving and completing objectives through various means. RPGs usually involve several boss stages, which are challenging puzzles or enemies the player has to overcome in order to proceed with the game. These boss stages help to gauge player's progress and help them be ready for upcoming stages. An example of a well-known RPG is *Pokémon*. Since RPGs are generally progression and quest based, possible game metrics analysts could consider include character progression, number of failed attempts, number of scenes skipped, and question completion time.

An example of completing a quest in RPG game Pokémon. (Source: pokemondungeon.net)

6.2.3 **First Person Shooter**

First Person Shooter (FPS) Games simulate a gun fight in first-person perspective. The player controls the protagonist and has to navigate stages while defeating enemies with the weapon. Many of these games also fall under role-playing games as the player has to follow objectives and clear a stage by performing several tasks. Since FPS games rely on the player's reflexes and decisions made, useful game metrics include weapon selected, hits to shots fired ratio, and death spots.

In FPS game Team Fortress, the player controls the character in first-person view and has to defeat enemies before they do the same to you. (Source: teamfortress.com)

6.2.4 **Mobile Social Games**

Mobile social games are growing in popularity due to the availability of smartphones and game simplicity. Gamers of various skills and background can easily download and jump right into the game without being constrained to playing it on a console or computer. As added incentive, most mobile games are free-to-play, with optional in-app purchases for gamers to achieve the full experience or to gain competitive edge in the game. These free-to-play games such as *Candy Crush* rely on in-app purchases for revenue. Since these games focus on player interaction and game recommendations, analysts are interested in modelling the player profile by their average session game time, spending habits (both virtual currency and in-app purchases), interaction with other players and game progression. With these data, analysts aim to maximize retention rate, the measurement of how effective the game is at keeping the user interested. Analysts will also want to study trends or features which will affect conversion rate, the rate of converting non-paying users to paid users.

Free-to-play mobile social game Candy Crush allows players to compare their scores with friends and leaderboard. (Source: bgbox.com)

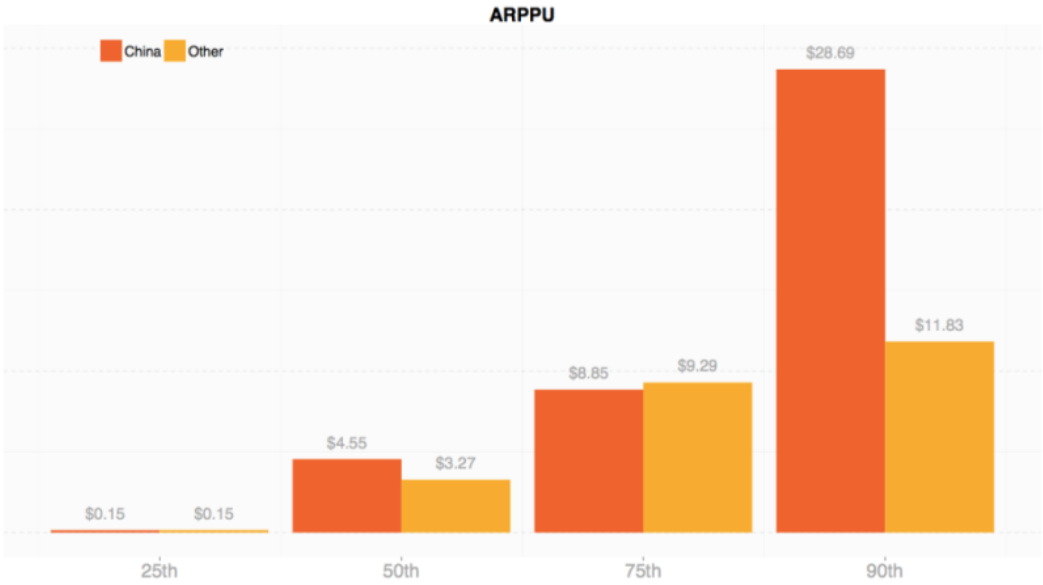
6.3 Mining the "gold" instead of the "rocks"

Consider a gold miner who wants to maximize the haul of the day. The gold mine contains an unlimited supply of heavy rocks and some gold ores. Within the time (8 hour work schedule) and space (storage container size) constraint, the miner will want to get as much gold and as little rocks as possible. Similar analogy can be used on data mining. Querying large amount of data will take a lot of time for data transfer and analysis, which is a waste of time and resources especially when some of these data will not be used at all. Furthermore, game data could easily go up to terabytes for more complex games.

