6-bit Final Report

Team Members

Steven Alan Scott	43853333
Junchuan Xue	43792964
Changyi Zhuang	43788266
Chunzi Lyu	43425570
Dan Dai	43993903

Table of Contents

1 Summary

Over the last 3 months 6-Bit has undertaken the colossal task of developing a Roguelike game called Project Diegeon in unity. Roguelike games is a subgenre of the RPG game. The player usually explores dungeon like environments that are generated procedurally. In this dungeon players will fight monsters, solve puzzles and avoid traps. The main feature of roguelike is that when the player dies they lose most of their progress and are reset at the beginning of the game.

The Product

Project Diegeon is a rogue like game where the dungeons are not created procedurally but rather by other players. This means that we have two main users, explorers who run the dungeon and builders who design the rooms.

Project Diegeon also incorporates a dice based stat system where explorers can roll dice and add the value of this dice to their attack or defence. The player loses a dice every time they die and when they run out of dice it is game over. This adds an element of random chance and resource management.

An explorer gains more points the faster they complete a room and a builder gains points the more time explorers spend in their rooms. These points are used to showcase the ability of each player. If the room is not solved, or the players dies, no one gets any points.

Context of Use

The overall aim of Project Diegeon is twofold. First it is designed to help facilitate a collaborative environment where explorers and builders come up with their own rules of play. The second is that is serves as a tool that helps users understand some of the rules of game development and what goes into making a game.

Intended Users

The target users of this game are 15 to 30-year-old gamers who enjoy playing and creating games. While we do expect them to have knowledge of the roguelike genre, they do not have to have experience playing roguelike games in the past. Gender has no impact as we expect an equal amount of both male and female players.

Product Value

Project Diegeon at its core is a game and as such we see the main value based on how fun the game is to play. On top of this we also see it being a useful tool in teaching people the initial steps on how games are created. At a higher level it can be used to teach students how to develop games within unity. As such we have developed Project Diegeon in a modular fashion where people can easily add their own features in the future. Finally, the core dynamic of builder vs explorer in a collaborative environment where the players design their own challenges allows us to collect data in relation to online collaboration. This means that the overall gameplay will continue to evolve in the years to come.

The Client

The client is Jason Weigel who is currently undertaking his PHD at the University of Queensland. His area of interest is collaboration online through the use of software. This project is the first step in developing a full game that lets users create their own experiences and as such we focus on the core gameplay of building and exploring. In the future the client will use our code to further develop an online collaborative tool that can be release open source to the public.

Overview

TODO AFTER ALL OTHER SECTIONS ARE DONE

2 Frequently Asked Questions

Why should I play this game? What is the selling point?

The game introduces the concept of creating your own version of dungeons and exploring dungeons generated by other players. You are challenged to build a dungeon that is difficult but not impossible to pass, and finish other people's dungeon as fast as you can. If you are interested in this concept, the game is for you.

Are there any other characters to play other than the rogue?

Although haven't been incorporated into the game, models of archer and mage are designed for future reference. Have a look at them in the appendix.

How do I pick up the item?

Press E. For other key binding details, please refer to the control manual.

I notice that one character can pick up and use every other weapon. Why design it this way? How do you differentiate the uniqueness of characters?

First of all, we find it entertaining to see a rogue casting spells, a mage shooting arrows and an elf swinging dagger (although mage and elf are not implemented). It is harmless, new and interesting. With this spirit, we implemented a weapon system that allows any moving characters, be it players or enemies, to pick up and use any weapon. Enemies in our game can be really intelligent, which is one of the delaying tactics to be utilised by dungeon builders.

Since we don't have any other classes in the game, there is no need to differentiate them.

Why is the dungeon in total darkness when I entered explore mode?

Please check your internet connection. The problem is mostly caused by loss of internet connection.

Why can't I get my dice out of the slots after allocation? Is that a bug?

No you can't. It is intentionally made so. Players will have to reroll to get the dice out. Knowing that reroll is expensive forces the players to be cautious for dice allocation. This is to enforce some difficulty over dice mechanics.

When will the room I created be available for other people? Where can I find the room?

Once you clicked the submit button after testing the builded room, it is of certain possibility to be instantiated for the next explorer's dungeon walkthrough. If you'd like to experience the room as an explorer, you will have to go through the level multiple times. If the database is filled with hundreds of rooms, some luck is needed for that to happen.

What will happen if I lost connection in the middle of building or exploring? Is there any way to salvage my progress?

Nothing will happen. The progress will be there regardless of the connection state, since it is stored locally on your PC. If the connection is lost mid-way, please reconnect before submitting the room or entering the next level.

Can people only play one mode? How do you balance people's tendency of playing one mode more than the other?

Yes, you can, but it would depreciate the fun. Although haven't been implemented, building packs with advanced features will be unlocked by levelling up, and levelling up requires enough exploring experience.

Is the game available on other platforms?

No, it is limited to PC.

What PC system supports the game? What is the requirement?

Mainly the windows environment. IOS is suspected to be error-prone. Please check the installation manual for requirement.

Can I use a controller for control?

No it is not supported.

What engine and programming language is used for development?

Unity and C#.

What application(s) is involved during the development?

We have Adobe Photoshop and Illustrator for sprite design, Pyxel for animation, BTV solo for music production and Unity for the pipeline.

What is the size of the development team? What kind of development strategy do you employ?

The team is of 5 people. We employ agile development with two weeks per sprint. Client meeting closes up the sprint and creates new tasks for the next sprint.

3 Deployment/installation manual

In this section, we provide start-up manual for different audience that will make use of our final product: the end-users (players), the server administrator and future developers that will maintenance or continue develop the project.

3.1 Game Installation Guide for End-users

Minimum System Requirement

To enjoy the Project Diegeon game, your computer should at least satisfy the following requirements in Table X:

Table 1 Minimum System Requirement

Operating System	Windows XP SP2+.
Graphics Card	DX9 (shader model 3.0) or DX11 with feature level 9.3 capabilities.
Screen Resolution	1280*720 px
CPU	SSE2 instruction set support.
RAM	2GB.
Hard Disk Space	500MB
Input Device	Mouse and Keyboard
Others	Internet connection is essential when playing the game. Requires any software that can extract ZIP file.

Installation

1. Extract the game client ZIP file (Project_Diegeon.zip) to a folder.
2. Execute the Project Diegeon.exe to run the game.

3.2 Server Deployment Guide for Server Administrator

Prerequisites

1. PHP version 5.5 and above
2. phpMyAdmin for database management
3. Any software that can extract ZIP file

Deployment

1. Extract the game server ZIP file (Project_Diegeon_Server.zip) to any folder.
2. In the extracted folder, move the “game_server” folder to the DOCUMENT_ROOT of the server. For example, the file structure might like “/var/www/htdocs/game_server/”.
3. Open the phpMyAdmin, create a new database called “game_server” (Image 1), and import the game_server.sql file to this database using the default import settings (Image 2). The game_server.sql file can be found in the extracted folder.

