Role-Playing Game Creation Using Procedural Generation

Dizon, Sam Louis

lhdizon@up.edu.ph

Favorito, Johanna Maria Antoinette P. jpfavorito@up.edu.ph

Mendoza, Jeffrey B.

Monserrat, Toni-Jan Keith tonijanmonserrat@gmail.com

1. Abstract

Up to this date, there are no computer role-playing games that are totally procedurally generated. Current computer video games still show preference for artist-rendered graphics and handwritten storylines. The proposed solution will take a procedural approach in creating a single-player role playing game with the main elements being the story, quests, and dungeons. The story will be broken into chunks and randomized, while the maps and dungeons will be generated using checkpoints formed from the story. The solution will be evaluated by calculating how much play throughs can be obtained from the generated content. Each play through scenario must be unique in some way to give the game more replay value.

1. Introduction

Computer role playing games today have now ascended to another level—life-like graphics and rich soundtracks add another dimension to the gaming experience. Complex games such as World of Warcraft and StarCraft provide players with a rich environment for exploration, quests, and interaction. These are important elements in making the games more appealing and enjoyable for players.

These elements, however, come at a cost. The story and plot are usually handwritten, they are planned and designed and are generally fixed, limiting the number of times a player can play the game without encountering the same scenario. Life-like graphics with sharp details take up much memory and need to be designed by artists, making the production process more expensive. Such games may take up several gigabytes of space because of the graphics that have to be stored and loaded when the game is played. One approach in lessening this burden (if not totally solving it) is the use of procedural content generation in making the elements of the game. Togelius et al. (2013) presented several reasons in favor of procedural content generation (PCG): first, it can provide variety, second, reduce development time and development costs, third, save space in transmission or on disk, fourth, extend human creativity, and lastly help games become more adaptable.

The proposed solution in this paper aims to make use of PCG to dictate how the game’s story would progress, and how the maps and dungeons would be generated. Story generation would be done by laying out the next event based on what the player chooses to do. To lend an element of randomness, some elements like characters and places would be swapped. Then, based on certain keywords from the plot, the map for the story would be generated. The quests will then be based on the generated map. The proposed solution may be used in creating learning systems that could aid teachers in teaching computer science subjects.

1. Related Work

Procedural generation has been explored by several people. Buck (2013) made use of a combination of fractal based-algorithm and a Markov chain matrix process to generate a world for his turn-based strategy/role-playing game entitled “Project Death and Taxes”. Content generated included land masses, terrain features such as localized forests, marshes, and rivers, start/end seeds, and political maps. Snodgrass & Ontañón (2014) also used Markov Chains but this time in the generation of maps. Using statistical patterns obtained from high quality human-authored maps, they train the Markov chains, then generate maps from those chains. The highly acclaimed video game known as “Minecraft” is well known for this (procedural generation) kind of gameplay feature. It generates a world in such a way that no world is generated twice unless generated with a use of a seed, seeds make it so that two different players with the same seed would generate the same map.

The Library of Babel is a popular site that is well known for its capability of procedurally generating every possible combination of strings of length 3200 with the use of 28 symbols plus the space which equates to 293200 unique possible combinations. Seeds and the block of string code can be generated back and forth by reversing the algorithm used in the site. A study from Hartsook (2011) procedurally generated maps based from player preferences and user or computer generated story. The story is in a format of a plot points, and these plot points are the main basis for building the procedurally generated world in their study. The world generated and the behavior of various non-playable characters depend on the plot points of the story. Meanwhile, Matthews & Malloy (2011) used procedural generation to build a map for a city or town based game. Their approach allowed for developers to specify the constraints of the town or city like distance, direction, navigation, and placement. Their approach also considered the context of the story to make sure that the generated map was appropriate for the story.

The method proposed in the paper differs in its medium of generation and display. The game will be a web application. Also, the maps and quests will be dependent on the progression of the story.