Choosing the optimal game metrics is a challenging task, especially working with large data sets. The analyst has to brainstorm and select appropriate variables depending on their hypothesis or objective. For example, if the objective is to determine if a specific stage is too hard for players, the analyst could get player data on the number of retries for that stage and cause of death. Some data such as player's idle time or game audio settings have significantly less effect on the goal. With these data, the analyst can use statistical modelling and regression analysis to analyze spot trends and make improvements to the game.

A study of in-game behaviour of players from China made by Rosa Colom from GameAnalytics aims to compare the gaming and spending habits of Chinese players compared to players from the rest of the world [16]. The analysis is done on 205 million monthly active users spanning over 6234 active games.

Players are separated into two categories, non-paying users and paid users. Game metrics measured include retention rate, conversion rate, average revenue per paying user (ARPPU) and average revenue per paid user (ARPPU). These data are then graphed using business intelligence tools, which makes it easier to visualize the data for analysis.



The ARPPU chart above shows that paid users from China tend to spend significantly more compared to paid users from other countries. The study concludes that although it is harder to retain gamers from China, a game with good game experience will eventually allow conversion, with high probability of obtaining more revenue from the players in China compared to other paid users. [16]

6.4 Player frustration analysis

A game is primarily meant for entertainment. The consumer plays games to relax and destress. However, some games might cause frustration and stress for the players. Causes of player frustration include bugs, game crashes, and for some players, stages which are difficult to beat and long, un-skippable cut scenes or dialogues.

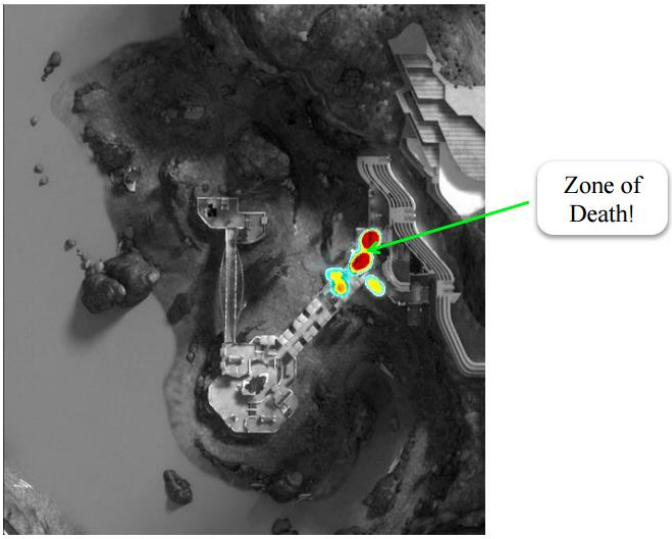
Consider a player who is enjoying a relaxing RPG after a long day at work. After a 10 minute long cut-scene which cannot be skipped, the game encountered a bug and crashes. Not only does the player have to reload the game, he has to sit through the entire cut-scene again.

Frustration can be measured by a variety of methods, with the easiest way being the physical actions of the player. Groaning, repeatedly pressing a button and abandoning the game totally are common signs of frustration. It is harder to measure frustration using telemetry. When the player has less interest in exploration, uses less strategy and performs suicide in the game, it might signal that the player is encountering frustration. Although this fact cannot be affirmed with 100% certainty with standalone data, it is possible to hypothesize the claims, or verify these suspicions using interviews, questionnaires or observations.

6.5 Data Analysis

Data analysis is the process of discovering and communicating patterns in data. Various modelling tools are used to analyse data obtained from players. The goal of modelling is to consolidate raw data, transform them into useful information, make it easier to spot trends and make it visually appealing and understandable to non-analysts in the form of tables and charts.

A commonly used visualization technique for games with maps is the heat map. A heat map is a graphical representation of data; it contains areas of various colors depending on the occurrence rate of a specific event. In the example below, data of players' death were obtained and consolidated them into a heat map. The colored spots highlight areas in-game where players have died. Heat maps use a spectrum of colours to denote severity, and in this case, the red color indicates the highest concentration of death. This might be caused due to a possible bug or difficulty spike. The analyst can investigate and make appropriate changes to the game or fix the bug by developing a patch which the gamers can download.



