# Final Project

Design Document

# Procreate Design Document

Author: Jared Schroeder CIT025989

Due: 12 Nov. 2014

## Glossary of Terms

| Acronym | Meaning |
|---|---|
| LG | Level Generation |
| OS | Operating System |
| PG | Procedural Generation |
| PLG | Procedural Level Generation |
| NPC | Non-Playable Character |
| WPF | Windows Presentation Foundation |

## Document History

| Date | Version | Author | Description |
|---|---|---|---|
| **22/10/2014** | 2.0 | Jared Schroeder | Updated to Pages document, updated first half of the document |
| | | | |
| | | | |
| | | | |

# 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 replay-ability 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.

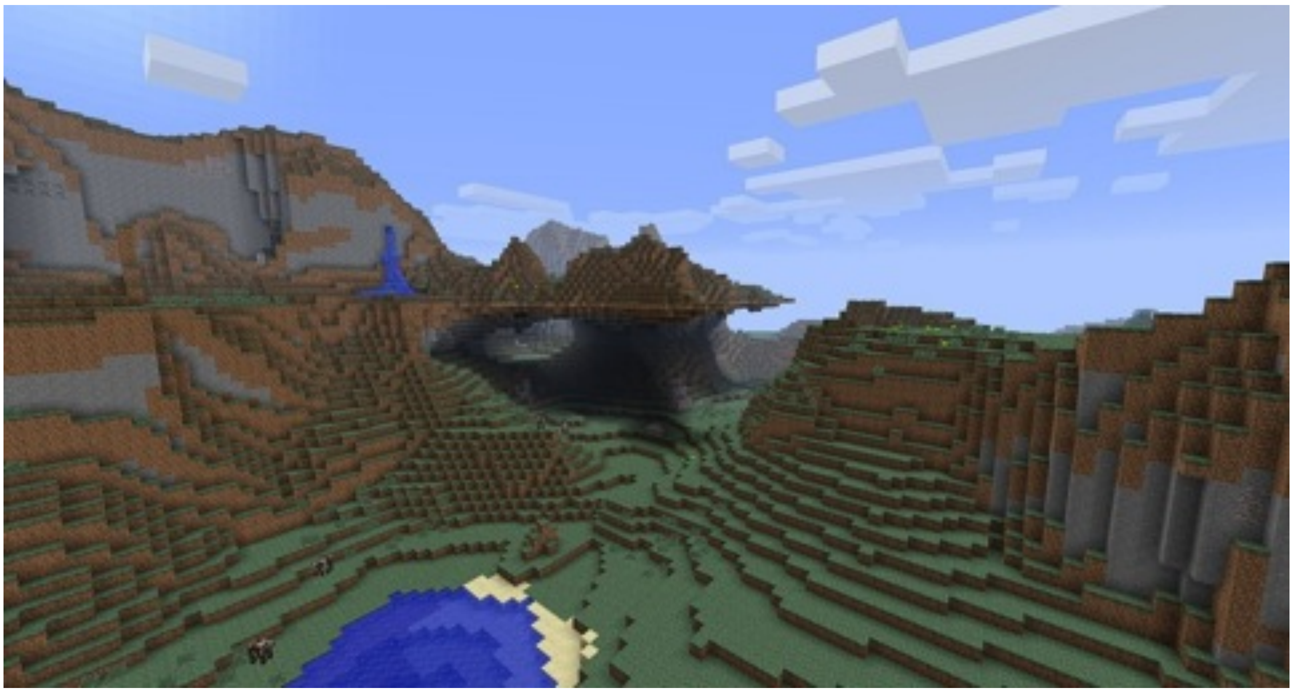


**FIGURE 1: MINECRAFT (CLIFFE 2011) USES PG TO CREATE LARGE AND DIVERSE WORLDS TO BE EXPLORED (2011)**

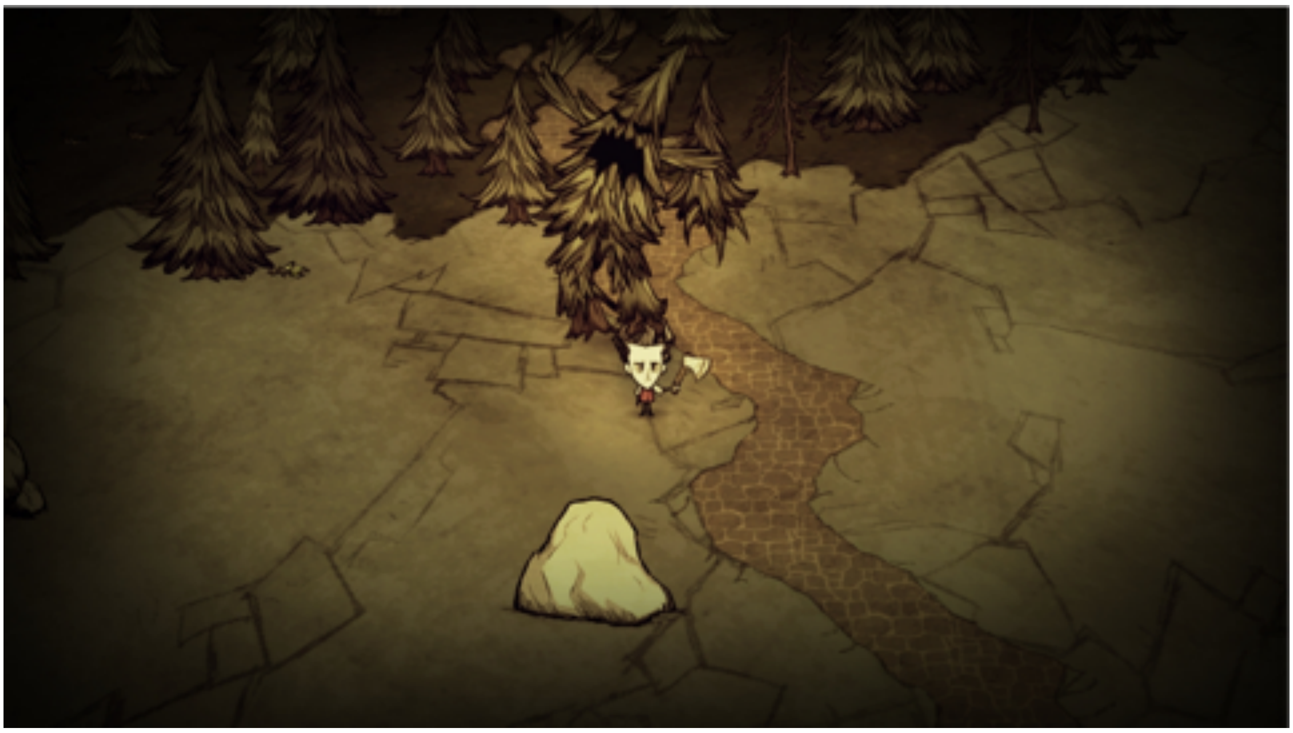


FIGURE 2: DON’T STARVE (DIRTYTABS 2014) USES NARRATIVE-BASED PG TO GENERATE VARIED PLAYER TASK-BASED LEVELS (2013)

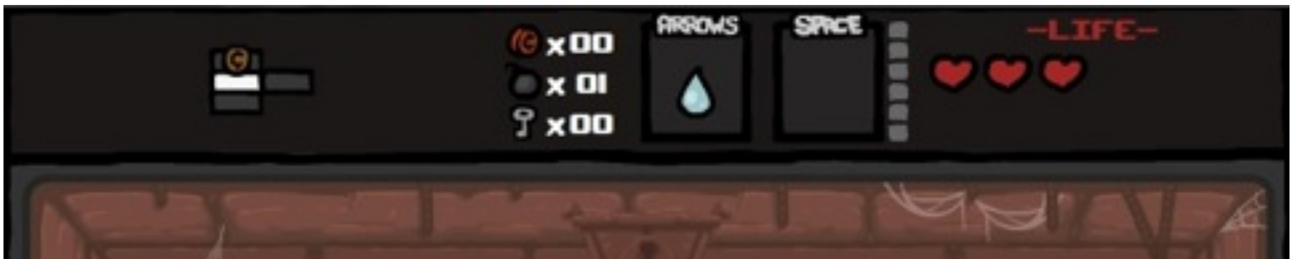


FIGURE 3: THE BINDING OF ISAAC GENERATES A LARGE NUMBER OF VARIED LEVELS COMPRISED OF CONNECTED ROOMS (2011)