1. Methodology

The problem of low replay value in most video games nowadays will be solved by building the game contents such as the story, dungeons, and quests procedurally. Several stories will first be hand-authored and broken down into chunks. Each chunk will correspond to a certain ‘checkpoint’ for the plot. The checkpoint is a value that represents what visual elements are needed for that a particular scene (story chunk). These chunk-checkpoint pairs will be stored in a data dictionary for later use. The chunks will then be randomized to create an initial starting scenario. Players will then be given options to choose from on how they want to respond to the scene. The story will classify the player’s response as positive, neutral, or negative, and will use that as a basis for selecting what the next event will be. The story will progress based on the initial start point, and the player’s preferences.

Each checkpoint, which will be represented by a word, will correspond to a certain feature of the dungeon. When the player makes a decision, the checkpoint corresponding to the decision is selected. From the selected checkpoint, the corresponding features of the dungeon will be generated. This will ensure that the dungeon is different in some way for each play through, since the story may be different each time it is played. Similarly, main quests are also generated from the checkpoints. The values contained in each checkpoint would correspond to certain key features to form a quest. Side quests would be generated based on the previously created dungeon.

Each chunk of story selected will be processed by an algorithm that will take its corresponding check point value and generate the needed elements specified. How the story progresses lends to the randomness of the main quest and its components.

The dungeon map will be drawn using fractals pixel by pixel to form a 3200 x 3200 frame for each world that will be created.

1. Evaluation

Computer role playing games are typically valued for the number of hours a player can get out of the game. From linear storylines, a player can get one play through since the story only takes one path. Since the proposed method is procedurally generated and should generate a number of different scenarios, a player should be able to play the game a number of times before the scenarios are exhausted.

Twenty-five persons will be asked to play the game at least twice. They will be asked to rate the game based on the story’s coherence, visuals, and uniqueness of each play through. To evaluate the game’s replay value, there must be positive feedback based on the following criteria:

* Variability – the game must not be repetitive
* Player retention – players must be interested enough to play the game again
* Flexibility – the game must offer a certain degree of freedom to the players such that they feel they have an impact on the game and the game must be dynamic enough to present game plays

1. Timeline

2 weeks – creation of mini-stories

1 month – implementation for process of removing certain portions of the stories and replacing them to make them reusable and generation of checkpoints

2 months – implementation of algorithm that will make sure stories are coherent and logical

1 month – connection of checkpoints to dungeons and maps

1 month – algorithm for creating the map

2 months – algorithm for creating the dungeon and its features

2 weeks – integration of work

2 weeks – testing of the game/ survey

2 weeks – bug fixes and minor improvements

2 weeks – summary of results and integration of paper

1. References

Hartsook, K., Zook, A., Das, S., & Riedl, M. O. (2011). Toward Supporting Stories with Procedurally Generated Game Worlds, 2011 IEEE Conference on Computational Intelligence and Games (CIG), Seoul, South Korea, 2011. doi: http://dx.doi.org/10.1109/CIG.2011.6032020

Kremers, R. (2009). Introduction to Level Design. In *Level Design: Concept, Theory, and Practice* (p. 47). Florida: CRC Press.

Lee, J. (2014, November 26). How Procedural Generation Took Over The Gaming Industry. Retrieved September 24, 2015, from http://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/

Matthews, E. A. & Malloy, B. A. (2011). Procedural Generation of Story-Driven Maps, 2011 16th International Conference on Computer Games (CGAMES), Louisville, Kentucky, 27-30 July 2011. doi: http://dx.doi.org/10.1109/CGAMES.2011.6000324

Snodgrass, S. & Ontañón, S. (2014). Experiments in Map Generation using Markov Chains, 9th International Conference on the Foundations of Digital Games, Florida, 2014.

Togelius, J., Champandard, A. J., Lanzi, P. L., Mateas, M., Paiva, A., Preuss, M., & Stanley, K. O. (2013). Procedural Content Generation: Goals, Challenges and Actionable Steps. *Dagstuhl Follow-ups, 6,* 61-75. doi: 10.4230/DFU.Vol6.12191.61