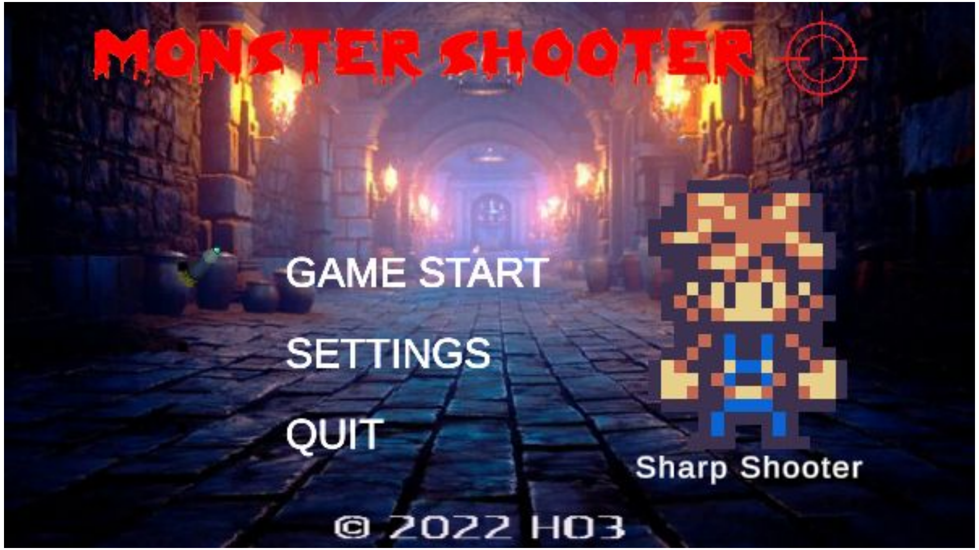
HO3 - Monster Shooter



By

Chan Foo Him, Himi Hon Tsz Pang, Andy Lee Kwan Hang, Joe

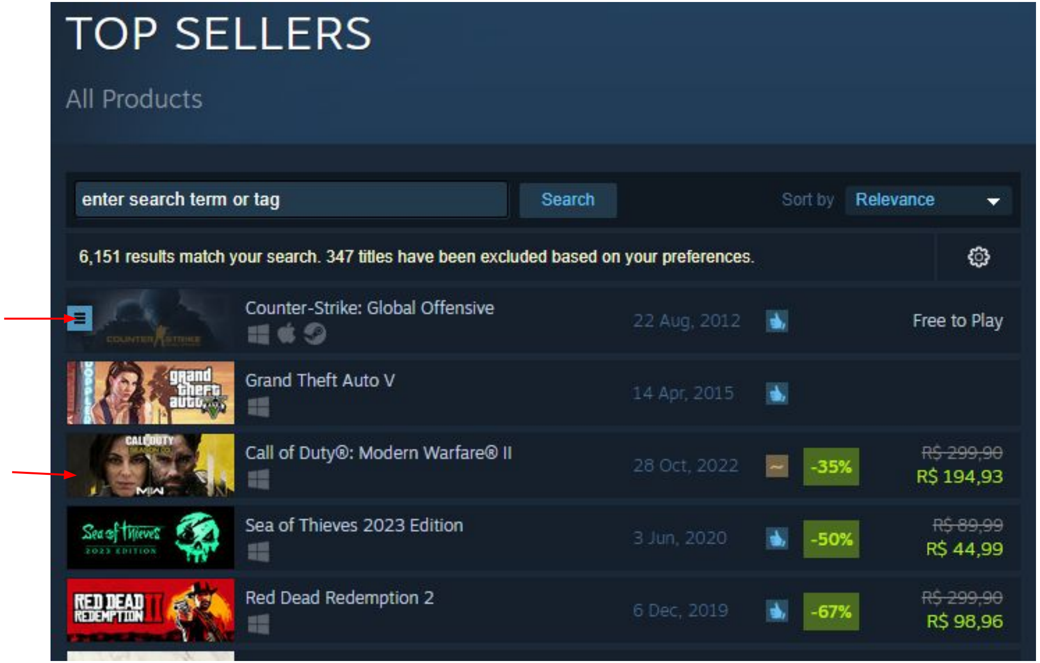
Advised by Andrew B. HORNER

Game Introduction

- Shooting game is one of the **most popular** game types

- In STEAM Sales list:

- **40%** of the **Top 5 games** are shooting game



TOP SELLERS

All Products

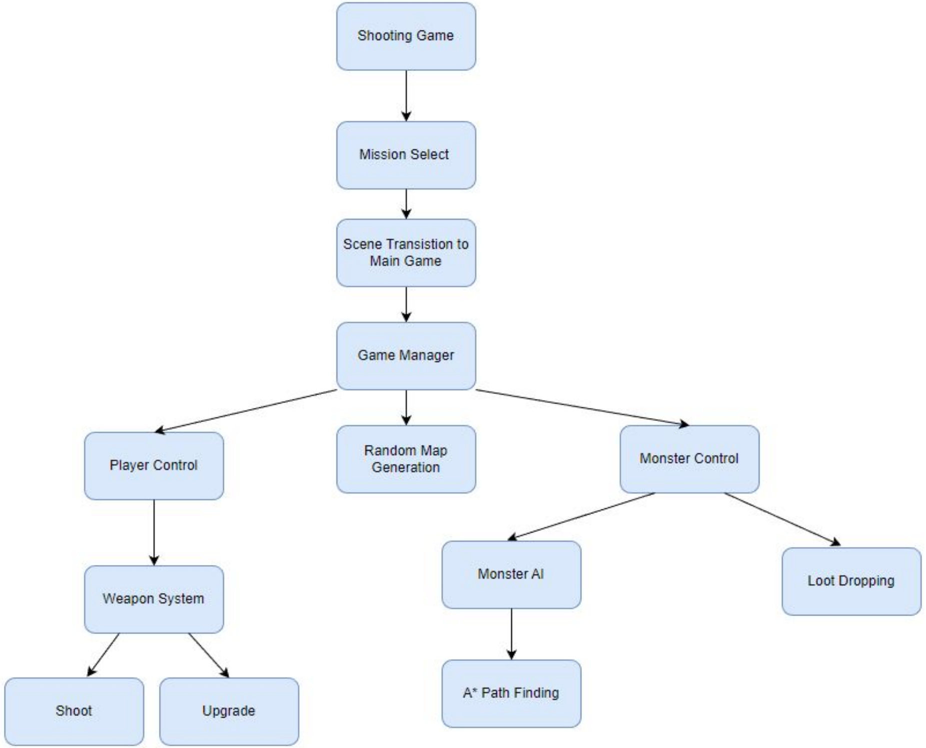
6,151 results match your search. 347 titles have been excluded based on your preferences.

