台北科技大學 資訊工程系

物件導向程式設計
書面報告

組　別：第二十一組

題　目：Eternal Singularity

組　員：104590024 蔡一玄

104360098 梁皓鈞

指導老師：陳偉凱

目錄

一、簡介：

(A)遊戲選擇動機：這款遊戲是屬於經典類型的角色扮演遊戲（Classical Role-Playing Game)。Classical Role-Playing Game 在沒有電子計算機之前就已經存在，當時是視為一種桌上角色扮演遊戲。到了電子計算機開發後，開始有一些經典的 RPG 遊戲出現。例如幾乎是 RPG 遊戲始祖的 <<龍與地下城>>，<<Dungeon>> 等。到了 1987 年，Final Fantasy（最終幻想)第一代的出現，由於其故事的原創性以及令人感動的劇情，加上易懂的操作方式，造成 RPG 在世界上的風潮。

而我們希望參考舊時代的 Final Fantasy 的操作，同時配合新的 Final Fantasy 元素，例如 物品拾取，任務流程，任務操作之類的。製作出經典風格的 RPG 遊戲。

遊戲名字中的 Singularity 是指奇異點。其中近年被大幅報道是由 Raymond Kurzweil 提出的技術奇異點逼近，指出技術發展會出現 Exponential growth 狀態，最後會到了一個神奇的世紀，就是發展速度近似無限。在那個時候，所有幻想都成為可能。而我們遊戲就是希望帶出，當玩家在一個一切幻想都成為可能時的探險。

名字中的 Eternal 代表的是永恆。想表達出玩家永恆地被困在這個一切神話及幻想都成真的奇異點，唯一可以突破的就是把創造這個奇異點，控制這個向量空間維度的機器打破。基本上我們這個遊戲的風格在於史詩以及迷茫。讓玩家在遊戲中遇到不同的困境以及學習敵人攻擊的節奏進行躲避。

(B)組員分工方式：

1. 蔡一玄：遊戲部分負責玩家基本素質(包括技能、HP、MP、EXP・・・等)的設計及程式，怪物技能及基本素質，以及遊戲內的各種特效(轉場、下雪)。報告部分負責程式架構部分以及統整問題與解決方法。

2. 梁皓鈞：遊戲部分負責任務程式及設計(獎勵、達成條件)，地圖規劃及程式設計，準備遊戲所需的素材(音樂、圖片)。報告部分負責一些說明的部分與動機。

二、遊戲內容介紹：

(A)遊戲說明：此遊戲含有部分 RPG 的元素，玩家需要透過解任務或打怪來獲得經驗值，達到一定的等級會解鎖新技能。在玩家通過最後一個任務及通過考驗地圖後，將會得到強大的能力，並進入魔王關卡作戰。

詳細說明如下：

1.當玩家血量歸零時，遊戲結束。

2.遊戲勝利條件為玩家擊敗最後關卡的魔王。

3.玩家可使用方向鍵操控角色行動，按下 Z 鍵可以撿取物品。攻擊型技能：X 鍵為基本攻擊；C 鍵為技能 V-Cut；V 鍵為技能 Burst Cut；B 鍵為技能 Sword Light；Space鍵為Cross Cut。特殊攻擊技能：Alt鍵為技能Hero Slash；Ctrl鍵為技能Frozen Sword。移動、輔助型技能：A 鍵為技能 Wind Move；S 鍵為技能 Dark Force；D 鍵為技能 Potential Power。藥水部分：Q 鍵為 HP Potion；W 鍵為 MP Potion。

(※注意：詳細的解鎖等級可於官方網站查詢)

4.玩家可以到任務表內查看當前可執行的任務，而每個任務都有接取等級限制，若玩家等級未到則任務不會顯示在任務表內。當任務中含有擊敗怪物的條件時，將滑鼠移至任務按鈕上，會出現當前已擊殺的怪物數量。

5.玩家可於背包欄位查看當前擁有道具的數量。

6.技能表內會顯示玩家的基本技能，未解鎖的技能會呈現紅字。

7.特殊技能因較強大，故有冷卻時間的設定，玩家可於冷卻時間欄位中查看。

8.一般地圖中，怪物死亡後約 20 秒後會復活。怪物血量、攻擊力與玩家血量、攻擊力的比例有經過設計，這邊建議玩家不要越級打怪，否則很容易死亡。

9.玩家達到等級 10 之時，可進入考驗地圖。必須到達考驗地圖終點(右下角)並拿到卷軸後，才可進入魔王地圖。若玩家在過程中不慎死亡，會回到考驗地圖的起點，並自動補滿 HP 及 MP，因此玩家可持續挑戰。

10.魔王關卡中，會有兩名古代賢者(Ancient Sage)、兩隻黑惡魔(Black Devil)以及魔王---審判者(Inquisitor)。其中，魔王以固定時間產生暗黑球來攻擊玩家，每施放五次黑暗球技能會施放迴力鏢，大約五秒會施放寒冰箭(Frostbolt)技能(產生十支寒冰箭於地圖上隨機位置)。古代賢者每七秒會使魔王的暗黑球數量加一。當玩家接近黑惡魔時，他們會偵測玩家位置並追蹤，當玩家離開偵測範圍時則變為隨機移動。另外魔王身上會帶有紫色盾牌，若盾牌存在則玩家無法對魔王造成攻擊，而玩家必須擊敗兩隻黑惡魔，紫色盾牌才會消失。當魔王血量低於 50%時，雨量會增加，場上每秒會產生閃電，玩家被擊中會受到大量傷害。

11.畫面右上方有個小按鈕，點擊即可前往官方網站。

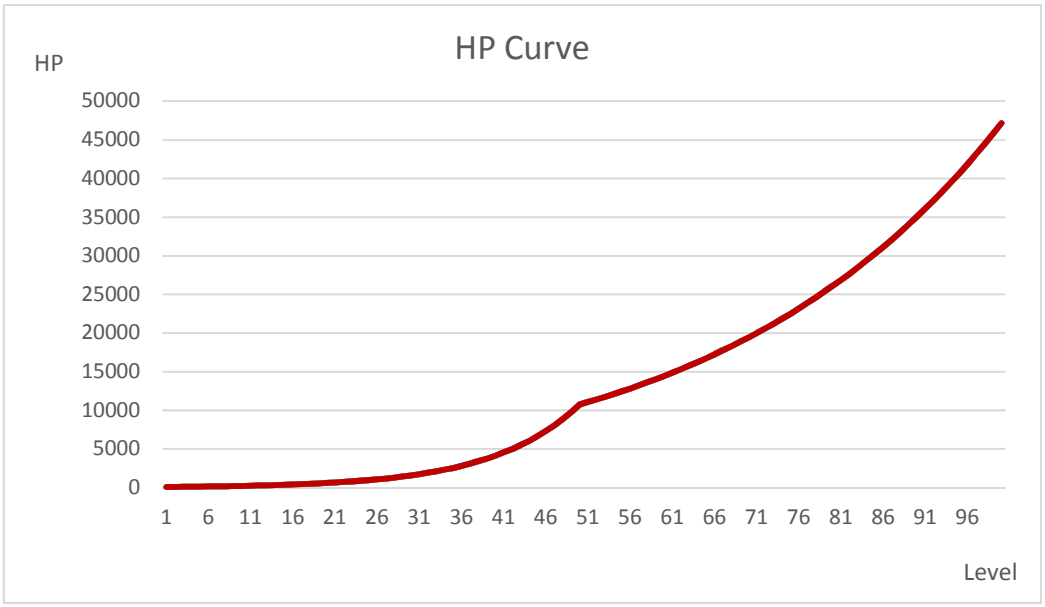
12.按下起始畫面下方的按鈕(Skill Showcase)可進入技能展示模式，玩家可直接使用所有技能。

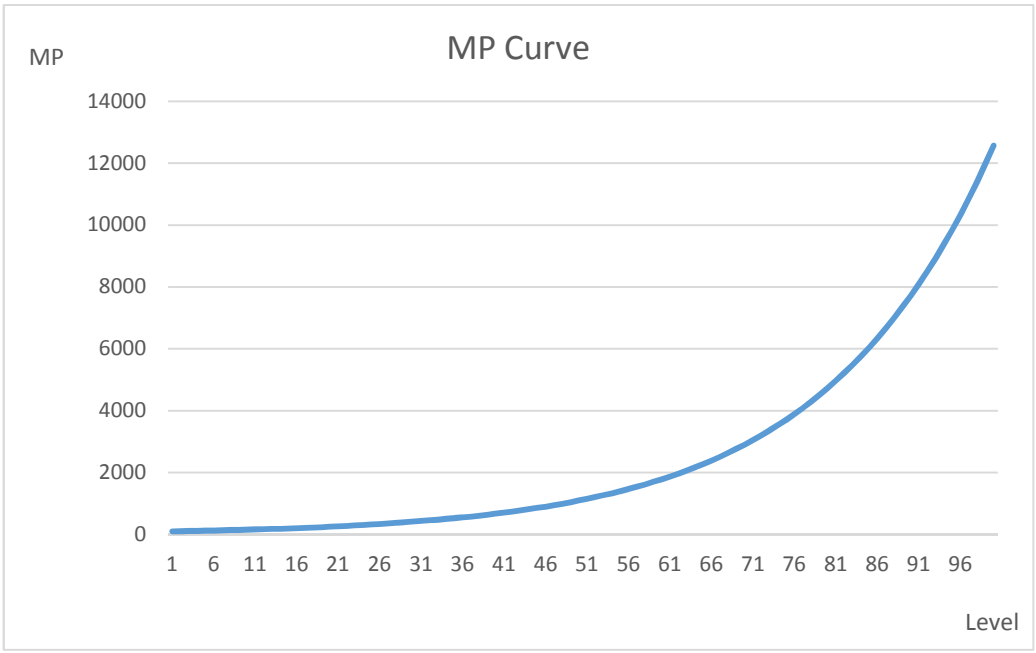
13.密技：按下數字鍵 1 可以讓防禦力變為 100%(即怪物無法對角色造成傷害)，並讓攻擊力變為 99999；按下數字鍵 2 可以讓 HP 及 MP 回滿；按下數字鍵 3 可以讓等級提升一。(※注意：場上的閃電攻擊會無視防禦力)

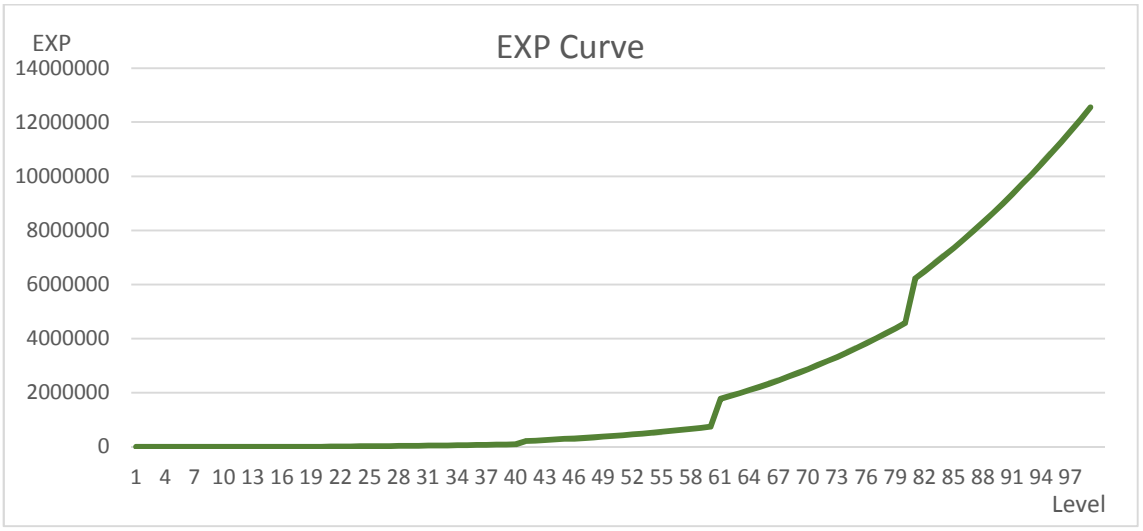
(B)遊戲設計：我們在遊戲中有大多元素是自己設計的，像是玩家的攻擊力、血量(Health Point)、魔力(Mana Point)、行動力(Action Point)、經驗值、任務以及技能。另外，怪物的血量以及攻擊力我們是依照玩家的成長曲線來調配的。

以下為一些範例:

1.玩家能力及經驗值曲線：







$$f(x) = \begin{cases} \text{ROUND(POWER(x,3)+9,1)} & \text{if } 1 <= x < 21 \\ \text{ROUND(POWER(x,3)*1.5,0)} & \text{if } 21 <= x < 41 \\ \text{ROUND(POWER(x,3.3),0)} & \text{if } 41 <= x < 61 \\ \text{ROUND(POWER(x,3.5),0)} & \text{if } 61 <= x < 81 \\ \text{ROUND(POWER(x,3.5)*1.3,0)} & \text{if } 81 <= x < 100 \end{cases}$$

x : Level

2.怪物能力表：

	Black Devil		
Level	HP	ATK	EXP
50	170000	300	177147
Special skill :	Fire Strike(ATK : 500)		
Special ability :	1. Tracing (When character close to it) 2. Generate a magic guard for the inquisitor(When the magic guard exists, character can't deal any damage to it)		
	Inquisitor		
Level	HP	ATK	EXP
50	500000	1000	0
special skills	1. Black ball(1 time/2s) 2. Boomerang(1 time/10s) 3. FrostBolt(1 time/5s) 4. Thunder light(1 time/1s)		
Skill description :	When the skill "Frostbolt" is dealt, it will produce 10 frostbolts randomly on the field. When player's health point is below 50%, it will deal the skill "Thunder light" per second(The position of thunder light is random)		

3.角色技能：

skill 1 : V-Cut

Method to gain	When the character reaches level 5				
Type	Active Skill				
Range	32 X 96				
Level	1	2	3	4	5
Damage	110%	120%	130%	140%	150%
MP Cost	3	4	5	6	7

skill 2 : Micro Shield

Method to gain	When the character reaches level 8				
Type	Passive Skill				
Level	1	2	3	4	5
Reduce damage received from monsters	5%	10%	15%	20%	25%

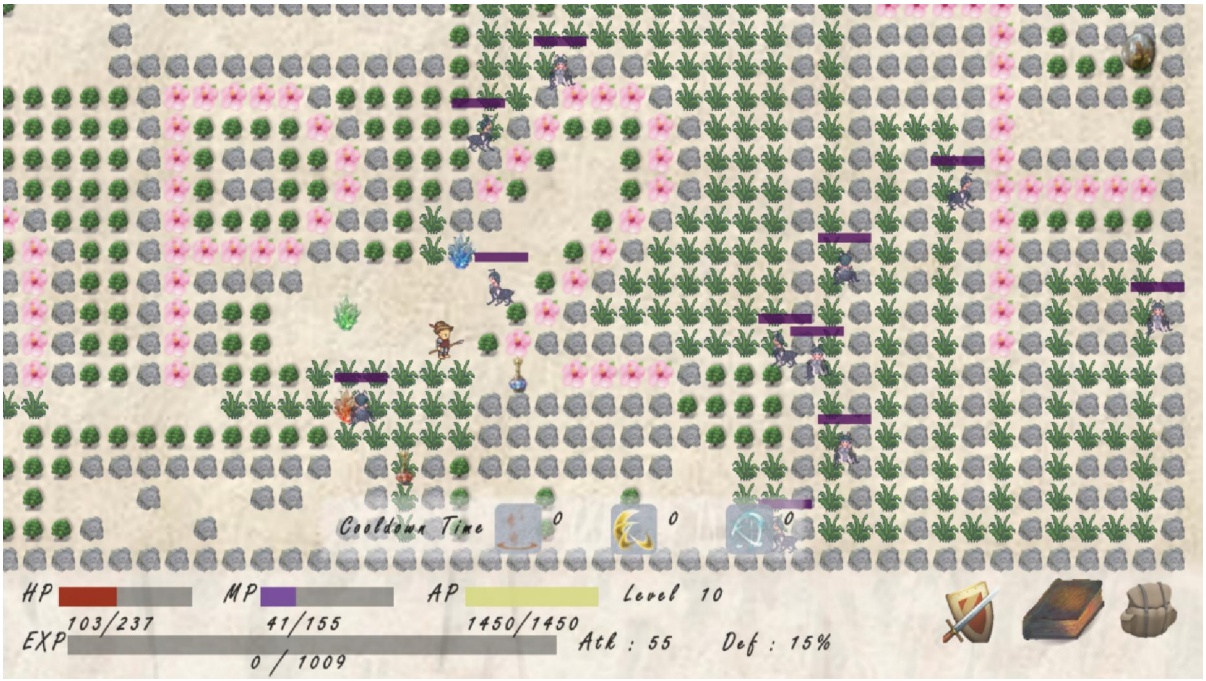
skill 3 : Burst Cut

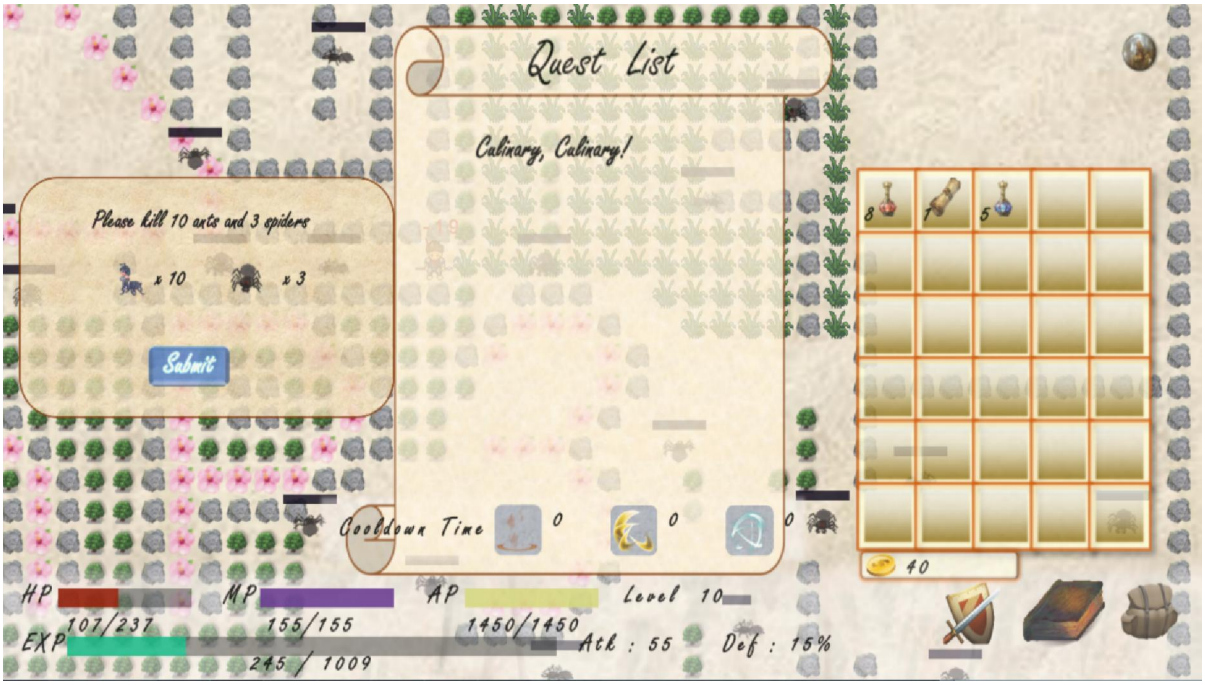
Method to gain	When the character reaches level 10				
Type	Active Skill				
Range	96 X 96				
Level	1	2	3	4	5
Damage	160%	170%	180%	190%	200%
MP Cost	6	7	8	9	10

