Design and Implementation of Random Word Generator using Backtracking Algorithm for Gameplay in Ambrosia Game

Imam Kuswardayan
Department of Informatics, Information and Technology Faculty, ITS Surabaya

Ridho Rahman H.
Department of Informatics Information and Technology Faculty, ITS Surabaya

Nanik Suciati
Department of Informatics Information and Technology Faculty, ITS Surabaya

ABSTRACT

Playing games is one way to spend our free time. Game was created to entertain users who play it. One favorite genre for some gamers is a Role Playing Game (RPG). RPG is favored because players assume the roles of characters in a fictional setting. Game development requires new innovations to fulfill the market's requirement. One of the innovations in the game lies in the gameplay. In the RPG game that adopts Turn Base Strategy (TBS), improvised gameplay is very minimal. TBS system is focus in strategy, while for the action of each player is very little, that makes some novice players feel bored. Therefore, to solve these problems, the TBS gameplay need to improved. Ambrosia is RPG game with TBS gameplay that improved. The improvement gameplay is creating a word from random letters (Anagram) to attack the enemy. The randomization process is considering by the weight of the initial letter. And for extraction of words solution which can be created, is using Backtracking Algorithm. After testing this game, it shows that words solution can extracted by using Backtracking Algorithm. The improved gameplay also successfully attract users to play Ambrosia again.

General Terms

Human-Computer Interaction

Keywords

Anagram, Backtracking Algorithm, Improvisation, Role Playing Game (RPG), Turn Base Strategy (TBS)

1. INTRODUCTION

Playing game is one of way to spend leisure time. Game itself have been created to entertain users who play it. One genre of game that is very entertaining, fun and favourite of game lover is a Role Playing Game (RPG). Many people like RPG Game because players can customize and develop the characters in the game according the player's wishes. One of RPG game that is quite popular is the Pokémon series which developed by Nintendo [1].

Game development is rapidly increasing in line with growth of the people who love the game. All ages can be covered with many varieties of games that are on the market. However, new innovation is required to adjust the needs of the market. One of the innovations in the game lies in the gameplay. In the RPG game that adopts Turn Base Strategy (TBS), improvised gameplay is very minimal. Gameplay on TBS system focus about the strategy, while the action of each player is very small, making some ordinary players are bored in the early game. Therefore, to solve that problem, the improvement in TBS gameplay is needed.

Focus on this research is improvement new gameplay such as arrange random words (anagram) to attack the enemy. In this gameplay, the player must make a word as long as possible to attack the enemy. The longer words are arranged, the greater attack is given. Player can only arrange words in English. There are two main components in this gameplay, the first is randoming the words and the second is checking words. For randoming words, will be selected words from the database or reference English dictionary. When checking the word, there is two steps, creating and checking word list. Word list is some words that can be composed using random words before and come from English dictionary. Creating list of words is an implementation of the algorithm of Backtracking. After creating the list , the game will wait input from the player. When word of the player has been arranged, it will be checked using the list that has been created. This checking method is using Brute Force Algorithm. This game will built with C# language using Unity3D as Game Engine with Android SDK as the add-on so that this game can run on Android devices. For the filtering process in dictionary reference is made with C ++ language using Code::Blocks IDE. In this research will using approach of software system design.

Using this improvement is expected for beginner player who have never played RPG game does not feel bored and tired. In addition, the game is expected can train English language skills of the players.

This research is a research proposal which try to improve the methodology of playing anagram. We found previous research design waka (a Japanese poem) composing and playing interface "Iroha Pad" [11]. The proposed system supports users to have less literacy of Japanese literature for composing and appreciating a waka. Users can compose the original waka like playing a puzzle [11].

In this research, we use English words. There are two main components in this game. They are component for random the alphabets and component for checking the generated words. And we use backtracking algorithm to extract the words.

This is an implementation research. We validate our implementation research by testing the developed application based on its requirement. This validation research will be explained in the next section.

