Final Project

Design Document

Procreate Design Document

Author: Jared Schroeder CIT025989

Due: 12 Nov. 2014

Abstract

Glossary of Terms

Acronym	Meaning
LG	Level Generation
OS	Operating System
PG	Procedural Generation
PLG	Procedural Level Generation
WPF	Windows Presentation Foundation

Table of Contents

1. Introduction

This design document is for Procreate - a ‘procedural generation-based level creator’ application being developed by Jared Schroeder for the Final Project course in the Bachelor of Games and Virtual Worlds. The document first describes the differences between traditional and procedural level creation, and notes the history, strengths and weaknesses of each method. It then introduces Procreate, and proposes its great worth for 2D game developers creating modern game levels.

Following this, the functionality and features of the tool are detailed, which are then further embellished with application conceptual layout images and written walkthroughs.

The aim of this document is to introduce Procreate to the reader, help them understand the benefits that make it worth developing, and then give them a basic knowledge of what they can do with it, what it looks like, and how they can use it.

1.1. Traditional Level Creation

A game level is traditionally created by one or more digital artists, with the following steps:

1. Review the gameplay that the level will present to the player (the level's requirements)
2. Research and decide what the level will be like (what it looks like, what it contains, etc.)
3. Plan and design the actual level
4. Create the level with level editing software

This process achieves a level that is fully man-made – every part of it has been crafted by the artists. Levels created in this way contain a quality that is based on the amount of creativity, effort and volume of content that has gone into them. However, these three factors can take time to achieve, and some game developers may not have this time to produce high quality levels.

The time can be reduced by hiring more artists, however doing so might increase the cost of development, which might be unaffordable to developers with less money.

A game may have no digital artists working on it. In this case, non-artists have the difficult job of making game levels with a similar quality to games created with artists. Artists have more experience in their line of work, meaning that the levels they create will be of higher quality than non-artists.

1.2. Procedural Level Creation

Procedural Level Creation is a solution to these problems. Instead of using human artists, it uses processes completed by a computer to generate a game level. To do it, the following steps are followed:

1. Research and decide what the level will be like
2. Research, brainstorm and decide which procedural generation algorithms (processes) to use that are suitable for the level
3. Code and test the algorithms, to make sure they're working correctly
4. Tweak the algorithms to produce the desired level
5. Repeat step 4 (with or without tweaking) to produce as many levels as desired

Procedural level generation solves many of the traditional game level creation problems. Firstly, once it is implemented, it can produce a great variety and volume of levels in a short time period, that if done with artists, would require more time and money to produce. It also needs to be noted that a group of artists can only create a finite set of concrete levels (levels that don't change in the game) whereas procedural level generation is capable of making countless levels that could provide the player with a new level and therefore a new experience every time they play.

Secondly, procedural level generation can be achieved by developers with minimal experience with art or art creation programs, because they can provide the computer with a small amount of digital art and it generates the level for them.

Thirdly, while game level data created by artists need to be stored with the game to be reproduced, procedurally-generated levels can be created when the game starts, and deleted when the game quits. This can cause the game's software files to be smaller in size, and thus be quicker to download, transfer, install and uninstall, because the level files do not need to persist.

In summary, procedural generation has the capacity to reduce the cost and time required to create levels, it can increase game replayability by producing different levels each time the game is played, increases the throughput of game developers with little artistic experience, and can reduce the size of a video game's software files. These benefits have pushed procedural game level generation into the spotlight throughout the previous decade.

1.2. Procedural Level Generation Examples

A variety of modern games have used procedural level generation, which suggests that it is currently popular. A few examples have been provided below to describe the procedural method used, the results that it produced, and reasons why it was used. However the main reason for referencing these games is to show that procedural level generation is a relevant tool for making modern games.