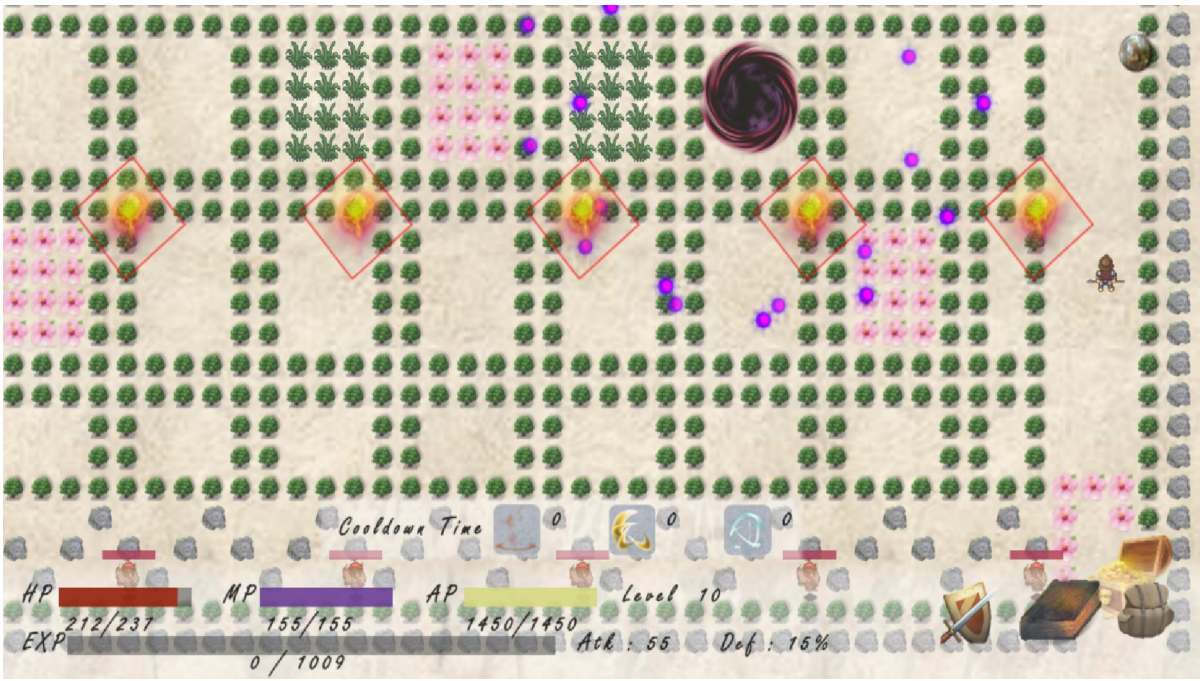
(C)遊戲場景畫面及介面：

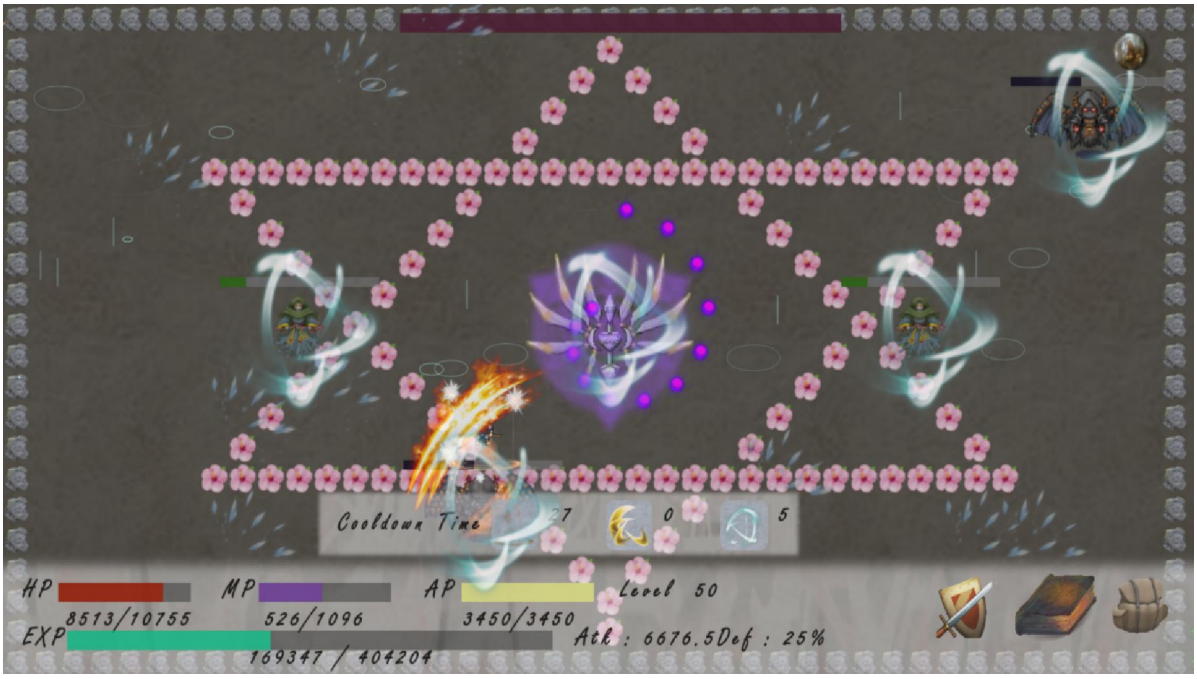
＃遊戲開始畫面

＃一般地圖場景



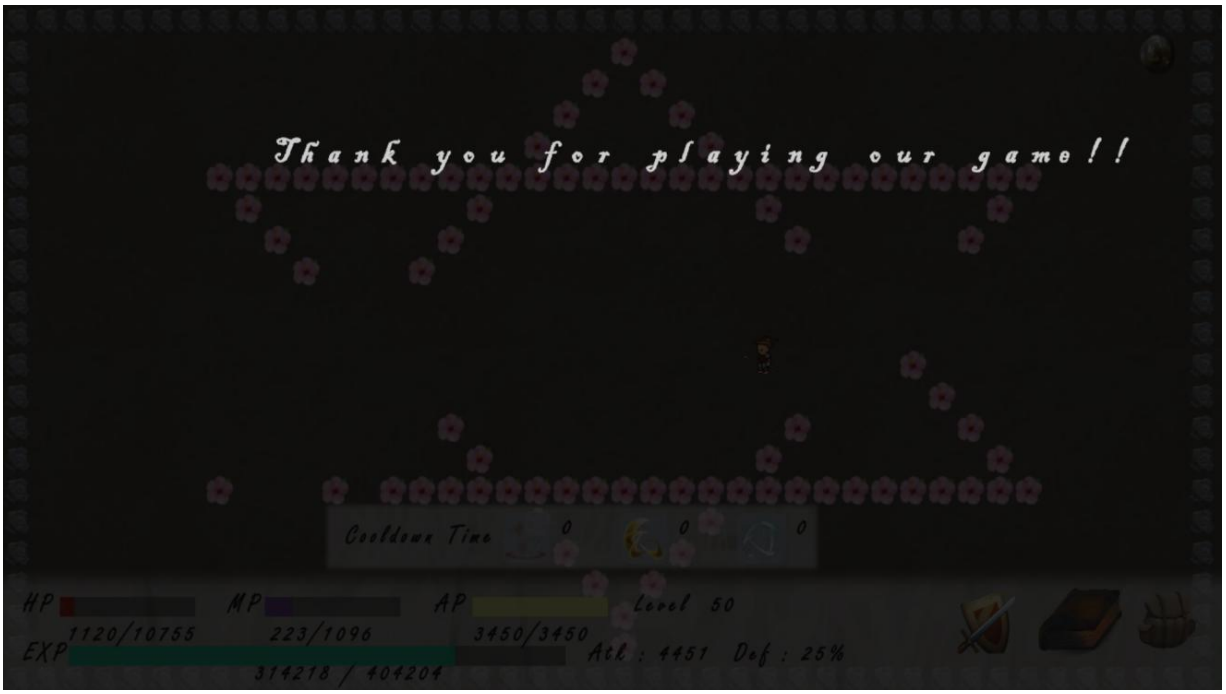

Quest List
Culinary, Culinary!
Please kill 10 ants and 3 spiders
x 10
x 3
Submit
Cooldown Time
0
0
0
8
1
5
40
HP
107/237
MP
155/155
AP
1450/1450
Level 10
EXP
245 / 1009
Atk : 55
Def : 15%


Cooldown Time
0
0
0
HP
212/237
MP
155/155
AP
1450/1450
Level 10
EXP
0 / 1009
Atk : 55
Def : 15%


Cooldown Time
HP
MP
AP
Level 50
8513/10755
526/1096
3450/3450
EXP
169347 / 404204
Atk : 6676.5Def : 25%


Thanks for your sacrifice!!
Cooldown Time
HP
MP
AP
Level 50
0/10755
526/1096
3450/3450
EXP
169347 / 404204
Atk : 4451 Def : 25%


Cooldown Time
19
1
0
HP
MP
AP
Level 50
1278/10755
293/1096
3420/3450
EXP
314211 / 404204
Atk : 4451 Def : 25%


Thank you for playing our game!!
Cooldown Time
0
0
0
HP
MP
AP
Level 50
1120/10755
223/1096
3450/3450
EXP
314218 / 404204
Atk : 4451 Def : 25%

(D)遊戲音效：

我們在遊戲中只用了三種音樂，主要原因是因為我們時間花比較多在程式設計上，在技能之類的各種遊戲音效都比較難找到不同有特色的來使用。因此我們遊戲最後決定注重背景音樂配合遊戲模式所帶來的感受。我們的遊戲是參考 Final Fantasy (最終幻想)系列，遊戲中的技能之類的注重點放的比較少，因此我們認為比起各種技能花俏的音效更加來的有我們希望做到的史詩感。

・遊戲開始畫面音樂：To Zanarkand

出處：Final Fantasy X HD Remaster

使用原因：Final Fantasy X 是一個單機劇情遊戲，他的故事內容十分史詩，這首曲是女主角在男主角死後，重建世界時的背景音樂，即使重建世界是一個快樂的事，但在沒有男主角的世界上，女主角也不知道該開心還是悲傷，然後慢慢淡出遊戲。在遊戲最後，男主角從海底醒來回到世上。這音樂在場境中造出了寧靜，安寧的感受。配合到我們遊戲開始畫面是 Final Fantasy XIV： A Realm Reborn 的 Gridania 的藝術圖以及下雪的畫面，我們希望可以帶出一種不知為何來到這個世界，但沒有討厭這個世界，反而是因此而感到安寧的感覺。

資源：https://www.youtube.com/watch?v=Ght23jESOpA

・遊戲平常的音樂：Wailers and Waterwheels

出處：Final Fantasy XIV : A Realm Reborn

使用原因：我們的遊戲有大量的樹林以及郊外，而這個背景音樂是在 FFXIV:ARR 中的 Gridania 日間背景音樂。Gridania 是一個充滿樹林橋樑湖泊的自然環境。FFXIV:ARR 是一個 MMORPG(大型多人線上遊戲)，他與一般的 Final Fantasy 系列不同，他是一個多人遊戲而不是單機劇情遊戲，所以十分注重玩家長時間在遊戲中沉浸，把遊戲畫面帶給玩家的感覺。Gridania 這個地方有分成日間與晚間的背景音樂，我們選擇了日間是因為我們的遊戲設定是日間。

資源：https://www.youtube.com/watch?v=VSKNntvdRJk

・遊戲平常的音樂：Torn from the Heavens

出處：Final Fantasy XIV : A Realm Reborn

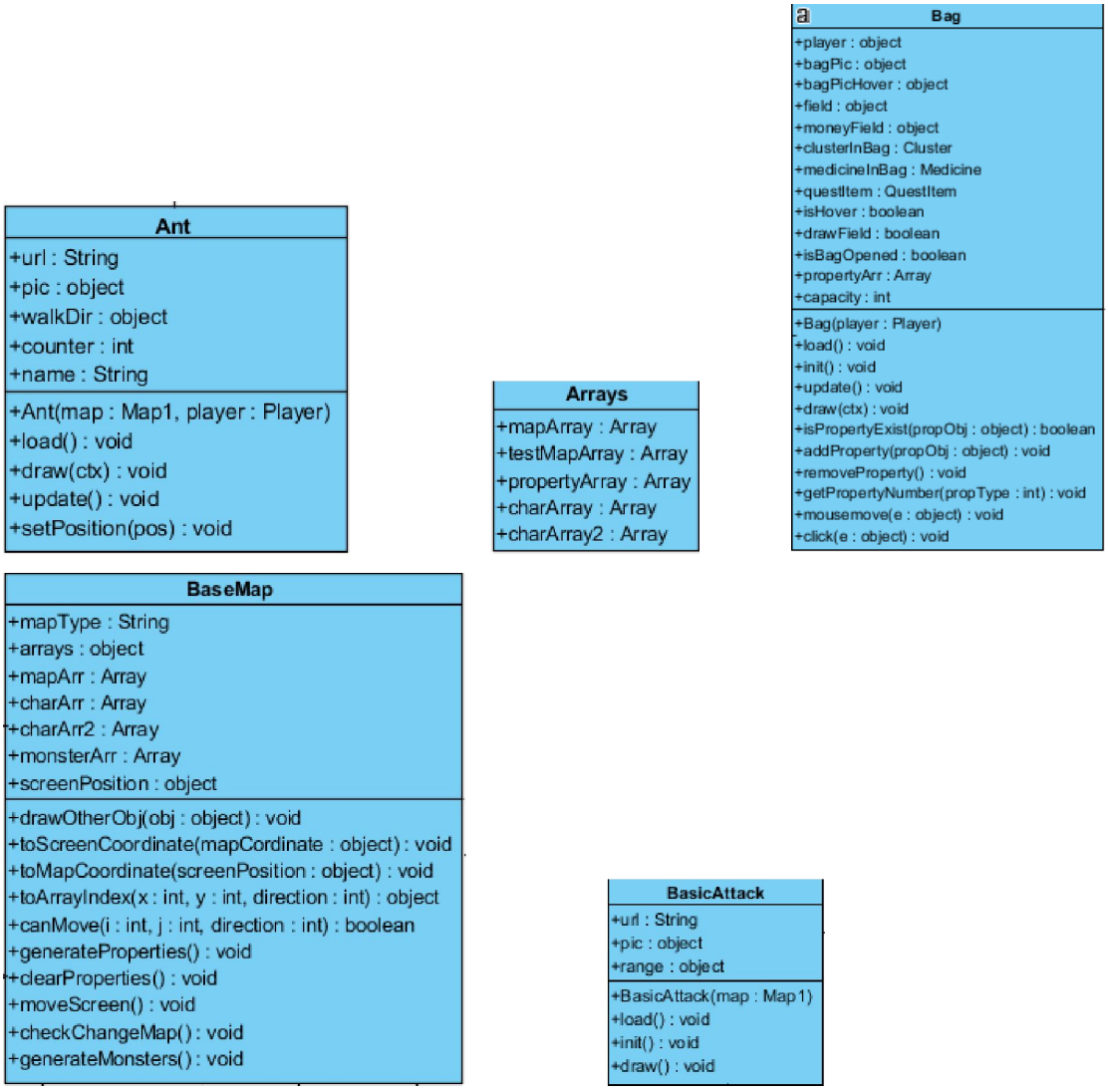
使用原因：這首曲是在 FFXIV:ARR 中的 Trial 類型的副本中打 Hydra / Chimera 或者在地圖日常任務 F.A.T.E.中的背景音樂。使用這首的原因很顯然就是因為這個很配合打怪物 Boss 的情境。這首曲前段有著緊張的節奏，帶出了戰爭的氣氛。到中間開始有弦樂以及慢慢的音階漸高的，有著戰爭白熱化的感受。因此我們認為這個曲是最能代表出 RPG 遊戲那種孤注一擲打 Boss 的場境。