An example of a heat map , with the red patch showing an abnormally high concentration of deaths. [17]

6.6 Dynamically Adjusting Game Design

Many gamers do not enjoy repetition. In order to keep them interested, dynamic game difficulty balancing is introduced such that the game difficulty scales depending on the player's performance. With dynamic difficulty adjustment, players of varying skills can enjoy the game without the frustration of it being too hard or too easy. It also keeps the game fresh, as each play through might be different, encouraging multiple attempts.

With this game design, game parameters such as probability of an enemy executing a move, or difficulty of a task, are adjusted as users progress through the game. Upon reaching a certain milestone in-game, the program will evaluate the heuristics of a specific game state, and follow an algorithm to adjust the environment's parameters. This might be paired with probabilistic models to include further uncertainty to enhance the player's interest, making the game less predictable.

Adaptive AI is also used in some strategy games. These game agents obtain data through the player's actions and calculate a strategic counter to it. This, together with randomness of using specific strategies, will make the game challenging even to veteran players.

An example of game with dynamically changing content is Infinite Mario. The stage difficulty scales depending on player performance. (Source: themarysue.com)

6.7 Constraints and Ethical Issues

Getting game data from gamers is no different from being videotaped when playing a game, or being key logged (programs which records what the users enter on the keyboard, usually without player's knowledge). This might make some players feel uncomfortable, and raise privacy concerns.

It is imperative that the user should be informed that their game data will be collected, which a large number of video games having it listed in their "terms and conditions" page [18]. These data should also be kept confidential in a database. When users consent to having their game data logged, they expect that the company keep this information safe and secure. The company has a duty of care to ensure that these data do not get leaked. Data transmission should be encrypted to ensure that third parties will not easily tap this information. Personal information of players should also be kept confidential and should not be distributed.

Research has shown that player gameplay behaviours have a strong correlation to their real-world identity [19]. When enough data of a user is collected, companies can use profile-modelling to have a good estimate of the player's age, personality and behavior.

6.8 Sampling

For complex games with a large number of players, it is not feasible to log every single action of every single player as the search space grows exponentially with the number of variables in the data set. For example, large MMOs (such as World of Warcraft) or online social games (Candy Crush) have millions of users and telemetry data could easily reach terabytes.

A solution is using sampling techniques. These methods extract only a subset of the dataset, and also sample only a portion of the users. Although it is not as accurate as collecting and using all data, the space saved for data storage and time saved for complex analysis and computation is significant. This trade-off between accuracy and cost and limitations differs for each company, depending on their budget and goals.

6.9 Parallelization

Parallel programming is increasingly popular in computation and analysis of large datasets, saving significant amount time by dividing tasks into subtasks. Regular programs are run sequentially. One task has to complete before it can proceed to the next command. However, parallelizing a program allows it to be run concurrently and thus save significant amount of time by not waiting for other tasks to complete.

The basic idea of parallelization is that the main program creates multiple threads, which are distinct paths of execution, and each thread performs a specific task concurrently, instead of sequentially. Parallelization of a code might not be easy as the programmer might have to consider data races (multiple threads accessing and modifying data at the same time, causing errors), and might require more testing to ensure that program does not introduce more bugs. However, if implemented correctly, it can both improve time taken for computation.

Data analysis usually involves big data, with algorithms taking a significant amount of time, by using parallel techniques, valuable time taken to model the metrics can be reduced.

7 Analysis of Multi-Armed Bandit on Diamond Hunter

7.1 Introduction of the game

This study will test a multi-armed bandit approach on an open-source Java based game *Diamond Hunter* developed by an independent game developer Mike [20]. The gameplay involves role playing as the protagonist, a young man, traversing a maze and collecting all the diamond pieces. Initially, there will be obstacles in the map such as shrubs and lakes which will prevent him from reaching hard to reach areas containing diamonds. As the player progress through the dungeon, he will be able to find tools such as an axe or a boat to overcome these obstacles. This game is simple and short and players can expect to complete it in approximately 4 -7 minutes.

7.2 Study Design

The study process involves the participant getting a chance to play the game, followed by a survey online.

The volunteer is first given a URL to an online consent form which describes the study, procedures and disclaimers. In this page, he will be able to download the Consent Form and Certificate of Approval to his local machine. In order to proceed with the study, he has to click on the appropriate button to consent to the survey. The participant will then be redirected to the game page, which contains simple instructions on how to play the game. The player has the option to abort the game anytime.

The application will first connect to an online database, query values of interest and perform the selected algorithm to select a variant (game theme) which the user will experience. The gamer will continue playing until he completes the stage or closes the application. Game data will then be sent to a database before it terminates.

