<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Procedural content generation horror game</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Wong, Ho (黃濠)</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Wong, H. (2016). Procedural content generation horror game (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.</td>
</tr>
<tr>
<td><strong>Issue Date</strong></td>
<td>2016</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/2031/8700">http://hdl.handle.net/2031/8700</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.</td>
</tr>
</tbody>
</table>
15CS124

Procedural Content Generation Horror Game

(Volume 1 of 1)

Student Name : WONG Ho
Student No. : 
Programme Code : Bachelor of Science (BSC3)

Supervisor : Dr. Yuen, Chun Hong Joe
1st Reader : Dr. Li, Zhenjiang
2nd Reader : Dr. Yu, Yuen Tak
Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Procedural Content Generation Horror Game

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: __________________________ Signature: __________________________

Student ID: __________________________ Date: __________________________
Extended Abstract

Summary
This project can be split into two parts including game engine implementation and procedural content generation (PCG) algorithms development. Game engine provides the base element for a game function as well as a platform where the PCG algorithms can take place.

Aims
The project is aimed at creating a survival horror game as the final product. This game will be taken as the result of using the game engine and testing the goodness of the algorithms. It is expected to see a fully functional survival horror game implemented with the reasonable house and room environment setting with a good enough tension experience from this project.

Game Engine
The game engine has implemented the necessary elements for a basic 2D-platformer game including base the movement and control, base character, objects interaction, character status, lighting, sound, animation system, messenger system, game element pool, AI system, pacing system and environment generation system.

These parts of game elements provide a functional game engine where the PCG algorithms can take place (environment generation system & pacing system).

PCG Algorithm
This project will introduce 3 algorithms from the big picture that generating the house structure down to rooms’ detail structure and the pacing generation algorithm which controlling the tension level of the game. With these algorithms, the game is able to provide different game run experience to player.

House generation algorithm constructs the full 2D house structure including distribution and the connections of rooms based on the concept public and private room. Also, house structure is expected to be similar to the real life house.

Room generation algorithm construed the structure inside a room including the furniture placement and other items based on the private level which is extended from public and private room concept.
Pacing generation algorithm acts as a director who deciding when and what elements to cue in such as background sound, sound effect, plots or events to the game to create the ideal tension level the game introduced.

**Acknowledgments**

I would like to have a special thanks to my final year project supervisor – Dr. Joe Yuen from the Computer Science department of City University of Hong Kong. Dr. Yuen has kindly helped a lot on guiding me finding the topic, instructing the researching direction and providing suggestions on the test plan. Thank you.
# Table of Contents

1 Introduction .................................................................................................................. 6  
   1.1 Background .............................................................................................................. 6  
   1.2 Motivation & Objectives ......................................................................................... 6  
   1.3 Literature Reviews ................................................................................................. 7  
       1.3.1 Elements in Horror Game ............................................................................ 7  
       1.3.2 House Generation Algorithm ..................................................................... 7  
       1.3.3 Room Generation Algorithm ..................................................................... 7  
       1.3.4 Pacing Generation Algorithm .................................................................. 8  
       1.3.5 Development Tool ..................................................................................... 8  

2 Game Engine & Algorithms .............................................................................................. 9  
   2.1 Game Engine ........................................................................................................... 9  
       2.1.1 Use Case Diagrams ..................................................................................... 9  
       2.2.1 Game Elements ........................................................................................... 10  
   2.2 Algorithm ............................................................................................................... 13  
       2.2.1 House Generation Algorithm ..................................................................... 13  
       2.2.2 Room Generation Algorithm .................................................................... 15  
       2.2.3 Pacing Generation Algorithm .................................................................. 17  
   2.3 Test Cases ................................................................................................................ 16  
       2.3.1 House Generation Algorithm ..................................................................... 19  
       2.3.2 Room Generation Algorithm .................................................................... 21  
       2.3.3 Pacing Generation Algorithm .................................................................. 25  

3 Conclusion .................................................................................................................... 27  
   3.1 Summary ................................................................................................................. 27  
   3.2 Suggest Improvement ............................................................................................. 27  
   3.3 Critical review ......................................................................................................... 28  

4 References ...................................................................................................................... 29  

5 Appendices .................................................................................................................... 30
1. Introduction

1.1 Background
With the raise of funding and distribution platforms like Kickstarter and Steam, indie game developers are getting easier to gather more funding and selling their products with less cost. This created an indie games bloom and taking a greater share of the game market against those AAA games in recent years. Those great indie games like Spelunky (Mossmouth, 2008)[1], The Binding of Isaac (Edmund MaMillen and Florian Himsl, 2011)[2], Risk of Rain (Hopoo Games, 2013)[3] and Rogue Legacy (Cellar Door Games, 2013)[4] are brought to the main stream.