資源：https://www.youtube.com/watch?v=C0t7-CPvUuk

三、遊戲程式設計

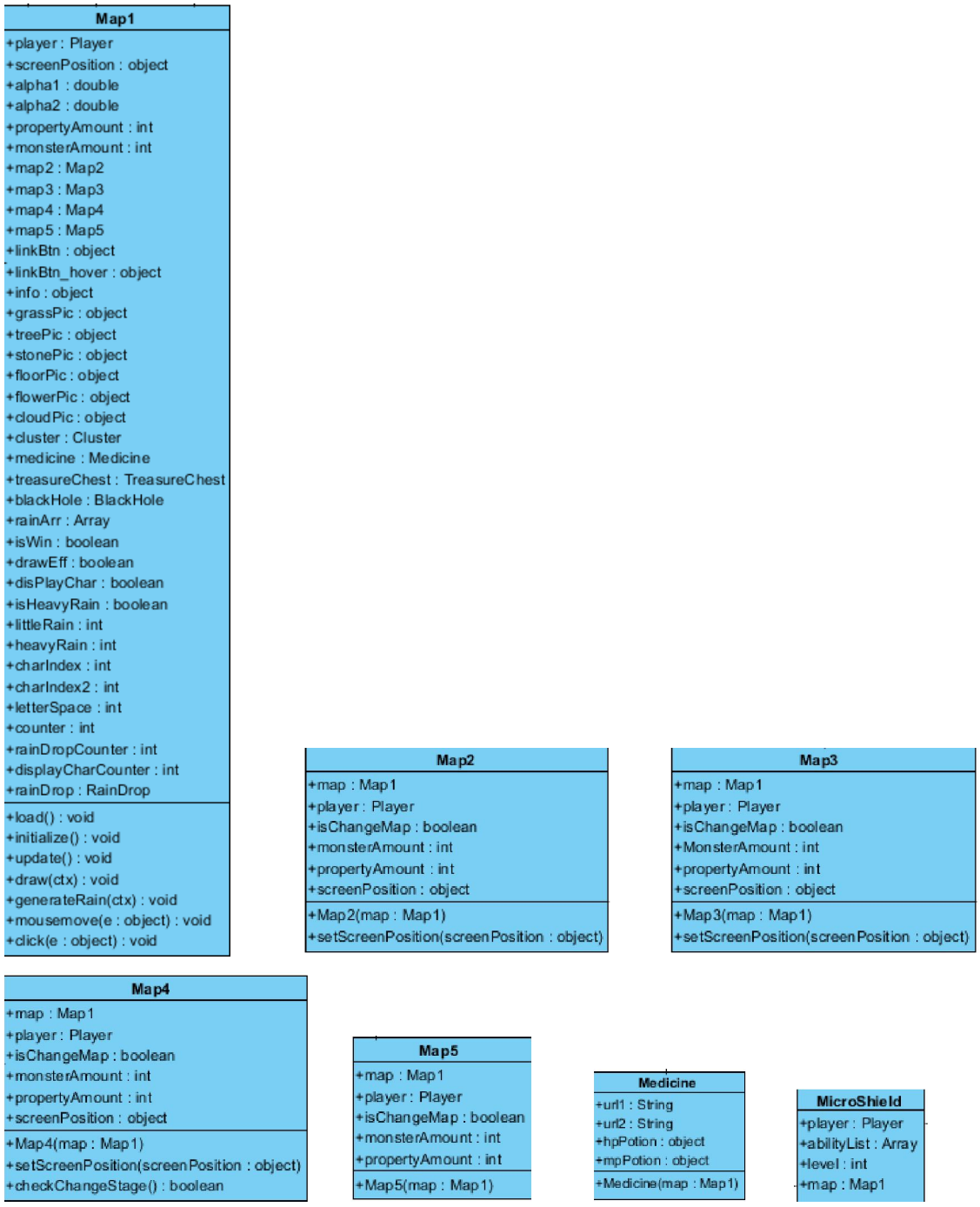
(A) Class diagram：

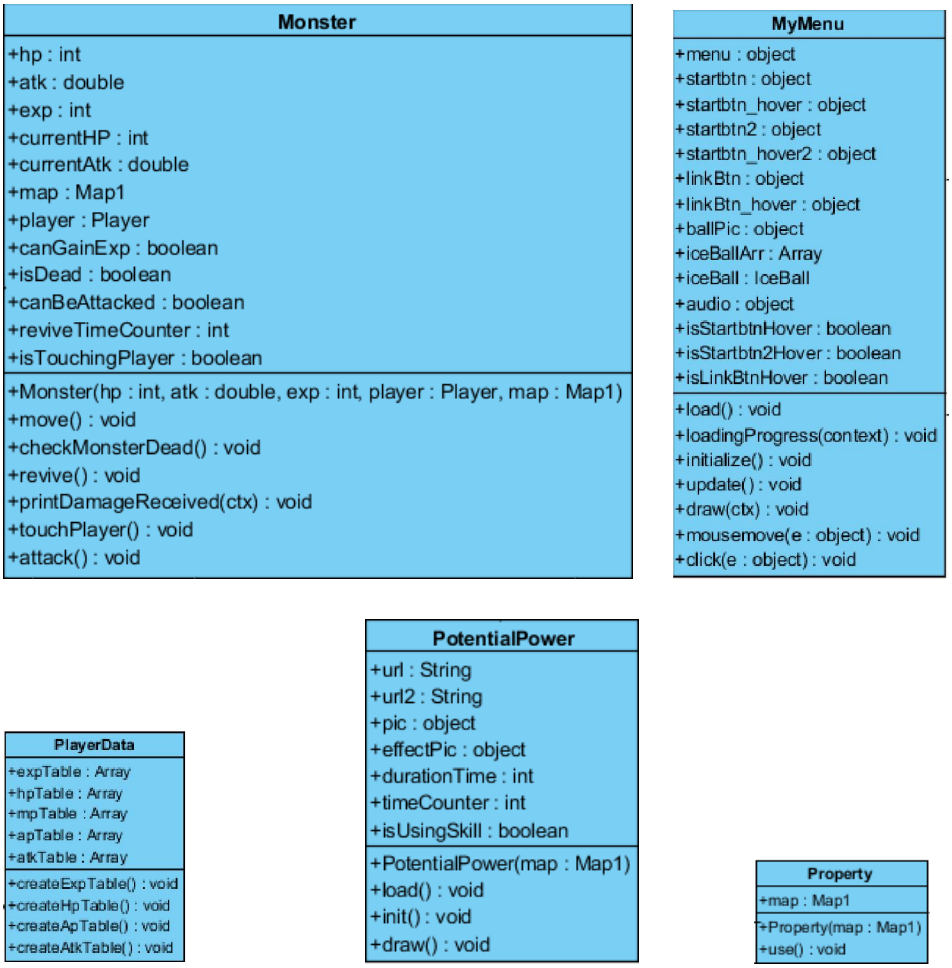
1.各 Class 內容：

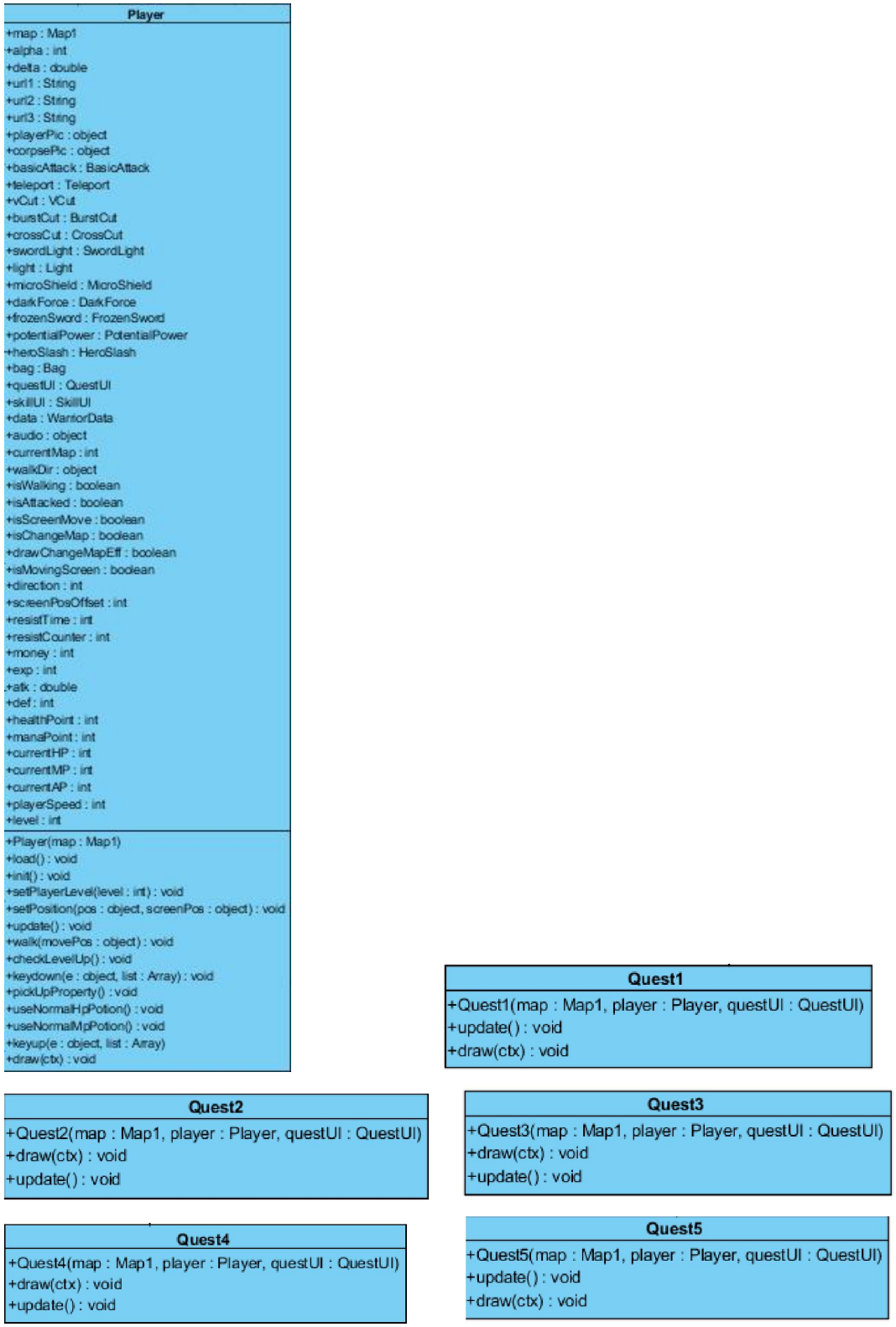


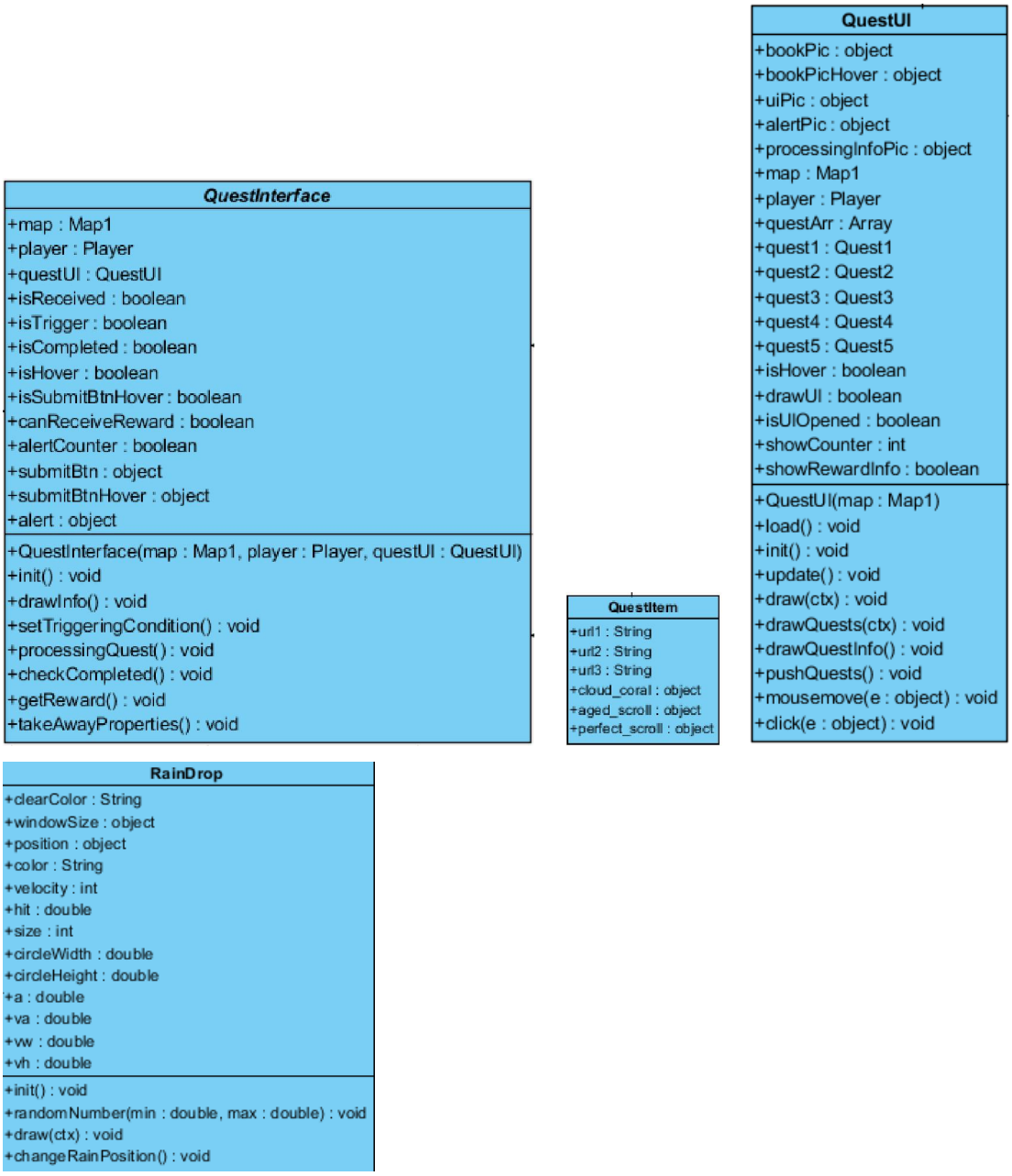

BlackBall
+url : String
+pic : object
+ballNumber : int
+picArr : Array
+counter : int
+angleOffset : int
+monster : object
+BlackBall(map : Map1, monster : object, ballNumber : int)
+load() : void
+setBallPosition(angleOffset : int) : void
+pushBall() : void
BlackHole
+url : String
+pic : object
+blackBall : BlackBall
+counter : int
+ballAngleOffset : int
+map : object
+player : object
+BlackHole(map : Map1)
+load() : void
+draw(ctx) : void
+update() : void
+usingBlackBall() : void
Boomerang
+url : String
+pic : object
+ballNumber : int
+picArr : Array
+counter : int
+monster : object
+Boomerang(map : Map1, monster : Boss)
+load() : void
+setBallNumber(number : int) : void
+setBallPosition() : void
+back() : void
Boss
+url1 : String
+url2 : String
+url3 : String
+pic : object
+icePic : object
+guardPic : object
+frostbolt : Frostbolt
+blackball : BlackBall
+boomerang : Boomerang
+thunder : Thunder
+counter : int
+alpha : int
+defenderArr : Array
+deadDevilCounter : int
+ballAngleOffset : int
+isUsingSkill : boolean
+canBeAttacked : boolean
+name : String
+Boss(map : Map1, player : Player)
+load() : void
+init() : void
+initDefender() : void
+draw(ctx) : void
+update() : void
+usingThunder() : void
+usingBoomerang() : void
+usingBlackBall() : void
+usingFrostBolt() : void
+setPosition(pos : object) : void
BurstCut
+url : String
+pic : object
+BurstCut(map : Map1)
+load() : void
+init() : void
+draw() : void
Calavera
+url : String
+pic : object
+fireBall : FireBall
+direction : int
+counter : int
+name : String
+Calavera(map : Map1, player : Player)
+load() : void
+draw(ctx) : void
+update() : void
+usingFireBall() : void
+setPositionAndDirection(pos : object, dir : int) : void
Cluster
+url1 : String
+url2 : String
+url3 : String
+url4 : String
+url5 : String
+blue_cluster : object
+red_cluster : object
+green_cluster : object
+tourmaline : object
+zircon : object
+Cluster(map : Map1)
+load() : void
+update() : void
CrossCut
+url : String
+pic : object
+picArr : Array
+CrossCut(map : Map1)
+load() : void
+init() : void
+draw() : void
DarkForce
+url : Strnig
+url2 : String
+pic : object
+effectPic : object
+durationTime : int
+timeCounter : int
+isUsingSkill : false
+DarkForce(map : Map1)
+load() : void
+init() : void
+draw() : void
+decreaseCoolDownTime(skill : object) : void
+restoreCoolDownTime(skill : object) : void

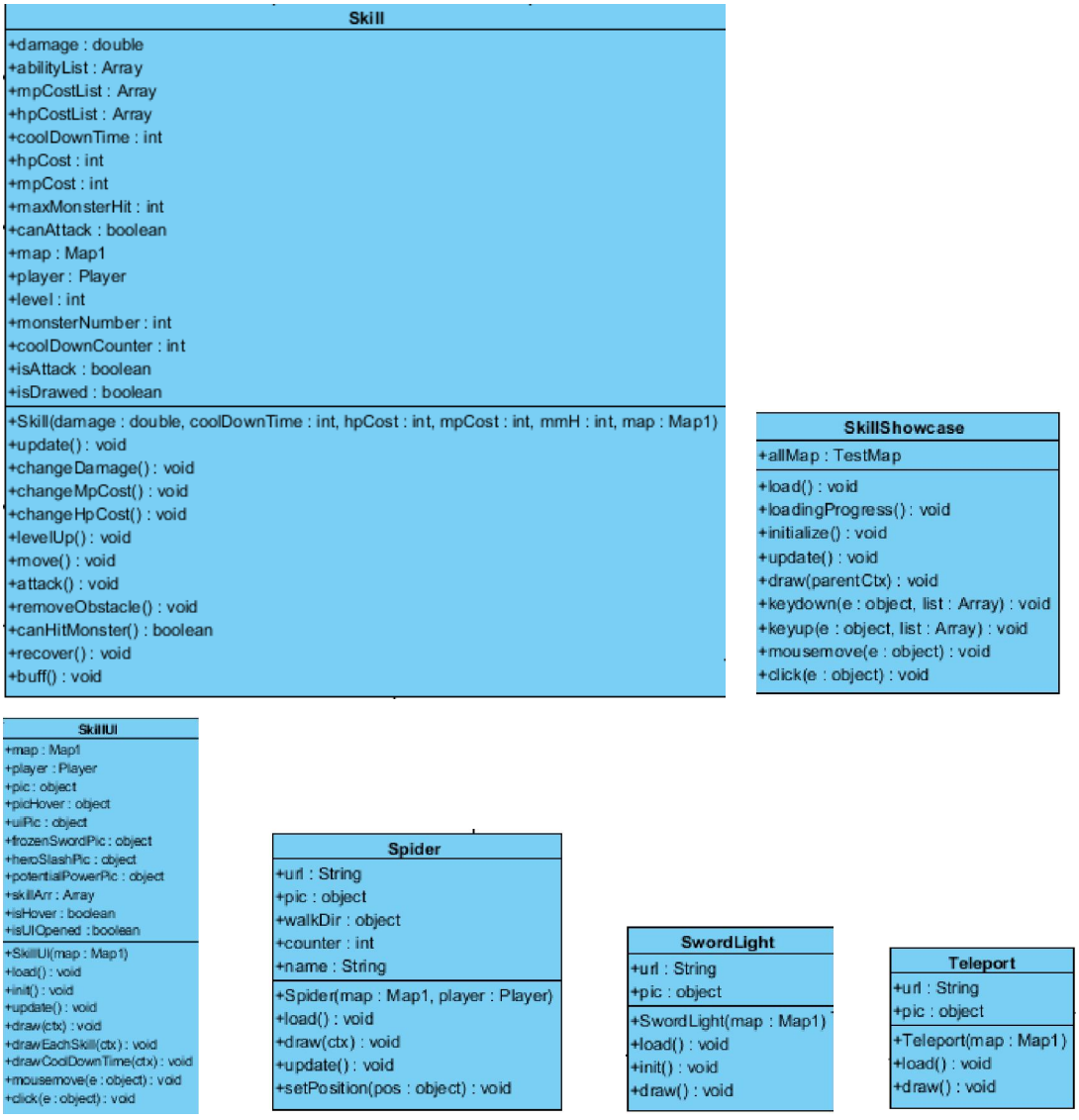

Devil
+url1 : String
+url2 : String
+pic : object
+strikePic : object
+fireStrike : FireStrike
+walkDir : object
+direction : int
+walkCounter : int
+fireStrikeCounter : int
+isUsingSkill : boolean
+boss : object
+name : String
+Devil(map : Map1, player : Player)
+draw(ctx) : void
+update() : void
+closeToPlayer() : boolean
+usingFireStrike() : void
+setPosition(pos : object) : void
+tracePlayer() : void
+traceMove(dir : object) : void
Defender
+url : String
+pic : object
+counter : int
+name : String
+Defender(map : Map1, player : Player)
+load() : void
+draw(ctx) : void
+update() : void
+setPosition(pos : object) : void
Dragonfly
+url : String
+pic : object
+walkDir : object
+counter : int
+name : String
+Dragonfly(map : Map1, player : Player)
+load() : void
+draw(ctx) : void
+update() : void
+setPosition(pos : object)
FrozenSword
+url : String
+url2 : String
+pic : object
+effectPic : object
+picArr : Array
+FrozenSword(map : Map1)
+load() : void
+init() : void
+draw() : void
Frostbolt
+url1 : String
+url2 : String
+pic : object
+effectPic : object
+picArr : Array
+counter : int
+monster : object
+Frostbolt(map : Map1, monster : Boss)
+load() : void
FireStrike
+url : String
+pic : object
+monster : object
+FireStrike(map : Map1, monster : Devil)
+load() : void
HeroSlash
+url : String
+url2 : String
+pic : object
+effectPic : object
+durationTime : int
+timeCounter : int
+direction : int
+hit : boolean
+HeroSlash(map : Map1)
+load() : void
+init() : void
+draw() : void
GameLevel1
+allMap : Map1
+load() : void
+loadingProgress() : void
+initialize() : void
+update() : void
+draw(parentCtx) : void
+keydown(e : object, list : Array) : void
+mousemove(e : object) : void
+touchstart(e : object) : void
+click(e : object) : void
IceBall
+url : String
+pic : object
+windowSize : object
+velocity : int
+hit : double
+load() : void
+init() : void
+draw() : void
+update() : void
+randomNumber() : double
Light
+url : String
+pic : object
+direction : int
+Light(map : Map1)
+load() : void
+draw() : void

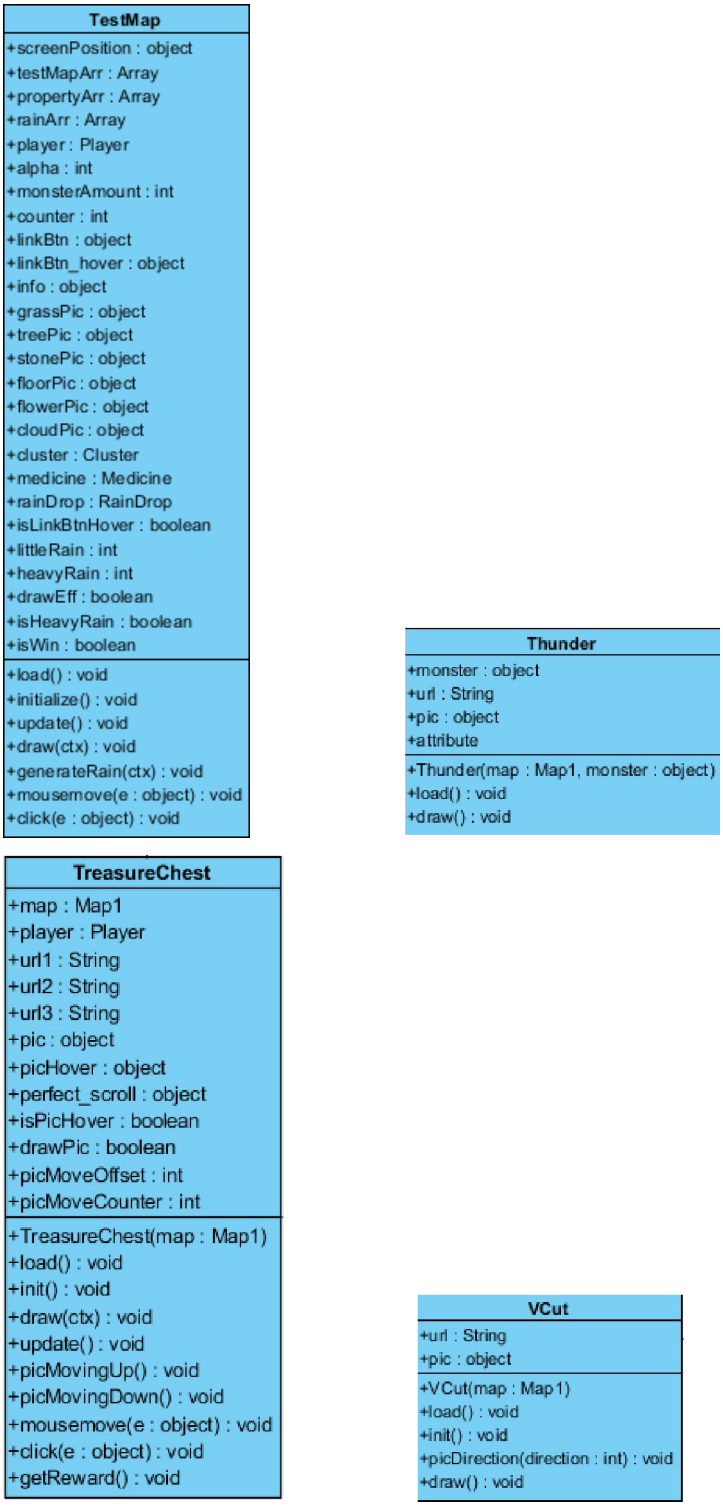

Map1
+player : Player
+screenPosition : object
+alpha1 : double
+alpha2 : double
+propertyAmount : int
+monsterAmount : int
+map2 : Map2
+map3 : Map3
+map4 : Map4
+map5 : Map5
+linkBtn : object
+linkBtn_hover : object
+info : object
+grassPic : object
+treePic : object
+stonePic : object
+floorPic : object
+flowerPic : object
+cloudPic : object
+cluster : Cluster
+medicine : Medicine
+treasureChest : TreasureChest
+blackHole : BlackHole
+rainArr : Array
+isWin : boolean
+drawEff : boolean
+disPlayChar : boolean
+isHeavyRain : boolean
+littleRain : int
+heavyRain : int
+charIndex : int
+charIndex2 : int
+letterSpace : int
+counter : int
+rainDropCounter : int
+displayCharCounter : int
+rainDrop : RainDrop
+load() : void
+initialize() : void
+update() : void
+draw(ctx) : void
+generateRain(ctx) : void
+mousemove(e : object) : void
+click(e : object) : void
Map2
+map : Map1
+player : Player
+isChangeMap : boolean
+monsterAmount : int
+propertyAmount : int
+screenPosition : object
+Map2(map : Map1)
+setScreenPosition(screenPosition : object)
Map3
+map : Map1
+player : Player
+isChangeMap : boolean
+MonsterAmount : int
+propertyAmount : int
+screenPosition : object
+Map3(map : Map1)
+setScreenPosition(screenPosition : object)
Map4
+map : Map1
+player : Player
+isChangeMap : boolean
+monsterAmount : int
+propertyAmount : int
+screenPosition : object
+Map4(map : Map1)
+setScreenPosition(screenPosition : object)
+checkChangeStage() : boolean
Map5
+map : Map1
+player : Player
+isChangeMap : boolean
+monsterAmount : int
+propertyAmount : int
+Map5(map : Map1)
Medicine
+url1 : String
+url2 : String
+hpPotion : object
+mpPotion : object
+Medicine(map : Map1)
MicroShield
+player : Player
+abilityList : Array
+level : int
+map : Map1