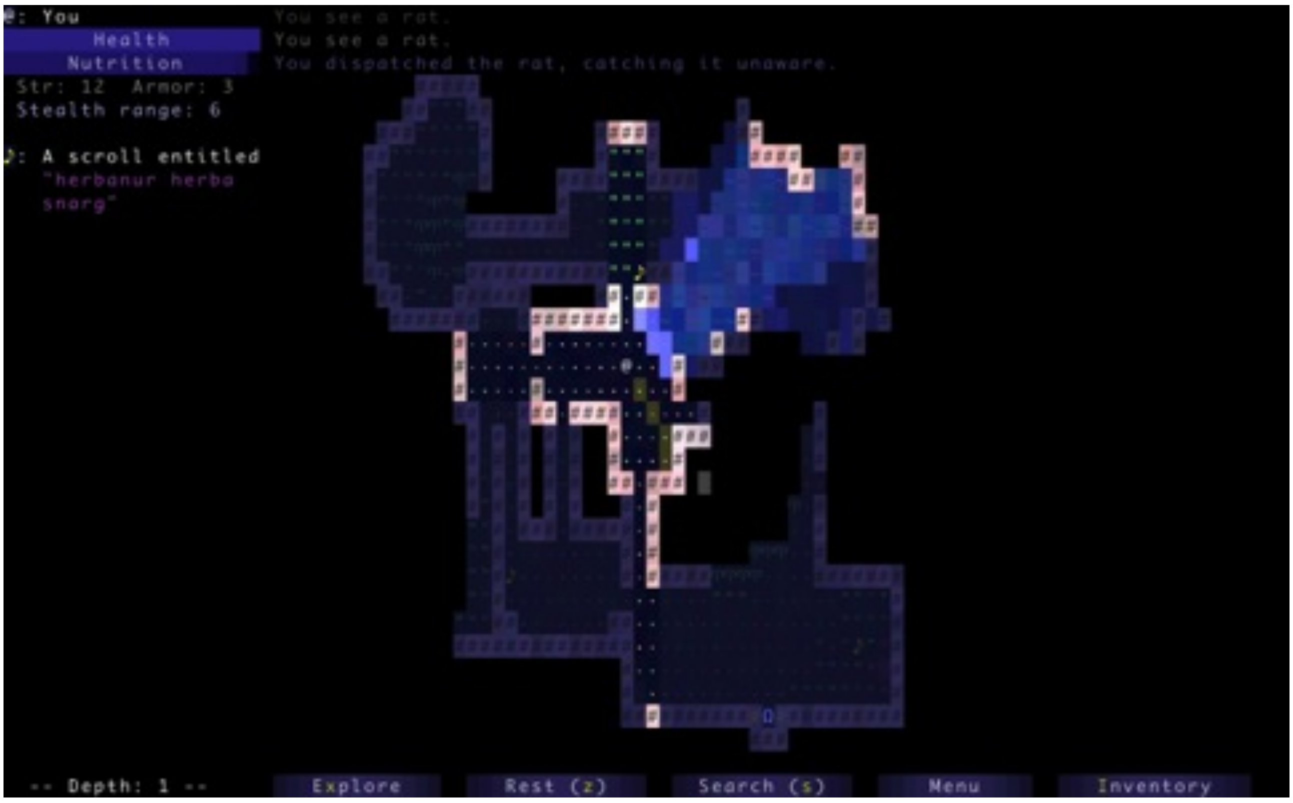

You see a rat.
You see a rat.
You dispatched the rat, catching it unaware.