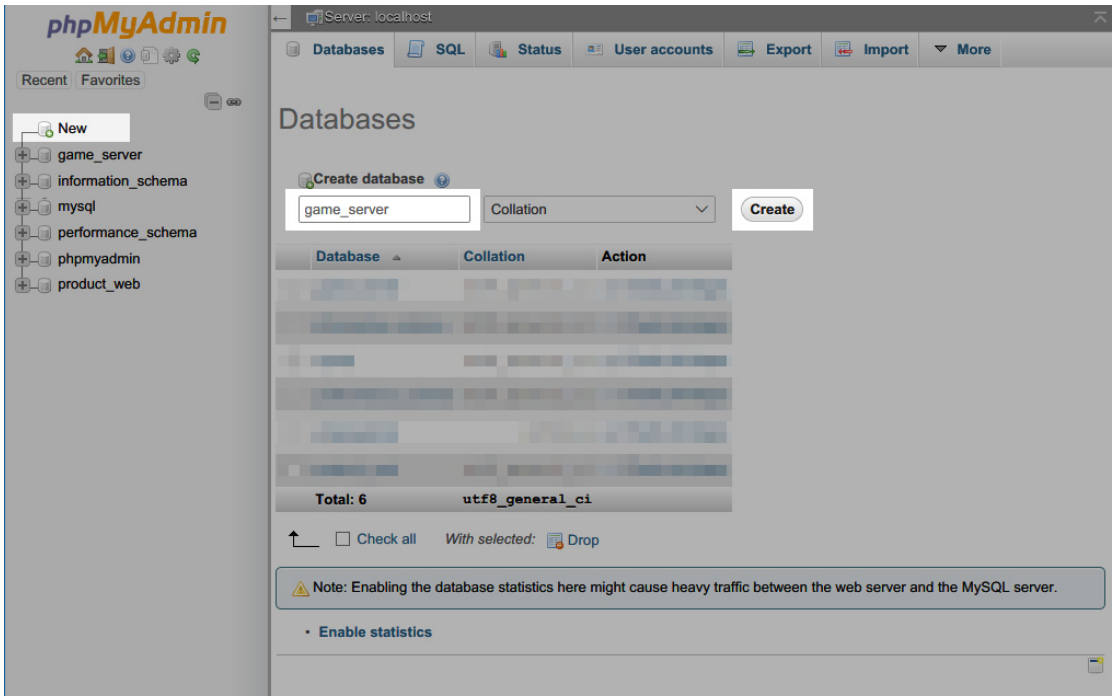


Image 1 Create database

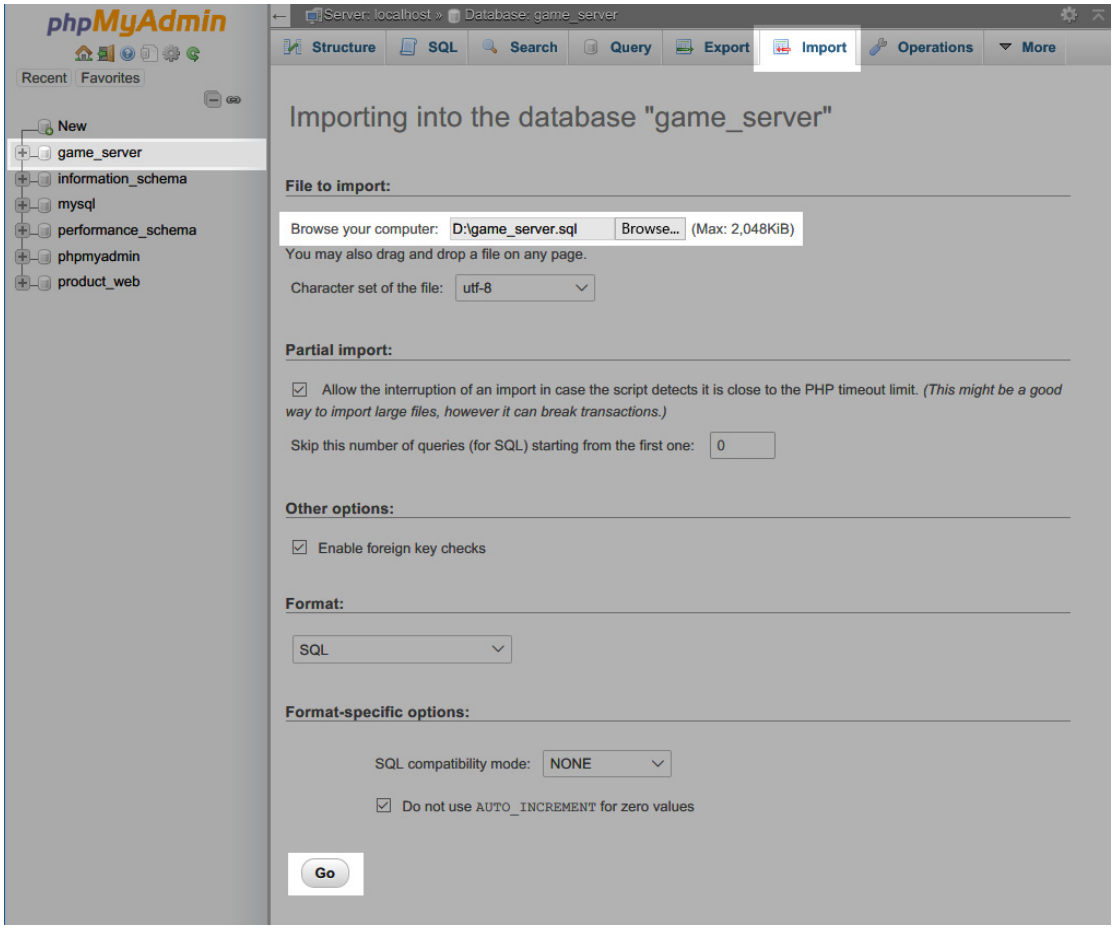


Image 2 Import database

4. In phpMyAdmin, create a new user account, with the following details:

User name	game_client
Host name	Local
Password	game_client

And the Global privileges should have at least SELECT, INSERT, UPDATA privileges for the data (Image 3 and 4).

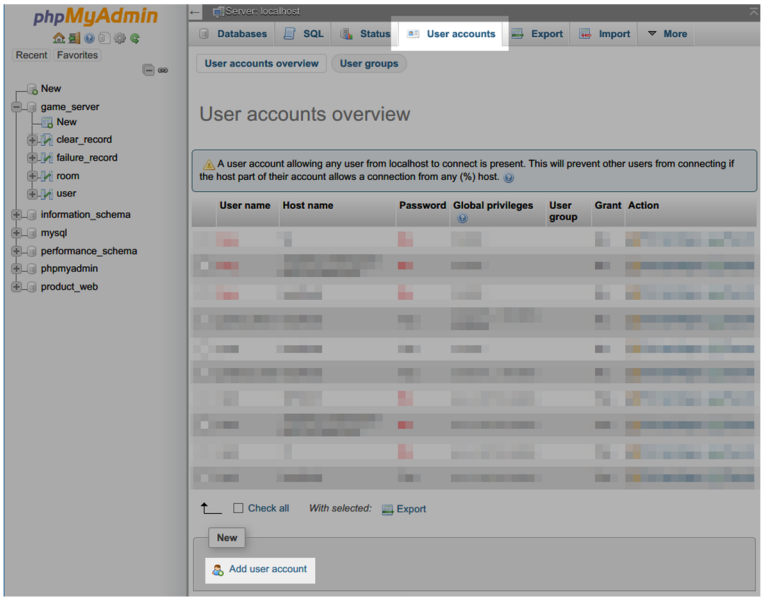


Image 3 Create user account A

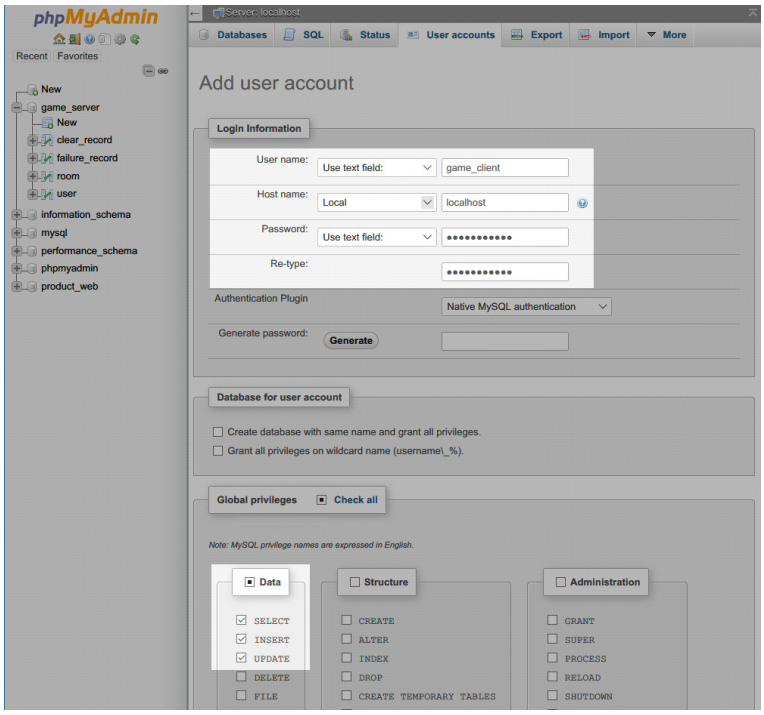


Image 4 Create user account B

Now the server should be able to work nicely.

Note:

1. Public access to the web server should be enabled to handle HTTP requests from the game server.
2. The user password for the “game_client” account in phpMyAdmin can be changed. Please make sure you also modified the password defined in the database_config.php file in your deployed game_server folder.

3.3 Project Import Guide for Future Developers

6-bit team delivers the whole Unity project folder for future development and maintenance. The Unity project folder is zipped in the file called “Project_Diegeon_Source.zip”. Thus, all essential editor configurations are maintained. For future development, we strongly suggest that use Unity version 5.4.0f3 for full compatibility. Simply extract the zip file and open the folder using Unity (Image 5), then you should be able to work on the project.

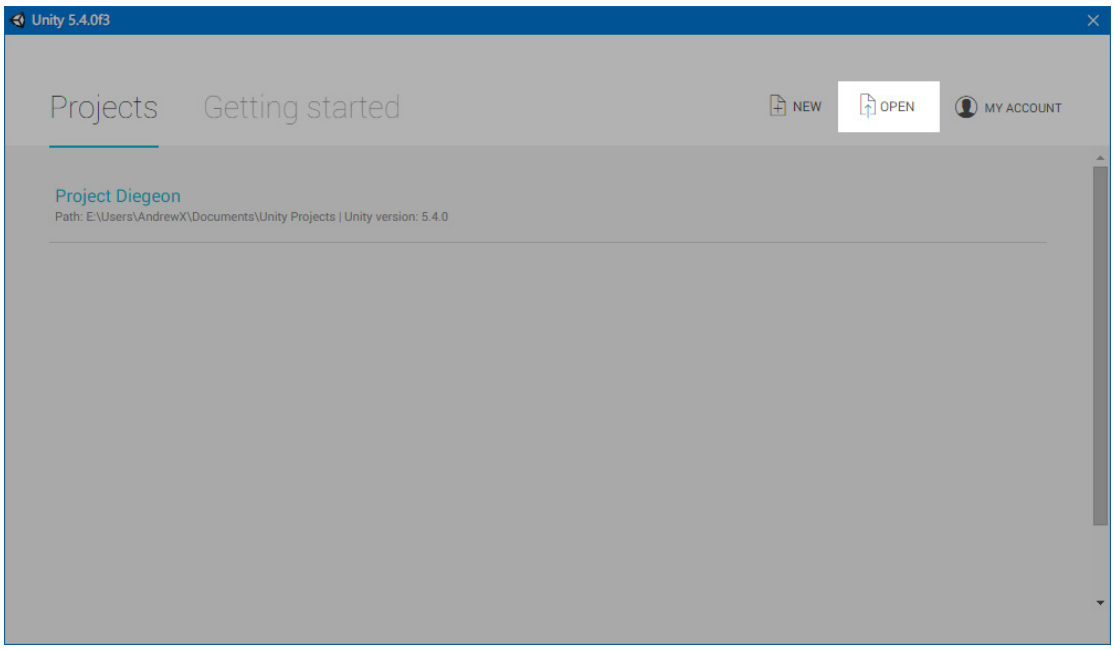


Image 5 Import the project

4 Functional Testing

To maintain the high quality of the product, functional testing is essential during the whole project development process. In this section, we illustrate the testing method, test cases and report the test results of our final product.

4.1 Test Method

Project Diegeon is a game project, in which components connect with each other tightly. In addition, the server-side scripts are mainly used for writing data to the database and retrieve data from the database randomly, other than handling static user interactions. Thus, it is not practical and suitable to conduct automated unit testing. In this project, we manually test the functionalities of our product to ensure the product quality.

We integrated both black-box and white-box testing in our manual tests, where we simulated possible user interactions and scenarios, and also generated test cases to cover various branches in our code.

4.2 Test Cases and Results

Detailed test cases and corresponding test results are listed in Table X to X. The test cases are grouped basing on the scenes they applied to in the Unity project.