Monster
+hp : int
+atk : double
+exp : int
+currentHP : int
+currentAtk : double
+map : Map1
+player : Player
+canGainExp : boolean
+isDead : boolean
+canBeAttacked : boolean
+reviveTimeCounter : int
+isTouchingPlayer : boolean
+Monster(hp : int, atk : double, exp : int, player : Player, map : Map1)
+move() : void
+checkMonsterDead() : void
+revive() : void
+printDamageReceived(ctx) : void
+touchPlayer() : void
+attack() : void
MyMenu
+menu : object
+startbtn : object
+startbtn_hover : object
+startbtn2 : object
+startbtn_hover2 : object
+linkBtn : object
+linkBtn_hover : object
+ballPic : object
+iceBallArr : Array
+iceBall : IceBall
+audio : object
+isStartbtnHover : boolean
+isStartbtn2Hover : boolean
+isLinkBtnHover : boolean
+load() : void
+loadingProgress(context) : void
+initialize() : void
+update() : void
+draw(ctx) : void
+mousemove(e : object) : void
+click(e : object) : void
PotentialPower
+url : String
+url2 : String
+pic : object
+effectPic : object
+durationTime : int
+timeCounter : int
+isUsingSkill : boolean
+PotentialPower(map : Map1)
+load() : void
+init() : void
+draw() : void
PlayerData
+expTable : Array
+hpTable : Array
+mpTable : Array
+apTable : Array
+atkTable : Array
+createExpTable() : void
+createHpTable() : void
+createApTable() : void
+createAtkTable() : void
Property
+map : Map1
+Property(map : Map1)
+use() : void

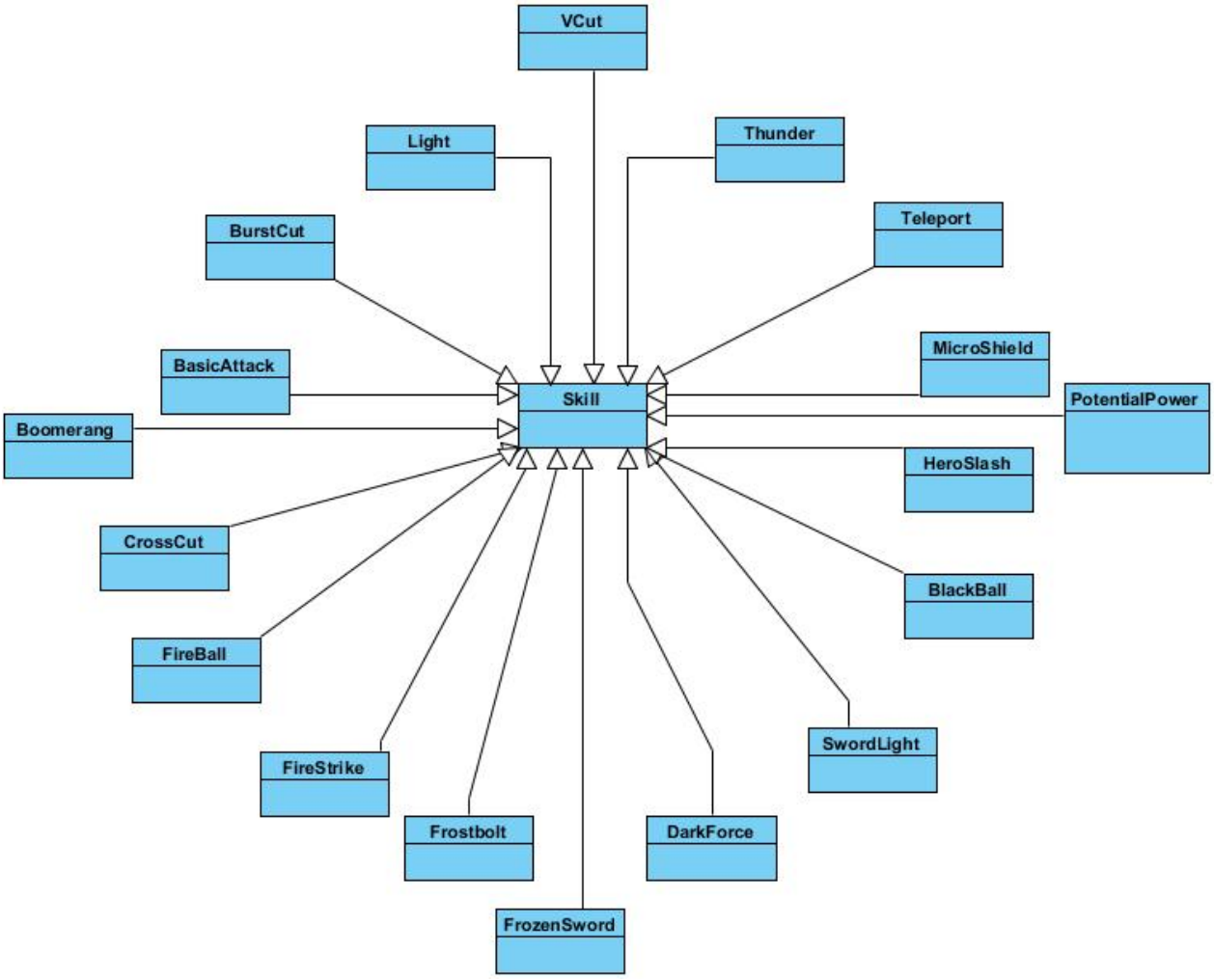

Player
+map : Map1
+alpha : int
+delta : double
+url1 : String
+url2 : String
+url3 : String
+playerPic : object
+corpsePic : object
+basicAttack : BasicAttack
+teleport : Teleport
+vCut : VCut
+burstCut : BurstCut
+crossCut : CrossCut
+swordLight : SwordLight
+light : Light
+microShield : MicroShield
+darkForce : DarkForce
+frozenSword : FrozenSword
+potentialPower : PotentialPower
+heroSlash : HeroSlash
+bag : Bag
+questUI : QuestUI
+skillUI : SkillUI
+data : WarriorData
+audio : object
+currentMap : int
+walkDir : object
+isWalking : boolean
+isAttacked : boolean
+isScreenMove : boolean
+isChangeMap : boolean
+drawChangeMapEff : boolean
+isMovingScreen : boolean
+direction : int
+screenPosOffset : int
+resistTime : int
+resistCounter : int
+money : int
+exp : int
+atk : double
+def : int
+healthPoint : int
+manaPoint : int
+currentHP : int
+currentMP : int
+currentAP : int
+playerSpeed : int
+level : int
+Player(map : Map1)
+load() : void
+init() : void
+setPlayerLevel(level : int) : void
+setPosition(pos : object, screenPos : object) : void
+update() : void
+walk(movePos : object) : void
+checkLevelUp() : void
+keydown(e : object, list : Array) : void
+pickUpProperty() : void
+useNormalHpPotion() : void
+useNormalMpPotion() : void
+keyup(e : object, list : Array)
+draw(ctx) : void
Quest1
+Quest1(map : Map1, player : Player, questUI : QuestUI)
+update() : void
+draw(ctx) : void
Quest2
+Quest2(map : Map1, player : Player, questUI : QuestUI)
+draw(ctx) : void
+update() : void
Quest3
+Quest3(map : Map1, player : Player, questUI : QuestUI)
+draw(ctx) : void
+update() : void
Quest4
+Quest4(map : Map1, player : Player, questUI : QuestUI)
+draw(ctx) : void
+update() : void
Quest5
+Quest5(map : Map1, player : Player, questUI : QuestUI)
+update() : void
+draw(ctx) : void


QuestUI
+bookPic : object
+bookPicHover : object
+uiPic : object
+alertPic : object
+processingInfoPic : object
+map : Map1
+player : Player
+questArr : Array
+quest1 : Quest1
+quest2 : Quest2
+quest3 : Quest3
+quest4 : Quest4
+quest5 : Quest5
+isHover : boolean
+drawUI : boolean
+isUIOpened : boolean
+showCounter : int
+showRewardInfo : boolean
+QuestUI(map : Map1)
+load() : void
+init() : void
+update() : void
+draw(ctx) : void
+drawQuests(ctx) : void
+drawQuestInfo() : void
+pushQuests() : void
+mousemove(e : object) : void
+click(e : object) : void
QuestInterface
+map : Map1
+player : Player
+questUI : QuestUI
+isReceived : boolean
+isTrigger : boolean
+isCompleted : boolean
+isHover : boolean
+isSubmitBtnHover : boolean
+canReceiveReward : boolean
+alertCounter : boolean
+submitBtn : object
+submitBtnHover : object
+alert : object
+QuestInterface(map : Map1, player : Player, questUI : QuestUI)
+init() : void
+drawInfo() : void
+setTriggeringCondition() : void
+processingQuest() : void
+checkCompleted() : void
+getReward() : void
+takeAwayProperties() : void
QuestItem
+url1 : String
+url2 : String
+url3 : String
+cloud_coral : object
+aged_scroll : object
+perfect_scroll : object
RainDrop
+clearColor : String
+windowSize : object
+position : object
+color : String
+velocity : int
+hit : double
+size : int
+circleWidth : double
+circleHeight : double
+a : double
+va : double
+vw : double
+vh : double
+init() : void
+randomNumber(min : double, max : double) : void
+draw(ctx) : void
+changeRainPosition() : void

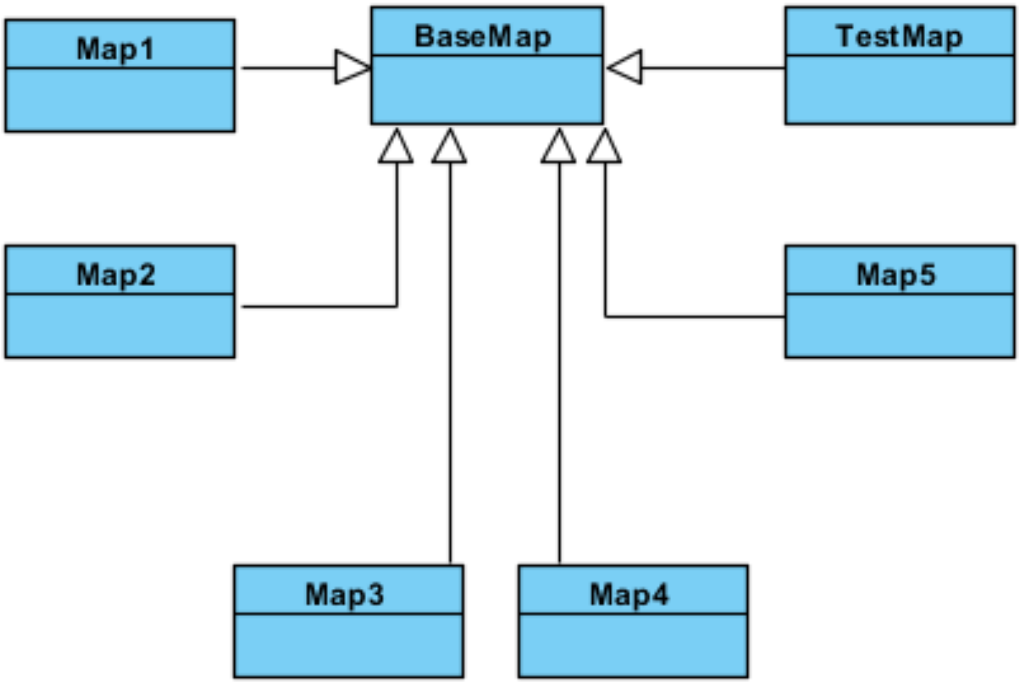

Skill
+damage : double
+abilityList : Array
+mpCostList : Array
+hpCostList : Array
+coolDownTime : int
+hpCost : int
+mpCost : int
+maxMonsterHit : int
+canAttack : boolean
+map : Map1
+player : Player
+level : int
+monsterNumber : int
+coolDownCounter : int
+isAttack : boolean
+isDrawed : boolean
+Skill(damage : double, coolDownTime : int, hpCost : int, mpCost : int, mmH : int, map : Map1)
+update() : void
+changeDamage() : void
+changeMpCost() : void
+changeHpCost() : void
+levelUp() : void
+move() : void
+attack() : void
+removeObstacle() : void
+canHitMonster() : boolean
+recover() : void
+buff() : void
SkillShowcase
+allMap : TestMap
+load() : void
+loadingProgress() : void
+initialize() : void
+update() : void
+draw(parentCtx) : void
+keydown(e : object, list : Array) : void
+keyup(e : object, list : Array) : void
+mousemove(e : object) : void
+click(e : object) : void
SkillUI
+map : Map1
+player : Player
+pic : object
+picHover : object
+uiPic : object
+frozenSwordPic : object
+heroSlashPic : object
+potentialPowerPic : object
+skillArr : Array
+isHover : boolean
+isUIOpened : boolean
+SkillUI(map : Map1)
+load() : void
+init() : void
+update() : void
+draw(ctx) : void
+drawEachSkill(ctx) : void
+drawCoolDownTime(ctx) : void
+mousemove(e : object) : void
+click(e : object) : void
Spider
+url : String
+pic : object
+walkDir : object
+counter : int
+name : String
+Spider(map : Map1, player : Player)
+load() : void
+draw(ctx) : void
+update() : void
+setPosition(pos : object) : void
SwordLight
+url : String
+pic : object
+SwordLight(map : Map1)
+load() : void
+init() : void
+draw() : void
Teleport
+url : String
+pic : object
+Teleport(map : Map1)
+load() : void
+draw() : void


TestMap
+screenPosition : object
+testMapArr : Array
+propertyArr : Array
+rainArr : Array
+player : Player
+alpha : int
+monsterAmount : int
+counter : int
+linkBtn : object
+linkBtn_hover : object
+info : object
+grassPic : object
+treePic : object
+stonePic : object
+floorPic : object
+flowerPic : object
+cloudPic : object
+cluster : Cluster
+medicine : Medicine
+rainDrop : RainDrop
+isLinkBtnHover : boolean
+littleRain : int
+heavyRain : int
+drawEff : boolean
+isHeavyRain : boolean
+isWin : boolean
+load() : void
+initialize() : void
+update() : void
+draw(ctx) : void
+generateRain(ctx) : void
+mousemove(e : object) : void
+click(e : object) : void
Thunder
+monster : object
+url : String
+pic : object
+attribute
+Thunder(map : Map1, monster : object)
+load() : void
+draw() : void
TreasureChest
+map : Map1
+player : Player
+url1 : String
+url2 : String
+url3 : String
+pic : object
+picHover : object
+perfect_scroll : object
+isPicHover : boolean
+drawPic : boolean
+picMoveOffset : int
+picMoveCounter : int
+TreasureChest(map : Map1)
+load() : void
+init() : void
+draw(ctx) : void
+update() : void
+picMovingUp() : void
+picMovingDown() : void
+mousemove(e : object) : void
+click(e : object) : void
+getReward() : void
VCut
+url : String
+pic : object
+VCut(map : Map1)
+load() : void
+init() : void
+picDirection(direction : int) : void
+draw() : void

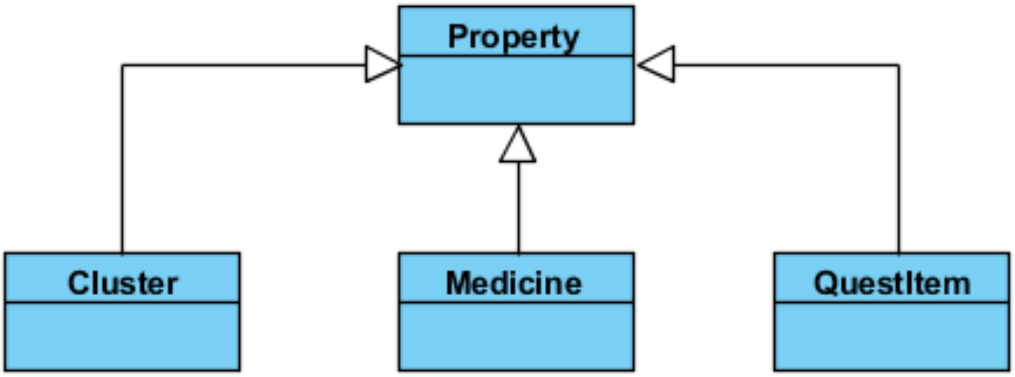
2.Class 之間的關係(程式架構)：



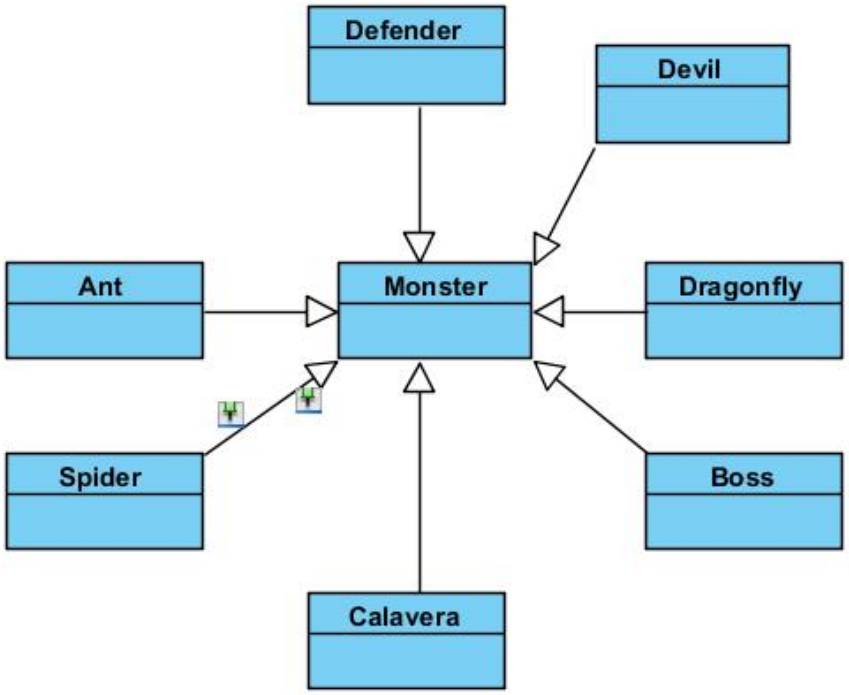
#上圖為技能(包含玩家技能與怪物技能)的繼承架構



#上圖為地圖的繼承架構



#上圖為道具的繼承架構



#上圖為怪物的繼承架構

(B) Class 功能說明：

Class Name	Number of lines(.js)	Description
Ant	95	Monster 的子類別
Arrays	256	用來儲存各種 Array
Bag	176	管理角色背包
BaseMap	78	所有 Map 的父類別
BasicAttack	69	玩家的基本攻擊
BlackBall	98	技能
BlackHole	54	考驗關卡中製造黑球的黑洞
Boomerang	112	Inquistior 的技能
Boss	214	管理魔王基本資訊及技能
BurstCut	154	玩家技能(攻擊型)
Calavera	92	考驗關卡中的怪物
Cluster	23	地圖上的道具
CrossCut	136	玩家技能(攻擊型)
DarkFoce	91	玩家技能(輔助型)
Defender	51	魔王的守衛
Define	54	定義各種路徑(像是圖片、音效)

Class Name	Number of lines(.js)	Description
Devil	265	魔王的守衛
Dragonfly	98	Monster 的子類別
FireBall	78	Calavera 的技能
FireStrike	60	Devil 的技能
Frostbolt	98	Inquistior 的技能
FrozenSword	114	玩家技能
GameLevel1	80	遊戲的普通模式
HeroSlash	124	玩家技能
IceBall	38	起始畫面特效設定
Light	101	玩家技能
LoadGame	100	將所有遊戲中會用到的 Class 載入
MainGame	8	管理各個 Level
Map1	518	遊戲初始地圖、設定遊戲內所需要用到的物件及元素(例如:角色、道具、怪物)
Map2	95	管理第二張地圖的條件判斷
Map3	93	管理第三張地圖的條件判斷
Map4	65	管理考驗關卡的條件判斷
Map5	37	管理魔王地圖的條件判斷
Medicine	16	管理各種的藥水
MicroShield	31	玩家技能(輔助型)
Monster	65	所有怪物的父類別，定義基本的怪物資料(例如：血量、攻擊力)
MyMenu	146	管理遊戲初始畫面會觸發的事件
Player	604	管理角色會使用到的物件(例如：技能、任務、背包)
PlayerData	50	定義所有職業的基本資料(血量、魔力、行動力及經驗值表)，每種職業會有各自的曲線
PotentialPower	82	玩家技能（輔助型）
Property	6	所有道具的父類別
Quest1	116	管理任務一的內容
Quest2	141	管理任務二的內容
Quest3	159	管理任務三的內容
Quest4	158	管理任務四的內容
Quest5	172	管理任務五的內容
QuestInterface	61	任務介面，定義任務中會發生的事件
QuestItem	13	管理任務中會獲得的道具