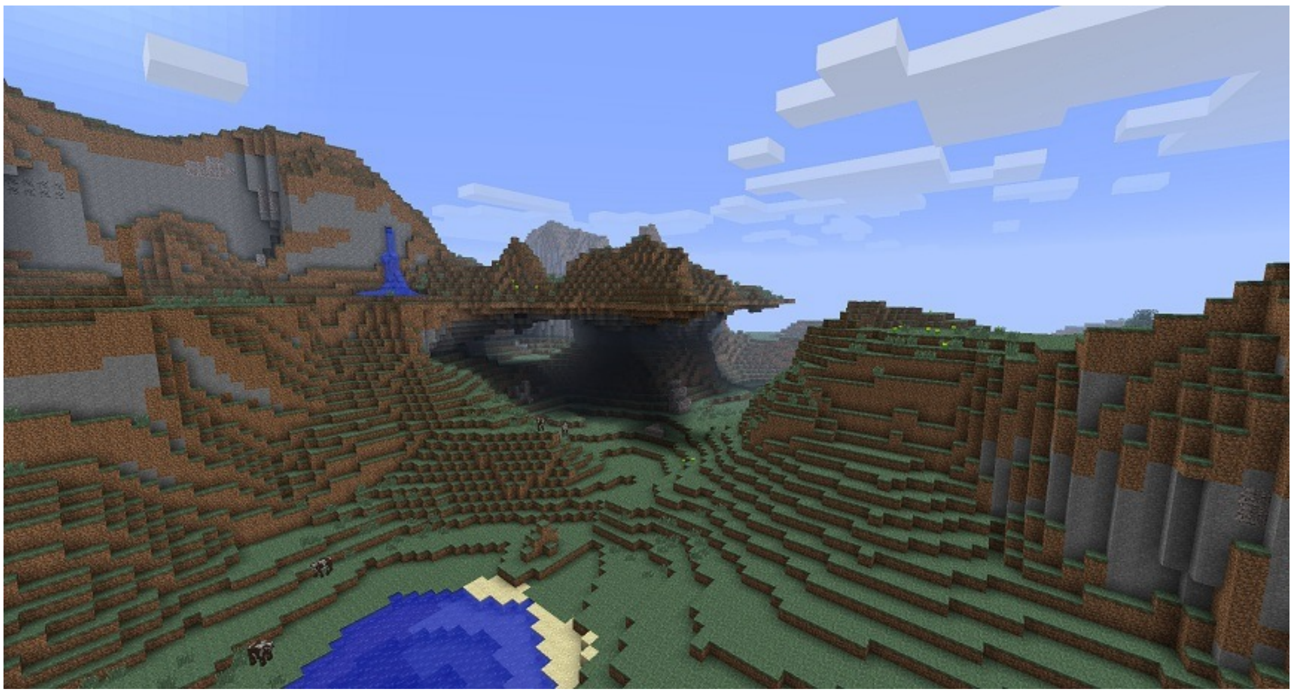
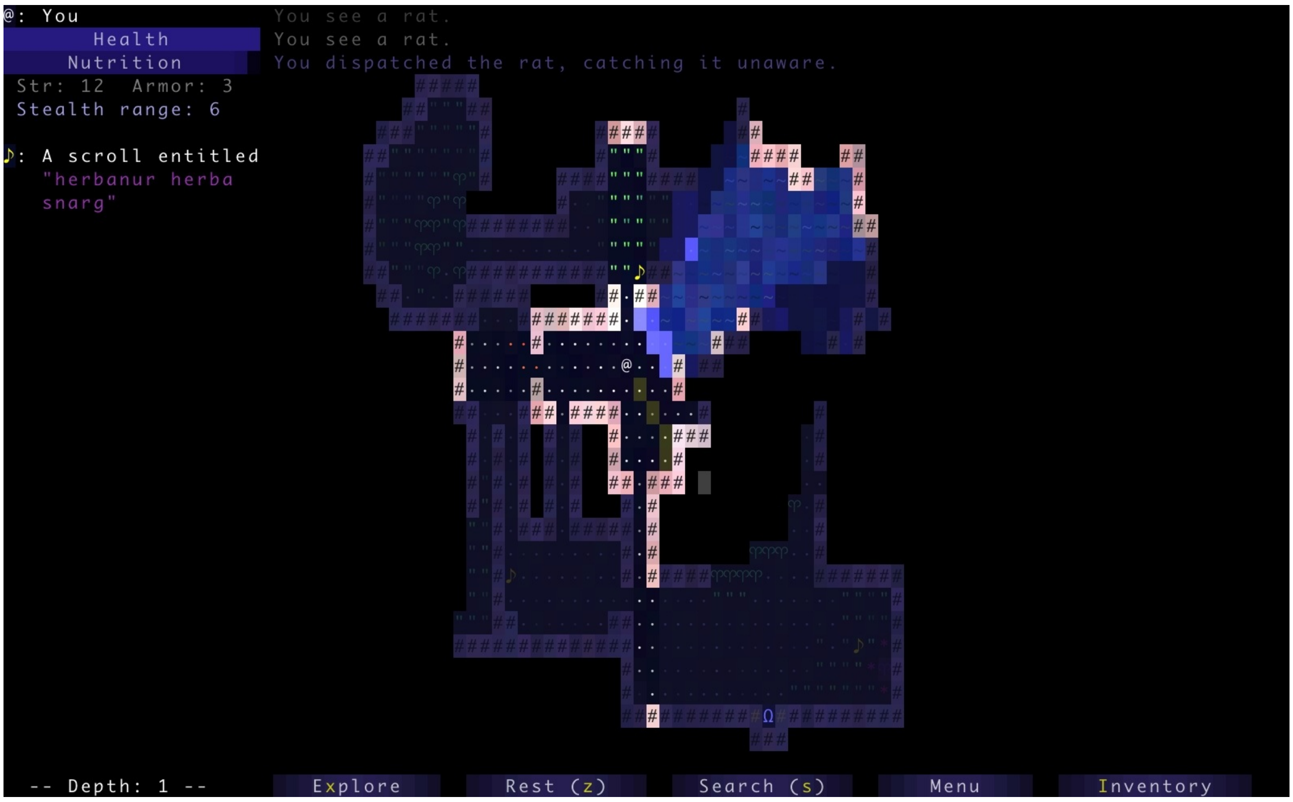