These games have incredible long game life when compared to the games contained the same among of game elements. One of the keys is the procedural content generation, PCG in short, generated many different various game environment for players to experience each time.

1.2 Motivation & Objectives
Meanwhile, there are a lot of great horror games in the game history. They have successfully buried their twisted plots, horrible monsters, unpredictable jump scares into the players’ deepest mind and became their life-long nightmares. Ironically, the success of printed the horror element into players’ mind makes those great games almost not re-playable because of players knew it so well. Therefore, horror games are inevitably consumable because of its very nature.

However, with PCG, the life cycle of horror games can be extended or even bring a more horrible experiences to players by re-triggering their fear. Therefore, horror games could be a good experiencing target to put the PCG on it, and this could be a new direction to the horror game genre.

In this project will develop a survival horror game with generated a reasonable environment including building structure (the house) and the sub unit structures (the rooms) for the hunting house. Also, the PCG will also be used on creating game pacing.

Apart from the PCG algorithms, a game engine will be composed to develop this survival horror game and let the algorithms take place.
1.3 Literature Review

1.3.1 Elements in Horror Game
Analyzed and summarized into a game language by the blogger Gdrealms’s article “Survival Horror Design” in 2012 [5], a good survival horror game should focus on vulnerability, isolation, scarcity and the unknown. In a simple explanation, the unknown, the disempowerment, the isolation and the tension are the keys to a good horror game.

For keys like the unknown, the isolation and the disempowerment, those more related to the game design, script design or even creative writing will not be not our scope in this project. However, the tension can be in the target testing the goodness of the pacing algorithm.

1.3.2 House Generation Algorithm
There are 3 main type of algorithms currently used for procedural environment generation. Algorithms for 3D houses, algorithms for top view 2D houses and algorithms for 2D side view action platformer which is irrelevant to ours.

For the first 2 algorithms, they cannot be applied to our game because of the missing ‘depth’. In 3D houses or even top view houses, there are plenty options to connect the rooms at the same floor level through the walls surrounded. However, in our 2D side view house, only 2 directions along the x-axis (left and right) and 1 z-axis (into the screen) can used for connecting rooms because of the view limitation.

Even though these algorithms cannot be directly applied to solve the issue, some of the concepts are very inspirational. In 2006, Jess Martin in “Procedural House Generation: A method for dynamically generating floor plans” [6] divided the floor plan into different types of area such as public room and private room due to certain conditions, and grown the next hierarchy rooms from it. This concept can be abstracted and apply to our house.

1.3.3 Room Generation Algorithm
Martin Jennings-Teats, Gillian Smith and Noah Wardrip-Fruin in “Polymorph: A Model for Dynamic Level Generation” in 2010 [7] have implemented the action-rhythm for 2D platformer levels created by Smith G. in 2009 [8]. This approach even if have not much related to this project’s neither theme nor genre, there could be a direction to transform this algorithm with other rhythm base to generate the room setup.
In 2009, T. Germer and M. Schwarz propose an algorithm to put the furniture in “Procedural Arrangement of Furniture for Real-Time Walkthroughs” [9]. By attaching every item on another item, including the wall, a cluster will be formed. For example, the table is attached to one of the wall, the chair is attached to table’s left, and the light is attached to table’s top. This could form reasonable clusters for the rooms.

1.3.4 Pacing Generation Algorithm

In 2009, a publication written by Michael booth in Value Software [10] has showed how their AI system in Lead 4 Dead worked to create the pacing. They used AI-director to monitor the tension level of players and decide when to add stress or let them relax. They have ranked the tension point for different events, so once these events are happened, they can estimate the current tension level.

Mike Lopez points out the pacing curve structure for game form movie and TV dramas in his article ‘Gameplat Fundamentals Revisited: Harnessed Pacing & Intensity’ in 2008 [11]. It shows how the pacing curve should be featured. This could help to build the expected game pacing curve in the game.

There are many algorithms currently, but most of these are not directly hit to the project title. Therefore, how to transform these concepts to fit the project is important.