Class	Number of lines(.js)	Description
QuestUI	189	顯示任務內容、達成條件及獎賞
RainDrop	74	管理下雨動畫
Skill	81	所有技能的父類別
SkillShowcase	48	管理遊戲技能展示模式的內容
SkillUI	146	顯示技能的資訊
Spider	97	Monster 的子類別
SwordLight	153	玩家技能(攻擊型)
Teleport	39	玩家技能(移動型)
TestMap	239	管理技能展示模式的地圖
Thunder	72	Inquistior 的技能(玩家血量低於50%時會觸發)
TreasureChest	78	考驗關卡終點的寶箱
Vcut	140	玩家技能(攻擊型)
WarriorData	48	管理職業---戰士的基本資料
Total number of lines	6949	

四、結語

(A)問題及解決方法：

問題	對應的解決方法
畫地圖時使用二維陣列很難處理，因為我們需要處理跨地圖，而每個地圖的障礙物都不一樣。	改用三維陣列(Ex:arr[currentMap][row][col])。
實作卷軸模式時螢幕移動異常(人物跑出螢幕外、螢幕移動不連續)。	螢幕與玩家移動方式相同。
改為卷軸模式後，有時候人物移動會有延遲、停頓的現象。	找出大概原因(我們一次畫太多物件)，因此改為只畫螢幕範圍內的物件。
在卷軸模式下，圖片位置異常。	發現是因為座標轉換的關係，因此就先把物件轉成螢幕座標，然後畫出來，之後再轉為地圖座標。
因管理角色的 Class 同時處理技能及道具，導致整個 Class 的內容過於龐大。	使用繼承將道具及技能獨立。
產生道具的方式與產生障礙物相同，都是用一個陣列儲存，因此陣列會變很多。而且道具的位置是固定的，玩家很方便記憶。	改為隨機產生道具，用 Math.random()的函式產生數字後，再 push 到陣列裡面，這樣既可以解決陣列很多的問題，同時也解決了道具位置固定的問題。
怪物死亡後玩家仍然會受到攻擊，由於怪物是以圖片方式顯現，我們在怪物死亡後只是將圖片改為不顯示，但實際上怪物的攻擊力依然會存在，殺死也能得到經驗值。	除了讓圖片消失之外，將經驗值以及攻擊力都調整為零，然後額外新增一個復活機制，待怪物復活後再將其攻擊力及經驗值恢復。
怪物有無觸碰到玩家的判斷是放在 update 處理，因此會於一秒之間執行很多次，導致玩家會連續受到攻擊。	讓玩家有無敵時間，無敵時間內不會進到判斷。
由於所有地圖在同一個 Class 處理，因此會遇到變數容易重複的問題，同時也導致 Class 內容過多。	創建一個 BaseMap 的 Class 來定義所有地圖會使用到的功能，並將所有 Class 繼承它。
想要讓物件呈圓形排列，並讓它們往外擴散移動(實作魔王的黑球技能)，但卻不知道如何下手。	上網搜尋發現可以用三角函數及圓的方程式來達成。
實作下雨、下雪特效時，雨雪不夠分散，擬真度也不夠。	後來想到可以設定隨機速度跟隨機初始位置，這樣看起來就會有些交錯變化了。
實作怪物追蹤玩家的部分時，原本想要讓怪物維持上下左右行動，但發現這樣做的話在判斷與選擇路徑上會很麻煩。	讓怪物的方向不僅限於上下左右(可以斜走)，當玩家進入追蹤範圍時才會觸發，若遠離的話變為隨機行走。

(B)時間分配表：

週次	花費時間(小時) 蔡一玄+梁皓鈞	進度
1	5	構想遊戲內容，了解老師提供的Framework。
2	6	上網找尋遊戲內所需的一些基本素材，試著將初始畫面做出來。
3	7	設計地圖的障礙物分布，並實作遊戲的第一張地圖，人物可在地圖上行走，遇到障礙物不能走。
4	10	人物改為連續行走(持續按壓按鍵可一直走路)，並且可以透過基本的攻擊來移除障礙物，移除後可以得到物品。
5	9	設計第二張地圖並實作出來，且人物能夠切換地圖。此外，增加新的技能及特效。
6	20	擴增並重新設計地圖，將地圖改為卷軸模式。
7	9	由於改為卷軸模式，地圖變大，圖片數量也隨之增加，導致速度變慢，因此本周著重於增進效能。
8	8	加入怪物，顯示人物的基本素質(HP、MP、AP)，並優化界面。
9	25	人物可以將地圖上的物品撿入包包內，且包包會顯示物品的數量。人物可以透過基本攻擊或技能來殺死怪物，而怪物為隨機移動。調整 Class，實作多型與繼承。
10	37.5	人物可透過打怪來獲得經驗值，並升等，若怪物碰觸到人物會顯示其造成的傷害。設計人物的經驗、攻擊力以及 HP、MP 成長曲線並實作。怪物死亡後過一段時間內會復活，並增加怪物的數量。將地圖上的道具改為隨機產生。人物可以使用藥水。

11	35	設計並加入任務系統，新增轉場動畫。新增一些新介面並將所有介面再次優化。更新起始畫面背景及按鈕，新增技能展示模式。
12	8	新增人物技能，規劃新任務。
13	8	規劃新地圖，實作新任務。
14	25	設計 Boss 及其手下技能、特殊能力，並實作出來，新增人物特殊技能。設計考驗關卡及裡面的元素，並實作。
15	38	新增下雨及下雪動畫特效。再度新增 Boss 技能及人物技能。加入冷卻技能顯示的介面。完成遊戲的勝利及失敗條件。
16	35	進行遊戲測試，並完成整個遊戲及書面報告。
17	0	期末展示成果。

(C)貢獻比例：

蔡一玄：50%

梁皓鈞：50%

(D)檢核表：

項目	是否完成	說明
解決 Memory leak		JavaScript 無此問題
自訂遊戲 Icon		網頁遊戲無 Icon
全螢幕啟動	是	
修改 Help->About		
初始畫面說明按鍵、滑鼠用法及密技	是	由於說明較繁複，故改成在一個網頁中顯示。
上傳 Setup 檔		網頁遊戲無 Setup 檔
報告字型、點數、對齊、行距、頁碼等格式皆正確	是	
報告封面及側面格式正確	是	

(E)收穫:

1.蔡一玄：在程式撰寫的部分，我最大的收穫就是更加有物件的概念，包括繼承跟多型。另外，由於我在遊戲設計及美工特效上也都有花點時間，因此以整個遊戲開發來說，我不僅學習到許多程式撰寫上的技巧，同時也對一個遊戲所需的元素有了大概的方向，而特效的部分我也是試著用老師的 Framework 加上網路上參考的資料製作，接觸這部分後我才發現，原來用 Canvas 也能做出如此逼真的特效。而也因為這個遊戲是花一學期做的，整個架構一定會比之前寫過的程式還大許多，因此，我在 class 整合、溝通、統整的能力大大的提升，加上許多問題不是一下子就能解決的，所以在擴充及除錯的能力也進步許多。

在書面報告的部分，我第一次學習到如何繪出 Class Diagram，這部分以前只有照著別人設計好的圖去撰寫程式，並沒有親手製作。所以也因為報告中需要這部分，我特別上網找尋有關 class 之間關係的描述方法(Generation、Association、Realization・・・等)。

2.梁皓鈞：對我來說，收獲絕對比一般同學還要大。最主要是因為我大一時是在電子系，修讀的物件導向沒有資工系這麼深入。因此有很多概念我都是新學到的。像是 Polymorphism 的概念，在遊戲做到道具時，我本來以為是要為可以使用的道

具，如藥水，建立一個 Usable Tool 的 Class 跟 交任務用的道具的 Class 之類。後來跟老師講了才發現原來是我對 Polymorphism 的觀念不熟悉。可以直接在 Class 中建立一個 Method 叫 Use ，然後再透過 Polymorphism 的特性變成有不同的 Use 方式。由於我在這門課之前，沒有學太多 JavaScript，因此在這門課時，我算是邊做邊學。所以這門課學到最大收獲，應該就是學會點 JavaScript 了。在網上季到的 JavaScript 大部份都是 DOM 類型的，很少會像這種真的是去執行遊戲程式這種，所以算是接觸了一個新的領域新的認知。

在除錯方法就更加有趣了，我的認知只有像 IDE 按一下跑出來那種感覺。可是在現在 JavaScript 中，因為他是一個 Scripting Language，所以本身就是用 Browser 去跑，然後我就打開了人生第一次的 Debug mode 去看那些分析跟錯誤回報，後來發現原來 JavaScript 就算寫錯了，也是可以跑的動，只是在 Debug mode 上會有錯誤回報而已。這算是一種新的認知，畢竟我從來都沒有碰過 Scripting Language 與 Browser 的相關工具。

(F)心得：

1.蔡一玄：這是我第一次接觸開發一個小遊戲，由於對遊戲開發及運作不是很熟悉，因此我覺得如果開學再開始著手的話，進度可能會太慢，於是我在寒假時就開始接觸老師提供的 Framework。一開始，我先選擇比較熟悉的 C++來練習，試著改一些變數數值，看看會有甚麼變化。花了一些時間嘗試後，我便打算選擇做網頁遊戲，原因是 C++的部分在上學期已經碰過了，而 Java 的部分我之前也有稍微接觸，剛好網頁的部分主要用到的是 JavaScript，也是我完全沒學過的語言，所以我想藉這個機會嘗試看看新的開發平台。既然選擇了新的語言，我就得開始熟悉它，於是我就到 Codecademy 找了 JavaScript 的練習，後來發現其實它並不會太難，跟 C++比起來少了變數的型態、指標，我們也不用擔心有 memory leak 的問題。把 Codecademy 上面的教學完成差不多後，我便開始試著了解老師給的範例遊戲，但是，過程中發現它的規模比我想像中還大，加上有些地方的語法我不是很懂，尤其是函式的部分，JavaScript 與其他語言不同的是物件基本上是以函式的形式存在，

加上函式還有很多種表達方式，像是 function expression、function declaration、generator function expression、arrow function expression、anonymous function expression，所以很容易會搞混。開學後，我與隊友討論後決定要做類似 RPG 風格的遊戲，不過我覺得要做這類型的遊戲說實在要做得很完整才會好玩，所以對我們來說是相當有挑戰性的。過程中我遇到的第一難關就是由普通地圖轉換為卷軸地圖，由於我們剛開始的規劃及安排都是以普通地圖設計，因此在轉換為卷軸地圖的過程中很多物件都因此發生異常，所以我們只好將部分物件重新設計。接著，玩家以及怪物之間的設計部分也是個麻煩點，因為要考慮到的因素蠻多的，我們需要配合玩家的攻擊力及血量成長曲線來設計怪物的強度，還有怪物獲得經驗值的部分也需要與玩家升級經驗配合，來達到彼此間的平衡。而我們的遊戲設計是以任務為主軸，原本想要加上劇情使內容更豐富，也更貼近 RPG 遊戲的理念，但因為劇情的部分要自己構想，這部分會花很多時間，加上這門課主要的重點是物件導向程式設計，所以我果斷放棄劇情這塊。而在特效及美工的部分我原本不是很在意，想說還是以程式邏輯為主，但後來在每次測試的時候都會感覺畫面不太對，少了點感覺，於是我又額外花些時間在處理特效的部分。在完成遊戲之時，我真的覺得很有成就感，第一次完成規模稍微大的程式，之前頂多只有寫過大概 6~7 個 Class 的程度，在這個遊戲中居然用到了接近 60 個 Class。其實我一開始有點擔心程式規模大會承受不住，但後來發現做起來其實也沒那麼難，多去嘗試幾個方法就有辦法解決問題，到遊戲後期，很多物件都是參照前期製作的框架去改變，因此 Class 的數量也越來越多，而且越到後面會覺得遊戲越來越有趣，也就更有動力繼續做下去。最後，很感謝老師以遊戲的方式讓我們學習物件導向程式，讓整個課程變得更有趣！

2.梁皓鈞：對我來說 <<物件導向程式設計實習>>，是一個非常大的挑戰。不論是在教育背景還是課程背景也是。我因為是來自香港的僑生的關係，本身教育背景就不像系上某些優秀的同學早就已經有非常豐富的實作經驗及能力。有的早就已經在外面接了幾年的案子，有的幫科技部寫軟體。相比之下，沒有實作經驗的我對於物件導向大概也只有少量理論上的知識且談不上是了解。更嚴重的問題是於大學第一年時我是電子工程系的學生，雖然我在電子系已經修讀過 <<物件導向程式設計

>> 的理論課程，並且也把前一學期資訊工程系的 <<物件導向程式設計課>> 給抵免了。但畢竟是在電子工程系所學的，難度相對比較低，且知識上也只有學習到 Overloading 部份。站在這樣的背景立場上，加上不論是 C++，JavaScript 還是 Java 我都沒什麼接觸。這等於我是站在既不會任何一種課程需求的程式語言，又不會物件導向的理論基礎。

由於我個人的志向關係，我不希望成績因為我的知識未達到已不如理想，同時也希望把這些知識學習起來。在寒假時我開始自學 JavaScript，同時也把陳偉凱教授於前一個學期教授的 <<物件導向程式設計>> 課程的用書買回來自修一下。當然也不時請教了好幾位在該課程取得好成績學習充足的同學。在這個寒假中，雖然自修出來的效果，絕對不如上教授的課完整，也必然會有一些漏看或沒有注意到的重點，但我確實已經盡了力了。

在課程一開始時，我也跟陳教授談到我的情況，教授也很樂意解答我的疑問。甚至之後在遊戲開發初期對於這種設計不太熟悉時，預約時間前往研究室請教教授，教授也非常願意教我。這幫助了初期的我的疑惑。到後來慢慢遊戲成熟起來後，就開始問題愈來愈少了。我在整個遊戲中主要負責了角色，部份地圖，少量物品的程式碼，前期素材找尋，地圖分佈以及任務設計。在地圖設計上請教過教授有關多張地圖是否要放在多個 GameLevel 中，但這樣會影響到變成每一次回到 GameLevel 時所有東西都會刷新到原始狀態，要解決這個問題，就只能開一個很大的 global 去存放資訊，每一次回到 GameLevel 時就要重新載入。這樣對於運算量很大，可能每一次進入關卡地圖都要卡一下。教授最後建議我們可以做三維陣列，其中的二維是地圖，而第三維就是用來切換第幾張地圖。這樣的做法的確很省時，而且在之後的增加項目時也非常快速。我覺得這讓我學到盡可能減低所有非必要運算，想方法去以最簡單的方式去開發。

在角色設計方面，開始用到比較多的 Class，因此我也不斷向比較有經驗的同學請教 Class 的設計要怎樣才可以做到比較之後可擴充性。幸好同學們都願意指導我一下，雖然我花時比較長才完成，但總算可以順利製作。在角色行走時的一些限

制上一開始的運算比較多 bug。但缺乏經驗的我也只能不斷去使用數學公式計算出哪個一位置不能行走。最後數學公式愈堆愈多，卻還是有一點少 bug。最後請教了教授才知道原來是要計算出人物的四點有沒有超越障礙物。經過教授的提點馬上就成功了。

與程式設計方面比較沒關係不過比較有心得的是任務設計上。雖然我沒有設計出很多任務，但卻充分讓我感受到 RPG 類遊戲的任務設計十分複雜。要估算玩家在這個任務時大概是多少等級，要給予多少經驗值以及道具。而且還要配合到角色等級的曲線，在某些等級可能會因為希望玩家多去解分支任務或者去打怪，而在主線任務中給予比較少的經驗值讓玩家無法單純透過主線任務升等。讓我不禁覺得遊戲公司特別是 RPG 遊戲的關卡設計者真是辛苦。我很慶幸我的組員是一個學習能力高的優秀學生，體諒我因為沒有相關知識基礎的背景，因此我在前期的工作效率比較低，主要都把時間花在學習上。在中後期我的理論基礎開始成熟且開始熟悉了 JavaScript 令我效率開始提高。

總括而言，這一門課雖然我一開始十分吃力，但經過教授的指導以及看書自修後，開始慢慢熟悉物件導向的原理以及 JavaScript。十分高興可以從這門課透過製作遊戲去熟悉物件導向以及遊戲的機制。這門課對我十分有意義。

附錄：

1.Ant.js：

```
var Ant = function(map,player){
  Monster.call(this, 50, 15, 3, player, map);//Call the parent's constructor(hp,atk,exp,player)
  this.load = function(){
    this.url = define.imagePathMonster + "ant.png";
    this.pic = new Framework.AnimationSprite({url:this.url, col:3, row:4, loop: true, speed: 3});
    this.pic.scale = 0.8;
    this.walkDir = {x:0, y:0};
    this.counter = 0;
    this.name = 'ant';
  }

  this.draw = function(ctx){
    var picPos = this.pic.position;
    var screenPos = this.map.screenPosition;
    var playerPos = this.player.playerPic.position;
    ctx.fillStyle = 'rgb(76, 13, 89)';
    ctx.fillRect(picPos.x-screenPos.x-30, picPos.y-screenPos.y-30, 60*(this.currentHP/this.hp), 10);
    if(this.isTouchingPlayer){
      this.printDamageReceived(ctx);
    }
  }

  this.update = function(){
    if(this.counter > 90){
      var randomDir = Math.floor(Math.random() * 4);
      this.counter = 0;
    }
    this.counter++;
    this.move(randomDir);
    this.checkMonsterDead();
    this.revive();
    this.pic.update();
```

```
    if(this.player.currentHP > 0 && this.player.isAttacked === false){
      this.touchPlayer();
    }

    if(this.player.resistCounter === 0){
      this.player.isAttacked = false;
      this.isTouchingPlayer = false;
      this.player.resistCounter = this.player.resistTime;
    }
   }

   this.setPosition = function(pos){
    this.pic.position = {
      x: pos.x * 32 + 16,
      y: pos.y * 32 + 16
    };
   }
}

Ant.prototype = Object.create(Monster.prototype);
Ant.prototype.constructor = Monster;

Ant.prototype.move = function(randomDir){
 switch (randomDir) {
   case 0://up
    this.walkDir = {x: 0, y: -1};
    this.pic.start({from:9, to:11, loop:true});
    break;
   case 1://left
    this.walkDir = {x: -1, y: 0};
    this.pic.start({from:3, to:5, loop:true});
    break;
   case 2://down
    this.walkDir = {x: 0, y: 1};
    this.pic.start({from:0, to:2, loop:true});
```

```
      break;
    case 3://right
      this.walkDir = {x: 1, y: 0};
      this.pic.start({from:6, to:8, loop:true});
      break;
  }
  if(this.map.canMove(this.pic.position.x + this.walkDir.x, this.pic.position.y + this.walkDir.y, randomDir)){
      this.pic.position.x += this.walkDir.x;
      this.pic.position.y += this.walkDir.y;
  }
};


Ant.prototype.touchPlayer = function(){
    var picPos = this.pic.position;
    var playerPos = this.player.playerPic.position;
    if(Math.abs(picPos.x - playerPos.x) < 33 && Math.abs(picPos.y - playerPos.y) < 33 && !this.isDead){
        if(this.player.currentHP <= this.currentAtk){
            this.player.currentHP = 0;
        }
        else{
            this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
        }
        this.player.isAttacked = true;
        this.isTouchingPlayer = true;
    }
};
```

2.Arrays.js：

```
var Arrays = function(mapType){

   this.mapArray =  [

      [   //Map1

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 1, 1, 1, 1, 1, 1, 7, 7, 8, 0, 0, 0, 8, 7, 7, 0, 0, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 8, 1, 8, 1, 1, 1, 1, 1, 0, 0, 7, 7, 8, 0, 0, 0, 8, 7, 7, 0, 0, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 8, 0, 8, 0, 0, 0, 0, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 8, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 8, 0, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 8, 1, 8, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 8, 0, 8, 0, 0, 0, 8, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 8, 1, 1, 1, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 7, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 8, 8, 0, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 7, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 7, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 7, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 7, 1, 0, 1, 0, 1, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 7, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 7, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 7, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 1, 1, 1, 8, 7, 7, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 7, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 8, 7, 7, 7, 1, 0, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 7, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 8, 7, 0, 7, 1, 0, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 8, 7, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 8, 7, 7, 7, 1, 0, 0, 0, 0, 0, 8, 8,
7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],
```

```
            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 8, 1, 1, 1, 8, 0, 0, 0, 0, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 8, 8, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 0, 0, 7, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 0],

            [8, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7,
7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 0, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 0],

            [8, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7,
7, 7, 7, 0, 7, 7, 7, 7, 7, 0, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 0],

            [8, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 0, 7, 7, 7, 0, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 8, 8, 8],

            [8, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 0, 0, 0, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0,
0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7,
7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 7, 7, 7, 7, 7, 7, 0, 7, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 8, 1, 1, 1, 8, 0, 0, 0, 0, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 0, 7, 7, 7, 7, 0, 7, 7, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 8, 1, 1, 1, 8, 0, 0, 0, 0, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 8, 1, 1, 1, 8, 0, 0, 0, 0, 0, 0, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8],

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]

      ],

      [   //Map 2_1

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

            [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 8, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 8, 8, 8, 8, 1, 1, 1, 1, 1, 1, 7, 7, 8, 0, 0, 0, 8, 7, 7, 0, 0, 0, 0, 0, 8,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 1, 1, 1, 8, 1, 1, 1, 8, 6, 0, 0, 0],

            [8, 0, 0, 0, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 7, 1, 1, 1, 1, 8, 8, 8, 8, 1, 1, 1, 8, 0, 0, 7, 7, 8, 1, 1, 1, 8, 7, 7, 0, 0, 0, 0, 0, 8,
0, 8, 8, 8, 8, 8, 8, 8, 8, 0, 8, 0, 0, 0, 8, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 8, 1, 8, 6, 0, 0, 0],

            [8, 6, 8, 0, 8, 1, 1, 1, 1, 7, 7, 7, 7, 7, 0, 0, 8, 8, 8, 0, 7, 1, 1, 1, 8, 8, 8, 8, 1, 1, 8, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 8,
0, 8, 0, 0, 0, 0, 0, 0, 8, 0, 8, 1, 1, 1, 8, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 8, 1, 8, 6, 0, 0, 0],
```