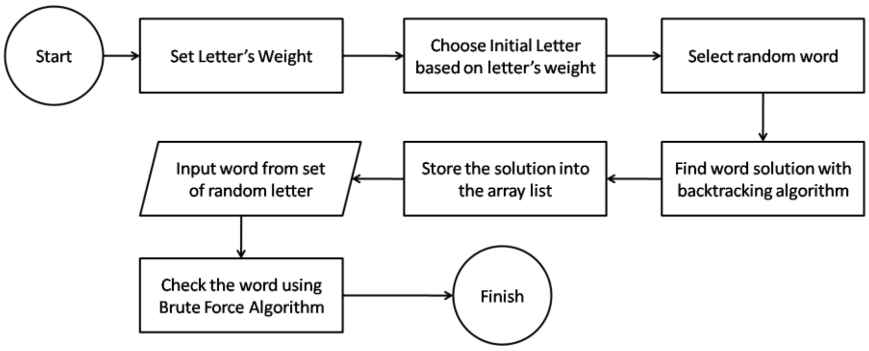
Start
Set Letter's Weight
Choose Initial Letter based on letter's weight
Select random word
Find word solution with backtracking algorithm
Store the solution into the array list
Input word from set of random letter
Check the word using Brute Force Algorithm
Finish

Fig. 1 Flowchart of randomization and checking word

2. LITERATURE

2.1 Backtracking Algorithm

Backtracking is Depth First Search based algorithm to find solution for problem. Backtracking is an improvement over brute-force algorithm that systematically only searching for possible solution [5]. Backtracking is a recursive algorithm that also available as iterative algorithm. It described in these steps:

1. Solution will be searched by generating a path from root to the leaf. The resulting node will be called living node.
2. If the path is not directing to the solution then the living node will be killed into death node. The function that will be used to kill the function is the limiting function.
3. If the last constructed node is a death node, then the search process will be resumed by creating another child node. If there's no more child node then the search will be backtracked to the parent node.
4. The search ends when a solution is found or there's no more alive node to be backtracked.

2.2 Brute Force Algorithm

Brute Force is a straightforward approach to find a solution of the problem. It based on the problem statement and the definition of the concept [6]. Brute force finds the solution of the problem by using a simple, straightforward and obvious way. The steps that will be taken by this algorithm is to process data one by one until n data that can be concured as its worst case scenario.

2.3 Roulette Wheel Selection

Roulette Wheel Selection is a selection method based on the value of the object [7]. This method use the percentage of every object. The value of the percentage will be adapted with the object value. Bigger percentage means bigger chance for the object to be selected. This selection concept is very similiar to a Roulette machine. This algorithm is written as these steps below:

1. Set the percentage or range of every object.
2. Randomize the number from the smallest value in the range until the last range.
3. If the result located within an object range. that object will be choosen.

3. DESIGN AND IMPLEMENTATION

3.1 Problem Analysis

Innovation in battle system on RPG genre with TBS mechanism is very little. Most improvement for that kind games only in story line or type of enemies that are more varied and increasingly flexible character customization. Moreover, a game with TBS mechanism is lack of action, so beginner players will be bored in the beginning.

Games that will be build will answer those problems by create a new battle system where player must compose a word in English to attack an enemy. Hopefully this game can help a beginner player to know fun aspect of RPG game and make player practice their ability to know the vocabulary in English.

3.2 General Description

Games that will be built is a game with RPG genre and TBS mechanism combined with educational elements, recognizing word in English. This game tells about someone who wants to find Ambrosia, the food of the gods. Someone who eats Ambrosia will have godlike power and could defeat a destructive god.

The main focus of this game lies in the mechanism of fight or gameplay. Players will be confronted with a set of letters, then the player is asked to make a word in the English correctly to attack the enemy. More longer the words that can be formed, more damage given to the enemy. In this game system, random letters are displayed to the player comes from random words taken from a dictionary reference, then a random word is searched for possible words that can be formed using Backtraking Algorithm. After obtaining a list of words that is formed, then the input word given form players will be checked using Brute Force.

There are some dungeons that can be played. Since this is a RPG game, then the player can do some customization. However, there is some limit to customize, such as only adds attributes but does not change the appearance. In addition, players can improve their character's level up to level 20. For the development of character and the difficulty in fighting the enemy is set automatically by the system.