Counter-Strike: Global Offensive	22 Aug, 2012			Free to Play
Grand Theft Auto V	14 Apr, 2015			
Call of Duty®: Modern Warfare® II	28 Oct, 2022	-35%	~~R$ 299,90~~ R$ 194,93	
Sea of Thieves 2023 Edition	3 Jun, 2020	-50%	~~R$ 89,99~~ R$ 44,99	
Red Dead Redemption 2	6 Dec, 2019	-67%	~~R$ 299,90~~ R$ 98,96	

Objectives

1. **Develop a 2D shooting game using Unity**
2. **Provide a satisfactory shooting feeling to the player by having several shooting effects for the weapons**
3. **Create numerous of monster attack modes and attack effects on the player.**

Overview of the Game flow



Shooting Game

Mission Select

Scene Transistion to Main Game

Game Manager

Player Control

Random Map Generation

Monster Control

Weapon System

Monster AI

Loot Dropping

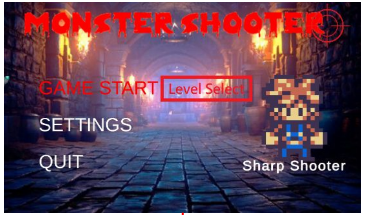
Shoot

Upgrade

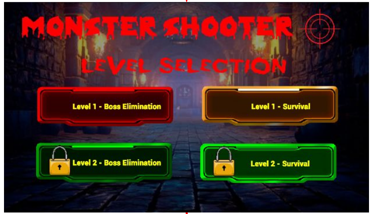
A* Path Finding

Transition between Scene

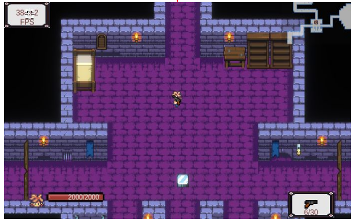
Starting Menu



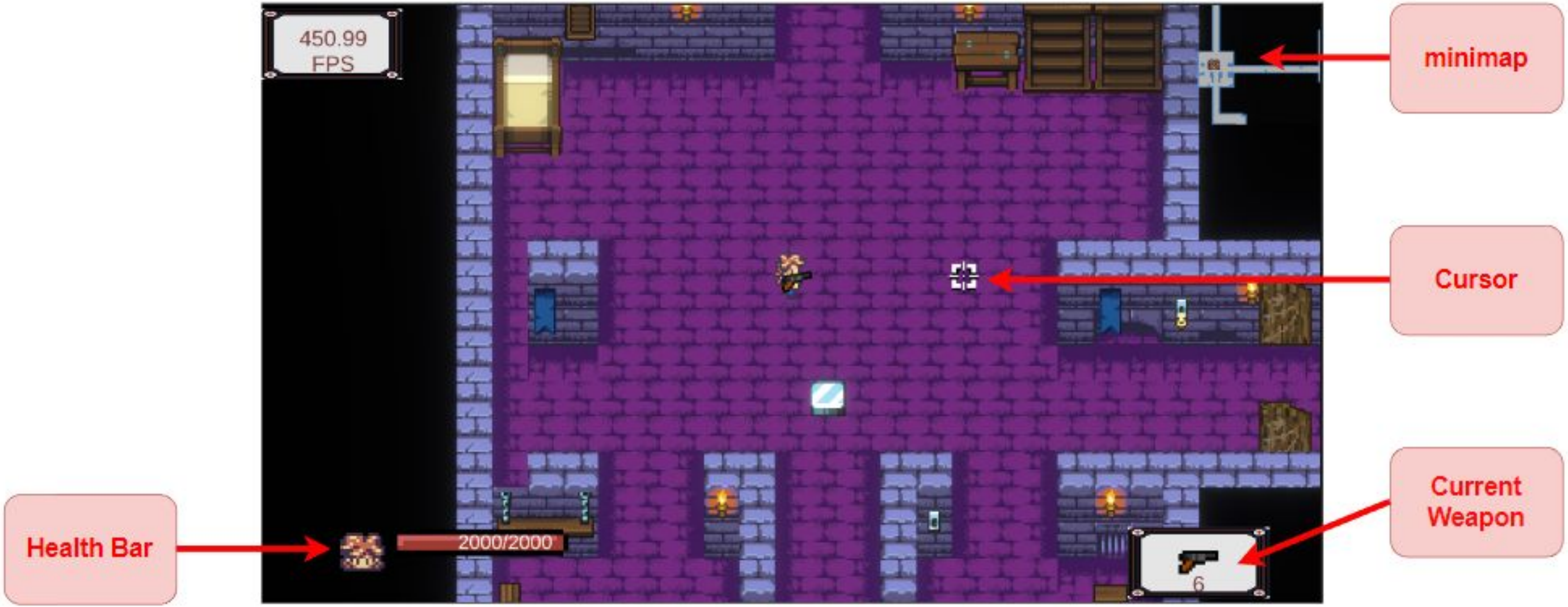