[8, 6, 8, 0, 8, 1, 1, 1, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 1, 8, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, 8, 0, 8, 0, 8, 8, 8, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 8, 1, 8, 1, 8, 8, 8],

[8, 6, 8, 0, 8, 1, 1, 1, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 0, 8, 8, 1, 8, 8, 8, 8, 8, 8, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 8, 1, 8, 1, 1, 1, 8],

[8, 8, 8, 0, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 7, 8, 8, 8, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 8, 1, 8, 1, 1, 1, 8],

[8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 8, 1, 1, 8, 1, 1, 8],

[8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 8, 6, 0, 0, 6, 6, 6, 0, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 1, 1, 8, 1, 1, 8, 1, 8, 1, 1, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 8, 6, 0, 0, 6, 6, 6, 0, 0, 0, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 1, 1, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 0, 0, 0, 6, 6, 6, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 8, 7, 0, 8, 8, 8, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 8, 8, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 7, 0, 0, 0, 7, 0, 8, 7, 0, 8, 8, 8, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 0, 0, 8, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 7, 0, 7, 0, 0, 8, 7, 0, 8, 8, 8, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 7, 7, 7, 7, 7, 7, 7, 8, 1, 8, 8, 8, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 8, 0, 0, 0, 7, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 8, 1, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 7, 7, 7, 8, 0, 0, 0, 8, 8, 8, 8, 0, 0, 8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 7, 0, 7, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 8, 1, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 0, 0, 0, 0, 8, 8, 8, 0, 0, 0, 0, 0, 8, 8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 7, 0, 0, 0, 7, 0, 8, 7, 0, 0, 0, 0, 0, 7, 7, 0, 0, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 8, 1, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 7, 7, 7, 0, 0, 6, 6, 6, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 8, 7, 8, 6, 6, 6, 8, 7, 8, 8, 8, 8, 0, 0, 8, 0, 8, 0, 8, 0, 0, 0, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 8, 8, 8, 8, 1, 8, 8, 8, 1, 8, 1, 8, 1, 8, 1, 1, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 0, 0, 0, 6, 0, 0, 6, 6, 6, 0, 0, 0, 7, 8, 8, 8, 8, 6, 8, 8, 8, 8, 7, 8, 6, 6, 6, 8, 7, 8, 7, 7, 0, 0, 0, 8, 0, 8, 0, 8, 8, 8, 8, 8, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 0, 0, 0, 6, 1, 6, 6, 8, 1, 1, 1, 8, 1, 8, 1, 1, 8],

[8, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 8, 0, 0, 0, 6, 0, 0, 6, 6, 6, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 8, 8, 8, 8, 8, 7, 8, 0, 7, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 8, 8, 8, 8, 1, 8, 6, 8, 8, 8, 8, 8, 1, 8, 1, 1, 8],

[8, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 8, 0, 7, 0, 7, 0, 7, 0, 7, 0, 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 7, 7, 7, 0, 0, 0, 8, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 6, 6, 6, 6, 6, 8, 1, 1, 1, 1, 8],

[8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 7, 7, 8, 0, 0, 0, 7, 7, 7, 7, 7, 0, 7, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 8, 8, 8, 8, 6, 8, 7, 8, 8, 8, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 7, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 8, 6, 8, 8, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 1, 8, 8, 8, 1, 1, 1, 1, 1, 8, 1, 8, 8, 8, 8, 8, 6, 8, 7, 7, 7, 7, 8],

[8, 7, 0, 7, 7, 8, 6, 6, 8, 7, 0, 7, 0, 7, 0, 7, 7, 7, 0, 7, 0, 7, 0, 7, 8, 6, 6, 8, 7, 7, 7, 7, 7, 0, 7, 7, 7, 8, 6, 7, 6, 8, 7, 7, 7, 7, 7, 7, 7, 8, 6, 6, 6, 6, 6, 8, 7, 7, 7, 7, 7, 1, 1, 8, 6, 6, 6, 8, 1, 1, 1, 1, 8, 1, 8, 8, 8, 8, 8, 6, 8, 8, 8, 8, 7, 8],

[8, 7, 0, 7, 7, 8, 6, 7, 6, 8, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 8, 6, 7, 6, 8, 0, 0, 0, 0, 0, 7, 7, 7, 8, 6, 7, 7, 7, 6, 8, 7, 7, 7, 7, 7, 7, 8, 6, 7, 7, 7, 7, 6, 8, 7, 7, 7, 7, 1, 8, 6, 7, 7, 7, 6, 8, 1, 1, 1, 8, 1, 8, 1, 1, 1, 8, 6, 0, 0, 0, 0, 7, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 6, 8, 7, 0, 7, 0, 7, 0, 7, 0, 7, 0, 7, 8, 6, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 7, 8, 6, 7, 7, 7, 7, 7, 6, 8, 7, 7, 7, 7, 7, 8, 6, 7, 8, 8, 7, 7, 6, 8, 7, 7, 7, 8, 6, 7, 0, 0, 7, 6, 8, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 8, 8, 8, 8, 8, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 6, 8, 7, 0, 7, 0, 7, 0, 7, 0, 7, 8, 6, 8, 7, 8, 6, 8, 0, 0, 0, 0, 7, 7, 8, 6, 7, 7, 7, 7, 7, 7, 7, 6, 8, 7, 7, 7, 7, 8, 6, 7, 8, 8, 7, 7, 6, 8, 7, 7, 8, 6, 7, 0, 0, 0, 7, 6, 8, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 6, 6, 6, 6, 6, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 6, 8, 7, 0, 7, 0, 7, 0, 7, 8, 6, 7, 8, 7, 8, 6, 8, 7, 7, 7, 7, 7, 8, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 8, 7, 7, 7, 8, 6, 7, 7, 7, 7, 6, 8, 8, 7, 7, 8, 8, 0, 0, 0, 7, 6, 8, 1, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 7, 7, 7, 7, 7, 8],

```
        [8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 7, 6, 8, 7, 0, 7, 0, 7, 8, 6, 7, 0, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 8, 7, 7,
8, 6, 6, 6, 6, 6, 8, 8, 7, 7, 7, 7, 0, 0, 0, 7, 6, 8, 1, 1, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 8, 8, 8, 8, 1, 8],

        [8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 7, 7, 6, 8, 7, 7, 7, 8, 6, 7, 0, 7, 8, 7, 8, 6, 8, 7, 7, 7, 7, 8, 6, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 8, 7, 7,
8, 6, 8, 8, 8, 8, 7, 7, 0, 0, 0, 0, 0, 0, 7, 6, 8, 1, 1, 1, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 8, 1, 1, 1, 1, 8],

        [8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 6, 8, 7, 8, 6, 7, 0, 7, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 8, 7, 7,
8, 6, 8, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 7, 6, 8, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 8, 1, 8, 8, 1, 8],

        [8, 7, 0, 0, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 7, 6, 8, 6, 7, 0, 7, 7, 7, 8, 7, 8, 6, 8, 7, 7, 7, 7, 8, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 8, 7, 7,
8, 6, 8, 7, 7, 7, 7, 7, 0, 0, 0, 0, 7, 6, 8, 8, 8, 8, 8, 1, 1, 1, 1, 8, 1, 8, 1, 8, 1, 8, 6, 8, 1, 8, 8, 1, 8],

        [8, 7, 7, 0, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 7, 7, 6, 7, 0, 7, 7, 7, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 8, 7, 7,
8, 6, 8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 8, 7, 7, 7, 8, 1, 8, 1, 8, 1, 8, 1, 8, 1, 1, 1, 1, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 8, 7, 8, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 7, 7, 7, 7, 8, 1, 8, 1, 8, 1, 8, 1, 8, 1, 8, 8, 1, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 0, 7, 7, 7, 0, 7, 7, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 7, 8, 1, 8, 1, 8, 1, 8, 1, 8, 1, 1, 1, 1, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 7, 0, 7, 0, 7, 0, 7, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 8, 1, 1, 1, 8, 7, 7, 7, 7, 7, 7, 8, 8,
8, 8, 8, 8, 8, 8, 8, 0, 8, 7, 8, 7, 8, 8, 8, 8, 8, 8, 8, 0, 0, 1, 1, 8, 1, 8, 1, 8, 1, 8, 1, 8, 1, 8, 8, 1, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7, 0, 7, 0, 7, 0, 7, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 8, 0, 0, 0, 8, 7, 7, 7, 0, 7, 0, 0, 0,
8, 0, 0, 0, 8, 8, 0, 0, 8, 7, 8, 7, 0, 0, 7, 0, 0, 7, 0, 0, 0, 1, 1, 8, 1, 8, 1, 8, 1, 8, 1, 8, 1, 8, 8, 1, 8],

        [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 0, 7, 0, 7, 0, 7, 8, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 8, 1, 1, 1, 8, 7, 7, 7, 7, 7, 7, 8, 0,
0, 0, 8, 0, 0, 0, 0, 8, 8, 7, 8, 7, 0, 0, 7, 0, 0, 7, 0, 0, 0, 1, 1, 8, 1, 1, 1, 8, 1, 1, 1, 8, 1, 1, 1, 1, 8],

        [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]

    ],

    [    //Map3_1

        [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 8, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 8, 8, 8, 8, 1, 1, 1, 1, 1, 1, 7, 7, 8, 0, 0, 0, 8, 7, 7, 0, 0, 0, 0, 0, 8,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

        [8, 0, 0, 0, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 7, 1, 1, 1, 1, 8, 8, 8, 8, 1, 1, 1, 8, 0, 0, 7, 7, 8, 0, 0, 0, 8, 7, 7, 0, 0, 0, 0, 0, 8,
0, 8, 8, 8, 8, 8, 8, 8, 8, 0, 8, 0, 0, 0, 8, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

        [8, 6, 8, 0, 8, 1, 1, 1, 1, 7, 7, 7, 7, 7, 0, 0, 8, 8, 8, 0, 7, 1, 1, 1, 8, 8, 8, 8, 1, 1, 8, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 8,
0, 8, 0, 0, 0, 0, 0, 0, 8, 0, 8, 1, 1, 1, 8, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

        [8, 6, 8, 0, 8, 1, 1, 1, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 1, 8, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8,
0, 8, 0, 8, 8, 8, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 6, 8, 0, 8, 1, 1, 1, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 0, 8, 8, 1, 8, 8, 8, 8, 8, 8, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8,
0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 8, 8, 0, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 8, 8, 8, 0, 0, 8,
0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 8,
0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 8, 6, 0, 0, 6, 6, 6, 0, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 8,
0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 8, 6, 0, 0, 6, 6, 6, 0, 0, 0, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 8,
0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 7, 0, 0, 6, 6, 6, 0, 0, 0, 0, 8, 7, 0, 0, 0, 0, 0, 7, 8, 7, 0, 8, 8, 8, 0, 0, 0, 0, 0, 8, 0, 0, 8,
0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],
```

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 8, 7, 7, 0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 8, 8, 0, 0, 0, 0, 0, 6, 0, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 7, 1, 1, 1, 8, 1, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 7, 7, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 7, 7, 7, 7, 7, 7, 7, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 8, 8, 8, 8, 8, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 8, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 7, 7, 8, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 8, 0, 8, 0, 0, 8, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 8, 0, 0, 0, 0, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 8, 8, 8, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 8, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 8, 8, 8, 8, 8, 8, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 7, 7, 7, 7, 7, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 8, 8, 8, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0, 0, 7, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 8, 8, 8, 8, 8, 8, 8, 8, 7, 1, 1, 7, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 8, 6, 8, 0, 7, 7, 7, 7, 7, 0, 7, 7, 7, 8, 6, 8, 8, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 0, 8, 8, 8, 0, 0, 1, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 6, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 6, 8, 0, 0, 0, 0, 7, 0, 7, 7, 7, 8, 6, 7, 6, 8, 7, 7, 7, 7, 7, 7, 7, 8, 6, 6, 6, 6, 6, 8, 7, 7, 7, 7, 7, 8, 6, 6, 6, 8, 0, 0, 1, 1, 1, 1, 8, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 7, 6, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 7, 6, 8, 0, 0, 0, 0, 0, 7, 7, 7, 8, 6, 7, 7, 7, 6, 8, 7, 7, 7, 7, 7, 7, 8, 6, 7, 7, 7, 7, 6, 8, 7, 7, 7, 8, 6, 0, 0, 0, 6, 8, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 6, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 7, 8, 6, 7, 8, 6, 8, 0, 0, 0, 0, 7, 7, 7, 8, 6, 7, 7, 7, 7, 7, 6, 8, 7, 7, 7, 7, 7, 8, 6, 7, 8, 8, 7, 7, 6, 8, 7, 7, 7, 0, 0, 0, 0, 6, 8, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 6, 8, 7, 0, 0, 0, 0, 0, 0, 7, 8, 6, 8, 7, 8, 6, 8, 0, 0, 0, 0, 7, 7, 8, 6, 7, 7, 7, 7, 7, 7, 7, 6, 8, 7, 7, 7, 7, 8, 6, 7, 8, 8, 7, 7, 6, 8, 7, 7, 0, 0, 0, 0, 6, 8, 0, 0, 0, 0, 0, 7, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 6, 8, 7, 0, 0, 0, 0, 7, 8, 6, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 7, 8, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 8, 7, 7, 7, 8, 6, 7, 7, 7, 7, 6, 8, 8, 7, 7, 0, 6, 6, 6, 8, 0, 0, 0, 0, 0, 0, 7, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 7, 6, 8, 7, 0, 0, 0, 7, 8, 6, 7, 0, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 8, 7, 7, 8, 6, 6, 6, 6, 6, 8, 8, 0, 0, 0, 0, 0, 0, 6, 8, 0, 0, 7, 0, 0, 7, 7, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 7, 7, 6, 8, 7, 0, 7, 8, 6, 7, 0, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 8, 7, 7, 8, 6, 8, 8, 8, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 8, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 7, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 6, 8, 7, 8, 6, 7, 0, 7, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 8, 7, 7, 8, 6, 8, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 0, 0, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 7, 6, 8, 6, 7, 0, 7, 7, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 8, 7, 7, 8, 6, 8, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 7, 0, 7, 8, 6, 8, 7, 7, 7, 7, 7, 7, 7, 7, 6, 7, 0, 7, 7, 7, 7, 8, 7, 8, 6, 8, 0, 0, 0, 0, 8, 6, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 6, 8, 7, 7, 8, 6, 8, 7, 7, 7, 0, 0, 0, 0, 0, 0, 6, 6, 6, 8, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 0, 7, 7, 7, 8, 7, 8, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 8, 8, 8, 8, 0, 0, 0, 0, 7, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 0, 7, 7, 7, 0, 7, 7, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 7, 0, 7, 0, 7, 0, 7, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 8, 1, 1, 1, 8, 7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7, 0, 7, 0, 7, 0, 7, 0, 8, 7, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 8, 0, 0, 0, 8, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 0, 7, 0, 7, 0, 7, 8, 7, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 8, 1, 1, 1, 8, 7, 7, 7, 7, 7, 7, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]

],

[

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

[0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8],

[0, 0, 6, 0, 6, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 6, 0, 6, 7, 8],

[0, 0, 6, 0, 6, 0, 8, 6, 0, 6, 7, 8],

[8, 8, 8, 6, 6, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 6, 6, 6, 7, 8],

[8, 7, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 6, 0, 0, 6, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 6, 0, 0, 0, 0, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 6, 0, 0, 0, 0, 6, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 6, 0, 0, 6, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 6, 6, 6, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 6, 6, 6, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 6, 0, 0, 6, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 6, 6, 6, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 8, 8, 8, 8, 7, 6, 0, 0, 0, 0, 6, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 6, 6, 6, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 8, 8, 8, 8, 7, 6, 0, 0, 0, 0, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 6, 0, 0, 6, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 6, 6, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 7, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 7, 8],

[8, 7, 6, 0, 6, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 6, 0, 6, 7, 8],

[8, 7, 6, 0, 6, 0, 8, 6, 0, 0, 0, 8],

[8, 7, 6, 6, 6, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 6, 0, 0, 0, 8],

[8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8]

],

[

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 0, 0, 0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0, 0, 0, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 6, 6, 6, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 1, 1, 1, 7, 7, 6, 6, 6, 7, 7, 1, 1, 1, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 6, 6, 6, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

```
            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 6, 6, 6, 7, 7,
0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 6, 6, 6, 6, 7, 7,
0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 6, 6, 6, 7, 7,
0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 6, 6, 6, 7, 7, 0, 7, 0, 0, 0, 7, 8],

            [8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

            [8, 7, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 7, 8],

            [8, 7, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7,
0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

            [8, 7, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7,
0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 0, 0, 7, 7, 0, 7, 0, 0, 0, 7, 8],

            [8, 7, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6, 6, 6, 7, 8],

            [8, 7, 6, 0, 6, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0,
0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 6, 0, 6, 7, 8],

            [8, 7, 6, 0, 6, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8,
0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 0, 8, 6, 0, 0, 0, 8],

            [8, 7, 6, 6, 6, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0,
8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 6, 0, 0, 0, 8],

            [8, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 0, 0, 0, 8],

            [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 8]

    ]

  ];

  this.testMapArray = [

        [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8],

        [8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 8],
```

```
      [8, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8],
      [8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 8],
      [8, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 0, 0, 0, 0, 0, 8],
      [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
      [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
      [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
      [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
      [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
      [8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
  ];
  /*Property:
  1: blue_cluster
  2: red_cluster
  3: green_cluster
  4: hpPotion
  5: mpPotoin
  */
  this.propertyArray = [];
  if(mapType === 'scroll'){
    for(var row = 0; row < 38; row++){
      this.propertyArray[row] = [];
      for(var col = 0; col < 86; col++){
        this.propertyArray[row].push(0);
      }
    }
  }

  this.charArray = 'Thank you for playing our game!!';
  this.charArray2 = 'Thanks for your sacrifice!!';
}
```