Table 2 Test cases and results for MainMenu

#	Description	Inputs	Expected Output	Actual Output	P/F	Comments
1	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 640*400	The main menu UI displays correctly; no element is out of the view port and no elements are overlayed	The buttons overlayed with the logo	F	
2	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1280*720	The main menu UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
3	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1920*1080	The main menu UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
4	[UI] Tests that button works correctly	Click the “Explore Mode” button	The ExploreMode scene should be loaded	The results were as expected.	P	
5	[UI] Tests that button works correctly	Click the “Build Mode” button	The BuildMode scene should be loaded	The results were as expected.	P	
6	[UI] Tests that button works correctly	Click the “Credit” button	The credit window should appear	The results were as expected.	P	
7	[UI] Tests that button works correctly	Click the “Exit” button	The game application closes	The results were as expected.	P	
8	[UI] Tests that button works correctly	Click the close button in the	The credit window should close	The results were as	P	

		credit window		expected.		
9	[Audio] Tests that BGM plays correctly	Launch the game	The BGM plays and loops	The results were as expected.	P	

Table 3 Test cases and results for both ExploreMode and TestMode

#	Description	Inputs	Expected Output	Actual Output	P/F	Comments
1	[Player Movement] Tests that the player can move according to the user input	Press WSAD keys	The player moves towards the corresponding direction	The results were as expected.	P	
2	[Player Movement] Tests that the player movement is blocked by walls	When player facing the wall, press WSAD keys	The player cannot move through the wall	The results were as expected.	P	
3	[Player Movement] Tests that the player can aim according to the user input	Move mouse cursor	The player always faces towards the mouse cursor	The results were as expected.	P	
4	[Player Pickup] Tests that the player can pick up items	When the player is equipping nothing, move the player to an item, and press E	The item is picked up, and the player's equipped item changed; corresponding dice is added; all dice get reset	The results were as expected.	P	
5	[Player Pickup] Tests that the player can pick up items	When the player is equipping any weapon, move the player to an item, and press E	The item is picked up, the old equipment is dropped to the ground, and the player's equipped item changed; corresponding dice is added; all dice get reset	The results were as expected.	P	
6	[Player Pickup] Tests that the player can drop items	When the player is equipping any weapon, press G	The equipped item is dropped; corresponding dice is removed	The results were as expected.	P	
7	[Player Attack] Tests that the player can attack properly	When equipping no weapon, click the left mouse button	The player performs default attack towards the direction it is facing to	The results were as expected.	P	
8	[Player Attack] Tests that the player can attack properly	When equipping dagger, click the left mouse button	The player performs dagger attack towards the direction it is facing to	The results were as expected.	P	
9	[Player Attack] Tests that the player can attack properly	When equipping gun, click the left mouse button	The player shoots towards the mouse cursor	The results were as expected.	P	
10	[Player Attack] Tests that the player can attack properly	When equipping no weapon or the dagger, hold the left mouse button	The player attacks continuously	The results were as expected.	P	
11	[Player Attack] Tests that the player can attack properly	When equipping gun, hold the left mouse button	The player attacks continuously, and when bullets run out, automatically reload	The results were as expected.	P	
12	[Player Attack] Tests that the player can attack properly	Attack on the enemy	The HP of the target enemy decreases	The results were as expected.	P	
13	[Player Death] Tests that the player death can be triggered correctly	Let the player takes damage until HP gets 0	The death animation plays, and the player can no longer be controlled; Game Over menu shows.	The results were as expected.	P	

14	[Monster AI – Movement] Tests that the monster can move properly	The player moves into the monster's sight	The monster starts to chase the player	The results were as expected.	P	
15	[Monster AI – Movement] Tests that the monster can move properly	When the monster chasing the player, move out of its sight	The monster returns to its original position	The results were as expected.	P	
16	[Monster AI – Movement] Tests that the monster can move properly	When the monster chasing the player, a wall blocks the monster	The monster cannot move through the wall, and it will move around the wall	The results were as expected.	P	
17	[Monster AI – pickup] Tests that the monster can pick up items properly	When a weapon is placed within the sight of a monster	The monster will pick up the weapon	The results were as expected.	P	
18	[Monster AI - Attack] Tests that the monster can attack properly	When the monster is equipping no weapon, move the player into its attack range	The monster performs default attack towards the player	The results were as expected.	P	
19	[Monster AI - Attack] Tests that the monster can attack properly	When the monster is equipping dagger, move the player into its attack range	The player performs dagger attack towards the player	The results were as expected.	P	
20	[Monster AI - Attack] Tests that the monster can attack properly	When the monster is equipping gun, move the player into its attack range	The player shoots towards player	The results were as expected.	P	
21	[Monster AI - Attack] Tests that the monster can attack properly	The player gets hit by the monster	The HP of the player decreases	The results were as expected.	P	
22	[Monster AI - Death] Tests that the monster death can be triggered correctly	Attack the monster until its HP gets 0	The monster death animation plays, and then the monster disappears	The results were as expected.	P	
23	[Trap] Tests that the spike trap works	A room with spike trap; Move the player to the spike trap	The spike trap activates, and after short time, spikes launches. If the player is still on the trap, it gets damage and will be pushed back	The results were as expected.	P	
24	[Trap] Tests that the arrow trap works	A room with arrow trap; Move the player into its attack range	The arrow trap shoots arrows. When the player touches the arrow, it gets damage	The results were as expected.	P	
25	[Trap] Tests that the p trap works	A room with spike trap; Move the player to the pit trap	The pit trap activates, and after short time, the floor breaks. If the player is still on the trap, game is over	The game over triggered, but the player HP did not changed	F	
26	[Dice] Tests that the dice menu can be displayed	Hold the space bar	The dice menu should appear	The results were as expected.	P	
27	[Dice] Tests that the dice can be rolled	While in the dice menu, press the roll button	All dice gets rerolled, and the player stats returns to the original values	The results were as expected.	P	
28	[Dice] Tests that the dice can be rolled	While in a room, roll a second time	The player loses one free dice, and all dice gets rerolled, and the player	Free dice is not lost	F	

			stats returns to the original values; if the player has no free dice, it cannot reroll			
29	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the free dice to any slot	The dice is assigned to that slot, and corresponding player stats increases	The results were as expected.	P	
30	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the armor dice to armor slot	The dice is assigned to that slot, and corresponding player stats increases	The results were as expected.	P	
31	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the armor dice to slots other than the armor slot	The dice cannot be assigned to that slot	The results were as expected.	P	
32	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the strength dice to strength slot	The dice is assigned to that slot, and corresponding player stats increases	The results were as expected.	P	
33	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the strength dice to slots other than the strength slot	The dice cannot be assigned to that slot	The results were as expected.	P	
34	[Dungeon] Tests that the room is rendered correctly	Enters a new room	The room layout is correct; the player is placed at the entrance, and there is an exit; the current level and room number is displayed correctly	The results were as expected.	P	
35	[Dungeon] Tests that the room is rendered correctly	Move the player to the exit	The next room is loaded	The results were as expected.	P	
36	[Dungeon] Tests that the level is generated correctly	Start the game and run through several rooms	The player starts at the start room of level 1; after completing all rooms in the current level, it reached the end room, and there is a reward item that can be picked up; when the player touches the stairs, it goes to the next level	The results were as expected.	P	
37	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 640*400	The game menu UI displays correctly; no element is out of the view port and no elements are overlayed	The dice menu was out of the view port	F	
38	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1280*720	The game menu UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
39	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1920*1080	The game menu UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
40	[UI – Pause Menu] Test that pause menu can be triggered correctly	In Explore Mode, press ESC	The pause menu appears; if the player is currently in an end room, the “Reset Room” button is deactivated	The results were as expected.	P	
41	[UI – Pause Menu] Test that pause menu can be triggered correctly	In Explore Mode, and when the pause menu is already shown, press ESC	The pause menu disappears	The results were as expected.	P	

42	[UI – Pause Menu] Test that pause menu can be triggered correctly	In Test Mode, press ESC	The test mode ends and returns to the build mode	The results were as expected.	P	
43	[UI – Pause Menu] Tests that button works correctly	Click the “Resume” button	The pause menu disappears	The results were as expected.	P	
44	[UI – Pause Menu] Tests that button works correctly	Click the “Reset Room” button	The room is reset and the player returns to the entrance	The results were as expected.	P	
45	[UI – Pause Menu] Tests that button works correctly	Click the “Main Menu” button	Returns to the main menu	The results were as expected.	P	
46	[UI – Pause Menu] Test that pause menu can be triggered correctly	In Explore Mode, the player is dead	The pause menu appears; if the player currently has no free dice, the “Retry” button is deactivated	The results were as expected.	P	
47	[UI – Game Over Menu] Tests that button works correctly	Click the “Retry” button	The room is reset; the player’s health restored; the player lost one free dice; the player is moved to the room entrance	The results were as expected.	P	
48	[UI –Game Over Menu] Tests that button works correctly	Click the “Main Menu” button	Returns to the main menu	The results were as expected.	P	
49	[Audio] Tests that BGM plays correctly	Enter the Explore/Test Mode	The BGM plays and loops	The results were as expected.	P	
50	[Audio] Tests that BGM plays correctly	Let the player die	The game over BGM plays	The results were as expected.	P	
51	[Audio] Tests that SE plays correctly	Enter the Explore/Test Mode	The SE for player movement, attacks, objects plays correctly	The results were as expected.	P	
52	[Audio] Tests that SE plays correctly	Decrease the player’s HP to low level	The heartbeat SE plays and loops	The results were as expected.	P	

Table 4 Test cases and results for ExploreMode only

#	Description	Inputs	Expected Output	Actual Output	P/F	Comments
1	[Dice] Tests that the dice menu can be displayed	Hold the space bar	The dice menu should appear	The results were as expected.	P	
2	[Dice] Tests that the dice can be rolled	While in the dice menu, press the roll button	All dice gets rerolled, and the player stats returns to the original values	The results were as expected.	P	
3	[Dice] Tests that the dice can be rolled	While in a room, roll a second time	The player loses one free dice, and all dice gets rerolled, and the player stats returns to the original values; if the player has no free dice, it cannot reroll	Free dice is not lost	F	
4	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the free dice to any slot	The dice is assigned to that slot, and corresponding player stats increases	The results were as expected.	P	
5	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the armor dice to armor slot	The dice is assigned to that slot, and corresponding player stats increases	The results were as expected.	P	
6	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the armor dice to slots other than the armor slot	The dice cannot be assigned to that slot	The results were as expected.	P	

7	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the strength dice to strength slot	The dice is assigned to that slot, and corresponding player stats increases	The results were as expected.	P	
8	[Dice] Tests that the dice can be assigned to the stats slots correctly	After dice gets rolled, drag the strength dice to slots other than the strength slot	The dice cannot be assigned to that slot	The results were as expected.	P	
9	[Dungeon] Tests that the room is rendered correctly	Enters a new room	The room layout is correct; the player is placed at the entrance, and there is an exit; the current level and room number is displayed correctly	The results were as expected.	P	
10	[Dungeon] Tests that the room is rendered correctly	Move the player to the exit	The next room is loaded	The results were as expected.	P	
11	[Dungeon] Tests that the level is generated correctly	Start the game and run through several rooms	The player starts at the start room of level 1; after completing all rooms in the current level, it reached the end room, and there is a reward item that can be picked up; when the player touches the stairs, it goes to the next level	The results were as expected.	P	
12	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 640*400	The game menu UI displays correctly; no element is out of the view port and no elements are overlayed	The dice menu was out of the view port	F	
13	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1280*720	The game menu UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
14	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1920*1080	The game menu UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
15	[UI – Game Menu] Test that pause menu can be triggered correctly	In Explore Mode, press ESC	The pause menu appears; if the player is currently in an end room, the “Reset Room” button is deactivated	The results were as expected.	P	
16	[UI – Game Menu] Test that pause menu can be triggered correctly	In Explore Mode, and when the pause menu is already shown, press ESC	The pause menu disappears	The results were as expected.	P	
17	[UI – Game Menu] Tests that button works correctly	Click the “Resume” button	The pause menu disappears	The results were as expected.	P	
18	[UI – Game Menu] Tests that button works correctly	Click the “Reset Room” button	The room is reset and the player returns to the entrance	The results were as expected.	P	
19	[UI – Game Menu] Tests that button works correctly	Click the “Exit Game” button	Returns to the main menu	The results were as expected.	P	