Level Select



Main Game Scene



Game Interface



minimap

Cursor

Current Weapon

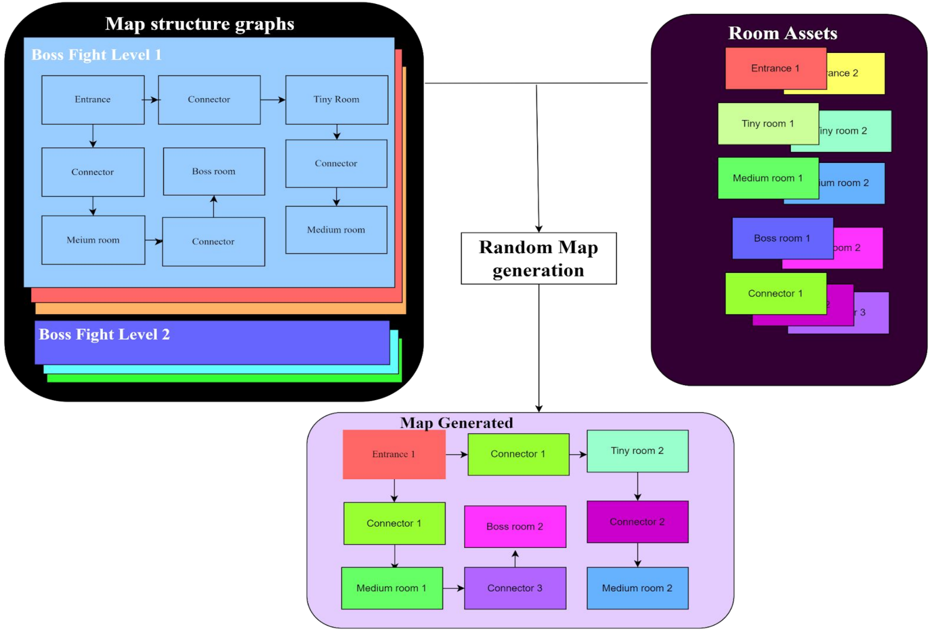
Health Bar

Design and Implementation

1. Random Map Generation Technique
2. Monster
3. Weapon System

Random Map Generation Technique

Overview:



Map structure graphs

Boss Fight Level 1

Entrance

Connector

Tiny Room

Connector

Boss room

Connector

Meium room

Connector

Medium room

Boss Fight Level 2

Room Assets

Entrance 1

ance 2

Tiny room 1

iny room 2

Medium room 1

um room 2

Boss room 1

oom 2

Connector 1

r 3

Random Map generation

Map Generated

Entrance 1

Connector 1

Tiny room 2

Connector 1

Boss room 2

Connector 2

Medium room 1

Connector 3

Medium room 2

Combined 2 parts:

i. Room Assets

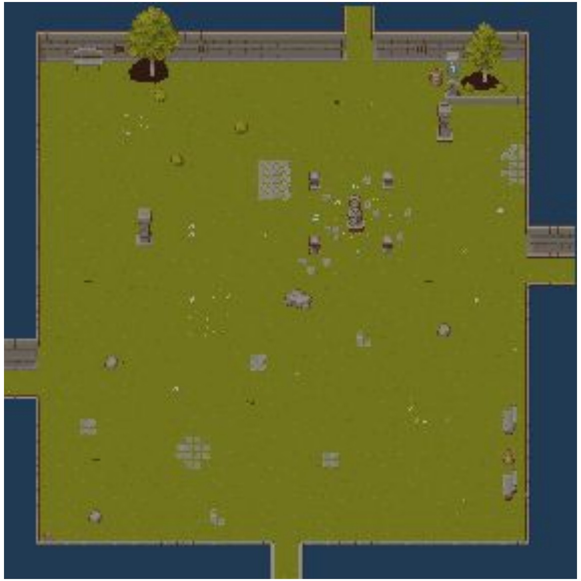
ii. Map structure graph

Goals:

- **Freshness** to the player(Higher uniqueness of map)
- **Decrease Repeatability** of the map
- **Increase reusability** of room assets