3.Bag.js：

```
var Bag = function(player){
   this.load = function(){
      this.player = player;
      this.bagPic = new Framework.Sprite(define.imagePathCharacter + 'bag/bag.png');
      this.bagPicHover = new Framework.Sprite(define.imagePathCharacter + 'bag/bag_hover.png');
      this.field = new Framework.Sprite(define.imagePathCharacter + 'bag/field.png');
      this.moneyField = new Framework.Sprite(define.imagePathCharacter + 'bag/moneyField.png');
      this.clusterInbag = new Cluster();
      this.clusterInbag.load();
      this.medicineInBag = new Medicine();
      this.medicineInBag.load();
      this.questItem = new QuestItem();
      this.questItem.load();
   }

   this.init = function(){
      this.bagPic.scale = 0.5;
      this.bagPicHover.scale = 0.5;
      this.moneyField.scale = 0.35;

      this.isHover = false;
      this.drawField = false;
      this.isBagOpened = false;
      this.propertyArr = new Array();//Index 0 is the property type, index 1 is amount of the property
      this.capacity = 30;
      this.bagPic.position = {
        x: 1289,
        y: 630
      };
      this.bagPicHover.position = this.bagPic.position;
      this.moneyField.position = {
        x: 1055,
        y: 580
      };
```

```
}

this.update = function(){
    this.bagPic.update();
    this.bagPicHover.update();
    this.clusterInbag.update();
    this.medicineInBag.update();
    this.removeProperty();
}

this.draw = function(ctx){
    if(!this.isHover){
        this.bagPic.draw();
    }
    else{
        this.bagPicHover.draw();
    }
    if(this.drawField){
        for(var row = 0; row < 6; row++){
          for(var col = 0; col < 5; col++){
            this.field.position ={
              x: 65 * col + 1000,
              y: 65 * row + 210
            };
            this.field.draw();
          }
        }
        this.moneyField.draw();
        ctx.fillStyle = 'rgb(34, 27, 31)';
        ctx.fillText(this.player.money, this.moneyField.position.x-38, this.moneyField.position.y+10);
    }
    if(this.isBagOpened){
      for(var i = 0; i < this.propertyArr.length; i++){
        var picPosition = {
          x: 65 * (i%5) + 995,
          y: 65 * Math.floor(i/5) + 200
```

```
};
switch (this.propertyArr[i].type) {//30~ => quest item
  case 1:
    this.clusterInbag.blue_cluster.position = picPosition;
    this.clusterInbag.blue_cluster.draw();
    break;
  case 2:
    this.clusterInbag.red_cluster.position = picPosition;
    this.clusterInbag.red_cluster.draw();
    break;
  case 3:
    this.clusterInbag.green_cluster.position = picPosition;
    this.clusterInbag.green_cluster.draw();
    break;
  case 4:
    this.medicineInBag.hpPotion.position = picPosition;
    this.medicineInBag.hpPotion.draw();
    break;
  case 5:
    this.medicineInBag.mpPotoin.position = picPosition;
    this.medicineInBag.mpPotoin.draw();
    break;
  case 10:
    this.clusterInbag.tourmaline.position = picPosition;
    this.clusterInbag.tourmaline.draw();
    break;
  case 11:
    this.clusterInbag.zircon.position = picPosition;
    this.clusterInbag.zircon.draw();
    break;
  case 30:
    this.questItem.aged_scroll.position = picPosition;
    this.questItem.aged_scroll.draw();
    break;
  case 31:
    this.questItem.perfect_scroll.position = picPosition;
```

```
            this.questItem.perfect_scroll.draw();
            break;
        }
        ctx.fillStyle = 'rgb(8, 17, 3)';
        ctx.fillText(this.propertyArr[i].number, picPosition.x-25, picPosition.y+25);

      }
    }
}

this.isPropertyExist = function(propObj){
    if(this.propertyArr.length === 0){return false;}
    for(var i = 0; i < this.propertyArr.length; i++){
        if(this.propertyArr[i].type === propObj.type){
            return true;
        }
    }
    return false;
}

this.addProperty = function(propObj){
    for(var i = 0; i < this.propertyArr.length; i++){
        if(this.propertyArr[i].type === propObj.type){
            this.propertyArr[i].number += propObj.number;
        }
    }
}

this.removeProperty = function(){
    for(var i = 0; i < this.propertyArr.length; i++){
        if(this.propertyArr[i].number === 0){
            this.propertyArr.splice(i, 1);
        }
    }
}
```

```
    this.getPropertyNumber = function(propType){
        for(var i = 0; i < this.propertyArr.length; i++){
            if(this.propertyArr[i].type === propType){
              return this.propertyArr[i].number;
            }
        }
    }

    this.mousemove = function(e){
        if(e.x > 1262 && e.x < 1315 && e.y > 599 && e.y < 655){
            this.isHover = true;
        }
        else {
            this.isHover = false;
        }
    }

    this.click = function(e){
        if(e.x > 1262 && e.x < 1315 && e.y > 599 && e.y < 655){
            if(!this.drawField){
                this.drawField = true;
            }
            else{
                this.drawField = false;
            }
            if(!this.isBagOpened){
                this.isBagOpened = true;
            }
            else{
                this.isBagOpened = false;
            }
        }
    }
}
```

4.BaseMap.js：

```
var BaseMap = function(mapType){
  this.screenPosition = {//on the map
      x: 0,
      y: 0
  };
  this.mapType = mapType;
  this.arrays = new Arrays(mapType);
  this.mapArr = this.arrays.mapArray;
  this.charArr = this.arrays.charArray;
  this.charArr2 = this.arrays.charArray2;
  this.monsterArr = [];
  if(mapType === 'scroll'){
    this.propertyArr = this.arrays.propertyArray;
  }
}
BaseMap.prototype.drawOtherObj = function(obj){
    this.toScreenCoordinate(obj.position);
    obj.draw();
    this.toMapCoordinate(obj.position);
};

BaseMap.prototype.toScreenCoordinate = function(mapCoordinate) {
    mapCoordinate.x -= this.screenPosition.x;
    mapCoordinate.y -= this.screenPosition.y;
};

BaseMap.prototype.toMapCoordinate = function(screenPosition) {
    screenPosition.x += this.screenPosition.x;
    screenPosition.y += this.screenPosition.y;
};

BaseMap.prototype.toArrayIndex = function(x, y, direction){//Transform absolute coordinate to relative coordinate(2d array index)
    if(direction === 0){
        this.arrRow = Math.floor((y-15)/32);
```

```
    }
    else{
        this.arrRow = Math.floor((y+15)/32);
    }
    this.arrCol = Math.floor(x/32);
    return {row: this.arrRow, col: this.arrCol};
};

BaseMap.prototype.canMove = function(i, j, direction) {
    this.toArrayIndex(i, j, direction);
    if(i < 16 || i > 2736){ return false; }
    if(j < 16 || j > 1200){ return false; }
    if (this.mapArr[this.player.currentMap][this.arrRow][this.arrCol] > 1) {
        return false;
    }
    else {
        return true;
    }
};

BaseMap.prototype.generateProperties = function(){

};

BaseMap.prototype.clearProperties = function(){
    for(var row = 0; row < 38; row++){
      for(var col = 0; col < 86; col++){
          this.propertyArr[row][col] = 0;
      }
    }
};

BaseMap.prototype.moveScreen = function(){

};
```

```
BaseMap.prototype.checkChangeMap = function(){


};


BaseMap.prototype.generateMonsters = function(){


};
```

5.BasicAttack.js：

```
var BasicAttack = function(map){
   Skill.call(this, 1, 0.5, 0, 0, 1, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)


   this.load = function(){
     this.url = define.imagePathEffect + 'slash.png';
     this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:2, loop:true, speed:5});
   }


   this.init = function(){
     this.pic.scale = 0.35;
     this.range = {x:32, y:32};
   }


   this.draw = function(){
     if(this.isDrawed){
       this.map.drawOtherObj(this.pic);
     }
   }
}


BasicAttack.prototype = Object.create(Skill.prototype);
BasicAttack.prototype.constructor = Skill;


BasicAttack.prototype.changeDamage = function(){
   this.damage = this.player.atk;
};
```

```
BasicAttack.prototype.changeMpCost = function(){

};

BasicAttack.prototype.attack = function(i, j, direction){
    var index = this.map.toArrayIndex(i, j, direction);
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;

    this.pic.start({from:0, to:8, loop:false});
    for(var i = 0; i < monsterArr.length; i++){
      if(Math.abs(playerPos.x - monsterArr[i].pic.position.x) <= this.range.x &&
         Math.abs(playerPos.y - monsterArr[i].pic.position.y) <= this.range.y &&
         monsterArr[i].canBeAttacked){
           if(monsterArr[i].currentHP >= this.damage){
             monsterArr[i].currentHP -= this.damage;
           }
           else{
             monsterArr[i].currentHP = 0;
           }
      }
    }

    if(this.map.mapType === 'scroll'){
      if(this.map.mapArr[this.player.currentMap][index.row][index.col] === 7) {
         this.map.mapArr[this.player.currentMap][index.row][index.col] = 1;
      }
      if(this.map.mapArr[this.player.currentMap][index.row][index.col] > 1 &&
         this.map.mapArr[this.player.currentMap][index.row][index.col] < 8){
         this.map.mapArr[this.player.currentMap][index.row][index.col] = 0;
      }
    }
    else{
      if(this.map.testMapArr[index.row][index.col] === 7) {
         this.map.testMapArr[index.row][index.col] = 1;
      }
```

```
    if(this.map.testMapArr[index.row][index.col] > 1 &&
       this.map.testMapArr[index.row][index.col] < 8){
        this.map.testMapArr[index.row][index.col] = 0;
    }
  }
};
```

6.BlackBall.js：

```
var BlackBall = function(map,monster,ballNumber){
    Skill.call(this, 2, 2, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.monster = monster;
    this.load = function(){
      this.url = define.imagePathEffect + 'blackball.png';
      this.picArr = [];
      this.ballNumber = ballNumber;
      for(var i = 0; i < this.ballNumber; i++){
        this.pic = new Framework.Sprite(this.url);
        this.pic.scale = 1.5;
        this.picArr.push(this.pic);
      }
      this.counter = 0;
      this.isAttack = true;
      this.angleOffset = 0;
    }

    this.setBallPosition = function(angleOffset){
      this.angleOffset = angleOffset;
      for(var i = 0; i < this.picArr.length; i++){
        var angle = (Math.PI/12) * i + this.angleOffset;
        this.picArr[i].position = {
          x: monster.pic.position.x + Math.cos(angle) * 50,
          y: monster.pic.position.y + Math.sin(angle) * 50
        };
      }
    }
```

```
    this.pushBall = function(){
      this.pic = new Framework.Sprite(this.url);
      this.pic.scale = 1.5;
      this.picArr.push(this.pic);
    }
}

BlackBall.prototype = Object.create(Skill.prototype);
BlackBall.prototype.constructor = Skill;

BlackBall.prototype.update = function(){
   var playerPos = this.player.playerPic.position;
   for(var i = 0; i < this.picArr.length; i++){
     this.picArr[i].update();
     this.picArr[i].rotation += 15;
   }
   this.move();
   for(var i = 0; i < this.picArr.length; i++){
     if(this.canHitMonster(i)){
       if(this.player.currentHP <= this.damage){
          this.player.currentHP = 0;
       }
       else{
          this.player.currentHP -= Math.floor(this.damage * (1-this.player.def));
       }
       this.player.isAttacked = true;
     }
   }

   if(this.isAttack && this.coolDownCounter > 0){
     this.coolDownCounter--;
   }

   if(this.coolDownCounter < this.coolDownTime * 100){
     this.canAttack = false;
   }
```

```
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
    }
};

BlackBall.prototype.canHitMonster = function(i){
    var playerPos = this.player.playerPic.position;
    var dirDifference = {
      x: playerPos.x - this.picArr[i].position.x,
      y: playerPos.y - this.picArr[i].position.y
    };

    if(Math.abs(dirDifference.x) < 33 && Math.abs(dirDifference.y) < 33){
        return true;
    }
    return false;
};

BlackBall.prototype.move = function(){
    for(var i = 0; i < this.picArr.length; i++){
      var angle = (Math.PI/6) * i + this.angleOffset;
      this.picArr[i].position.x += Math.cos(angle) * 3
      this.picArr[i].position.y += Math.sin(angle) * 3
    }
};

BlackBall.prototype.attack = function(angleOffset){
    var playerPos = this.player.playerPic.position;
    var screenRow = Math.floor(this.map.screenPosition.y/32), screenCol = Math.floor(this.map.screenPosition.x/32);
    this.setBallPosition(angleOffset);
}
```

7.BlackHole.js：

```
var BlackHole = function(map){
  this.load = function(){
    this.url = define.imagePathEffect + "032.png";
    this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:2, loop: true, speed: 3});
    this.pic.position = {
      x: 2254,
      y: 636
    };
    this.blackball = new BlackBall(map,this,36);
    this.blackball.load();
    this.counter = 0;
    this.ballAngleOffset = 0;
    this.map = map;
    this.player = this.map.player;
  }

  this.draw = function(ctx){
    var picPos = this.pic.position;
    var screenPos = this.map.screenPosition;
    var playerPos = this.player.playerPic.position;

    this.map.drawOtherObj(this.pic);

    if(this.blackball.isDrawed){
      for(var i = 0; i < this.blackball.picArr.length; i++){
        this.map.drawOtherObj(this.blackball.picArr[i]);
      }
    }
  }

  this.update = function(){
    this.counter++;
    this.pic.update();
    this.blackball.update();
```

```
    if(this.counter % 30 === 0){
      this.pic.start({from:0, to:3, loop:false});
    }

    if(this.counter % 360 === 0 && this.player.currentMap === 3){
      this.usingBlackBall();
    }
  }

  this.usingBlackBall = function(){
    if(this.blackball.canAttack){
      this.ballAngleOffset += 10;
      this.blackball.isDrawed = true;
      this.blackball.isAttack = true;

      this.blackball.attack(this.ballAngleOffset);
    }
  }
}
```

8.Boomerang.js：

```
var Boomerang = function(map,monster){
    Skill.call(this, 2, 10, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.monster = monster;
    this.load = function(){
      this.url = define.imagePathEffect + 'boomerang.png';
      this.picArr = [];
      this.ballNumber = 36;
      for(var i = 0; i < this.ballNumber; i++){
        this.pic = new Framework.Sprite(this.url);
        this.pic.scale = 1.5;
        this.picArr.push(this.pic);
      }
      this.counter = 0;
      this.isAttack = true;
```

```
    }

    this.setBallNumber = function(number){
      this.ballNumber = number;
    }

    this.setBallPosition = function(){
      for(var i = 0; i < this.ballNumber; i++){
        var angle = (Math.PI/18) * i;
        this.picArr[i].position = {
          x: monster.pic.position.x + Math.cos(angle) * 50,
          y: monster.pic.position.y + Math.sin(angle) * 50
        };
      }
    }

    this.back = function(){
        for(var i = 0; i < this.ballNumber; i++){
          var angle = (Math.PI/6) * i;
          this.picArr[i].position.x -= Math.cos(angle) * 2.5
          this.picArr[i].position.y -= Math.sin(angle) * 2.5
        }
    };
}

Boomerang.prototype = Object.create(Skill.prototype);
Boomerang.prototype.constructor = Skill;

Boomerang.prototype.update = function(){
    var playerPos = this.player.playerPic.position;
    for(var i = 0; i < this.picArr.length; i++){
      this.picArr[i].update();
      this.picArr[i].rotation += 15;
    }

    if(this.coolDownCounter > 1050){
```

```
    this.move();
  }
  else if(this.coolDownCounter > 850 && this.coolDownCounter < 1030){
    this.back();
  }

  for(var i = 0; i < this.picArr.length; i++){
    if(this.canHitMonster(i)){

      if(this.player.currentHP <= this.damage){
        this.player.currentHP = 0;
      }
      else{
        this.player.currentHP -= Math.floor(this.damage * (1-this.player.def));
      }
      this.player.isAttacked = true;
    }
  }

  if(this.isAttack && this.coolDownCounter > 0){
    this.coolDownCounter--;
  }

  if(this.coolDownCounter < this.coolDownTime * 120){
    this.canAttack = false;
  }
  if(this.coolDownCounter === 0){
    this.isAttack = false;
    this.canAttack = true;
    this.coolDownCounter = this.coolDownTime * 120;
  }


};

Boomerang.prototype.canHitMonster = function(i){
```

```
    var playerPos = this.player.playerPic.position;
    var dirDifference = {
      x: playerPos.x - this.picArr[i].position.x,
      y: playerPos.y - this.picArr[i].position.y
    };



    if(Math.abs(dirDifference.x) < 33 && Math.abs(dirDifference.y) < 33){
        return true;
    }
    return false;
};

Boomerang.prototype.move = function(){
    for(var i = 0; i < this.ballNumber; i++){
      var angle = (Math.PI/6) * i;
      this.picArr[i].position.x += Math.cos(angle) * 3
      this.picArr[i].position.y += Math.sin(angle) * 3
    }
};

Boomerang.prototype.attack = function(){
    var playerPos = this.player.playerPic.position;
    var screenRow = Math.floor(this.map.screenPosition.y/32), screenCol = Math.floor(this.map.screenPosition.x/32);
    this.setBallPosition();
}
```

9.Boss.js：

```
var Boss = function(map,player){
   Monster.call(this, 500000, 1000, 0, player, map);//Call the parent's constructor(hp,atk,exp,player)
   this.load = function(){
     this.url1 = define.imagePathMonster + 'inquisitor.png';
     this.url2 = define.imagePathEffect + 'ice_001.png';
     this.url3 = define.imagePathEffect + '005.png';
     this.pic = new Framework.AnimationSprite({url:this.url1, col:3, row:4, loop: true, speed: 1});
     this.icePic = new Framework.AnimationSprite({url:this.url2, col:5, row:6, loop: true, speed: 30});
     this.guardPic = new Framework.AnimationSprite({url:this.url3, col:5, row:4, loop: true, speed: 1});
     this.guardPic.scale = 1.8;
     this.pic.scale = 1.2;
     this.frostbolt = new Frostbolt(map,this);
     this.frostbolt.load();
     this.blackball = new BlackBall(map,this,12);
     this.blackball.load();
     this.boomerang = new Boomerang(map,this);
     this.boomerang.load();
     this.thunder = new Thunder(map,this);
     this.thunder.load();
     this.counter = 0;
     this.alpha = 0;
     this.defenderArr = [];
     this.deadDevilCounter = 2;
     this.ballAngleOffset = 0;
     this.isUsingSkill = false;
     this.canBeAttacked = false;
     this.name = 'boss';
   }
   this.init = function(){
     this.guardPic.position = {
       x: this.pic.position.x,
       y: this.pic.position.y
     }
   }
```

```
this.initDefenderArr = function(){
  for(var i = 0; i < this.map.monsterArr.length; i++){
    if(this.map.monsterArr[i].name === 'defender' && this.defenderArr.length < 2){
      this.defenderArr.push(this.map.monsterArr[i]);
    }
  }
}

this.draw = function(ctx){
  var picPos = this.pic.position;
  var screenPos = this.map.screenPosition;
  var playerPos = this.player.playerPic.position;
  var monsterArr = this.map.monsterArr;

  ctx.fillStyle = 'rgba(118, 116, 122, 0.9)';
  if(!this.isDead){
      ctx.fillRect(450, 10, 500, 20);
  }
  ctx.fillStyle = 'rgba(65, 9, 34, 0.93)';
  ctx.fillRect(450, 10, 500*(this.currentHP/this.hp), 20);
  this.printDamageReceived(ctx);
  for(var i = 0; i < this.frostbolt.picArr.length; i++){
    this.map.drawOtherObj(this.frostbolt.picArr[i]);
  }
  if(this.blackball.isDrawed){
    for(var i = 0; i < this.blackball.picArr.length; i++){
      this.map.drawOtherObj(this.blackball.picArr[i])
    }
  }

  if(this.boomerang.isDrawed && this.boomerang.coolDownCounter > 850 && !this.isDead)
  for(var i = 0; i < this.boomerang.picArr.length; i++){
    this.map.drawOtherObj(this.boomerang.picArr[i]);
  }
  this.frostbolt.effectPic.draw();
```

```
  this.icePic.draw();
  this.thunder.draw();
  for(var i = 0; i < monsterArr.length; i++){
    if(monsterArr[i].name === 'devil'){
      if(!monsterArr[i].isDead){
        this.map.drawOtherObj(this.guardPic);
      }
      else{
        if(monsterArr[i].reviveTimeCounter > 0 && monsterArr[i].reviveTimeCounter < 2){
          this.deadDevilCounter--;
        }
      }
    }
  }
  if(this.isDead){
    this.map.isWin = true;
  }
}

this.update = function(){
  this.counter++;
  this.frostbolt.counter++;
  if(this.counter % 100 === 0){
    this.pic.start({from:0, to:2, loop:false});
    this.guardPic.start({from:0, to:0, loop:false});
  }

  if(this.frostbolt.counter % 100 === 0){
    this.isUsingSkill = false;
  }

  if(this.deadDevilCounter == 0){
    this.canBeAttacked = true;
  }
  else{
    this.canBeAttacked = false;
```

```
    }

    this.checkMonsterDead();

    this.pic.update();
    this.frostbolt.update();
    this.blackball.update();
    this.boomerang.update();
    this.thunder.update();
    this.thunder.effectPic.update();
    this.icePic.update();
    this.guardPic.update();

    if(this.currentHP <= this.hp / 2){
        this.map.isHeavyRain = true;
        this.usingThunder();
    }

    if(this.player.currentHP > 0 && !this.isDead){
      if(this.player.isAttacked === false){
        this.touchPlayer();
      }
      this.usingBlackBall();
      if(!this.isUsingSkill){
        this.usingFrostBolt();
        this.usingBoomerang();
      }
    }

    if(this.player.resistCounter === 0){
      this.player.isAttacked = false;
      this.isTouchingPlayer = false;
      this.player.resistCounter = this.player.resistTime;
    }
  }
```

```
this.usingThunder = function(){
  if(this.thunder.canAttack && !this.isDead){
    this.thunder.isDrawed = true;
    this.thunder.isAttack = true;
    this.thunder.attack();
  }
}

this.usingBoomerang = function(){
  if(this.boomerang.canAttack && !this.isDead){
    this.boomerang.isDrawed = true;
    this.boomerang.isAttack = true;
    this.boomerang.attack();
  }
}

this.usingBlackBall = function(){
  if(this.blackball.canAttack && !this.isDead){
    this.ballAngleOffset += 10;
    this.blackball.isDrawed = true;
    this.blackball.isAttack = true;
    this.initDefenderArr();
    for(var i = 0; i < this.defenderArr.length; i++){
      if(!this.defenderArr[i].isDead && this.blackball.ballNumber <= 36 && this.counter % 700 === 0){
        this.blackball.pushBall();
      }
    }
    if(this.defenderArr.length === 0){
        this.blackball.picArr.splice(11, this.blackball.picArr.length-12);
    }
    this.blackball.attack(this.ballAngleOffset);
  }
}

this.usingFrostBolt = function(){
  if(this.frostbolt.canAttack){
```

```
        this.isUsingSkill = true;
        this.frostbolt.isAttack = true;
        this.icePic.position = {
          x: this.pic.position.x,
          y: this.pic.position.y
        }

        this.icePic.start({from:0, to:29, loop:false});
        this.frostbolt.attack();
      }
    }

    this.setPosition = function(pos){
      this.pic.position = {
        x: pos.x * 32 + 16,
        y: pos.y * 32 + 16
      };
    }
}
Boss.prototype = Object.create(Monster.prototype);
Boss.prototype.constructor = Monster;

Boss.prototype.touchPlayer = function(){
    var picPos = this.pic.position;
    var playerPos = this.player.playerPic.position;
    if(Math.abs(picPos.x - playerPos.x) < 90 && Math.abs(picPos.y - playerPos.y) < 80 && !this.isDead){
        if(this.player.currentHP <= this.currentAtk){
            this.player.currentHP = 0;
        }
        else{
            this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
        }
        this.player.isAttacked = true;
        this.isTouchingPlayer = true;
    }
};
```