After enjoying the game, the player will then be able to complete a short, where he can provide feedback on the game, and the option of being notified of the survey results.

7.3 Arms

A lot of options are available to be used as arms to test. Some possible arms include the background music, player movement speed and game design. For this study, the arm will be set as the background theme, also known as ‘the skin’ of the game. The difference between the variants is mainly aesthetic, gameplay and difficulty remains unaltered.

The original game involves a theme of nature, with trees representing permanent obstacles and shrubs representing temporary obstacles. For this study, the arm with original theme will be referred to the forest theme. The current design will be replaced with a winter themed skin, with snowman replacing the trees. This will be referred to as the winter theme. A third theme, a fiery volcanic theme, will also be included to add variability, and referred to as volcanic theme.

Screenshot of two variants of the game to be tested, the original being the forest theme (left) and a modified winter theme (right)

7.4 Reward

Measuring player satisfaction would require consideration of a number of factors, such as game time, the player's score, or reviews from a survey. For complex games, it will be possible to extract more variables through telemetry from the player, but the search space, computation time of the algorithm, will grow exponentially with the number of variables in the dataset. It is important to study and only select variables which are believed to affect the player satisfaction.

Most gamers today tend to have a short attention span and limited time [21]. The study aims to maximize customer satisfaction, and have decided to use the player's in-game time to quantify reward.

7.5 Choices of Algorithm

Upon analysis of the multi-armed bandit algorithm to use, the UCB1 algorithm has been discounted, as it is not the optimal choice, requiring a good understanding of the underlying distribution and is also limited to Bernoulli Arms.

The decision was to use the epsilon-greedy algorithm. Since there is no prior data, the first 9 iterations are evenly distributed at random amongst the 3 variants in order to obtain some knowledge (seed). After the initial explore phase, the algorithm perform ε-greedy (with $\varepsilon = 0.15$) to pick a background variant to use. The game selects a background at random (explore) with probability 0.15 and selects the variant with the maximum average game time (exploit) with probability 0.85.