1.3.5 Development Tool

In the review, JavaScript with HTML5 is a good option. First of all, it fits the requirement of making a good. For example, canvas can be used to draw the game image, and audio can used to play the game sound. Also, the events like keyboard, mouse move, mouse click are all available. In addition, there are APIs provided for those functions limited in JavaScript like locking the mouse.

Second, JavaScript is convenient to work and show at anywhere installed the web browser. Also, its development tool is only whatever a text-editing tool. This provides a good working environment no matter where you are.

Third, my familiar level to JavaScript and the experience of JavaScript game development before will save more time than using another language. This could help a better execution on applying the algorithm via JavaScript.
2. Game Engine & Algorithm

2.1 Game Engine

2.1.1 Game Use Diagram
2.1.2 Game Elements

Game Background

This game is about a pair of grandfather-daughter, Elbert and Emma, woke up and were trapped separately in a dark abandon haunting house. While more the house Elbert explored, the more wired things happened to him, and he released the real danger would come very soon. Now, Elbert has to find his granddaughter and found the way out before it was too late.

There are some base components are constructed in the game engine in order to handle the needs of a survival horror game. Here will introduce some main components’ function.

![Game Image Capture](image)

**Figure 2.1 Game Image Capture**

Control

The control part of game can be split into keyboard and mouse. For the mouse, it can control the direction of main character and select the pointing targets. Also the right click and left click are representing actions like pick up and use in the game. For the keyboard, it controls the movement like walk or run to a direction and some triggers including trigger the action of selected target, turn on or off the flash and select the items as current using item.

Basic Game Mechanism

The game mechanism are including the game initiation like loading game resource, game loop for object update and draw, object collision checking with hitting zone, game camera movement and some unity function used in the game.
Characters
There are 4 types of character including player, enemy, friendly character and neutral character. Player represents the character of player controlling and who the camera will following to. Enemy, friendly character and neutral character are AIs who will have different reaction to the other type of characters. For example, Enemy will treat player as their primary attack target, then friendly character as secondary target.

Character Status
For each character, they contain their own status object representing their health point, stamina point and attack point if able to attack. The status object take part in handling the change of character status. Based on the different status of character, it will change the state of them like set character to ‘tired’ when low stamina point and set it to ‘dead’ when health point is 0.

AI System
AI System is only applied to the Non-player Character (NPC) in this game. It takes the approach of stacking the AI actions into a list. For each AI action is done, it moves to the next AI action until the list is empty. When the list is empty, it will push a default AI action to it.

There are 4 types of AI actions represent the behaviors of characters’ need such as ‘MoveTo’, ‘Search’, ‘Wait’, ‘Patrol’ and ‘Attack’. For each AI action, it can be simplified as 2 main functions that checking this AI action can be executed and checking the execution is done.

For example, Enemy discover the player couple steps in front of it. It puts the ‘Attack’ to the AI action list. Then, ‘Attack’ object check if Enemy can attack the player, and it is false because the not close enough yet. So it clears the current action, and pushes 2 AI actions into the list: ‘MoveTo’ target and ‘Attack’. After ‘MoveTo’ is executed, ‘MoveTo’ object keep checking until Enemy arrives the target location. When Enemy arrived next to the player and so ‘MoveTo’ is done, now ‘Attack’ is finally able to execute.

Interaction
When player targeted an object, it will be treated as selected object. Player can trigger its actions including getting its information by pressing ‘Q’, picking it up by right-clicking the mouse, using it by left-clicking the mouse and calling its own action by pressing ‘E’. For calling their own action,
types of actions are contracture in the game. For exit object like doors, it will trigger the open or close. For objects in the room, it could get searched for items or provide a hiding place for callers.

**Lighting**
All lighting objects will be first stored in a global list. When draw the light, it will first check the lighting object if is available to display. For all available light objects will first union their boarders to form the lighting area, then draw all the objects inside the area and covered by the glowing light effect.

**Sound**
A global list will store all the sound resources after loaded into webpage. When the resource is needed, a global object named ‘SOUND’ will act as a class object to play, stop and pause the sound.

**Animation System**
For each object using image sprite like characters, room objects and items, it will create a set of animation action to them. Then the character walk, it calls the Animation Director to execute the walk animation by drawing the defined frame separately from the sprite to their owner’s location.