Table 5 Test cases and results for TestMode only

#	Description	Inputs	Expected Output	Actual Output	P/F	Comments
1	[Return to Build	In Test Mode,	The Build Mode is loaded	The results	P	

	Mode] Tests that the user can return to the build mode form test mode	press ESC		were as expected.		
2	[UI – Submission Menu] Test that submission menu can be triggered correctly	In Test Mode, move the player to the exit	The submission menu appears;	The results were as expected.	P	
3	[UI – Submission Menu] Tests that button works correctly	Click the “Cancel” button	The Build Mode is loaded	The results were as expected.	P	
4	[UI – Submission Menu] Tests that button works correctly	Click the “Submit” button	The room is submitted to the server, and the record in the database is added	The results were as expected.	P	

Table 6 Test cases and results for BuildMode

#	Description	Inputs	Expected Output	Actual Output	P/F	Comments
1	[UI – Room Size Menu] Tests that works correctly	Start the Build Mode	The room size menu shows up	The results were as expected.	P	
2	[UI – Room Size Menu] Tests that works correctly	Select all combinations of the width and height (i.e. 15*15, 15*25, 15*35, 25*15, 25*25, 25*35, 35*15, 35*25, 35*35), and click “Confirm”	The room of the corresponding size is displayed in the editor	The results were as expected.	P	
3	[UI – Room Size Menu] Tests that works correctly	Click the “Cancel” button	Returns to the main menu	The results were as expected.	P	
4	[Camera Control] Tests that the camera control is correct	Hold the right mouse button and drag or press WSAD keys	The camera follows the mouse/keys; and the room cannot be moved outside the viewport	The results were as expected.	P	
5	[Camera Control] Tests that the camera control is correct	Scroll up the mouse middle wheel or press E key	The camera zooms in; the camera cannot zoom in to too close	The results were as expected.	P	
6	[Camera Control] Tests that the camera control is correct	Scroll down the mouse middle wheel or press Q key	The camera zooms out; the camera cannot zoom out to too far away	The results were as expected.	P	
7	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 640*400	The editor UI displays correctly; no element is out of the view port and no elements are overlayed	The object buttons were out of the view port	F	
8	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1280*720	The editor UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
9	[UI] Tests that the UI elements are displayed correctly under various resolutions	Set the game window resolution to 1920*1080	The editor UI displays correctly; no element is out of the view port and no elements are overlayed	The results were as expected.	P	
10	[UI] Tests that button works correctly	Click the “Floor” button, and draw on the map	The floor tile is constructed at the correct position; all objects at that position is erased	The results were as expected.	P	

11	[UI] Tests that button works correctly	Click the “Wall” button, and draw on the map	The wall tile is constructed at the correct position; all objects at that position is erased	The results were as expected.	P	
12	[UI] Tests that button works correctly	Click the “Pit” button, and draw on the map	The pit tile is constructed at the correct position; all objects at that position is erased	The results were as expected.	P	
13	[UI] Tests that button works correctly	Click the “Monster” button, and draw on the map	The monster is placed at the correct position; if the position is not a floor, it cannot be placed	The results were as expected.	P	
14	[UI] Tests that button works correctly	Click the “Spike Trap” button, and draw on the map	The spike trap is placed at the correct position; if the position is not a floor, it cannot be placed	The results were as expected.	P	
15	[UI] Tests that button works correctly	Click the “Pit Trap” button, and draw on the map	The pit trap is placed at the correct position; if the position is not a floor, it cannot be placed	The results were as expected.	P	
16	[UI] Tests that button works correctly	Click the “Arrow Trap” button, and draw on the map	The arrow trap is placed at the correct position; if the position is not a floor, it cannot be placed	The results were as expected.	P	
17	[UI] Tests that button works correctly	Click the “Area Draw” button, and select an object to draw on the map	The selected object fills the rectangular area that the mouse drew	The results were as expected.	P	
18	[UI] Tests that button works correctly	Click the “Undo” button	The map restores to the previous status	The results were as expected.	P	
19	[UI] Tests that button works correctly	After undo, click the “Redo” button	The map changes to the next status	The results were as expected.	P	
20	[UI] Tests that button works correctly	Click the “Reset” button	The map returns to the initial state	The results were as expected.	P	
21	[UI] Tests that button works correctly	Click the “Test” button	The TestMode is loaded, and the room drawn in editor is shown	The results were as expected.	P	
22	[UI] Tests that button works correctly	Click the “Exit Game” button	Returns to the main menu	The results were as expected.	P	
23	[Audio] Tests that BGM plays correctly	Enter the Build Mode	The BGM plays and loops	The results were as expected.	P	

4.3 Implications from Functional Testing

The final product passed most of the functional tests, only failing some of them. The main problem is that the current UI design does not support low resolutions (e.g. 640*400 pixels). As it is rare that players play the game in such low resolution. To address this issue, we simply need to set a minimum resolution requirement, and disable the low resolution options in the built game client. As stated in the Game Installation Guide, we set the minimum resolution requirement to 1280*720 pixels.

5 User Evaluation Report

A user evaluation section was held on 6 October at the room 209 in Building 78. In this section, the details of the user evaluation and our findings are discussed.

5.1 The Testees

There were 6 users involved in this evaluation section. All of them were randomly selected to participate this event. They all agreed to play with our game and answer our questions and share their ideas. Detailed information about the participants are shown in table below.

Table 7 Five participants, having the following characteristics, evaluated Project Diegeon.

Education

Undergraduate	2
Postgraduate	3
TOTAL (participants)	**5**

RPG Experience

Yes	3
No	2
TOTAL (participants)	**5**

Age

18-25	3
26-39	2
TOTAL (participants)	**5**

Gender

Women	2
Men	3
TOTAL (participants)	**5**

5.2 Test Methods and Findings

5.2.1 Workshop

This workshop aims to collect ideas on the current traps and attack system from users, then implement the actionable ones in the next development stage.
The outcomes of users' suggestions for bettering trap and attack system will be drawn on an A2 size paper.

5.2.1.1 Test Procedure

Step 1
The facilitator introduced the workshop purpose to the users and asked their permission for taking photos.

Step 2
Demonstrate the game to the users, and evenly divided the 6 participants into 2 separate groups.

Step 3
Ask users to experience both of the Build Mode and Explore Mode of the game in turn.

Step 4
After that, two key questions focusing on traps and attacks system were asked, which were

- What do you think about the traps? What other traps would you like to see in this game?
- What do you think about the attacks? What other attacks would you like to see in this game?

Then the users would follow facilitator's instructions to write/draw their ideas on the answer sheets that we provided.

The facilitators were there to provide assistance and observe users' behaviour and interaction with the system. The whole workshop run for roughly 45 mins and the facilitator would ask users to stop in order to catch the schedule. Some snacks and drinks were provided during the workshop.

Detailed timetable can be found in Appendix 1 Table **Workshop Timetable**.

5.2.1.2 Test Results and Findings

Overall the current design of the traps and attacks were favourable to the users. Also, users expressed high enthusiasm to share their ideas of adding various traps to make the game more interesting. However, due to time limitation some creative suggestions to the trap system might not be able to implement during this course.

Most users mentioned the trap types are limited. They've suggested a volume of creative traps for our reference. Three of the users expressed the same thought on the monotony of the attack system. They would like the system to be upgraded with diversity. Dice mechanics on the other hand received positive feedback. Detailed ideas are listed in Appendix 2 User Evaluation Feedbacks.

5.2.1.3 Development Recommendations

The recommendations for this workshop section will be implementing the above ideas as more as possible in the future development stage.

5.2.2 Usability Testing

Usability test is performed to evaluate the product in terms of the intuitiveness of UI elements, to measure its usability by timing user's cognitive process.

5.2.2.1 Test Procedure

Step 1

Users are asked to perform tasks by themselves.

1. Build your own game room.
2. Test the room you just built and submit it.
3. Switch your role to play the room built by others.
4. Kill a monster.
5. Use Dice to facilitate your play.

Step 2

Each performance is clocked and the result is recorded.

5.2.2.2 Test Results and Findings

The test results are shown in Table X:

Table 8 Time spent for performing tasks

User	User 1(s)	User 2(s)	User 3(s)	User 4(s)	User 5(s)	AVERAGE(s)
Task 1	66	8	33	32	86	45
Task 2	107	92	88	53	74	82.8
Task3	10	8	5	3	5	6.2
Task4	15	48	25	15	47	30
Task 5	28	52	18	15	53	33.2
Finish Time	270	370	270	180	310	288

(* All the time above is net time that users operate the product. The performance is clocked right after making sure that the testees understand the question.)

Form the test, following issues are found:

- The button for drawing rectangle is unclear to 3 players. They asked the facilitator for its function.
- The performance varies according to player's past experience in gaming. For example, when performing task 4, inexperienced gamers spend an extra 25s on average to pick up a weapon on the ground. Four of them asked the facilitator for help.
- Two of the testees expressed confusion on dice unable to be removed from the dice slots after being allocated. The facilitator had to explain to them it was not a bug but

intentionally to be made so.

5.2.2.3 Development Recommendations

- Tooltips that indicate the functionalities of the UI buttons are needed in build mode.
- A tutorial is necessary for inexperienced gamers to get familiar with the controls.

5.2.3 Questionnaire

The questionnaire is designed to gather user's overall impression towards the game. Outcomes include user response to the questionnaire.

5.2.3.1 Test Procedure

After user experienced the whole game, they were asked to complete the questionnaire to express their ideas to our game. Detailed questions are listed below.

1. What parts of the game did you like?
2. What parts of the game did you dislike? And how you would like to change it?
3. Are there any features that confused you?
4. What is your impression of the Main Menu?
5. What did you think of the game overall in terms of playability?
6. What did you think of the challenge level of the game?
7. Did you have a moment that you want to drop off from the game?
8. Would you like to play this game again?

5.2.2.2 Test Results and Findings

"What parts of the game did you like?"
First impressions of the game were very favourable, especially the "Build your own game" mode. The attitude towards game difficulty level differed among users due to their different game strategies. Several features need to be further developed to clear users' confusion.