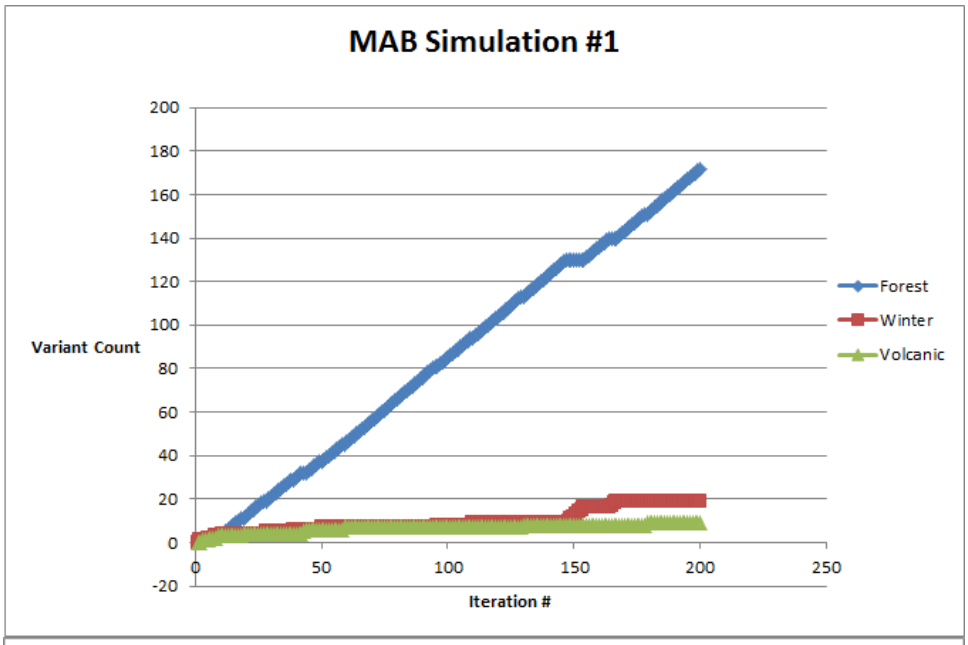
7.6 Pre-Test Simulations

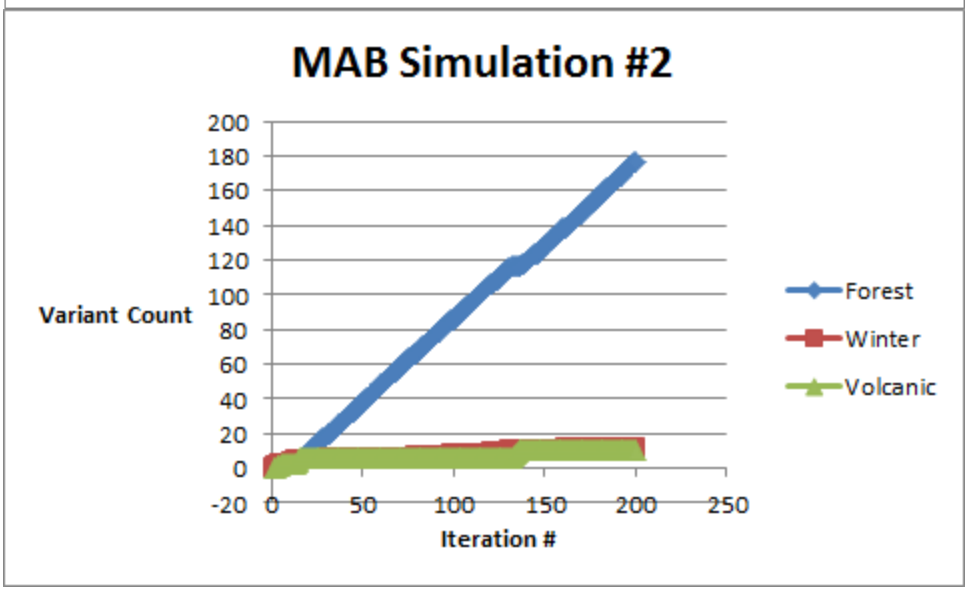
Prior to the actual study, simulations were performed to test the epsilon-greedy algorithm by using a script to simulate game time and to automate play. It is expected that the data will converge to the optimal variant given. A few simulation results are summarized in the form of graphs. Each variant is simulated using a Gaussian distribution with varying mean (Figure 5). In this simulation, the original theme (forest) has a slightly higher mean compared to the other two variants.

Theme	Mean	Variance
Forest	11000	5000
Winter	10500	5000
Volcanic	10000	5000

Figure 5. Simulated distributions for the arms

With a higher expected value, it is expected that the forest theme should be selected more often. The vertical axis (y-axis) denotes the number of times the variant has been selected and the horizontal axis (x-axis) denote the iteration number.