FIGURE 4: '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

## 4.1. Procedural 2D Level Generation

Procedural level generation is the act of using algorithms (methods) to create a game level, rather than having the level created from scratch by humans. Creating levels with this method requires coding the level generation algorithm, which once coded can produce as many diverse levels as the user wishes, and this can require less time and artists than doing the work by hand. This technique is quite relevant to indie game developers, who may not have the time, nor the required artists, to create a game with a suitable number of levels.

Procedural level generation is the basis for Procreate – if the program didn't have any level generation techniques, it wouldn't be of any special use. It's also important that the generation is included, because then a user won't have to code the level generation algorithms themselves, which will save them time, and let them experiment with the different types of level generation available.

Procreate will implement 3 procedural level generation algorithms: randomised level creation, Cellular Automata and walkers, a.k.a drunken walkers.

## 4.2. Modifying Level Generation Method’s Behaviour

Procedural generation methods often consist of some attributes that alter the look of the levels created. Using these, a user can drastically modify the types of levels that the generation methods create.

This is important for Procreate, because it gives users more control over the levels they're creating, and it also lets them create a more diverse range of levels. If each Procreate generation method used only default attribute values, the levels produced would be similar, and the user could become frustrated with the program because it can't help them make the level they want to put in their game.

Procreate will allow users to edit attributes for each of the 3 algorithms it supplies. These attributes take the form of numbers, boolean (true/false), and lists that the user adds to. Randomised level creation will have a list of Game Objects that it sample from, as well as a number value stating the likelihood of each Game Object to be generated. Cellular Automata will have a 'number of iterations' attribute, and the walkers method will have attributes for number of walkers, min/max walker life, chance of changing direction, if the walker creates a room when it dies, min/max values for the size of this room, random chance of death, and the type of Game Object that the walker creates.

## 4.3. Level Generation Ordering & Stacking

Sometimes a user might like to mix and match some level generation methods to get the kind of level they're looking for. Ordering lets the user specify the order in which they want generation methods to be used, and stacking lets them repeat particular methods to amplify the effects of the method.

Implementing this in Procreate will let the user create a more diverse range of levels, than if they could only, for example use a single generation method to create the level.

Procreate will have a 'Level Generation' menu that contains a list of the generation methods to be applied, and this will be in sequential order. To add a method to any area in the list, the user will only have to drag the method from the 'Algorithm' menu and drop it at their desired location in the list. This process can be repeated to use the generation method more than once. Each method in the list will have a delete symbol, and when clicked this will remove the method from the list.

## 4.4. Game Objects

Procreate will use game objects to represent each element in the game level. These are user-defined objects represented by an image/sprite, that fill the level and can be different parts of a game, including NPC, enemies, environment objects and man-made structures.

Game Objects are important to Procreate, because without them the level would be empty. To the user, they are important because they are the building blocks for the level. They let the user create both the level, and the objects in the level, so the user doesn't have to worry about placing these objects in the level within the actual game code.

Procreate Game Objects will contain the following values: Name, Type, Image, Image Path and Appearance Rate. Game Objects can be created, and these values can be set and modified in the 'Game Objects' menu. A user can specify that they want a generation method to use a particular Game Object, and a user can select Game Objects to be automatically populated into the level after all the generation methods have been applied. This population takes place according to the Game Objects' Chance of Appearance.

## 4.5. Level Drawing & Review

Providing lots of ways of creating a level with Procreate is no use if the user cannot see the results. By drawing the generated level to the screen, a user can experiment with different generation methods, game objects and level editing until they've created the right level. The drawn result gives a quick indication of how the level will actually look in the user's game, allowing users to be more productive and spot any level errors. It also lets the user experiment with Procreate, to see the range of levels they can create.

Procreate will draw the user's current level in the 'Level' menu. When the user generates a level, Procreate will take the Image value for each Game Object in the level, and render these to the screen. The user then be able to review the results of the generation. From this point they can decide whether to change any of the generation methods or Game Objects, and if they choose to do so, the program will automatically refresh the level for them as soon as they press the 'Generate' button.