"What parts of the game did you dislike? And how you would like to change it?"
Some users found the game too easy to play, but some mentioned it was too hard to continue. This is because the game system hadn't been completed yet, namely the atomic level generation mechanics and the reward system. Once these two features were finalised, users would be sent to appropriate game rooms according to their account levels.

Another disliked feature was using “E” to pick up the weapon. Two participants suggested to replace this with auto pick up action. However, we decided to keep the current status since this operation could give player the freedom to choose the weapon they want and continue to use the one they like.

“Is there any features that confused you?”

Three users mentioned they had no idea how to pick up the weapon. Two users said they didn’t know whether they died or not and felt confused about how to restart the game. One mentioned that didn’t know what the dice system is for.

Question 4 to 6

The majority of users felt satisfied with the game. But most participants suggested the game difficulty level needed to be upgraded. The average rating for the three aspects can be seen below.

Table 9 Average Rating for question 4~6

ITEM	SCORE (Out of 5)
Main Menu Impression	3.6
Overall Playability	3.8
Challenge Level	2

Question 7 and 8

None of the testee had a moment that want to drop off from the game during the play test, and 4 of the participants thinks that the game is quite engaging. When asked whether would like to play the game a second time, 80% of the testees answered “Yes”, and the other participant gave us several suggestions for improving the playability of the game.

The graphs of the results of question 4 to 8 can be found in Appendix 3 Questionnaire Results. And the user feedbacks are listed in Appendix 2 User Evaluation Feedbacks.

5.2.3.3 Development Recommendations

We need a rewarding system such as levelling up to motivate users for more playing; monsters with different levels of AI are needed to enrich the difficulty; it is better to have an erase icon to remove objects for users who couldn’t figure out the ‘delete’ button’s function themselves. We also need to prompt the user after their character is dead.

6 Project Reflective

In this section we will be taking a look at the overall process that we as a team undertook to turn the design document that we were handed at the beginning of the semester into a fully-fledged computer game. We will take a look at what aspect of the game we considered to be important, where we think we succeeded and where we consider ourselves to have failed. Finally based on the last fourteen weeks we will look to the future and postulate where the project could go from here. But before we begin we should take a look at the project and try and understand where we began.

6.1 Understanding the Scope of the Project

Over the last 14 weeks, Team 6-Bit has been working on developing a Roguelike single-player collaborative computer game called Project Diegeon. We were handed a design document that outlined the main features and suggestions on how to start implementation. The goal of the project was to develop a program for the client Jason Weigel, who wanted to create a game that served two main purposes.

The first purpose was to create a game experience where players could collaborate together to come up with new and interesting ways to challenge each other online. This meant that rather than creating a program that could procedurally generate a dungeon and then load in assets based on predefined algorithms, the rooms that the player faced, would in fact be designed by other players. This meant that the rules of the game were not set by the developer, but rather by the players themselves. It was our goal to simply provide them with the tools to do so.

The second most important goal was that the game could also serve as a creative entry point into the design process. Where players could start to learn how to develop games, this would take the form of a level editor that allowed users to create their dungeons. Ultimately this would lead to people being able to learn how to use the Unity game engine through modding in new content into the game.

It was clear to see that due to the nature of the project the overall scope was truly immense. On top of this we were also handing over the project under the open source license so everything had to be developed by ourselves. This meant that everything from programming to sound design and character animation had to be developed in house. The immense scope meant that we needed to have a fast paced iterative cycle that had us constantly moving forward without much time to pause and take stock of what we had already achieved.

6.2 Team Management and Process

Before we even started development for Project Diegeon, we had two main tasks to complete. First we had to plan out what each section of the game based on the meetings with the client and the document we were handed at the beginning of the semester. We had to develop class hierarchies, user stories and even assets lists such as sound effects and animation lists. We did this for two main reasons. The first was to make sure that everyone involved with the project, including the client were on the same page in relation to what the team would be delivering. The second was to highlight any key features that might cause potential issues down the line.

Based on the outline that we generated we listed the three core elements that we felt needed to be achieved for us to consider the game a success. The first was that the game had to be fun and complete. This meant than rather being able to build just one element of the project we had to build everything to a basic prototype level.

Secondly the game had to promote collaboration. This meant that we needed to have a server backend that allowed people to share their creations with each other. This meant developing a database that could not only store the details of the room, but also all of the players' metric data.

Finally, the client had to be able to take our project and build upon it. This meant we needed to develop the game in modular way that allowed for the easy addition of new content. Once these three core elements were defined as the core of our project we could use them as a guide and focus on what was important and move ahead with confidence.

The next step was to work out how we were going to start the processes and what development and communication pipelines we would eventually end up using during the semester. This meant working out how we were to communicate data to each other, how we would communicate online and what meetings we would have. All of this was written down in the team charter.

Back in Week 1, when we first formed the 6-Bit team, we were told that we had to adopt the SCRUM framework to help us manage our development processes. Since very few of the team members had experience working with SCRUM we were a little concerned that the framework might hinder development instead of aiding us. We feared that by implementing such a system would cause miscommunication in the team.

While adopting this system allowed us to remain agile and let all members of the team work in parallel with each other, it did introduce a few problems. The main one was cause by our lack of experience with SCRUM and our lack of experience working with each other. At the beginning we would suffer from miscommunication between team members. People would take cards and work without communicating with each other. We even had

a case where two people were working on different cards, but the same tasks. This came about from a problem with people creating their own cards and placing them into the sprint backlog. This problem was compounded with a project as complex as the one we were undertaking.

To help combat this we decided to implement a two-week sprint cycle, culminating with a meeting with the client. This meant that if we started to stray away from the core features of the game or we started to work in different directions we could be brought back on track by the each other and the client. It was these client meetings that allowed us to review our work for the previous two weeks and judge whether or not we were heading in the overall right direction.

We also increased the number of daily stand-ups from once a week to three times a week. These daily stand-ups served to increase communication between members and let each of us know what the other was doing. After a couple of weeks, we managed to settle into our established roles with each member selecting an area of development that they felt confident in developing.

To prepare for these client meetings we would merge our code together and come up with a stable alpha prototype that we could show the client Jason. This served as another check and balance to make sure that everyone was on the same page when it came to programming and to make sure that everyone was working on the latest build. It also allowed us to get feedback on our progress and to make sure that the client was happy with the work that we were doing.

The reason that we had to merge our code was because of the Unity game engine. The server that we were given allows us to store non binary files such as C# scripts. However, the Unity files themselves were not allowed to be stored on the server due to the fact that the file were considered binary files. If one person made a change to a unity file, then it would affect everyone. This meant we needed to keep these files local and only share the code with each other. To help combat this we would have what we called nightly builds that existed on a person's local computer. These builds were used to showcase certain features within the team. We would also create a stable alpha builds that we used for testing and to show to the client.

These demonstrations would take place at the client meeting at the end of the sprint. At this meeting we would talk with the client and discuss what we achieved, check if we were on the right track and with the client we would decide what the next tasks would be.

From this meeting we would generate cards and place them into the sprint backlog. These cards were placed into Trello which served as our tracking and management system. This fortnightly cycle of prototype and test meant that we could notice problems in both the design of the game, as well as the development pipeline early on and adapt quickly.
When you are creating a game it is not a matter of front end and back end. Every system

from the main combat, to enemy AI to the user interface interacts with each other. Without constant communication between team members, we would end up developing individual systems that could not engage with one another and essentially end up spending more time merging are work together than actual programming. This happened a lot at the beginning while we were programming the core features of the game.

It should also be mentioned that this was the first time that most of us had worked with the Unity game engine. Not only did this mean we had to work out a pipeline for sharing files with each other but it also meant that most of time we were developing features not knowing if they were going to work or not. Many times we decided to implement a feature or animate the characters in a certain way, only to come across a problem that limited us and forced us to waste time retracing our steps. The entire project was a learning experience and while this allowed us to grow as students it did cause us to go backwards rather than forwards. Ultimately what it boiled down to was that when we had to fix a mistake we needed to cut another asset from the game.

At the end of the day, the only reason we succeeded as well as we did was because we kept it simple. We focused on our core tenants of fun, complete, collaborative and serve as a base platform for further development. Instead of spreading the workload across a wide range of features we focused on developing our core features, arguing that they could be added later by the client only if we had a core system in place. It was our ability to focus on the important gameplay, adapt in the face of adversity and communicate efficiently within the team and with the client that allowed us to overcome our inexperience and develop a solid prototype that met the client's requirements.

While we failed many times during the process, we did manage to recover from most of it through testing and trial and error. However, there was one aspect of the process that we did not bounce back from. Near the end of the project we started to suffer from a lack of enthusiasm. To put it simply we started to burn out. We had already completed the minimum viable prototype (MVP) and we started to put less time into the project. The main reason was that we had to start focusing more on documentation such as the MVP video and the subsequent website. On top of this many other assignments from other subjects started to take the focus of the team members. But it was the interruption to our weekly process that caused us to lose the most focus and get side-tracked. However, this is not an easy problem to fix. Part of it is the nature of university, another part is due to our lack of experience. But the solution is not an easy one, it requires practice. In the future with stronger team dynamics and a better understanding of the process we can combat this together as a team. But besides all of the pitfalls and troubles we faced we still managed to get a final working prototype ready for the final presentation.

6.3 Overall Design and Final Product

The overall scope and design of Project Diegeon and its dual nature of gameplay and

level editor, lead interesting problem during the development process. There were three main areas of development that we had to divide up amongst the team. The first we dubbed explore mode that had the player controlling an avatar and moving them through the dungeon. It required Monster AI, traps, sound effects, music, art and animation to get it to a polished state that we could show it to others. The second mode we called build mode. Here we needed a user interface that would allow users to place down traps, and monsters. Finally, we needed to communicate between the two systems. Which meant we needed to implement a back end that could store the rooms on the server and load them dynamically into the game based on their difficulty rating.

The problem we faced was that we could not get away with just building a snapshot of one of these three elements. If we wanted to succeed in building Project Diegeon we needed all three. The potential for scope creep was rather high and remained a huge worry to us throughout the development process.

However, thanks to the SCRUM framework and the burndown chart we knew early on that we could not build the entirety of the game as it was initially pitched to us. To solve this, we needed to cut back and deliver a solid blank slate that showcased everything at a basic level. This decision was made four weeks after we started development and was approved by the client.

But even though we cut back on several key features we still ran out of time. This hit us the hardest with non-programing tasks. Things like animation and sprite assets, music and sound all had to be put aside while we continued to develop the core features. In fact, it could be argued that we left it to long as most of the planned features in relation to these assets was actually cut. We knew that the end result would be lacking in content, but it would showcase the value of the project and create a fun and engaging experience for the player as well. While this meant that the final product would not be considered complete we could still consider it a success.