MAB Simulation #1
Variant Count
Iteration #
Forest
Winter
Volcanic


MAB Simulation #2
Variant Count
Iteration #
Forest
Winter
Volcanic

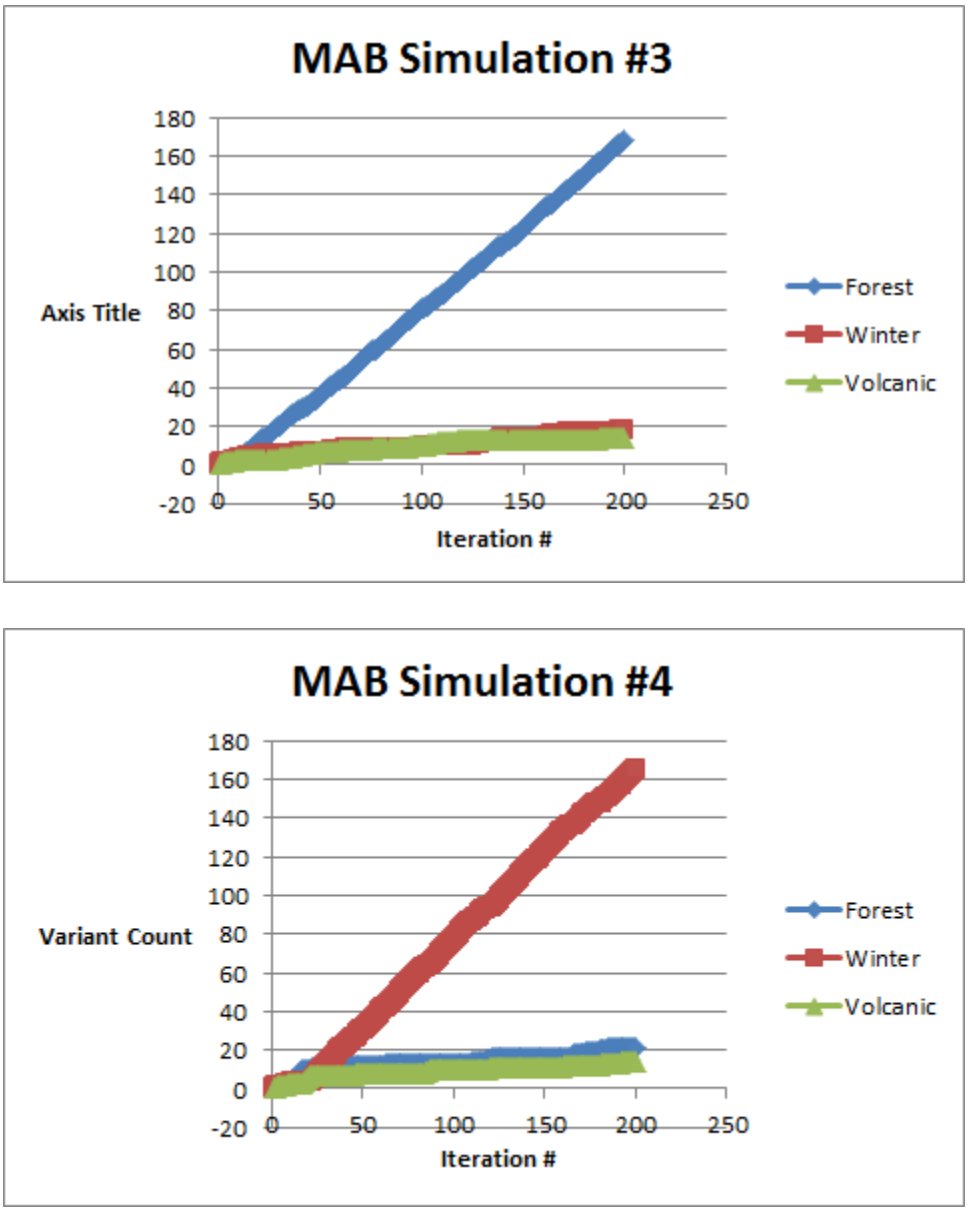


Figure 6. Graphical result of the simulations using epsilon-greedy algorithm.

It is observed that most of the iteration, the epsilon-greedy algorithm correctly picks the best distribution (forest theme) as the optimal arm. Also, note that the second optimal variant, winter theme, is consistently selected more often than the least optimal volcanic theme.

Statistically, according to the Law of large numbers (LLN), the algorithm will converge to the optimal variant after sufficiently large iterations. However, due to constraints, it might be difficult to obtain a large data set for this study. Using sample sizes of 200, the algorithm was repeated 100 times and the

number of times the forest theme is selected as the best arm is observed. The algorithm picked the favoured variant 81% of the time.

7.7 Live Study Results and Findings

Over a period of 20 days, 42 participants volunteered to take part in the study. In the figure is a graph of the frequency of the selected arm against the number of trials that have occurred. Since the study is started without any prior data, each arm is allowed to be selected three times to collect the seed up front for initialization. After 9 iterations, the multi-armed bandit algorithm then computes epsilon and optimal variant, using these values to determine the next arm to be shown to the user.