## 4.6. Saving Files & Level Exporting

Saving Procreate files and exporting a level is almost self-explanatory. Saving a file lets a user retain every piece of data associated with the level and its creation, to either be accessed with Procreate at a later date, or to be integrated into the user's game. Exporting a level is the minimal version of saving a Procreate file – it will only save the level and the Game Objects inside the level.

Saving and exporting are very important features to include - without them the user's levels would always be lost at the end of the Procreate session. Without these features, the user wouldn't be able to do anything with the levels they create.

Procreate will have both 'Save File' and 'Export Level' buttons that perform these functions. Saved data will be stored in XML format, so that it is human-readable, and fairly easy to integrate into a game. Saving a file will save the level, the generation methods, Game Objects and the order in which they are applied, while exporting a level will only save the level, and the Type, Tag and Image Path values for any Game Objects used in the level.

## 4.7. Opening Files & Level Importing

Opening files and importing levels is the reverse of saving and exporting them. Opening a PrProcreate file loads all data associated with the level and its generation, whereas importing a level gives the user access to the level, but not any data that was used to create it.

Opening files and importing levels lets a user continue where they left off, and it also lets users open, review and edit other users' levels.

Procreate will have 'Open File' and 'Import Level' buttons that let users do such. Upon opening a file, the user will have access to the level, its generation methods, Game Objects, and the order in which these were applied previously, however importing a level will only provide the user with the level, and the Game Objects inside it – any values used for generating the level will not be loaded.

## 4.8. Level Editing

Level editing is selecting an individual part of the level, and adjusting it to suit the user's requirements. It is intended to be done after the level has been generated.

This feature is important because often a user will want to add specific objects to a level that procedural generation cannot handle. An example is choosing a point in a level and placing a house object there, because it has to be there for to complete the game's story. Another reason for level editing is that Procreate might sometimes create a small error in the level, and the user will be able to patch it up on their own.

To edit Procreate levels, the user has to select the part they want to edit, and then a menu for editing it will appear. The user can make their changes, and then save them, to see the changes take effect immediately. A detailed example of this is provided in the 'Written Walkthroughs: Editing a Level Element' section.

## 5. Program Concept Images

The following concept images show the current layout of P2DLC and its menus. A numbering system is used to refer to the different elements of the main layout.