**Messenger System**
When there is a dialogue in the game or just some information shown for players, it will call the Messenger object to ‘say’ the information. The ‘say’ function requires the text need to be told and the teller object. Then there is an indicator moving along the text to bind the last char of each text display. So the text will not be displayed all at once, but more like the dialogue that ‘say’ word by word. Apart from that, under consideration of skipping text, players also can press ‘Q’ to skip or move to next text.

**Configuration Table & Game Element**
A configuration table stored the pre-defined constant variables in the game. Game element pool stores the gaming information such as the name of room, animation frame variables or character move speed in JSON format.

**PCG System**
Will be introduced in the next part.

**Game Pacing System**
Will be introduced in the next part.
2.2 Algorithms

2.2.1 House Generation Algorithm

2.2.1.2 Background & Idea

There are 3 main requirements for the house. First of all, all the rooms should be connected or reachable. Second, some of the wired room combinations like putting kitchen next door to the bedroom should be avoided. Third, the connections should be reasonable like it is in no way to find a stairs inside the washroom.

Inspired by the idea public and private zone concept from Jess Martin, the house can be divided into different type of areas. For each area, there will be a list of rooms can be assigned. Base on each type of areas, another level of area can be extended from it and this keeps going on until reach the last type of room or no rooms left.

In this algorithm, the house can be divided into 4 types of zones that public rooms, private rooms, attached rooms and hidden rooms. After extended to the hidden rooms, this algorithm will warp back to public rooms. The reason of allowing this is unexpected exits toward to other type of rooms is reasonable to happen in survival horror games. For example, player may find a secret path from the prison back to the hallway. Some other options are also available like wrapping back from attached rooms to public rooms or just stopped and removed those unreachable rooms.

In the original idea from Jess Martin picked a single starting room to extend the public zone to private room and go on. However, because the limited among of available exits in the 2D side view platformer, most of rooms will become public rooms like hallway or the house structure size will be so limited. Therefore, this algorithm will pick multiple rooms which are suitable to be converted as public rooms as starting rooms.

2.2.1.2 Implementation

Step 1 – Determine House Structure

The house width, house height, maximum room size and different room size ratio are needed to be defined as input. By determining these combination of values, can increase the reasonable level of the house. For example, it makes more sense if a house have 1 or 2 width unit rooms as the majority.
Step 2 – Setup Connections
For each floor, connect all the rooms in each floor using the spinning tree based on certain rule. For example, the rule could be more favor in connecting small and large rooms. Then, pick a qualified room as the first stair room based on the rule set to setup the connection with other floors.

Step 3 – Assign Different Zones
Picking a list of qualified room based on the rule set, say the highest exit number to room size ratio, as public rooms. Based on the list of public rooms and the stair rooms, assign their surrounding rooms as the next level rooms (public -> private -> attached -> hidden). Finally, all rooms will be assigned as public, private, attached, hidden and back to public recursively.

The result is indicated by the house structure below. The yellowed rooms are stair rooms which are the first picked room in the algorithm. Then, all public rooms (cyan in color) are assigned. Based on those public rooms and stair rooms, private rooms (pale gray in color) extends from the public. Attached rooms (dark gray in color) and hidden rooms (red in color) are extended from them accordingly. Also, those public rooms have high score on the exit number to room size ratio. For example, those public rooms with 1 width unit contains >1 exits.

Step 4 – Assign Room Information
A set of room information will be setup for determining where they should be put including the valid room size, the floor level, the type of room and their descending rooms. For example, a room information ‘Living Room’ will be a public or private type room with at least 2 width unit and only able to be found at the ground floor. By also giving them an importance order in the selecting list, those necessary room information can be increased the selected chance. After assigned all room information, there will be a default room information for those unassigned room, for example the abandoned room.
From the example below, the ‘Living Room’ is located at the ground level and fits the room size. ('Living Room’ is tagged as ‘客廳’ in the image)

2.2.2 Room Generation Algorithm

2.2.2.1 Background & Idea

Room generation algorithm needs to provide a reasonable environment for players to identify the room and those expected items.

Even though it is not much related to the sound algorithm presented by Martin Jennings-Teats, Gillian Smith and Noah Wardrip-Fruin in “Polymorph: A Model for Dynamic Level Generation”, it points out an inspiring direction. If users’ actions can determine the difficulty of game features, the geometry location of a room should also can determine the position of room objects inside. This could extend the concept used in the house generation algorithm: public and private zone.