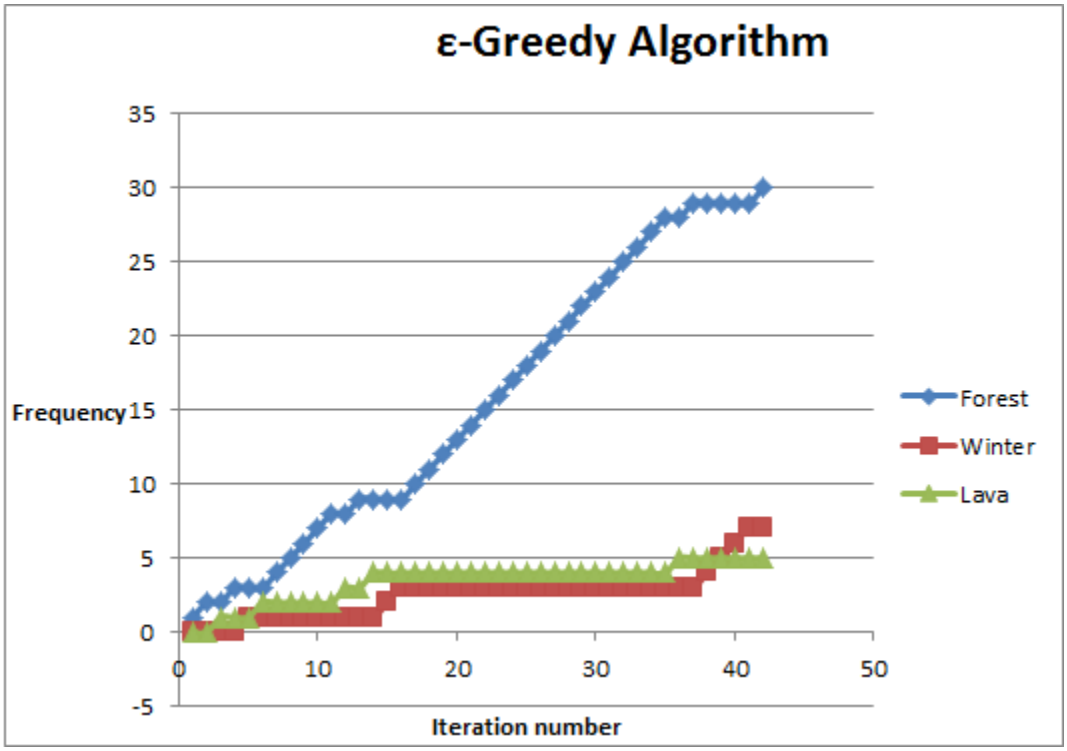


Figure 7. Results on application of epsilon-greedy algorithm on Diamond Hunter

As observed from the graph, the multi-armed bandit selected the forest theme as the optimal arm, which is selected 30 times out of 42 trials. This is no surprise as the forest theme is the original intended design made specifically for the game, whereas the other two variants, which are almost tied (with winter theme getting selected 7 times and lava theme getting selected 5 times) were only included for testing purposes.

	Forest theme	Winter theme	Volcanic theme	Total
Min of playtime	0.20	3.20	0.16	0.16

	Forest theme	Winter theme	Volcanic theme	Total
Max of playtime	11.81	8.02	13.39	13.39

	Forest theme	Winter theme	Volcanic theme	Total
Average of playtime	5.68	5.46	4.29	5.48

	Forest theme	Winter theme	Volcanic theme	Total
Count of playtime	30.00	7.00	5.00	42.00

Figure 8. Statistics of playtime for the different variants (in minutes)

When analyzing the data, note the extremely large variance amongst playtime. This might be caused by the fact that video games are subjective and not all players will enjoy it, or play long enough to complete the stage. Since the variants only differ aesthetically without any difference in difficulty, the average time taken to complete the game should be fairly close. From the data, the average time played is approximately 5 to 6 minutes. This falls within the expected game length of (4, 7) minutes. With this, it is fair to conclude that most players are able to complete the game. It can also be seen that even though the maximum recorded play time is 13 minutes from the volcanic theme; its poor performance on subsequent plays balances it out.

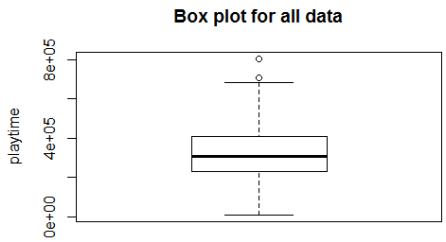


Figure 9. Box plot for combined data

By observing the boxplot for all combined data, notice the large variability and range, although the interquartile range is fairly small, discounting the outliers. A possible explanation for this is that while most players complete with game time within the interquartile range, a few players abandoned the game early, and a few players took a while longer to complete the stage, resulting in the outliers.

15 out of the 42 volunteers completed the survey. 21% of the participants who provided a feedback on the game length find it too short and the remaining 79% found the game length just right. Since no players complained about the game being too long, we could potentially increase the game length to better test abandon rate. Additionally, 3 feedbacks were regarding the game audio being too loud and would recommend an option to reduce it. It might be a good idea to allow the user more control over the game, such as volume of the game.

8 Future Works

Since the application of multi-armed bandit on game design and development is a new topic, there is infinite potential for discovery. First, different genres of games have their own characteristics. A good role-playing game requires an interesting story, a rhythm-based game requires a diverse, catchy soundtrack and a free-to-play mobile game requires features to retain players. It is also possible to consider exploring each genre further and select appropriate arms and rewards to build dynamically changing game to optimize user experience and revenue.