6.4 Future Development

Despite the fact that we finished developing a complete prototype containing all the key gameplay elements from exploration to building, there was still a lot of work that still needs to be done in relation to the project. Some of it is simple polish such as user tutorials and better UI, but most of it comes in the form of more content and control. The current prototype only contains one class, one avatar, one monster and a couple of traps. This basic prototype needs a lot more content in the future. But this is not where we suggest the client should focus their time. We feel that the most important next step is giving more control to the players.

During the last month of development, we started testing with users and running QA sessions. These informed us how well the games were being received but it wasn't till we ran a creative workshop in the last month that we found out the true drive behind our

builders. During this workshop we discovered that while builders enjoy the room creation process, they truly shine in coming up with news ways of challenging the player. They didn't just want to be able to create a room, they wanted to create their own traps and monsters. They wanted to draw and animate new monsters to coming up with different kinds of interesting traps.

The second most important thing to work on would be creating a online community. This would incorporate giving each user a unique log on and the ability to communicate and compete in some form or another. The easiest form would be a high score board for both builders and explorers. But it could also include the online forums, social media and even online communication. This community would serve to motivate the users, help them learn the system and ultimately control and balance the rules of the game.

Finally, we would focus on the data gathering side of the project. The client Jason wanted to be able to collect data about collaboration. We would increase the amount of data that we could collect and present it in an easy to understand method. The ability to create and customise, the online community and the tracking and display of metric data. No matter how many new assets you placed in, until you developed these three assets you couldn't take the game any further.

6.5 Conclusion

We were able to achieve as much as we did by having a clear goal from the very start and by adapting quickly when we stumbled or headed in the wrong direction. However due to our lack of experience as a team and our lack of experience working as game developers we ended up backtracking quite a bit. Our lack of experience also leads to a loss of productivity as time went on and as other external matters started to press on our time. But at the end of the day we were able to produce a bare bones prototype that showcased the core features of Project Diegeon.

7 Code Guide

The final product delivery includes the whole Unity project folder as well as the server-side scripts. In this section, an overview of the project files is provided. There are two main sections in our source code: the server scripts and the Unity project.

7.1 Server Scripts

game_server folder Contains PHP scripts that are used for handling HTTP requests and the SQL schema for constructing the current database.

File Name	Description
database_config.php	Defines the credentials need for connect to the database, which should be ignored by git.
connect_database.php	Contains a function that connect the game database.
submit_room.php	Handles the room submission requests from the client. Processes the room data and stores them into database.
load_level.php	Generate a random level data according to the level number contained in the POST request.
utils.php	Contains some frequently used functions. Mainly for login and session control.

7.2 Unity Project

The Project_Diegeon_Source folder is the complete Unity project which can be opened using Unity editor. In the project folder, folders other than the Assets folder are generated automatically by Unity editor. Thus, we only describe the Assets folder here. Table X lists the structure of the Assets folder.

Table 10 The structure of the Assets folder

Folder Name	Description
Audio	The audio files used in the game (background music and sound effects).
Fonts	The font file. PressStart2P-Regular is an open source font under SIL Open Font License.
Materials	The materials used on game objects (2D/3D).
Models	The models of the dice.
Plugins	The third party plugin. JsonFx[1] plugin is used for encoding and parsing JSON objects.
Resources	Preloaded dice resource files
Scenes	The scenes used in the game:

[1] https://bitbucket.org/TowerOfBricks/jsonfx-for-unity3d-git

	• MainMenu: the title menu of the game • ExploreMode: the game mode that the players explore the randomly generated dungeons • BuildMode: the game mode that the user creates the room • TestMode: used for testing the built rooms
Sprites	Image files used in the game
Prefabs	Templates for creating the game objects
Scripts	The C# source code used for controlling the game

Table X describes the source files in the Scripts folder:

Folder/File Name	Description
Dice	The scripts related to the dice mechanism, including the UI elements for the dice menu
MainMenu	Scripts that controls the main menu, including the UI Events definitions
RoomEditor	Scripts related to the Build Mode: • EditorManager.cs: Controls the whole logic of the room editor • EditorLoader.cs: Dynamically load the EditorManger into the game • EditorUIEvents.cs: Defines the UI interactions in the editor • EditorCameraController.cs: Controls the camera in Build Mode • TestManager.cs: Handles the logic of the play test • TestUIEvents.cs: Defines the UI interactions in the play test
Shaders	Used for game graphic rendering
UI	Defines the general game UI (in-game menus, health bar, cursor etc.)
Trap	Scripts that define and control traps
Weapon	Scripts that define and control weapon items
AI3.1	Defines and controls the monsters AI
AccountManager	An interface that handles the user account.
BoardManger	Controls the core logic of rendering a room map
CameraController.cs	Controls the camera in Explore Mode
Player.cs	Controls the player movement and attack etc.
PlayerOnEntranceExit.cs	Controls the events when player enters the entrance and exit
GameManager.cs	Controls the whole logic in the explore mode
Loader.cs	Dynamically load the GameManager into the game
RoomData.cs	The class that represent the data structure of a room
SoundManager.cs	Controls the audio in the game
Utils.cs	Some frequently used static functions (e.g. coordination and array index conversion)

All scripts are labelled with self-explanatory file names, and their functionality and usage can be found in the code comments.

8 Functional Coverage

Table X itemises the functionalities that are within the scope of the project, and their coverage status in the final product. The functionalities that are yet to be implemented are labelled with red colour.

Table 11 Functional coverage in final product

Category	Functionality	Description	Required by client	Status	Note
General	User sign-up	New user can create a new account		Leave Out	
	User log-in	User can login to the game		Leave Out	
	Main menu	The title menu of the game		Completed	
	In-game menus	Various menus in the game		Completed	
	Audio	Play BGM and SE in the game		Completed	
Explore Mode	Player movement	The user can move the game character using WSAD keys	Yes	Completed	
	Player attack: melee	The user can use mouse to execute short range attack	Yes	Completed	
	Player attack: long range	The user can use mouse to aim and execute long range attack		Completed	
	Player class: basic	The basic player class: rogue	Yes	Completed	
	Player class: advanced	Other player classes: hunter, mage, thief		Leave Out	The class system is changed to the weapon system
	Player growth	Player can level up and get stronger		Leave Out	
	Items	Items (new dices) can be acquired		Completed	
	Monster	The basic monster: zombie	Yes	Completed	
	Monster AI	The monsters have basic intelligence when moving, attacking	Yes	Completed	
	Traps: basic	The basic traps: spike trap, arrow trap etc.	Yes	Completed	
	Dice: roll	The user can roll dices in the room	Yes	Completed	
	Dice: manage	The user can drag dices to different slots	Yes	Completed	
	Dice: effect	The dice in slots will give advantages to the player	Yes	Completed	
	Dice: durability	The dice get damaged when rolling, and can be broken	Yes	Completed	Player lose dice when HP reaches 0 or roll dice multiple time in a room
	Dungeon: room	A game room can be constructed from given room data	Yes	Completed	
	Dungeon: level	A level containing multiple rooms can be generated from the given room data	Yes	Completed	
	Send play record	Send the play record when the player completes a room or dies in a room		Leave Out	
	Retrieve room data	Communicate with the game server and retrieve the room data	Yes	Completed	
	Score system	The user gets player scores for completing rooms	Yes	Leave Out	
Build	Editor: various	The user can build rooms in		Completed	

Mode	sizes	different sizes			
	Editor: add objects	The user can place objects to the room	Yes	Completed	
	Editor: set trigger	The user can set trigger for trap objects		Leave Out	
	Editor: modify objects	The user can move/change trigger/remove objects on the map		Leave Out	
	Editor: history	The user can redo or undo		Completed	
	Editor: reset	The user can reset the room		Completed	
	Playtest	The user can test the room he/she built	Yes	Completed	
	Save room	The user can save unfinished room locally		Leave Out	
	Submit room	The user can submit room to the server	Yes	Completed	
	Room statistics	The user can view the play records of the room he/she submitted		Leave Out	
	Score system	The user gets builder scores for trapping other players in his/her room	Yes	Leave Out	

9 System Architecture Diagram

Project Diegeon utilises a typical three-tier system architecture, which consists the game client application, the game server, and the database. No external API is need. Figure X demonstrates the system architecture of the project including the core functionalities. Unimplemented user stories are coloured in grey.

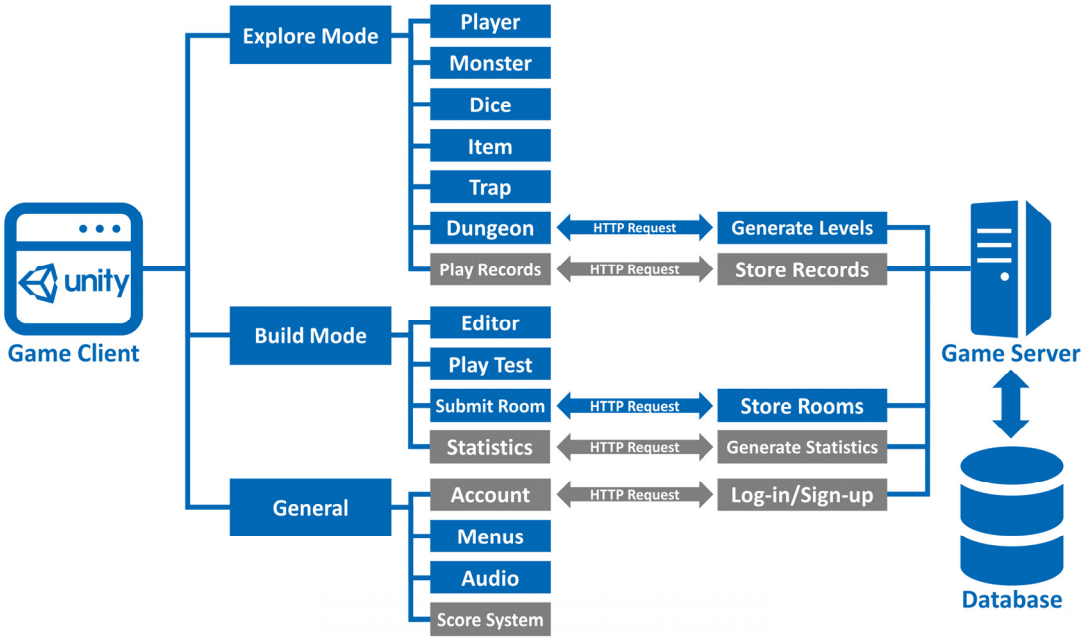


Figure 1 Project Diegeon System Architecture Diagram

9.1 System Overview

Game Client: The Unity game application will run on the user's PC. There are two main modes that the client has access to. In Explore Mode, the user explores procedurally generated dungeons, while in Build Mode, the user can build their own dungeon rooms.
Game Server: The game server is used to handle requests from the client. When the user requests commence, server-side scripts will be executed and the server will send back corresponding response to the client application. Where necessary, it will communicate with the database server to exchange the data needed. Scripts on the server are written in PHP.
Database: The game database stores the user accounts, play records and the data of the rooms. In this project, MySQL is used as the database management system.