By our observation, the shower in bedroom is most like away from the door or in the place or private. More example, people naturally tends to put the important things (measured by personal safe or price value) away from the door if possible like the safe or working desk in study room. Therefore, an indicator determining the public and private value, or p-Value in short, will decide the best location for a room object. And the distribution of p-Value in a room will be affected by its geometry location.

By abstracting the concept from the algorithm from T. Germer and M. Schwarz of forming objects into clusters. There will be a hierarchy to determine the reasonable parent child relationship when forming the clusters.
2.2.2.2 Implementation

**Step 1 – Assign Static Points p-Value**
First of all, a p-Value representing each type of room is needed to setup. For example, a higher p-Value representing the room type more public and a lower p-Value for more private in contrast. Then, apart from the p-Value for the current room, each exit in the current room will represent the p-Value of exits’ towards room.

**Step 2 – Assign Other Spaces p-Value**
After assigned these static points’ p-Value, the rest of unassigned space will be calculated their p-Value based on the distance between their closest static points’ p-Value. For example, an unassigned point in the exactly middle of 2 static points which are -1 and 1 p-Value, it could be simply assigned as 0 p-Value.

![Figure 2.4 Determining p-Value](image)

**Step 3 – Place Room Object**
Based on each room of their cluster hierarchy, room objects will be picked according to the important order or just other random rule set. For each picked room object, it will be assigned to the location which fits its own p-Value the most. Then, p-Value around the room object will be changed depends on the room object’s p-Value. For example, if a located room object with p-Value -2, the p-Value around it will be decreased and closer to -2.
Step 4 – Attach Room Item
Similarly, room items will be put on the room objects based on the cluster hierarchy of room objects and the available items list of the current room. The number of room items being put will be determined by the full ratio. For example, for 10 room items with 1 width unit putting to a room object with 10 width unit, there will be at most 7 items are put if the full ratio is 0.7. After room items are put, they will be randomly separated with spaces so they will not look like a giant complex.

2.2.3 Pacing Generation Algorithm

2.2.3.1 Background & Idea
As making a survival horror game demo, pacing in this game gets trimmed into only all about the tension built up, climax and rest state. In order to implement the concept of using AI-director introduced by Michael booth, there are 2 main elements are required: the current tension level and the expected tension level. After these 2 indicator are found, we try to fit the current tension level to the expected one.

It is hard to determine the current tension level, but it will be easier if start with the expected tension level. The first question is how should be fit the tension level to the expected one. A change will be needed and a new game element should be added while trying to push up the current tension level. Therefore, the answer pops up, the current tension level is the level after added new game element. It means the game element will be needed to assign a tension level value.

In addition, Mike Lopez points out the feature of a good pacing curve in this article, and this could shape the pacing curve for our game. By fitting the pacing element tension level in more like a curve shape, the tension duration and its pattern of decay are also needed to be determined. Besides, for each game element, it needs cool down time so it will not be overused in a short period of time.

What’s more, it will be confuse if same type of pacing elements are used at the same moment. For example, if 2 background music are played overlapped, players cannot hear clearly neither one of them. Therefore, different type of pacing elements are needed to be classified and only one is used for a type should be used each time.
2.2.3.2 Implementation

Step 1 – Generate Expected Pacing Curve
Based on the curve shape suggested, a formula is plotted to generate a draft of pacing curve. Then, a random variation is added the pacing curve so that the expected curve will not be the same each time.

![Figure 2.5 The expected pacing curve example](image)

Step 2 – Define Pacing Elements
A list of pacing elements are defined with well-tuned peak tension level, duration, cool down time and pacing type. When adding a pacing element to the current tension level, it will directly add the peak tension level to it, then it decays with a constant level in the duration and rapidly decade after it. Then it will enter a cool down time and cannot be picked within the period.

![Figure 2.6 The curve of pacing object perform](image)

Step 3 – Fitting the Curve
For each time, the current tension level will be check with the expected tension level. If the different is greater than a certain range like 2 tension level, it calculates and store the different value. Then, by adding the peak tension level of each available pacing element to the current tension level, the best
pacing element with the fittest value is picked to start. Finally, add the pacing curve of the picked element to the current pacing curve.