## 5.1. Main Layout

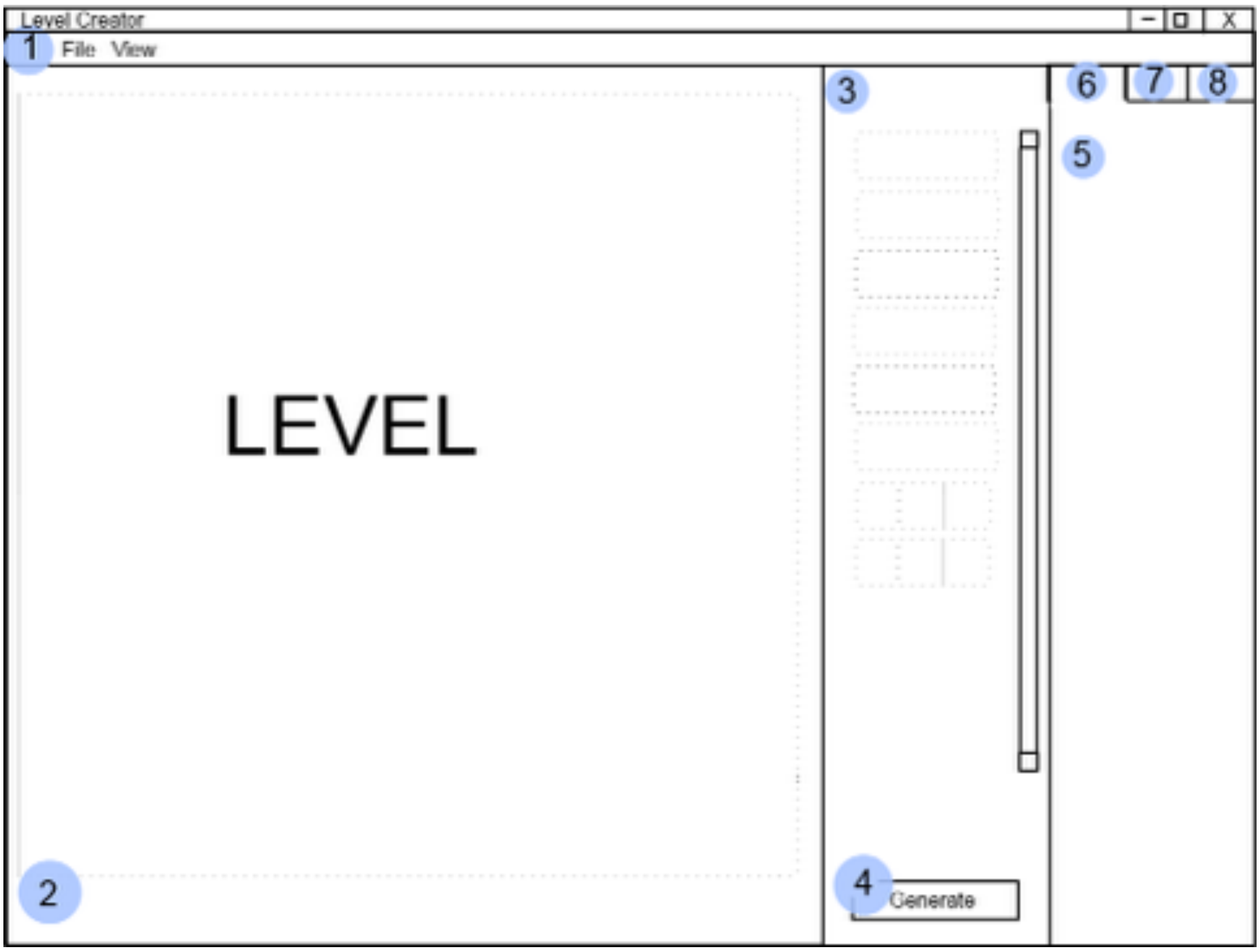


**FIGURE 5: PROCREATE MAIN LAYOUT**

### 5.1.1. Index

1. Toolbar
2. Level menu
3. Generation menu
4. Generate button
5. Context Menu (defaults to Game Object menu)
6. Method menu
7. Game Object menu
8. Level Element menu