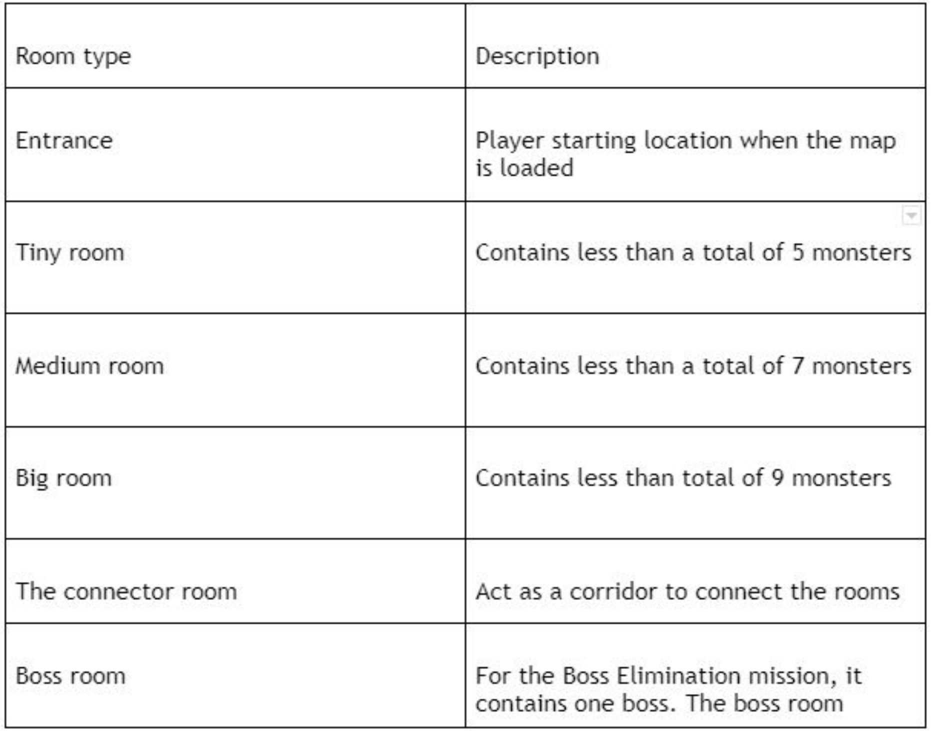
Random Map Generation Technique

Room Assets



Big Room

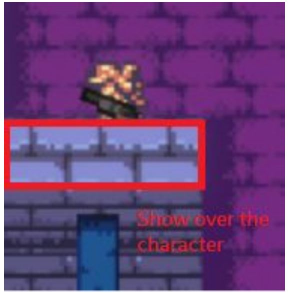
Room Type



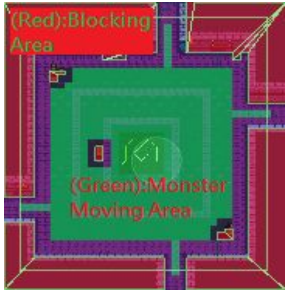
Room type	Description
Entrance	Player starting location when the map is loaded
Tiny room	Contains less than a total of 5 monsters
Medium room	Contains less than a total of 7 monsters
Big room	Contains less than total of 9 monsters
The connector room	Act as a corridor to connect the rooms
Boss room	For the Boss Elimination mission, it contains one boss. The boss room

Random Map Generation Technique

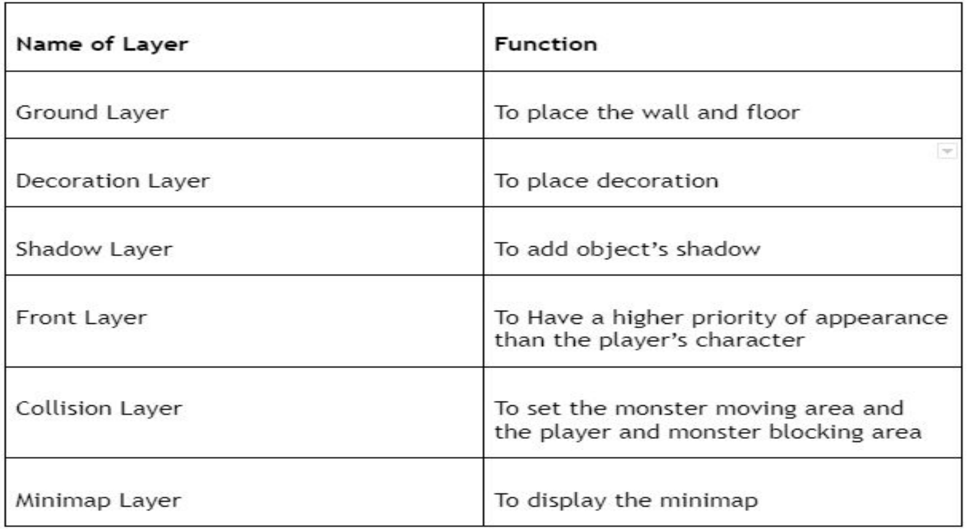
Layers of the map (Tilemap)



Front Layer

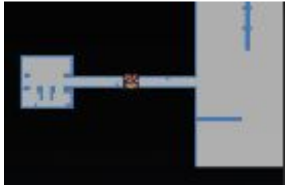


Collision Layer



Name of Layer	Function
Ground Layer	To place the wall and floor
Decoration Layer	To place decoration
Shadow Layer	To add object's shadow
Front Layer	To Have a higher priority of appearance than the player's character
Collision Layer	To set the monster moving area and the player and monster blocking area
Minimap Layer	To display the minimap

Collision: **Collider 2D**



minimap: **CinemachineVirtualCamera** and **SpriteRendered**. (Update the position of the miniplayer player which follows player movement.)

Random Map Generation Technique

Design of the Map Structure Graph

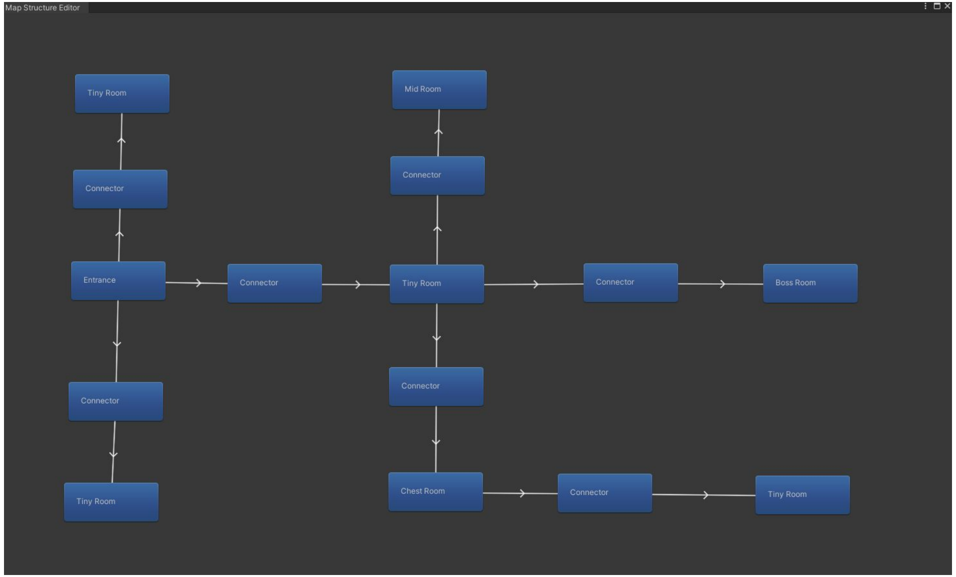
- Blueprint of the **whole map**

Goals:

- Ensure **FRESHNESS of the map**

- **easy to modify the structure of map**

  (adding/removing room assets on the map)

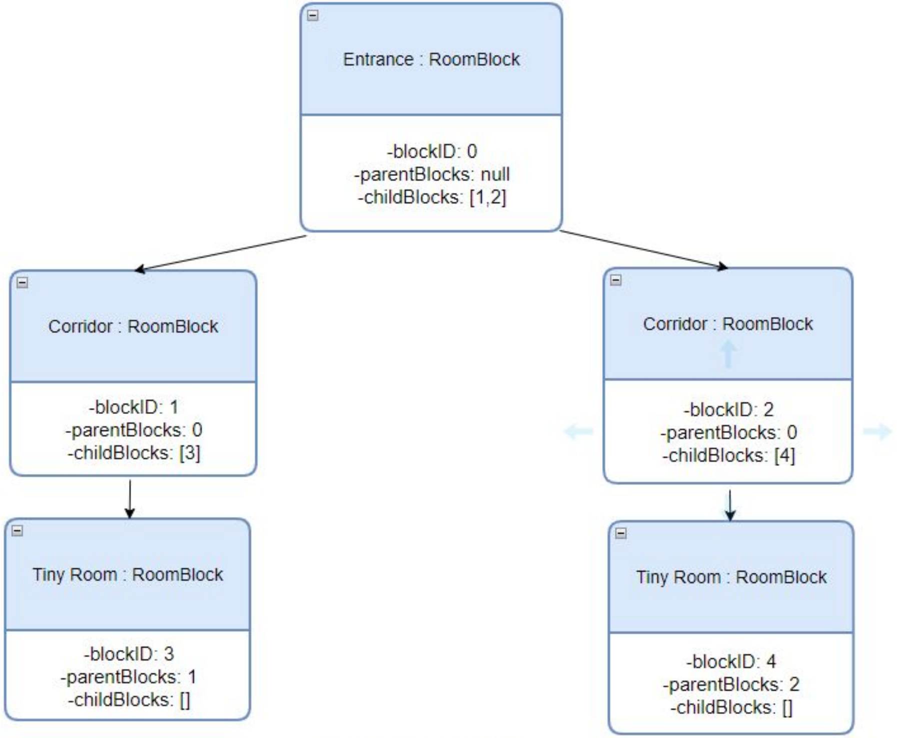


Random Map Generation Technique

Implementation of Map Structure Graph

- **Linked List** (parent and child)

- Easy to manage the **relationship between rooms**
- Ensure **connection** between rooms



Entrance : RoomBlock

-blockID: 0
-parentBlocks: null
-childBlocks: [1,2]

Corridor : RoomBlock

-blockID: 1
-parentBlocks: 0
-childBlocks: [3]

Corridor : RoomBlock

-blockID: 2
-parentBlocks: 0
-childBlocks: [4]

Tiny Room : RoomBlock

-blockID: 3
-parentBlocks: 1
-childBlocks: []

Tiny Room : RoomBlock

-blockID: 4
-parentBlocks: 2
-childBlocks: []

Map Structure Graph in details

Random Map Generation Technique

Procedure of the map generation function

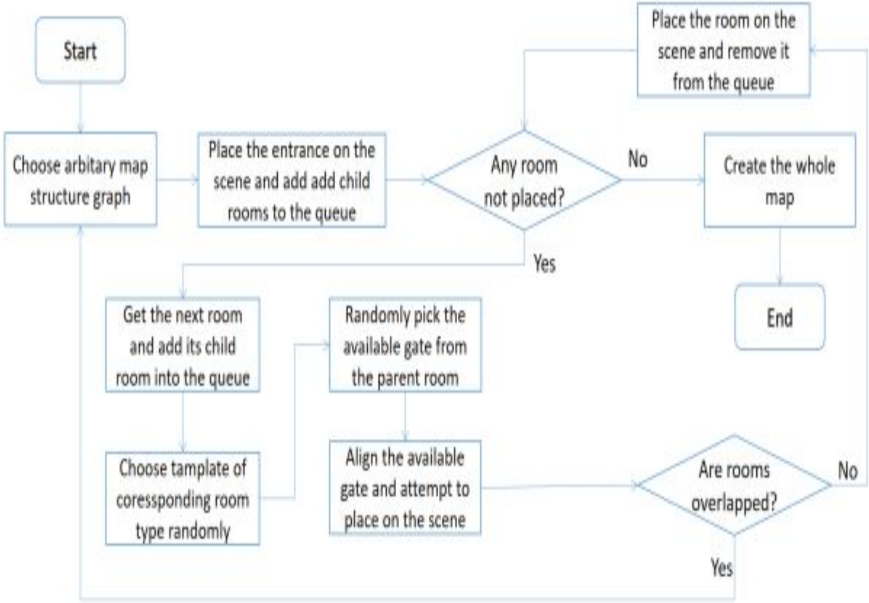
1. Select a **map structure graph** randomly (from List)
2. Place **Entrance Room** as the first room assets
3. Place the **child rooms** that matches the room type
4. Repeat the step 3 until the whole map is generated

Checking for each step 3:

- Coordinate (Overlap between room and room?)

Goals:

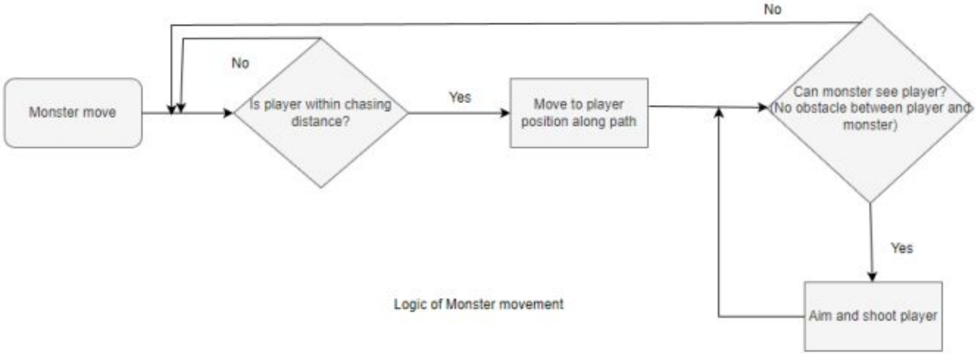
- Make sure the whole map is accurately connected.