For some of the pacing element, it might further trigger other pacing element while executing. This could add more variation to the game play and more fits the real situation. For example, after a pacing element which cue in the monster to loop for player, if the monster does find player, it will attack player and the current pacing curve should be increased.

2.3 Testing Cases

2.3.1 House Generation Algorithm

2.3.1.1 Measurement
The house generation algorithm will be tested with 3 size of house that small, middle and large. For the small house, the size is 4x2 unit (total width x total floor). For the middle house, the size is 9x2 unit. For the large house, the size is 12x3 unit.

Then, the set of middle house statistics will be used to generate the test cased for comparing with the random generated house. This case will point out the unreasonable part in the structure.

2.3.1.2 Test Cases
Case 1 – Different House Structures

Small House (Full Result in Appendix 1)

Figure 2.7 Small House Structure
Large House (Full Result in Appendix 2)

Middle House (Full Result in Appendix 3)

Case 2 – Algorithm Generated vs Random Generated

Algorithm Generation
(See Middle House in Case 1)

Random Generation (Full Result in Appendix 4)
2.3.1.3 Result Analyses
From test case 1, both 3 sizes of house cases show the algorithm ability to generation different size of structure. For those feature rooms in a house like ‘Living Room’ and ‘Entry’ (tagged as ‘客廳’ and ‘正門’ in the image) are put to the house. However, due to the shortage of rooms in small house, rooms not in the first priority list like bedroom are missing. For large house, there are more default room then the other size. This should due to not enough room information. Besides from these 2 issues, the house structure is reasonable that no rooms show up in wrong floor or wrong zone.

From test case 2, it is easy to find out the unnatural part from the random generated houses. For example, some wired room location like the ‘Kitchen’ and ‘Living’ showed up on the second floor and some unreasonable paths like from ‘Kitchen’ to ‘Study Room’. In contrast, the room combination in algorithm generated house are closer to the actual house structure.

2.3.2 Room Generation Algorithm

2.3.2.1 Measurement
The room generation algorithm will be tested with 3 rooms that bedroom, kitchen and study room to compare how much the p-Value affected on the room structure.

2.3.2.2 Test Cases
Case 1 – Algorithm Generated vs Random Generated

Algorithm Generation (Study Room)
Algorithm Generation (Kitchen)

Figure 2.12
Algorithm generated Kitchen

Algorithm Generation (Bedroom)

Figure 2.13
Algorithm generated Bedroom

Random Generation (Study Room)

Figure 2.14
Random generated Study Room
Random Generation (Kitchen)

Figure 2.15 Random generated Kitchen

Random Generation (Bedroom)

Figure 2.16 Random generated Bedroom

Case 2 – Variation in Algorithm Generation

Study Room

Figure 2.17 Algorithm generated Study Room
2.3.2.3 Result Analyses

From case 1, there are 2 main problem can be observed from random generated when comparing with the algorithm generated. The first one is the room items are not put on the reasonable parent. For example, from random generated kitchen, there is a boiler on the cupboard and a frame on the cooking stove, form random generated bedroom, there is also a frame on the bed as indicated. In contrast, algorithm generated room has formed reasonable clusters.

Second, there are some room object in random generated room placed in the unnatural location. For example, the random generated study room’s door is towards to the kitchen, but the safe is too close
to the door. Another example from the random generated bedroom, the bed is the closest room object
to the door, but the door is towards to a public room.

From case 2, the result shows there are enough different from the test cases. Some rooms show quite
a different which could due to the different room geometry. Even some rooms are put room objects
in a similar way or order, there are some details like the items making it different.

On the other side, there are some room object combinations commonly found. For example, the safe
and study desk in study room, and the bed and closet in bedroom. This makes good sense because of
their close p-Value.

2.3.3 Pacing Generation Algorithm

2.3.3.1 Measurement
Game Pacing Generation Algorithm will be tested with a random pacing generation and compared
with the how much are not fitting the expected pacing curve.

For the graph, black line represents the expected pacing curve and red line represents the current
pacing curve. The y-axis represents the tension level and x-axis represents the time. The total time of
pacing is used 100 time unit, which means it will check the pacing 100 time in total.

2.3.3.2 Test Cases

Case 1 – Algorithm Generated vs Random Generated

Algorithm Generated:
Random Generated:

![Graphs showing random generated expected pacing curves]

Figure 2.21 Random generated expected pacing curve

Case 2 – Comparison of Algorithm Generated
(Algorithm Generated Results from above)

2.3.3.3 Result Analyses
From the test case 1, it is no doubt that the algorithm generated is much more fitting to the expected curve. From the algorithm generated curve, there are some peaks suddenly higher than the expected which is due to the further pacing element triggered during a pacing element event.