9.2 Unimplemented Functionalities Description

In Project Diegeon, we implemented the product according to the user stories prioritised by our client. The core gameplay and level designing functionalities are complete. However, due to the time limitation, some user stories are yet to be implemented. These unimplemented functionalities are mainly for user data analysis, and are depend on the account system.

The end-user needs to create an account to sign in, so that the play records and the room he/she created will be stored under this account. Currently, the database supports the user accounts, and the relations between accounts and play records and rooms have been established. However, the server-side and client-side implementation is not completed. At the current stage, all room submitted to the server are stored under a default account (6bit).

Appendix

Appendix 1: User Evaluation Workshop Timetable

Time	Activity	Materials	Outcome
0:00	Greetings; Introduction explaining the game concept; Walk through the game components; Today's plan & Goal of this workshop	2 PCs with Game setup; Game Demo	None
0:05	Break into 2 groups and 3 members for each. Let participants play the Build Mode.	2 PCs with Game setup; Game Demo	Screen recording
0:10	Let participants play the Explore Mode.	2 PCs with Game setup; Game Demo	
0:15	Ask the users to write/draw their ideas of traps and attacks on the provided A2 sheets.	A2 blank sheets	Workshop notes
0:25	Each group present their messages and discuss.	Workshop notes	
0:45	Thank the participants --End--		

Appendix 2: User Evaluation Feedbacks

Interview Feedbacks

Trap System

- When you step on a floors, there will be a monster fall from the sky;
- Sliding ice on the floor, you'll be moving super-fast that you cannot control;
- The monster will mimic your movement (called stalker Awais);
- A flash bomb that if you step on it, you'll only be able to see limited sight in the dungeon;
- A chaos bomb that will reverse your control of movement;
- A reverse bomb that will turn your whole interface upside down;
- A bog that will slow down your movement, that will be easier for the monster to chase after you
- Firestorm
- Leader board / Reward system
- Ocean of grief / Deadly water
- Disease
- "Paralysed" / Cannot move
- Atomic bomb that kills all monsters
- Working dead with body parts falling apart
- Evil fly / Deadly fly / Bee evil bee
- Slower tiles where you move slower
- A "surprise" creature like the flowers in Super-Mario
- Mini game trap, a trap where when you step on it you have to play tetris or any other arcade game, once you finish with the mini game you get bonus probably extra dice or new weapon, when you lost you lose health
- I don't really like the lighting trap, coz it seems that it's on all the time, and I found it hard to conquer it.
- At the moment, there is only one type of monster moving around. You can also develop more types of monster such as: one can shoot towards you, one can only do random shooting, one can chase after you faster.

Attack System

- Attacks are pretty simple and they only have one type of attack available for the user
- Laser beam
- Tanks and airplane bomb drops
- Monster can be more aggressive, magic would be neat as well, maiming attack (blind, stun, sleep, etc.)
- The weapon switch mode is interesting and some interesting weapons could be added as my opinion

Dice

- I like the dice way of adding strength and luck and other properties, it provides

a more flexible way for me to customise the ability my character. It would be cool if there is a reward system built in the dice function, allowing player to roll a dice once the player kill certain amount of monster.
- The dice is a neat feature, limiting the reset and adding level up to gain a roll for a single dice to add or replace the status

Questionnaire Feedbacks

Like List

- I like the part with "creation" mode (where you can create your own environment/setting)
- Build your own game
- You can use your weapon to shoot
- Game style
- Exploration
- Interesting Characters
- I like how the game look - in pixelated way, it gave me retro and nostalgic feeling that I will enjoy playing the game as much as what I had in my childhood.
- The level maker is fun
- Killing enemy is also fun

Dislike List

- When a user dies, nothing happens
- It's very fast
- No control menu to change the combat
- Trap are easy to clear
- Zombies are easy to kill
- Easy to win
- Press E to pick up a weapon
- The traps are interesting
- The movement of the character seems not so agile and some undesired result of movement
- The traps sometimes too hard for me to continue
- The pressing "E" part of choosing the weapon, I'd prefer auto pick up.
- Some of the user-generated levels are impossible to pass, it needs restriction like amount of traps and monster can be put in the game.

System Confusions

- Not sure how to pick up weapons
- How to get the arms combat
- If I died in the game, where can I restart the mission?
- Why can I advance in the game without kill all the monsters?
- There was a moment that I'm not sure whether I died or not. Because in the game, when I was informed that I was killed, I could still see my character's feet

moving. It could be added more visual elements on the interface indicating I've been killed.

- The dice is a bit confusing in the beginning

Appendix 3: Questionnaire Results

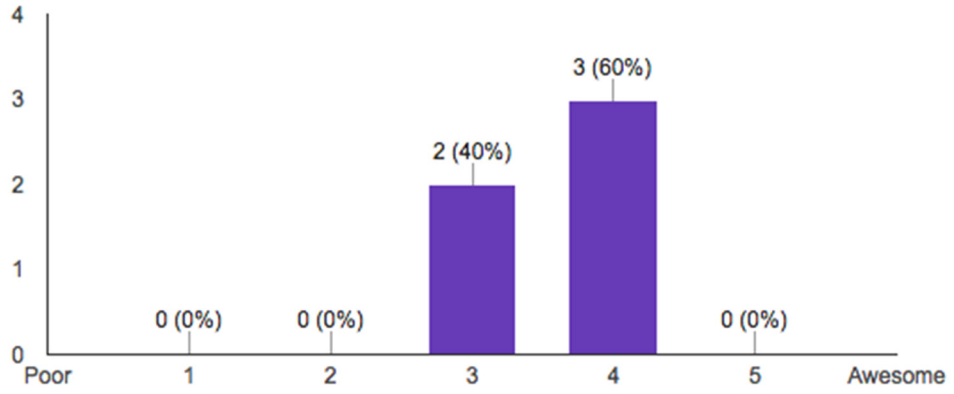


Figure 2 Result for "What is your impression of the Main Menu?"

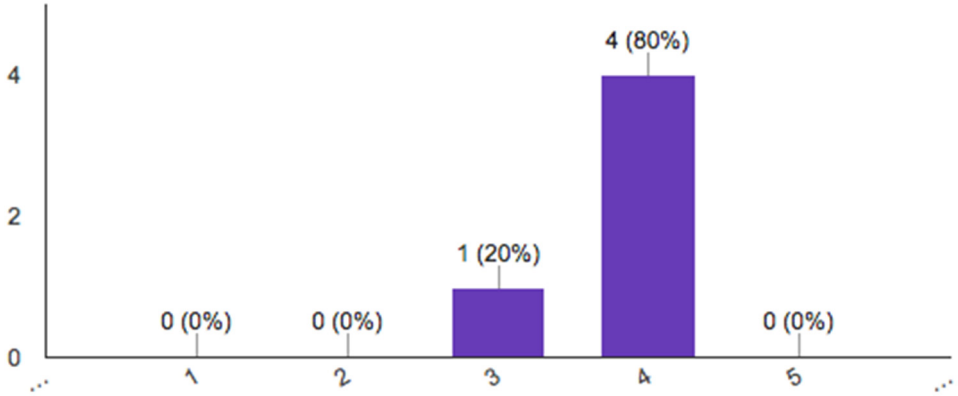


Figure 3 Result for "What did you think of the game overall in terms of playability?"

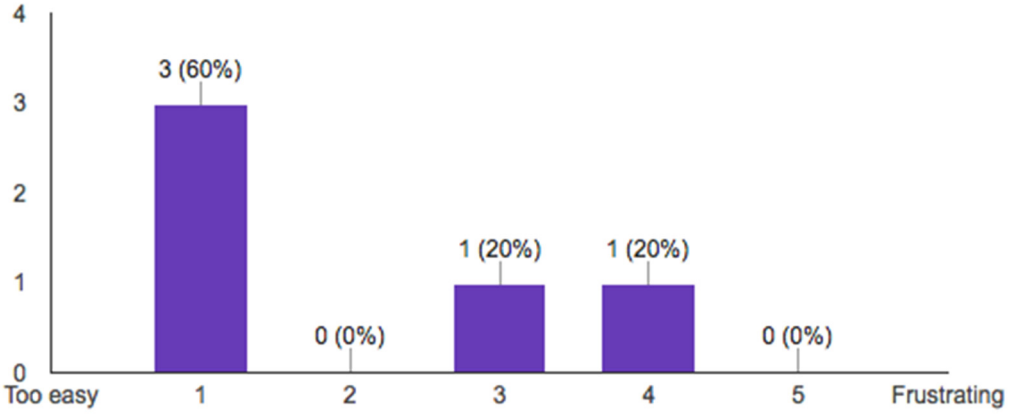


Figure 4 Result for “What did you think of the challenge level of the game?”

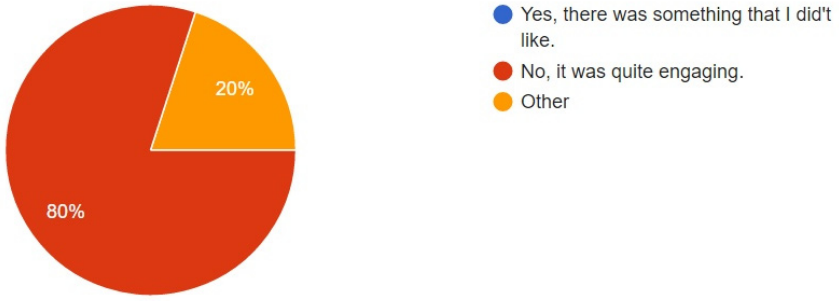


Figure 5 Result for “Did you have a moment that you want to drop off from the game?”

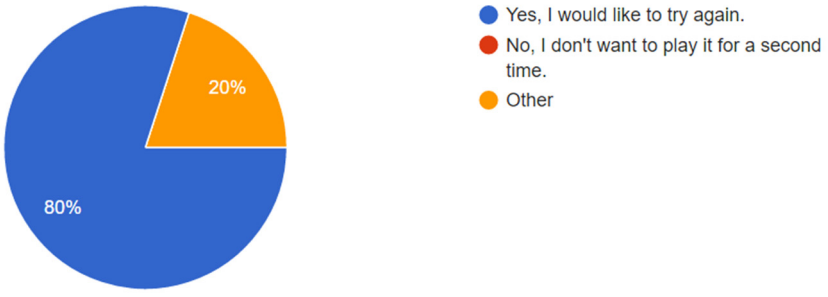


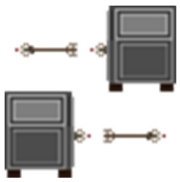
Figure 6 Result for “Would you like to play this game again?”