FIGURE 1: MINECRAFT (CLIFFE 2011) USES PG TO CREATE LARGE AND DIVERSE WORLDS TO BE EXPLORED (2011)

DON'T STARVE (DIRTYTABS 2014) USES NARRATIVE-BASED PG TO GENERATE VARIED PLAYER TASK-BASED LEVELS (2013)

THE BINDING OF ISAAC GENERATES A LARGE NUMBER OF VARIED LEVELS COMPRISED OF CONNECTED ROOMS (2011)



```
@: You
Str: 12  Armor: 3
Stealth range: 6

♪: A scroll entitled
   "herbanur herba
   snarg"
```

```
You see a rat.
You see a rat.
You dispatched the rat, catching it unaware.
```

'BROGUE' CREATES VARYING CAVES TO BE EXPLORED AND GENERATES ITEMS TO BE COLLECTED BY THE PLAYER, WITH PG TECHNIQUES (2009)

1.3. Problems with Procedural Level Creation

While procedural level creation provides many benefits, it also has some faults. These are detailed throughout this section.

The main problem with PLG is that it the levels it produces can eventually become meaningless to a player playing in them. This is because each level is very different, and doesn't have any meaningful elements that would usually be introduced by human artists creating the level. As such, a player can become used to the diversity in each level, and have no special experience with any of the levels.

Secondly, programmers implementing PLG in their game have to take the time to research the algorithms to use, code them, and then test them to make sure they're working . This takes time - a simple algorithm can be coded and tested in a day, however algorithms with greater complexity could take a few weeks to implement. Developers with less coding experience, for example artists, will require even more time to add a PLG technique to their game.

Thirdly, a game's PLG may require other development stages to have been completed. For example a game’s rendering system may have to be set up before a level generated with PG can be reviewed and evaluated. This means that PG levels cannot always be designed upfront.

Finally, if a game's procedurally-generated levels and level generation are embedded in the game code, they could be difficult to extract and reuse for other games. This is even more difficult if the game reusing old PLG code is written in a different computer language to the original game, because the code has to be perfectly translated from one language to another.

The Procreate application solves each of these problems, and is introduced in the next section.

2. Procreate Introduction

Procreate is a level editing program supplemented with procedural level generation techniques. It obtains the benefits of procedural generation, while solving the problems associated with it.

Procreate will be useful for 2D game developers who have lots, or no experience in level creation or procedural generation. It targets the market for games with procedurally-generated levels, with popular examples including Minecraft, Don't Starve and The Binding of Isaac. With Procreate, any developer with any amount of experience should be able to create a game that takes advantage of popular procedural game genres like RPG, roguelikes and dungeon crawlers.

Its only platform will be the Windows OS, because it is written in C# using WPF.