From the test case 2, some patterns are quite similar. For example, the first peak happened at the similar timing and it belongs to the cue in monster. This could be due to not enough pacing element type. When the background music and sound effect cannot provide enough tension, it has to cue in the monster. However, those small peaks are quite different among the cases, this shows a certain of randomness.
3 Conclusion

3.1 Summary
This project has implemented a game engine started from nothing in JavaScript. Couples of game elements are implemented in order to create a good platform to experience a survival horror game, as well as for the algorithm application.

In the first algorithm, a house generation algorithm is introduced to try providing a more suitable method for 2D side-view house. By considering the limitation of a 2D side-view house about the limited connection wall due to its flatness, this approach using a multiple growing point in a spinning tree under the connecting rule set and the defined hierarchy structure.

Then, a room generation algorithm extends from the house generation algorithm’s public and private zone concept is developed. By calculating the p-Value among the room, with the well-defined room object and items clusters can provide a reasonable room setup.

Apart from the environment generation, the pacing generation ensured the player experiences the expected pacing curve developers designed. Fitting and redefined the current pacing curve by introducing pacing object can create a proper pacing for the game.

These algorithms allow game developers to create a various experience to players and extends the game life cycle and its replay value.

3.2 Suggest Improvement
For the house generation algorithm, it could be improved by providing a proper handle when the growing zone reach to the last zone. For example, these zones can be either removed or just built as the secret region. Also, fine tuning the exits like adding or removing extra exits and adding more exits type such as hidden door or broken wall can let the house structure more convincing.

For the room generation algorithm, more layers can be added to enrich the room environment. For example, instead of just room object, it can be further divided into on-wall room object, back room object, middle room object and front room object. This can increase the layer of a room and more reasonable.
For the pacing generation algorithm, it can provide the relation between pacing elements to alter the pacing object tension level instead of just a fixed value. For example, when playing ‘Music A’, ‘Sound Effect B’ will have greater tension value.

3.3 Critical review

This project is one of the toughest I ever meet because of the huge effort and skills required.

This project contains many components and they are all tangled. Apart from the develop algorithm, a full game mechanism game engine is needed to apply the algorithms on it. Also, in order to test the algorithm, those game elements are needed for defining rule set take place. And making a convincing result required many game elements. In addition, it is not easy to find the proper and enough among of graphic with the same theme and type. So those graphics have to be drawn by myself. Therefore, in order to ensure algorithm work well, many works have to be done and all of them need to be good. So the requirement of this project is over my expectation both on its depth and wide.

In order to create a less complex game, 2D side-view game is picked. However, there is not much algorithm can be found on paper to apply on this special type of game. And how to abstract the concept and apply is very important.

This year not only strengthen my different part of technical skill, but also the mind set of abstracting concept and application. After this project, it will also make me take the pre-production more into the consideration.
4 References

1. Mossmouth 2008, Spelunky


3. Hopoo Games 2013, Risk of Rain

4. Cellar Door Games 2013, Rogue Legacy


5 Appendices

Appendix 1 – House Generation Algorithm: Small House Full Result

Remarks:
1. The upper part is the ‘inside’, the lower part is the ‘outside’ for each house.
2. Because of the space limit, rooms are indicated with Chinese name short form as follow:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Room Name</th>
<th>Tag</th>
<th>Room Name</th>
<th>Tag</th>
<th>Room Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>正門</td>
<td>Entry</td>
<td>客房</td>
<td>Guest Room</td>
<td>雜物室</td>
<td>Lumber Room</td>
</tr>
<tr>
<td>走廊</td>
<td>Hallway</td>
<td>主人房</td>
<td>Master Room</td>
<td>囚室</td>
<td>Prison</td>
</tr>
<tr>
<td>樓梯</td>
<td>Stair</td>
<td>書房</td>
<td>Study Room</td>
<td>祭壇</td>
<td>Altar Room</td>
</tr>
<tr>
<td>客廳</td>
<td>Living Room</td>
<td>廁所</td>
<td>Toilet</td>
<td>荒廢房</td>
<td>Abandoned Room</td>
</tr>
<tr>
<td>廚房</td>
<td>Kitchen</td>
<td>浴室</td>
<td>Bathroom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>飯廳</td>
<td>Dinner Room</td>
<td>電氣室</td>
<td>Control Room</td>
<td></td>
<td></td>
</tr>
<tr>
<td>客廁</td>
<td>Guest Toilet</td>
<td>溫室</td>
<td>Green Room</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
Appendix 2 – House Generation Algorithm: Large House Full Result
Appendix 3 – House Generation Algorithm: Middle House Full Result
Appendix 4 – House Generation Algorithm: Random Generated Full Result
Appendix 5 – Room Generation Algorithm: Algorithm Generation Full Result (Study Room)
Appendix 6 – Room Generation Algorithm: Algorithm Generation Full Result (Kitchen)