Start
Choose arbitary map structure graph
Place the entrance on the scene and add add child rooms to the queue
Any room not placed?
No
Create the whole map
End
Yes
Place the room on the scene and remove it from the queue
Get the next room and add its child room into the queue
Choose tamplate of coressponding room type randomly
Randomly pick the available gate from the parent room
Align the available gate and attempt to place on the scene
Are rooms overlapped?
No
Yes

Monster

Movements:

- Three states: Idle, Chase and Attack


Monster move
No
Is player within chasing distance?
Yes
Move to player position along path
No
Can monster see player? (No obstacle between player and monster)
Yes
Aim and shoot player

Logic of Monster movement

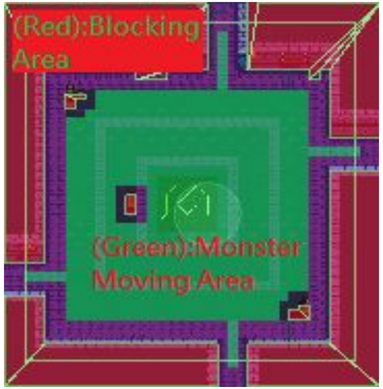
Logic:

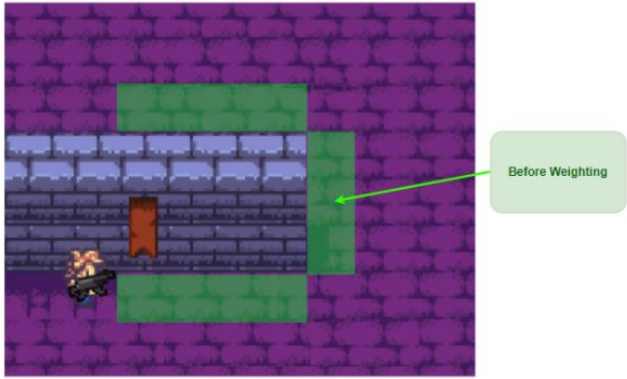
- Remains inactive when player is outside the chasing range
- Otherwise, move towards the player position
- Detect any obstacle between the monster and the player
- Attack player when the player is visible to the monster (using Physics Raycast)

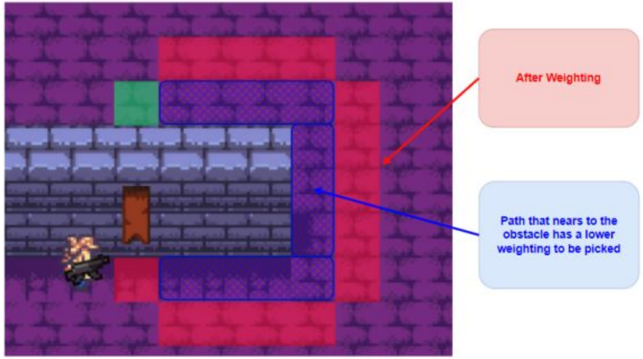
Pathfinding (A* algorithm)

When monster attempts to chase the player, a static method in Pafhfinding class will be called to generate a path between player and monster

By applying **weighting**, cell near the obstacle is less likely to be picked




Before Weighting


After Weighting

Path that nears to the obstacle has a lower weighting to be picked

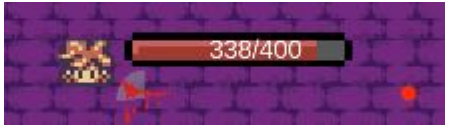
Attack Debuff

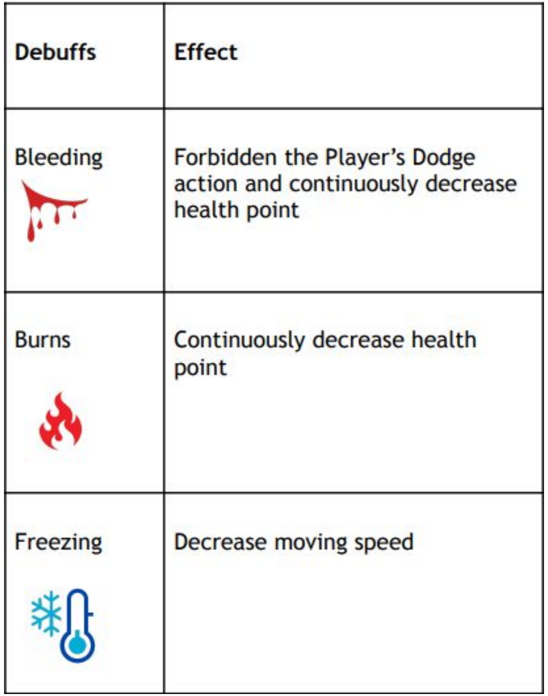
Monster’s attack hits player

- chance to produce special effect on player

Debuff Handler

- Use countdown timer to manage the bebuff status
- Broadcast OnDebuffChanged() event to update UI





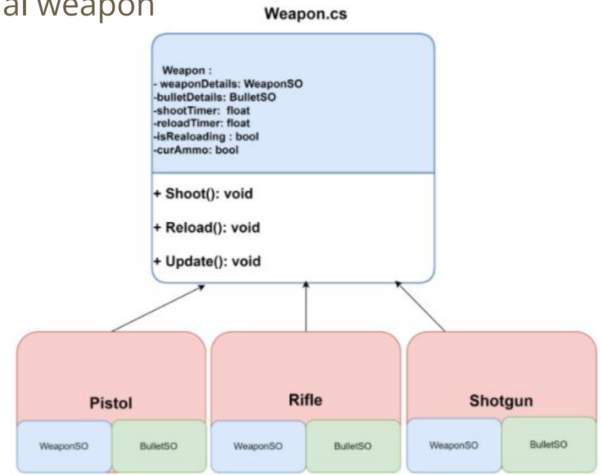
Debuffs	Effect
Bleeding	Forbidden the Player’s Dodge action and continuously decrease health point
Burns	Continuously decrease health point
Freezing	Decrease moving speed