## 5.2. Generation Menu (3)

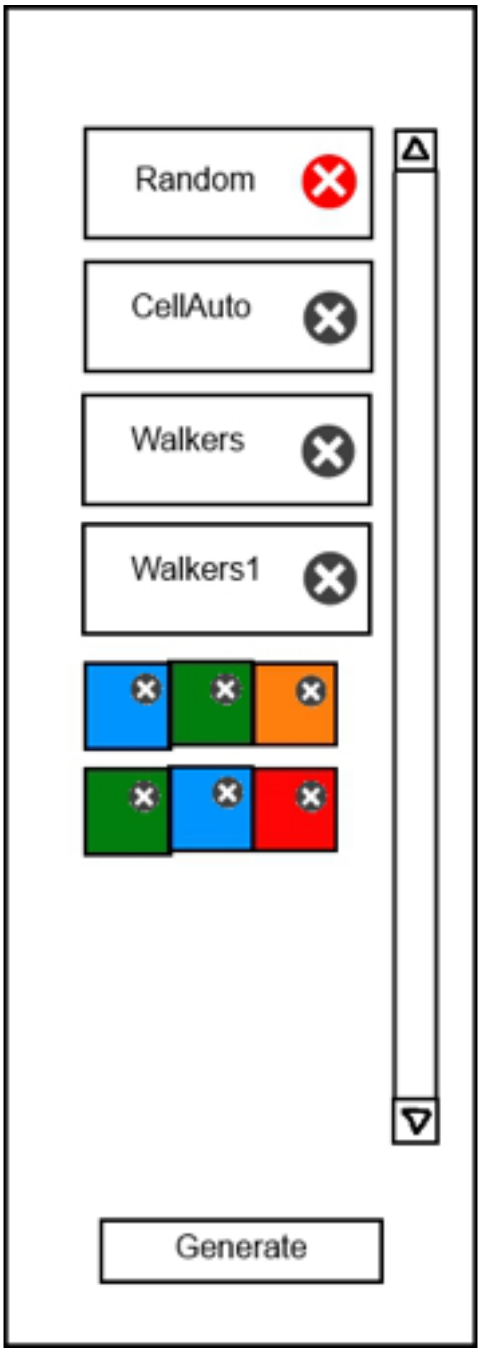
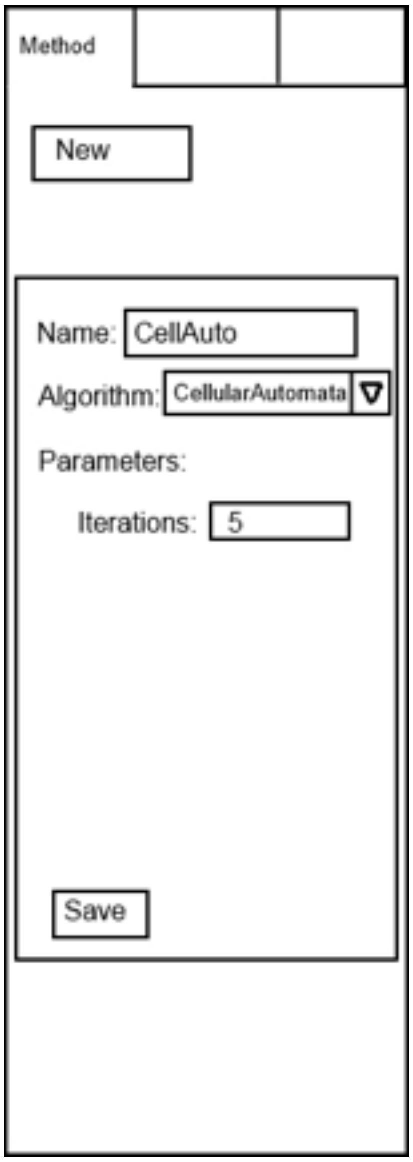
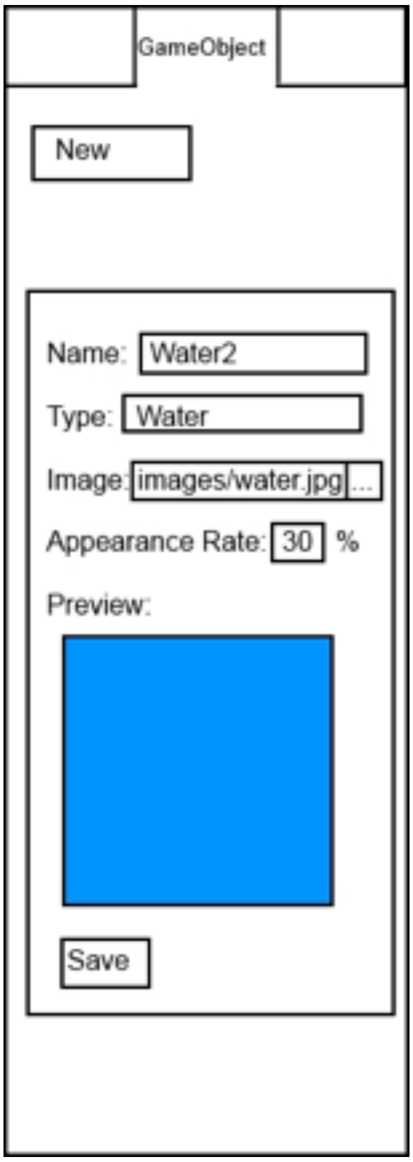