Appendix 4: Original Art Assets

Table 12 Original Character Sprites

	
Rogue	
	
Hunter (Unused)	Mage (Unused)

Table 13 Original Game Object Sprites

Traps				
	Pit Trap	Arrow Trap	Spike Trap	
Weapons				
	Knife			
Map				
	Ground	Wall 1	Wall 2	Stairs

Appendix 5: Third-party Resource References

Sprites

Monster

URL: https://unity3d.com/cn/learn/tutorials/projects/2d-roguelike-tutorial

Spell Book

URL: https://otland.net/threads/voting-16-spellbook.229083/

Crossbow

URL: http://tibia.wikia.com/wiki/File:Crossbow.gif

Dice

URL: http://xnafan.net/2012/07/

Blood Effect

URL: http://powstudios.com/

Slash Effect

URL: http://powstudios.com/

Fire Flash

URL: http://powstudios.com/

Fire Ball and Explosion

URL: http://powstudios.com/

Bullet

URL: https://16bitjusticesociety.wordpress.com/2010/11/15/game-update-asteroids-enemies-and-keeping-score/

Gun

URL: http://tdeleeuw.deviantart.com/art/Rimfire-A13-Spritesheet-pixel-art-601756393

Reload

URL: https://www.scirra.com/forum/suggestion-a-refresh-button-to-update-gfx_t85967

Arrow

URL: http://www.minecraftforum.net/forums/mapping-and-modding/resource-packs/1232237-dokucraft-the-saga-continues?page=323

Background Music

Echiptian Swaggah

Artist: RoccoW

URL: http://freemusicarchive.org/music/RoccoW/_1035/RoccoW_-__-_02_Echiptian_Swaggah

Accidental Skwugg

Artist: RoccoW

URL: http://freemusicarchive.org/music/RoccoW/_1035/RoccoW_-__-_03_Accidental_Skwugg

Pumped

Artist: RoccoW

URL: http://freemusicarchive.org/music/RoccoW/_1035/RoccoW_-__-_06_Pumped

SwingJeDing

Artist: RoccoW

URL: http://freemusicarchive.org/music/RoccoW/_1035/RoccoW_-__-_04_SwingJeDing

Sound Effect

Arrows.wmv

Splat Crush 01.wmv

Artist unknown

URL: http://www.freespecialeffects.co.uk/pages/fighting.html

WEAPON GUNSHOT Rifle Swish 02.wav

NEGATIVE Failure Descending Chime 05.wav

FOOTSTEPS (A) Walking Loop 01.wav

Artist: gameburp

URL: http://www.gameburp.com/free-game-sound-fx/

Bow_Fire_Arrow-Stephan_Schutze-2133929391.wav

Hearbeat_2-Mike_Koenig-143666461.wav

Artist unknown

URL: http://soundbible.com/tags-game.html

Appendix 6: Marketing Materials

A3 Banner

POSTER A3 UQ INFO

PROJECT DIEGEON

Team 6-Bit (email@StevenAlanScott.com), supervised by Dr Alex Pudmenzky

PROJECT OVERVIEW

Overview

Project Diegeon aims to create a collaborative environment that inspires players to create and play. It does this by creating a set of tools that allows players to create their own levels

Goals

The main goal for this project was to create an online resource that allowed players to create their own challenges. However the project is also designed to show new users how to develop in unity and show them how to build games.

Target Audience

The target users of this game are 15 to 40-year-old gamers who enjoy RPG's and have knowledge of games in the roguelike genre. However, they do not need to have experience playing roguelike games in the past.

Industry Client

The client is Jason Weigel who is currently undertaking his PHD at the University of Queensland. His area of interest is collaboration online through the use of software. Project Diegeon is designed to help him gather more data.

BACKGROUND

Roguelike Genre

The Roguelike genre is a subset of the role-playing game. However it is characterised by two main features. First the dungeons are usually generated procedurally based on algorithms. The second is that when the player dies its game over. Forcing the player to start again, though sometimes the player gets to retain some of their progress in the form of experience points.

Unity Game Engine

Project Diegeon has been developed using the Unity game engine and online database. The software is programmed in C# and PHP and has been built from scratch. In Figure 1 you can see the overall system architecture of how we developed Project Diegeon.

Figure 1: System Architecture Diagram

EXPLORE MODE

Figure 2: Example of Explore mode.

Gameplay

As an explorer, the player has to fend off hordes of enemies, navigate through deadly traps and survive in a randomly generated dungeon, the rooms in which are built by other player. They can survive by using their dice to roll their stats. Allowing them to plan for the dangers ahead.

Dice Mechanics

Project Diegeon also incorporates a dice based stat system where explorers can roll dice and add the value of this dice to their attack or defence. The player loses a dice every time they die and when they run out of dice it is game over. This adds an element of random chance and resource management.

BUILD MODE

Figure 3: Example of Explore mode.

Gameplay

The builder utilises traps, pits, walls and monsters to create a room. The more resources the builder the higher the rooms difficulty rating is. After testing the room, the builder can submit it to the server. You can see an overview of the user interface in Figure 3.

User Created Dungeons

While Project Diegeon is a rogue like game, the dungeons are not created procedurally but rather by other players. This means that we have two main users, explorers who run the dungeon and builders who design the rooms.

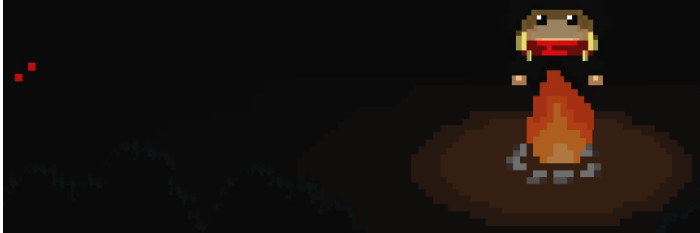
Team 6-Bit is made up of five UQ Students. Steven Alan Scott, Junchuan Xue, Changyi Zhuang, Chunzi Lyu, and Dan Dai

School of Information Technology & Electrical Engineering
Studio 3 Exhibition

A3 Poster A

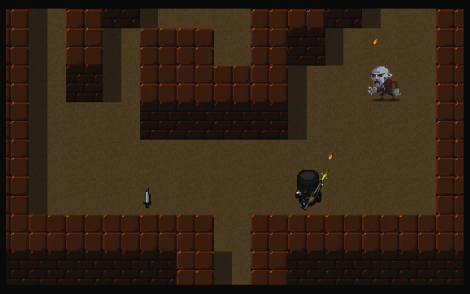
A3 Poster B

PROJECT DIEGEON



Project Diegeon is a rogue like computer game where the dungeons are created by other players. This means that we have two main users, explorers who run through the dungeon and builders who design the rooms. The target users of this game are 15 to 40-year-old gamers who enjoy playing and creating games. Project Diegon at its core is a game and as such we see the main value based on how fun the game is to play.

The overall aim of Project Digeon is twofold. First it is designed to help facilitate a collaborative environment where explorers and builders come up with their own rules of play. The second is that is serves as a tool that helps users understand some of the rules of game development and what goes into making a game. As such we have developed Project Diegon in a modular fashion where people can easily add their own features in the future. Finally, the core dynamic of builder vs explorer in a collaborative environment where the players design their own challenges allows us to collect data in relation to online collaboration. This also means that the overall gameplay will continue to evolve in the years to come.



Explore Mode: As an explorer, the player has to fend off hordes of enemies, navigate through deadly traps and survive in a randomly generated dungeon that loads in player created rooms based on their overall difficulty. They can survive by using their dice to roll their stats. Allowing them to plan for the dangers ahead.

Build Mode: The builder can utilise traps, pits, walls and monsters to create a dangerous labyrinth of rooms. The more resources the builder uses the higher the rooms difficulty rating is. However, the room stills has to be achievable by the explorer and as such a room can only be submitted to the server if the builder can complete it themselves.

A5 Information Flyer A

A5 Information Flyer B

Project Diegeon is a rogue like game where the dungeons are created by other players. This means that we have two main users, explorers who run through the dungeon and builders who design the rooms. The target users of this game are 15 to 40-year-old gamers who enjoy playing and creating games. Project Diegon at its core is a game and as such we see the main value based on how fun the game is to play.

The overall aim of Project Digeon is twofold. First it is designed to help facilitate a collaborative environment where explorers and builders come up with their own rules of play. The second is that is serves as a tool that helps users understand some of the rules of game development and what goes into making a game. As such we have developed Project Diegon in a modular fashion where people can easily add their own features in the future. Finally, the core dynamic of builder vs explorer in a collaborative environment where the players design their own challenges allows us to collect data in relation to online collaboration. This means that the overall gameplay will continue to evolve in the years to come.

Build Mode: The builder can utilise traps, pits, walls and monsters to create a dangerous labyrinth of rooms. The more resources the builder uses the higher the rooms difficulty rating is. However, the room stills has to be achievable by the explorer and as such a room can only be submitted to the server if the builder can complete it themselves.

Explore Mode: As an explorer, the player has to fend off hordes of enemies, navigate through deadly traps and survive in a randomly generated dungeon that loads in player created rooms based on their overall difficulty. They can survive by using their dice to roll their stats. Allowing them to plan for the dangers ahead.

A5 Game Guide A

PROJECT DIEGEON

Build Mode:

The object of the player is to get to the exit marked with an arrow.

Place a trap or monster in the room by clicking on the icons and then use the mouse to paint your masterpiece.

Design and build the ultimate room of doom. The only question is how evil do you want to be? But remember the player must be able to complete the room, it is only fair.

Test your build before submitting it to the server.

Controls:

Place Object

Zoom Camera

Pan Camera (HOLD)

Selection Mode:

Change to Selection Mode

Delete Object

A5 Game Guide B

PROJECT DIEGEON

Explore Mode:

Avoid monsters as they will hunt you down and kill you.

Keep an eye out for deadly traps.

When your health reaches zero you will die. Each death will cost you one dice. Run out of dice and its game over.

Avoid traps and kill monsters and try and make your way to the exit. How long can you survive?

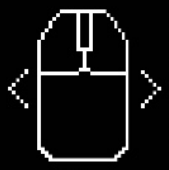
Drag dice here to increase your Strength or your Armor.

Click to Roll dice.

Controls:

Character Movement

Attack



Aim

Dice Menu:

Show Dice (HOLD)

Drag Dice

Extra:

Pick Up Item

Drop Weapon

Reload Gun