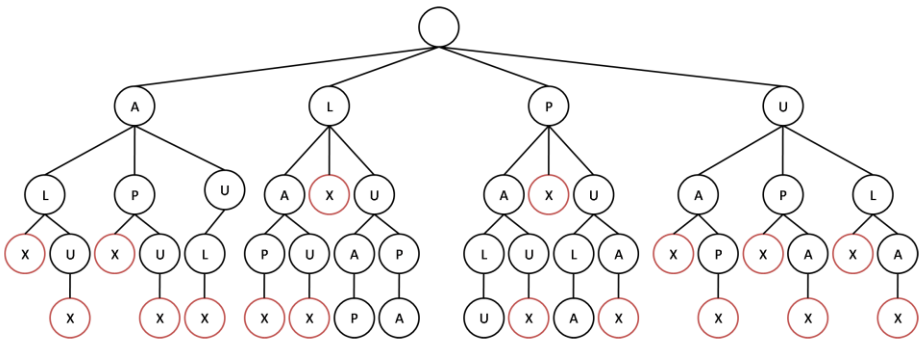

Fig. 2 Example of Backtracking Algorithm process

3.3 Reference Dictionary

Reference dictionary comes from a collection of English words derived by Infochimps [10]. There are 354 986 words. There's no plural word inside the dictionary. The words in the dictionary contain some acronyms and familiar names in English, along with past tense word. There will be filtering process which only take a word with a length at least 3 letters and a maximum 7 letters used in the game, and it will separating the dictionary into 26 sections based on the first letters.

3.4 Randomization and Checking Words

Focus of this research lies in the gameplay where there will be implementation of the algorithm Backtracking in checking word. The flow of the design of randomization and checking word is represented in Figure 1.

The first step is to determine weights for each letter. Weighting for each letter is based on points for each letter in a Scrabble game [11]. Every points that contained in a Scrabble game is assumed as appear frequency of a letter, so if the letter's points is low then that letter often to appear, otherwise if the points is high then that letter rarely come out. Points each letter can be seen in Table 1. After obtaining points each letter, then find the maximum value of all the letters by their frequency.

The next step is get the weight of each letter by dividing the frequency points of each letter with a maximum value. The formula to find the frequency of each letter can be seen in equation (1), while the formula to find the maximum value lies in Equation (2) and for a formula to find the weight of each letter can be seen in Equation (3). The weight of each letter is shown in Table 2.

Table I. Points of Each Letter

Letter	Points	Letter	Points
A	1	N	2
B	4	O	1
C	4	P	4
D	2	Q	10
E	1	R	1
F	4	S	1
G	3	T	1
H	3	U	2
I	1	V	5
J	10	W	4
K	5	X	8
L	2	Y	3
M	4	Z	10

$$frequency_i = \frac{10}{poin_i} \times 12. \quad (1)$$

$$MaxValue = \sum_{i=1}^{26} frequency_i. \quad (2)$$

$$Weight_i = \frac{frequency_i}{MaxValue} \times 100\%. \quad (3)$$

The next step is to determine the initial letters using Roulette Wheel Selection Algorithm, which will randomization based on the weight of each letter that has been obtained. In this process the letter with high weight has a chance of being selected higher than letters with low weights. After receiving the initial letter, the next step is to choose a random word based on the initial letter selected. Input from this process is the initial letters obtained in the previous process, and then reference dictionary representing the initial letter will be selected. Then select the word randomly with the range of the number of words in the dictionary of references that have been selected.

The next process is to find words solution that can be formed with Backtracking Algorithm. Input from this process is a word that has been selected in the previous process. For example, the word chosen is "LUPA". From that word, each letter will be turned into node corresponding to the alphabet, starting with the letter "A". From the node "A" and then go to node "L" as the child node. Then it is checked into a dictionary word with the prefix string "AL". If it is exist, then the search is continued by turning the node "P" and then checked again into the dictionary word with the prefix "ALP". If it is no existence, then node "P" will be disconnected and backtrack to the node "L". A solution is found if the string formed from the same node line with the words contained in the dictionary reference. The end of this process if all solution have been found and there's no line to created. Examples of words search process can be seen in Figure 2. The next process is any solution that found will be added to the array list.

Table II. Weight of Each Letter

Letter	Weight	Letter	Weight
A	8%	N	4%
B	2%	O	8%
C	2%	P	2%
D	4%	Q	1%
E	8%	R	8%
F	2%	S	8%
G	3%	T	8%
H	3%	U	4%
I	8%	V	1%
J	1%	W	2%
K	2%	X	1%
L	4%	Y	3%
M	2%	Z	1%

In the next process, players will enter a word based on a set of random letters displayed on the gameplay as the output of selecting word process. The word that has been entered by the players then checked with array list by using Brute Force Algorithm. Checking process will compare the string, and not comparing each character or letter.

3.5 Implementation

Ambrosia is built with C# language using Unity3D as Game Engine with Android SDK as the add-on so that this game can run on Android devices. For the filtering process in dictionary reference is made with C ++ language using Code::Blocks IDE. The interface on the main gameplay featuring two characters who are fighting, as well as the letter keys that can be selected to compose a word on the bottom. The interface is shown in Figure 3. Furthermore, if the word is composed by the players correctly, then the display interface is shown in Figure 4.