Weapon System

Responsible for managing the attributes of all weapons

- All weapons inherit Weapon class and overrides Shoot() method
- WeaponSO ScriptableObject to store the initial attributes of individual weapon

Pistol	Rifle	Shotgun	Laser	RPG



Weapon.cs

Weapon :
- weaponDetails: WeaponSO
-bulletDetails: BulletSO
-shootTimer: float
-reloadTimer: float
-isRealoading : bool
-curAmmo: bool

+ Shoot(): void

+ Reload(): void

+ Update(): void

Pistol

WeaponSO

BulletSO

Rifle

WeaponSO

BulletSO

Shotgun

WeaponSO

BulletSO

Crafting and Upgrading

Collect the required loot by defeating monster

Crafting:

- Restricts usage of uncrafted weapons
- Records the number of crafting components collected by player

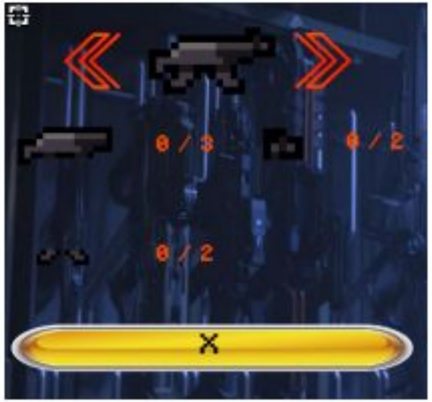
Upgrading:

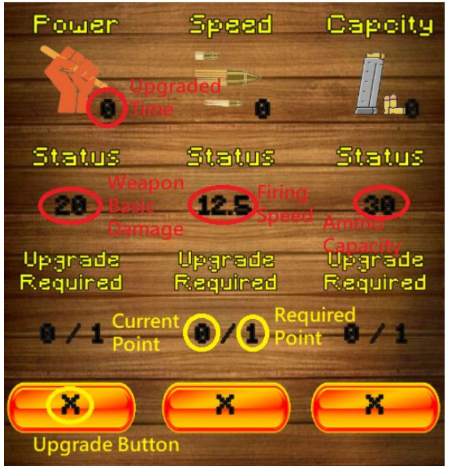
- Records the upgrade point of different attributes of a weapon
- Increase damage, fring speed and clip size

Maintian consistency over scene : Singleton pattern

DontDestroyOverLoad()