Dynamically changing games keeps players interested and improves player satisfaction and play time. Games with uncertainty and randomness and surprises grab both hard-core and casual gamers alike. Dynamic game balancing might be uncommon and hard to implement, but the multi-armed bandit can be potentially used as a solution.

It is also possible to test several variants of multi armed bandit algorithms such as UCB2, UCB-Tuned and the probability matching algorithm. By testing these algorithms and analyzing the results, we might derive new ideas for the application on game development. There might be possibility to conduct further study and come out with new algorithms which might be better catered to be used for game development.

9 Acknowledgements

I would like to thank my supervisor Dr. Ramon Lawrence for his guidance, suggestions and ideas regarding this project. I will also like to thank Giuseppe Burtini who provided valuable advice on multi-armed bandit. In addition, this project will not be complete without the original developer of Diamond Hunter, ForeignGuyMike [20], and his permission for me to modify the code and implement algorithms for research purpose.

10 References

[1] Dictionary.com (2012). Retrieved April 2016 from http://www.dictionary.com/browse/video-game

[2] Industry Facts. Retrieved April 2016 from http://www.theesa.com/about-esa/industry-facts/

[3] Violent Video Games Help Kids Manage Stress (July 2007). Retrieved from http://www.science20.com/news_articles/violent_video_games_help_kids_manage_stress-2511

[4] Video Game Controversies. Retrieved April 2016 from https://en.wikipedia.org/wiki/Video_game_controversies

[5] Multi-armed bandit. Retrieved April 2016 from https://en.wikipedia.org/wiki/Multi-armed_bandit

[6] Telemetry. Retrieved April 2016 from https://en.wikipedia.org/wiki/Telemetry

[7] Stian Berg (May 2010). Solving Dynamic Bandit Problems and Decentralized Games using the Kalman Bayesian Learning Automaton. Retrieved from http://grimstad.uia.no/ikt590/ikt10/g07/pdf/g7_report.pdf

[8] Solving Chess. Retrieved April 2016 from https://en.wikipedia.org/wiki/Solving_chess

[9] Richard Weber. Multi-armed Bandits and the Gittins Index Theorem. Retrieved April 2016 from http://www.statslab.cam.ac.uk/~rrw1/oc/ocgittins.pdf

[10] William H. Press (December 14 2009). Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research. Retrieved from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2793317/

[11] C. Hartland et al. Multi-Armed Bandit, Dynamic Environments and Meta-Bandits. Retrieved April 2016 from http://www.win.tue.nl/~aboer/sdt/Hartland.pdf

[12]Albert Bifet. Concept Drift. Retrieved April 2016 from http://epia2009.web.ua.pt/onlineEdition/353.pdf

[13] Jeremy Kun (November 8 2013). Adversarial Bandits and the Exp3 Algorithm. Retrieved from https://jeremykun.com/2013/11/08/adversarial-bandits-and-the-exp3-algorithm/

[14] Paul Fischer et al. Finite-time Regret Bounds for the Multiarmed Bandit Problem. Retrieved April 2016 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.57.4710&rep=rep1&type=pdf

[15] Peter Auer et al. Finite-time Analysis of the Multiarmed Bandit Problem. Retrieved April 2016 from http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

[16] Rosa Colom (January 20 2016). GameAnalytics.com. Retrieved from http://blog.gameanalytics.com/blog/chinese-gamers-in-game-behaviour-of-players-from-china.html

[17] Game Analytics. Cornell University. Retrieved April 2016 from http://www.cs.cornell.edu/courses/cs4152/2013sp/sessions/15-GameAnalytics.pdf

[18] Thestar news (December 29 2015). Video game companies are collecting massive amounts of data about you. Retrieved from http://www.thestar.com/news/canada/2015/12/29/how-much-data-are-video-games-collecting-about-you.html

[19] Chong-U Lim et al. Understanding Players' Identities and Behavioral Archetypes from Avatar Customization Data. Retrieved April 2016 from http://people.csail.mit.edu/culim/lim2015understanding.pdf

[20] ForeignGuyMike. Diamond Hunter. Retrieved April 2016 from https://www.youtube.com/watch?v=AA1XpWHhxw0

[21] Why most people don't finish video games (August 17 2011). CNN. Retrieved from http://www.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/

[22] Matthew Lai (September 2015). Giraffe: Using Deep Reinforcement Learning to Play Chess. Retrieved from http://arxiv.org/pdf/1509.01549v2.pdf