Elbert: ”I cannot remember anything...”

Elbert: ”...Have to find Emma and make sure she is okay...”

-- Press 'A' & 'D' to move --
Appendix 7 – Room Generation Algorithm: Algorithm Generation Full Result (Bedroom)
Appendix 8 – Monthly Log (Oct – Jan, May – Jun)

Monthly Log (Oct):

Game Design - Idea:
  ● Planning some the game background.

Game Design – Game Element:
  ● Base room types are constructing.

Game Art & Sound:
  ● Planned to use 2D Pixel side view art.

Game Engine - Control:
  ● Character’s base movement including walk, dash, and jump is implemented.

Game Engine – System:
  ● Camera’s base movement is implemented.
  ● Room display is implemented. (Need re-connect with new PCG structure)
  ● Map display is implemented. (Need re-connect with new PCG structure)

PCG – House Generation:
  ● Base algorithm for base house structure is implemented.
  ● Later will assign the suitable rooms and exits to the house.

Monthly Log (Nov)

Game Engine - Control:
  ● Character base interaction trigger implementing.

Game Engine – Interaction:
  ● Door interaction implementing.

Game Engine – System:
  ● Dramatic camera movement with cursor testing. (May remove if effect is poor)
  ● Room display is implemented. (connect with new PCG structure)
  ● Map display is implemented. (connect with new PCG structure)
Monthly Log (Dec)

Game Art & Sound:
- Try drawing some base elements to estimate if the time can make it.

PCG – House Generation:
- Start implementing to assign the suitable rooms and exits to the house.

Monthly Log (Jan)

Game Design – Game Element:
- Base room types are drafted. (used for house generation)

PCG – House Generation:
- Base assign the suitable rooms and exits to the house is implemented. (need to add more conditions to make the setting more reasonable)

PCG – Room Generation:
- Start implementing on attaching room objects on each other, say, chair will attach to table.

PCG – Event Generation:
- Drafting the idea of random quest generation.

Monthly Log (May)

Game Design - Idea:
- Game background planned.

Game Design – Game Element:
- Basically done, because most of the Game event will be cut due to time limit.

Game Art & Sound:
- 2D Pixel Art is drawing.
- (main character, friendly character done)
(room objects are drawing)

Game Engine - Control:
- Character jump is removed.
- Character base interaction trigger done.
- Character hiding added.

Game Engine – Interaction:
- Door interaction done.
- Room Object interaction done. (can search, can get information)
- Character interaction done. (can pick and drop friendly character)
- Item interaction system is done. (can pick up, drop, store, use)

Game Engine – System:
- Camera movement with cursor testing.
- Room moving and changing issue done.
- Character Status implemented. (HP, SP, hiding, dead)
- Character Actions implemented.
- Game messenger is done.
- Lighting is done.
- Sound wave effect is done.
- Character visibility related issue is done. (Base visibility, light visibility, room/diff room visibility, sound heard visibility)
- Animation system is done.

Game Engine – AI:
- AI is done, including route planning and action planning.

PCG – House Generation:
- additional exit is cut due to time limit

PCG – Room Generation:
- Base algorithm for room object generation done.

PCG - Item Generation:
- the basic system is done, but will only main item
• for the rest of items will be cut due to time limit

PCG – Event Generation:
• may be only changing the music and sending the monster

Monthly Log (Jun)

Game Art & Sound:
• Game Arts done
• Game Sound not yet

Game Engine – System:
• Adding Sound System

Monthly Log (July)

PCG - Item Generation:
• Done and fine tune the algorithm.

PCG – Event Generation:
• Done the algorithm and pacing object