Rifle

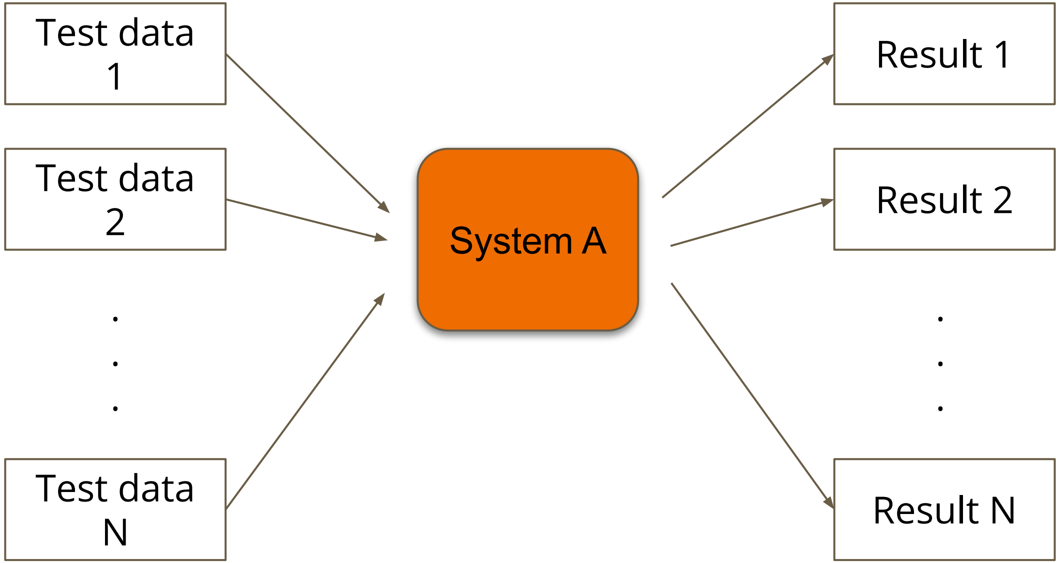




Testing

Functionality Test

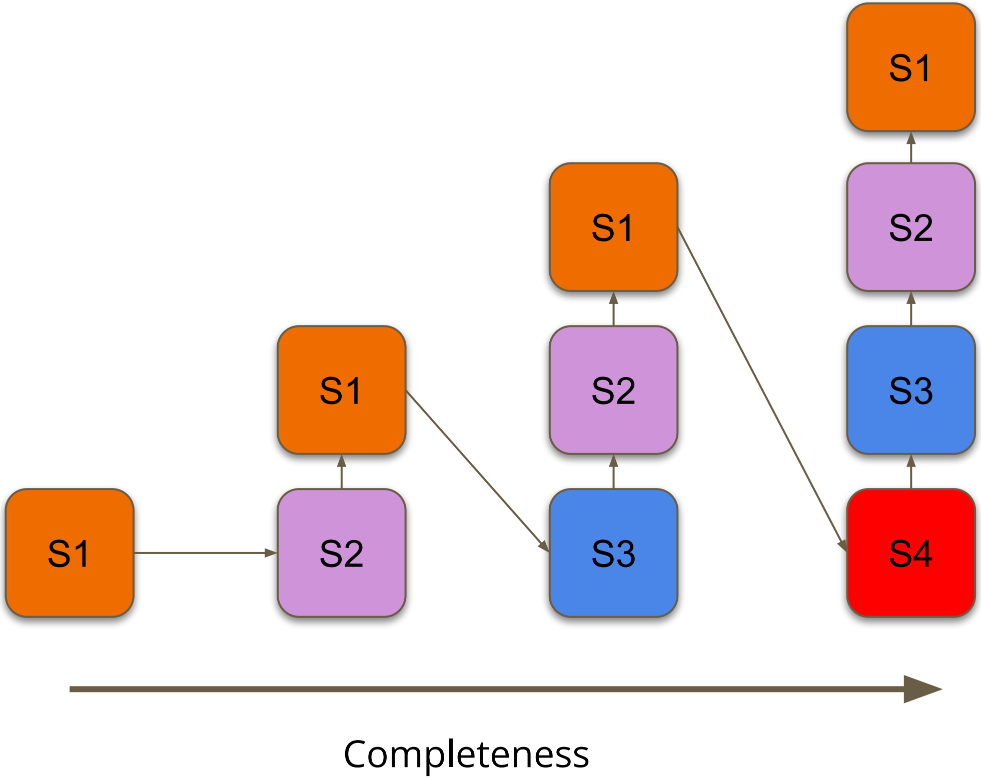
1. Prepare data (including extreme cases)
2. Run the system with input data
3. Record the result
4. Identifying bugs from unexpected results



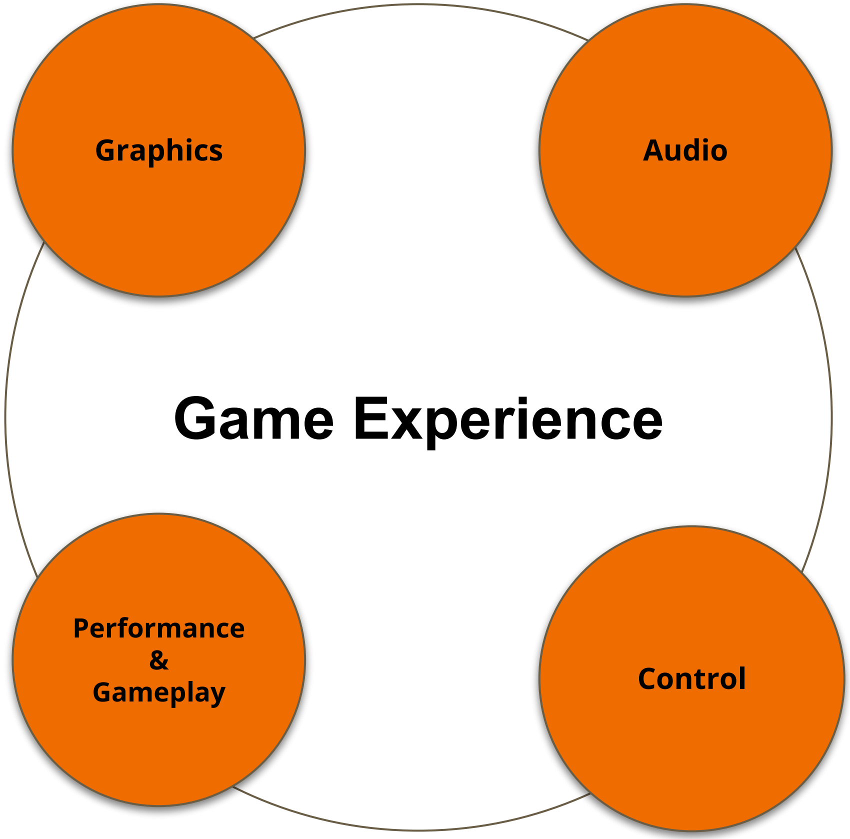
Testing

Regression Test

1. Perform functionality test on new system
2. Repeat testing on some/all previously completed systems in order
3. Record the result
4. Identifying bugs from unexpected results
5. Repeat when there are new system implemented



Evaluation



Graphics

Audio

Game Experience

Performance
&
Gameplay

Control

Evaluation

Graphics

✓ Each single object texture are highly featured

✓ Some animations are smooth

✗ No attack animation when emitting bullets

✗ Art styles are inconsistent

Audio

✓ Musics and effects fit the game

✓ Transition between musics are natural

✗ Too few musics, making the game monotonous

Evaluation

Control

✓ Apply mainstream keyboard settings

✗ Very inconvenience to switch weapon

Performance & Gameplay

✓ Strong feeling of shooting

✗ Unreasonable difficulty allocation between rooms

✗ Lack of tutorial

Discussion

Gameplay

1. Random Map Generation
   - All dungeon rooms are randomly picked, but there are too few room choices.
   - Players may feel boring after looping a level for a few more times.

2. Crafting System
   - We underestimated the difficulty of equipment crafting system and therefore replaced with upgrade system

Discussion

Graphics & Audio

1. Selection of audio
   - As developers, we are well aware of the “Developer Blindness” (aesthetic fatigue).
   - We can become desensitized to the auditory elements.
   - Therefore, we are particularly careful in our selection of these elements.
2. Selection of graphics
   - We fail to design exquisite artworks.
   - Therefore, artwork styles may appear inconsistent.

Conclusion

- Develop the game with top-down and bottom-up approach
  - Top-down: Break the whole game into small systems
  - Bottom-up: Start with single system and gradually build up to a comprehensive game
- Low coupling, high cohesion between game systems
  - Systems are highly modularized
- Follow DRY (Don’t Repeat Yourself) development principle
  - Utilize scriptable objects to reduce code duplication