**FIGURE 6: PROCREATE GENERATION MENU**

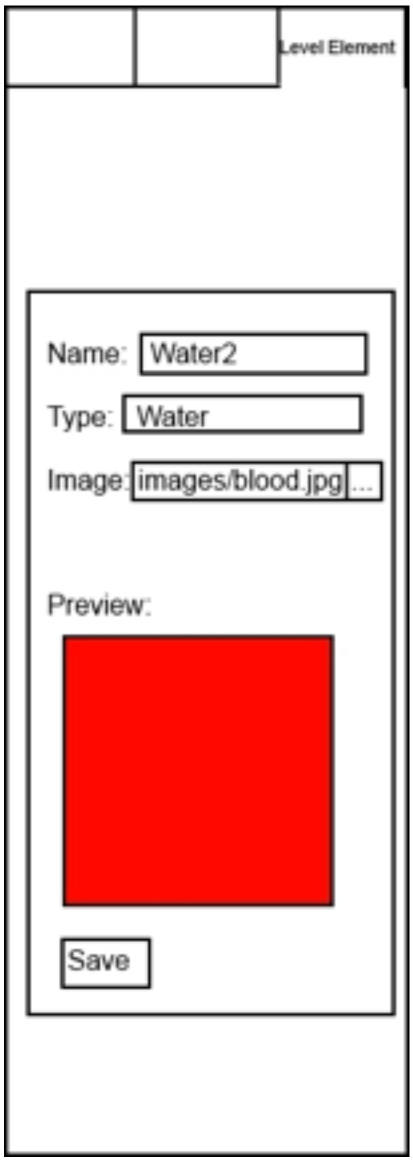
## 5.3. Context Menus (5)



Name: CellAuto
Algorithm: CellularAutomata
Parameters:
Iterations: 5



Name: Water2
Type: Water
Image: images/water.jpg ...
Appearance Rate: 30 %



Name: Water2
Type: Water
Image: images/blood.jpg ...

**FIGURE 7, 8 & 9: METHOD, GAME OBJECT & LEVEL ELEMENT MENUS**

# 6. Process Walkthroughs

The following sections give instruction as to how some of P2DLC's features can be achieved. Note that the numbering section used in the previous 'Concepts: Level Creator Main Layout' section are also used here, and refer to the same elements.

## 6.1. Creating a level generation method

1. Click (6)
2. Click the New button
3. Give the method a name (e.g. 'CellAuto')
4. Click the drop-down field next to Algorithm, and choose the generation algorithm you want to use (e.g. CellularAutomata).
5. After choosing the algorithm, its parameter fields will appear below. Modify them as you wish.
6. Click Save
7. All the method information is inside a box. To add the method to the level generation, click on any point inside this box, and drag it onto the methods area in (3)
8. Click (4) to see the result!

## 6.2. Creating a Game Object

1. Click (7)
2. Click New
3. Give the Game Object a name (e.g. 'Water2')
4. Set the Object's type to a general descriptive term (e.g. if its name is 'Water2', its type will probably be Water)
5. Click the '...' box next to the Image field. A file explorer dialog box will open, and from there you can choose the Object's image. Note that after select the image, a preview will appear in the Preview box
6. Set the Object's appearance rate (this is a percentage value between 0 & 100)
7. Click Save
8. All the Game Object information is inside a box. To add the Object to the level generation, click on any point inside this box, and drag and drop it inside (3)
9. Click (4) to see the result!

## 6.3. Saving a file

1. Click (1), and select 'Save'. A file explorer dialog box will appear.
2. Choose the name for your level file, and click the save button in the dialog box. If no errors occurred, our level file should have successfully saved.

## 6.4. Opening a file

1. Click (1), and select 'Open'. A file explorer dialog box will appear.
2. Choose the file you want to open, and click the open button in the dialog box. Your level file should now load into the Level Creator.

## 6.5. Editing a Level Element

1. Click on any part of the level inside (2). (8) should automatically appear in the (5), and will display the information about the level element you clicked.
2. Make your changes, and then click Save. If you changed the Level Element's image, you should see it change inside (2)

## 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