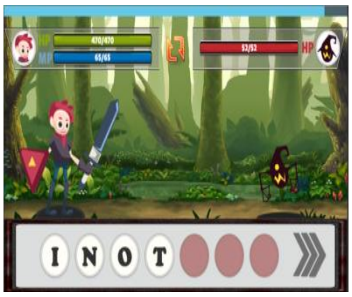

Fig. 3: Interface of attack with compose the word

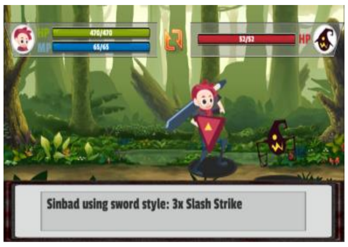
Sinbad using sword style: 3x Slash Strike

Fig. 4: Interface when attack success

4. TESTING

Once the game is build completely, the next step is testing the game. The test is to ensure the use of backtracking algorithm in finding a words solution. We use black box testing to test this application. This test takes 2 rounds of the game. There will be new random words in each round. Figure 5 displays the solution in the first round, while Figure 6 displays the solution in the second round.

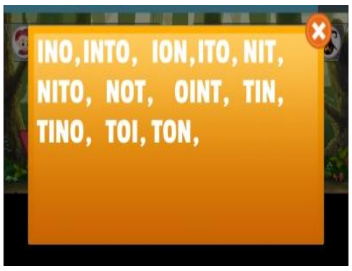
INO, INTO, ION, ITO, NIT,
NITO, NOT, OINT, TIN,
TINO, TOI, TON,

Fig. 5: Words solution that obtained in the first round

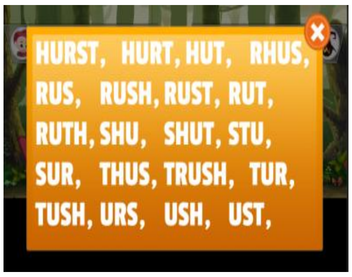
HURST, HURT, HUT, RHUS,
RUS, RUSH, RUST, RUT,
RUTH, SHU, SHUT, STU,
SUR, THUS, TRUSH, TUR,
TUSH, URS, USH, UST,

Fig. 6: Words solution that obtained in the second round

5. CONCLUSION

Ambrosia has been built applying randomization word that can be arranged in its gameplay. During testing, backtracking algorithm is capable to create a list of words solution based on the dictionary. For reference dictionary, Info chimps presents a pretty much word, but involves the acronyms and names in English that are not familiar to most Indonesian people. From the test result, this game is quite interesting and worth to play.

For the further works, another algorithm which can be used to create a list of word can be implemented. This research use English words. Another language can be used in the further works.

6. REFERENCES

[1] "Top 100 RPGs of All Time," IGN, [Online]. Available: http://www.ign.com/top/rpgs.

[2] I. Sommerville, Software Engineering 9th edition, Buston: Pearson Education, 2011.

[3] "Download Android Studio and SDK Tools | Android Developers," Google Android, [Online]. Available: http://developer.android.com/sdk/index.html.

[4] A. Levitin, "Backtracking," in Introduction to The Design and Analysis of Algorithms, New Jersey, Pearson Education, Inc, 2012, pp. 424-430.

[5] A. Levitin, "Brute Force," in Introduction to The Design and Analysis of Algorithms, New Jersey, Pearson Education, Inc, 2012, pp. 97-106.

[6] D. L. Adam Lipowski, "Roulette-wheel selection via stochastic acceptance," Physica A, vol. 391, pp. 2193-2196, 2012.

[7] "Code:Blocks," The Code::Blocks team, [Online]. Available: http://www.codeblocks.org/home.

[8] "Infochimps: Big Data - Cloud Services," Infochimps, Inc., 23 February 2012. [Online]. Available: http://www.infochimps.com/datasets/word-list-350000-simple-english-words-excel-readable.

[9] "Scrabble | Word Games | Board Games | Scrabble Online," Hasbro, [Online]. Available: http://scrabble.hasbro.com/en-us/faq. M. I. Assaat, "Aplikasi Algoritma Backtracking dalam Permainan Anagram," MAKALAH IF2251 STRATEGI ALGORITMIK, 2007

[10] Nishikawa. Naoki, "Iroha Pad: A Waka Composing and Playing Interface Using the Anagram of the Iroha Poem", second international conference culture and computing (culture computing), 2011, pp 153-154