Alongside the standard benefits of procedural level creation, Procreate will supply 2D game developers with these additional benefits:

- **Meaningful procedural 2D game levels**: The program will provide ways of customising the PG and editing levels so that they can have meaning.
- **Saving time**: The program supplies the PG algorithms for the user, so they can avoid researching and coding the PG, and jump straight into the level creation.
- **No coding required**: The program will do all of the PG code in the background, so it can be used by both coders and non-coders, for example digital artists.
- **Level creation isn't dependent on other game components**: The program fulfils all requirements to review the level, so levels can be created at any point in a game's development.
- **Language-independent levels**: Levels will be stored in a format that can be used and reused by any computer language for any game, rather than just in the language that a particular game was written in.

3. Functionality

Procreate consists of various types for functionality. Everything that Procreate is capable of doing (i.e. the functions) is listed below - grouped according to the relevant domains and in order of importance.

3.1. Level Review

3.1.1. Drawing the Level

After a level is generated, Procreate will automatically draw it for the user to see in the Level Menu (see section 5.1. for information about this menu). The level drawing will updated if the user makes any changes to the level.

3.2. Game Objects

3.2.1. Creation and Editing

Game objects can be created in the Game Object Menu (see section 5.1. for information about this level), where their type, image and appearance rate attributes can be set. Users can edit a game object by clicking on it in the Level Menu, which will cause the game object's attributes to be shown for editing in the Game Object Menu.

3.2.2. Loading a Game Object's Image

A dialog box will be opened when a user attempts to choose the image for their game object. This is analogous to opening a file for use in an arbitrary Windows program. Procreate can load png, bmp and jpeg images at least - any other load-able formats are subject to the capabilities of WPF.

3.2.3. Adding a Game Object to the Level Generation

A game object can be added to the level generation by 'drag and dropping' it form the Game Object Menu into the Generation Menu (see section 5.2. for more information about this menu). Any game object applied in this way will be automatically generated within the level after the procedural level generation techniques occur.

3.3. Procedural Level Generation

3.3.1. Three Procedural Generation Methods

Procreate will be capable of creating a level filled with a random selection of game objects (from a game object pool), creating cave-like levels with Cellular Automata, and creating other cave levels with the Walker algorithm (aka Drunken Walkers).

3.3.2. Applying Generation Methods to a Level in Sequence

Level generation methods can be mixed and matched in the user's desired order, in the Generation Menu (see section 5.2. for more information about this menu). They are then applied to the level in sequence when the user generates the level. Generation methods can also be applied to a level more than once.

3.3.3. Generating a Level

Once generation methods and game objects are added to the Generation Menu, the user can press the Generate button, and Procreate will generate the level for them.

3.4. Level Editing

3.4.1. Modifying Individual Level Elements

Once a level has been generated, any element in it can be modified. The user can initiate this process by clicking on any part of the level. This will bring up the Level Element Menu (see section 5.1. for more information about this menu), which presents the level element's attributes. These can be modified by the user, and the level will update to reflect any changes made.

These changes will not affect any other elements of the same game object type - they only affect the single level element. This means that changes made to a level element will be lost if the user re-generates the level. With this in mind, modifying level elements is suited only to adding the finishing touches to a level.

3.5. Saving/Opening Level Files

3.5.1. Saving the Procreate File

Procreate can save the current level to a file. This will save all information about the current level, the game objects, the level generation methods, and the level generation sequence (the order of LG methods and game objects used for generating the level). The file will be saved with a .pro file extension.

3.5.2. Exporting a Level

Procreate can export the current level, which saves the minimum information required to reproduce the level in a game. The only data exported here is the level, including the information about each level element in it. Exported levels will use a .lev file extension.

3.5.3. Opening a Procreate File

Procreate can open a file that was saved with the function described in section 3.5.1. This loads everything that was saved, thereby letting the user continue their work.

3.5.4. Importing a Level

Procreate can import a level that was exported with the function described in section 3.5.2. At this stage, any level imported by Procreate will only have information about itself, and the level elements in it. No game objects or generation methods can be imported, because are not exported by the export function.

4. Features

5. Program Concept Images

6. Process Walkthroughs

7. Bibliography

Cliffe 2011, *Minecraft 2.jpg*, viewed 6 June 2014, digital image, http://minecraft.gamepedia.com/File:Minecraft_2.jpg?version=810335ef249b76b83963c03f824a4d38

Dirtytabs 2014, *Don't Starve Gallery Treeguard.png*, viewed 6 June 2014, digital image, http://dont-starve-game.wikia.com/wiki/File:Don%27t_Starve_Gallery_Treeguard.png