10.BurstCut.js：

```
var BurstCut = function(map){
    Skill.call(this, 2, 0.8, 0, 6, 3, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)

    this.load = function(){
      this.url = define.imagePathEffect + 'Warrior/002.png';
      this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:2, loop:true, speed:5});
    }

    this.init = function(){
      this.abilityList = [0, 1.6, 1.7, 1.8, 1.9, 2.0];
      this.mpCostList = [0, 6, 7, 8, 9, 10];
      this.pic.scale = 0.8;
    }

    this.draw = function(){
      if(this.isDrawed){
        this.map.drawOtherObj(this.pic);
      }
    }
}

BurstCut.prototype = Object.create(Skill.prototype);
BurstCut.prototype.constructor = Skill;

BurstCut.prototype.levelUp = function(){
   for(var level = 10; level <= 14; level++){
     if(this.player.level === level){
       this.level = level - 9;
     }
   }
   if(this.player.level > 14 || this.map.mapType !== 'scroll'){//>19
     this.level = 5;
   }
};
```

```
BurstCut.prototype.removeObstacle = function(index,direction){
    var mapArr = this.map.mapArr;
    var currentMap = this.player.currentMap;

    if(direction === 0){//up
      for(var row = index.row - 1; row >= index.row - 3; row--){
        for(var col = index.col - 1; col <= index.col + 1; col++){
          if(this.map.mapType === 'scroll'){
            if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 && row >= 0 && col >= 0){
              this.map.mapArr[this.player.currentMap][row][col] = 0;
            }
          }
          else{
            if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
              this.map.testMapArr[row][col] = 0;
            }
          }
        }
      }
    }
    else if(direction === 1){//left
      for(var row = index.row - 1; row <= index.row + 1; row++){
        for(var col = index.col - 3; col <= index.col - 1; col++){
          if(this.map.mapType === 'scroll'){
            if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 && row >= 0 && col >= 0){
              this.map.mapArr[this.player.currentMap][row][col] = 0;
            }
          }
          else{
            if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
              this.map.testMapArr[row][col] = 0;
            }
          }
        }
```

```
    }
  }
  else if(direction === 2){
    for(var row = index.row + 1; row <= index.row + 3; row++){
      for(var col = index.col - 1; col <= index.col + 1; col++){
        if(this.map.mapType === 'scroll'){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 && row
>= 0 && col >= 0){
            this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
        }
        else{
          if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
            this.map.testMapArr[row][col] = 0;
          }
        }
      }
    }
  }
  else if(direction === 3){
    for(var row = index.row + 1; row >= index.row - 1; row--){
      for(var col = index.col + 1; col <= index.col + 3; col++){
        if(this.map.mapType === 'scroll'){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 && row
>= 0 && col >= 0){
            this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
        }
        else{
          if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
            this.map.testMapArr[row][col] = 0;
          }
        }
      }
    }
  }
};
```

```
BurstCut.prototype.canHitMonster = function(direction,i){
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    var dirDifference = {
      x: playerPos.x - monsterArr[i].pic.position.x,
      y: playerPos.y - monsterArr[i].pic.position.y
    };

    switch (direction) {
      case 0:
        if(Math.abs(dirDifference.x) <= 96 && dirDifference.y <= 96 && dirDifference.y >= 0){
            return true;
        }
        return false;
      case 1:
        if(dirDifference.x <= 96 && dirDifference.x >= 0 && Math.abs(dirDifference.y) <= 96){
            return true;
          }
        return false;
      case 2:
        if(Math.abs(dirDifference.x) <= 96 && dirDifference.y >= -96 && dirDifference.y <= 0){
            return true;
          }
        return false;
      case 3:
        if(dirDifference.x >= -96 && dirDifference.x <= 0 && Math.abs(dirDifference.y) <= 96){
            return true;
          }
        return false;
    }
};

BurstCut.prototype.attack = function(i, j, direction){
   var index = this.map.toArrayIndex(i, j, direction);
   var monsterArr = this.map.monsterArr;
```

```
    this.pic.start({from:0, to:7, loop:false});
    for(var i = 0; i < monsterArr.length; i++){//Determine if player hits the monster
      if(this.canHitMonster(direction,i) && monsterArr[i].canBeAttacked){
          if(monsterArr[i].currentHP >= this.damage){
            monsterArr[i].currentHP -= this.damage;
          }
          else{
            monsterArr[i].currentHP = 0;
          }
      }
    }
    this.removeObstacle(index,direction);
};
```

11.Calavera.js：

```
var Calavera = function(map,player){
    Monster.call(this, 10000000, 100000, 0, player, map);//Call the parent's constructor(hp,atk,exp,player)this.direction = direction;
    this.load = function(){
      this.url = define.imagePathMonster + "calavera.png";
      this.pic = new Framework.AnimationSprite({url:this.url, col:4, row:4, loop: true, speed: 1});
      this.fireBall = new FireBall(map, this);
      this.fireBall.load();
      this.direction = 0;
      this.counter = 0;
      this.name = 'calavera';
    }

    this.draw = function(ctx){
      var picPos = this.pic.position;
      var screenPos = this.map.screenPosition;
      var playerPos = this.player.playerPic.position;

      ctx.fillStyle = 'rgb(176, 60, 87)';
      if(this.map.mapType === 'scroll'){
        ctx.fillRect(picPos.x-screenPos.x-30, picPos.y-screenPos.y-30, 60*(this.currentHP/this.hp), 10);
```

```
  }
  else{
    ctx.fillRect(picPos.x-30, picPos.y-30, 60*(this.currentHP/this.hp), 10);
  }
  if(this.isTouchingPlayer){
    this.printDamageReceived(ctx);
  }
  this.map.drawOtherObj(this.fireBall.pic);
}

this.update = function(){
  this.counter++;
  this.checkMonsterDead();
  this.revive();
  this.pic.update();
  this.fireBall.update();

  if(this.counter % 120 === 0 && !this.isDead){
    if(this.direction === 0){//up
      this.pic.start({from:12, to:15, loop:false});
    }
    else if(this.direction === 2){
      this.pic.start({from:0, to:3, loop:false});
    }
    this.usingFireBall();
  }
  if(this.player.currentHP > 0){
    this.touchPlayer();
  }
  if(this.player.resistCounter === 0){
    this.player.isAttacked = false;
    this.isTouchingPlayer = false;
    this.player.resistCounter = this.player.resistTime;
  }
}
```

```
    this.usingFireBall = function(){
      if(this.fireBall.canAttack){
        this.fireBall.isAttack = true;
        this.fireBall.pic.position = {
          x: this.pic.position.x,
          y: this.pic.position.y
        };
        this.fireBall.attack(this.direction);
      }
    }

    this.setPositionAndDirection = function(pos,dir){
      this.direction = dir;
      this.pic.position = {
        x: pos.x * 32 + 16,
        y: pos.y * 32 + 16
      };
    }
}
Calavera.prototype = Object.create(Monster.prototype);
Calavera.prototype.constructor = Monster;

Calavera.prototype.touchPlayer = function(){
    var picPos = this.pic.position;
    var playerPos = this.player.playerPic.position;
    if(Math.abs(picPos.x - playerPos.x) < 33 && Math.abs(picPos.y - playerPos.y) < 33 && !this.isDead){
        if(this.player.currentHP <= this.currentAtk){
            this.player.currentHP = 0;
        }
        else{
            this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
        }
        this.player.isAttacked = true;
        this.isTouchingPlayer = true;
    }
};
```

12.Cluster.js：

```
var Cluster = function(map){
    Property.call(this, map);
    this.load = function(){
      this.url1 = define.imagePathItem + 'crafting_material/crystal/cluster/Deep_blue_cluster.png';
      this.url2 = define.imagePathItem + 'crafting_material/crystal/cluster/Deep_red_cluster.png';
      this.url3 = define.imagePathItem + 'crafting_material/crystal/cluster/Deep_green_cluster.png';
      this.url4 = define.imagePathItem + 'crafting_material/crystal/quartz/Tourmaline.png';
      this.url5 = define.imagePathItem + 'crafting_material/crystal/quartz/Zircon.png';
      this.blue_cluster = new Framework.Sprite(this.url1);
      this.red_cluster = new Framework.Sprite(this.url2);
      this.green_cluster = new Framework.Sprite(this.url3);
      this.tourmaline = new Framework.Sprite(this.url4);
      this.zircon = new Framework.Sprite(this.url5);
    }

    this.update = function(){
        this.blue_cluster.update();
        this.red_cluster.update();
        this.green_cluster.update();
    }
}
Cluster.prototype = Object.create(Property.prototype);
Cluster.prototype.constructor = Property;
```

13.CrossCut.js：

```
var CrossCut = function(map){
    Skill.call(this, 2, 1.5, 50, 0, 3, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
      this.url = define.imagePathEffect + '004.png';
      this.picArr = [];
      for(var i = 0; i < 4; i++){
        this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:5, loop:true, speed:10});
        this.picArr.push(this.pic);
      }
    }

    this.init = function(){
      this.abilityList = [0, 1.7, 1.9, 2.1, 2.3, 2.5];
      this.hpCostList = [0, 30, 35, 40, 45, 50];
      this.picArr[0].rotation = 90;
      this.picArr[1].rotation = 180;
      this.picArr[2].rotation = 270;
    }



    this.draw = function(){
      var playerPos = this.player.playerPic.position;
      if(this.isDrawed){
        this.picArr[0].position = {
          x: playerPos.x,
          y: playerPos.y - 64
        };

        this.picArr[1].position = {
          x: playerPos.x - 64,
          y: playerPos.y
        };

        this.picArr[2].position = {
```

```
            x: playerPos.x,
            y: playerPos.y + 64
          };

          this.picArr[3].position = {
            x: playerPos.x + 64,
            y: playerPos.y
          };

          for(var i = 0 ; i < 4; i++){
            this.map.drawOtherObj(this.picArr[i]);
          }
        }
      }
}

CrossCut.prototype = Object.create(Skill.prototype);
CrossCut.prototype.constructor = Skill;

CrossCut.prototype.levelUp = function(){
    this.level = 5;
};

CrossCut.prototype.update = function(){
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    for(var i = 0; i < 4; i++){
      this.picArr[i].update();
    }
    this.changeDamage();
    this.changeMpCost();
    this.levelUp();

    if(this.isAttack && this.coolDownCounter > 0){
      this.coolDownCounter--;
    }
```

```
    if(this.coolDownCounter < this.coolDownTime * 100){
      this.canAttack = false;
    }
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
    }
};


CrossCut.prototype.removeObstacle = function(index){
     for(var row = index.row - 1; row >= index.row - 3; row--){
       for(var col = index.col - 1; col <= index.col + 1; col++){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
             this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
       }
     }


     for(var row = index.row - 1; row <= index.row + 1; row++){
       for(var col = index.col - 3; col <= index.col - 1; col++){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
             this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
       }
     }


     for(var row = index.row + 1; row <= index.row + 3; row++){
       for(var col = index.col - 1; col <= index.col + 1; col++){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8  &&
row >= 0 && col >= 0){
             this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
       }
     }
```

```
    for(var row = index.row + 1; row >= index.row - 1; row--){
      for(var col = index.col + 1; col <= index.col + 3; col++){
        if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
          this.map.mapArr[this.player.currentMap][row][col] = 0;
        }
      }
    }
}


CrossCut.prototype.attack = function(i, j, direction){
  var index = this.map.toArrayIndex(i, j, direction);
  var playerPos = this.player.playerPic.position;
  var monsterArr = this.map.monsterArr;


  for(var i = 0; i < 4; i++){
    this.picArr[i].start({from:0, to:24, loop:false});
  }
  //Determine if player hits the monster
  for(var i = 0; i < monsterArr.length; i++){
    if(Math.abs(playerPos.x - monsterArr[i].pic.position.x) <= 120 &&
       Math.abs(playerPos.y - monsterArr[i].pic.position.y) <= 120){
      if(monsterArr[i].currentHP >= this.damage){
        monsterArr[i].currentHP -= this.damage;
      }
      else{
        monsterArr[i].currentHP = 0;
      }
    }
  }
  this.removeObstacle(index);
}
```

14.DarkForce.js：

```
var DarkForce = function(map){
  Skill.call(this, 0, 1, 0, 100, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
```

```
this.load = function(){
    this.url = define.imagePathEffect + '013.png';
    this.url2 = define.imagePathEffect + 'buff1.png';
    this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:4, loop:true, speed:5});
    this.effectPic = new Framework.AnimationSprite({url:this.url2, col:5, row:6, loop:true, speed:12});
}

this.init = function(){
    this.pic.scale = 0.7;
    this.durationTime = 2000;
    this.timeCounter = 0;
    this.isUsingSkill = false;
}

this.draw = function(){
  if(this.isDrawed && this.timeCounter <= this.durationTime && this.timeCounter !== 0){
    this.isUsingSkill = true;
    this.pic.position = {
      x: this.player.playerPic.position.x,
      y: this.player.playerPic.position.y + 20
    };
    this.effectPic.position = {
      x: this.player.playerPic.position.x,
      y: this.player.playerPic.position.y
    };
    if(this.map.mapType === 'scroll'){
      this.map.drawOtherObj(this.pic);
      this.map.drawOtherObj(this.effectPic);
    }
    else{
      this.pic.draw();
      this.effectPic.draw();
    }
  }

  else{
```

```
      this.isUsingSkill = false;
    }
  }

  this.decreaseCoolDownTime = function(skill){
    skill.coolDownTime /= 2;
    skill.coolDownCounter = skill.coolDownTime * 100;
  }

  this.restoreCoolDownTime = function(skill){
    skill.coolDownTime *= 2;
    skill.coolDownCounter = skill.coolDownTime * 100;
  }
}

DarkForce.prototype = Object.create(Skill.prototype);
DarkForce.prototype.constructor = Skill;

DarkForce.prototype.update = function(){
  var playerPos = this.player.playerPic.position;
  this.pic.update();
  this.effectPic.update();

  if(this.isAttack){
    this.timeCounter++;
  }

  if(this.timeCounter % 30 === 0 && this.timeCounter > 0){
    this.pic.start({from:0, to:7, loop:false});
  }

  this.canAttack = true;
  if(this.timeCounter === this.durationTime){
    this.isAttack = false;
    this.restoreCoolDownTime(this.player.basicAttack);
    this.restoreCoolDownTime(this.player.vCut);
```

```
    this.restoreCoolDownTime(this.player.burstCut);
    this.restoreCoolDownTime(this.player.swordLight);
    this.restoreCoolDownTime(this.player.crossCut);
    this.timeCounter = 0;
  }
};

DarkForce.prototype.buff = function(){
  if(!this.isAttack){
    this.decreaseCoolDownTime(this.player.basicAttack);
    this.decreaseCoolDownTime(this.player.vCut);
    this.decreaseCoolDownTime(this.player.burstCut);
    this.decreaseCoolDownTime(this.player.swordLight);
    this.decreaseCoolDownTime(this.player.crossCut);
  }
  this.effectPic.start({from:0, to:29, loop:false});
};
```

15.Defender.js：

```
var Defender = function(map, player){
  Monster.call(this, 200000, 100, 131072, player, map);//Call the parent's constructor(hp,atk,exp,player)
  this.load = function(){
    this.url = define.imagePathMonster + "boss4.png";
    this.pic = new Framework.AnimationSprite({url:this.url, col:3, row:4, loop: true, speed: 1});
    this.pic.scale = 0.8;
    this.counter = 0;
    this.name = 'defender';
  }

  this.draw = function(ctx){
    var picPos = this.pic.position;
    var screenPos = this.map.screenPosition;
    var playerPos = this.player.playerPic.position;

    if(this.map.mapType === 'scroll'){
      ctx.fillStyle = 'rgba(118, 116, 122, 0.69)';
```

```
      if(!this.isDead){
        ctx.fillRect(picPos.x-screenPos.x-90, picPos.y-screenPos.y-50, 180, 10);
      }
      ctx.fillStyle = 'rgb(35, 96, 10)';
      ctx.fillRect(picPos.x-screenPos.x-90, picPos.y-screenPos.y-50, 180*(this.currentHP/this.hp), 10);
    }
    else{
      ctx.fillStyle = 'rgba(118, 116, 122, 0.69)';
      if(!this.isDead){
        ctx.fillRect(picPos.x-90, picPos.y-50, 180, 10);
      }
      ctx.fillStyle = 'rgb(35, 96, 10)';
      ctx.fillRect(picPos.x-90, picPos.y-50, 180*(this.currentHP/this.hp), 10);
    }
  }

  this.update = function(){
    this.pic.update();
    this.checkMonsterDead();
    this.counter++;
    if(this.counter % 120 === 0){
      this.pic.start({from:0, to:2, loop:false});
    }
  }

  this.setPosition = function(pos){
    this.pic.position = {
      x: pos.x * 32 + 16,
      y: pos.y * 32 + 16
    };
  }
}
Defender.prototype = Object.create(Monster.prototype);
Defender.prototype.constructor = Monster;
```

16.Define.js：

```
(function(window){
    var define ={}, mainPath = '';
    Object.defineProperties(define, {
                'mainPath': {
                  value: mainPath,
                        writable: false
                },
                'jsPath': {
                        value: mainPath + 'js/',
                        writable: false
                },
                'musicPath': {
                        value: mainPath + 'music/',
                        writable: false
                },
                'imagePath': {
                        value: mainPath + 'image/',
                        writable: false
                },
                'imagePathStartPage': {
                        value: mainPath + 'image/StartPage/',
                        writable: false
                },
                'imagePathEffect': {
                        value: mainPath + 'image/Effect/',
                        writable: false
                },
                'imagePathMonster': {
                        value: mainPath + 'image/Monster/',
                        writable: false
                },
                'imagePathMap': {
                        value: mainPath + 'image/MapPicture/',
                        writable: false
                },
                'imagePathCanvas': {
```

```
                value: mainPath + 'image/canvas/',
                writable: false
            },
            'imagePathCharacter': {
                value: mainPath + 'image/Character/',
                writable: false
            },
            'imagePathByTeacher': {
                value: mainPath + 'image/byTeacher/',
                writable: false
            },
            'imagePathItem': {
                value: mainPath + 'image/item/',
                writable: false
            }
    });
    window.define = define;
})(window)
```

17.Devil.js：

```
var Devil = function(map,player){
    Monster.call(this, 170000, 300, 7, player, map);//Call the parent's constructor(hp,atk,exp,player)
    this.load = function(){
      this.url1 = define.imagePathMonster + "boss.png";
      this.url2 = define.imagePathEffect + '010.png';
      this.pic = new Framework.AnimationSprite({url:this.url1, col:3, row:4, loop: true, speed: 1});
      this.strikePic = new Framework.AnimationSprite({url:this.url2, col:5, row:3, loop:true, speed:12});
      this.strikePic.scale = 0.5;
      this.strikePic.rotation = 180;
      this.pic.scale = 1.5;
      this.fireStrike = new FireStrike(map,this);
      this.fireStrike.load();
      this.walkDir = {x:0, y:0};
      this.direction = 0;
      this.walkCounter = 0;
      this.fireStrikeCounter = 0;
```

```
  this.isUsingSkill = false;
  this.name = 'devil';
  for(var i = 0; i < this.map.monsterArr.length; i++){
    if(this.map.monsterArr[i].name === 'boss'){
      this.boss = this.map.monsterArr[i];
    }
  }
}

this.draw = function(ctx){
  var picPos = this.pic.position;
  var screenPos = this.map.screenPosition;
  var playerPos = this.player.playerPic.position;

  if(this.map.mapType === 'scroll'){
    ctx.fillStyle = 'rgba(118, 116, 122, 0.69)';
    if(!this.isDead){
        ctx.fillRect(picPos.x-screenPos.x-90, picPos.y-screenPos.y-50, 180, 10);
    }
    ctx.fillStyle = 'rgb(9, 4, 23)';
    ctx.fillRect(picPos.x-screenPos.x-90, picPos.y-screenPos.y-50, 180*(this.currentHP/this.hp), 10);
  }
  else{
    ctx.fillStyle = 'rgba(118, 116, 122, 0.69)';
    if(!this.isDead){
        ctx.fillRect(picPos.x-90, picPos.y-50, 180, 10);
    }
    ctx.fillStyle = 'rgb(9, 4, 23)';
    ctx.fillRect(picPos.x-90, picPos.y-50, 180*(this.currentHP/this.hp), 10);
  }
  this.map.drawOtherObj(this.fireStrike.pic);
  this.strikePic.draw();
}

this.update = function(){
  if(this.walkCounter > 120){
```

```
  var randomDir = Math.floor(Math.random() * 4);
  this.walkCounter = 0;
}

if(this.closeToPlayer()){
  this.tracePlayer();
}
else{
  this.move(randomDir);
}

if(!this.boss.isDead){
  this.revive();
}
this.walkCounter++;

this.checkMonsterDead();
this.pic.update();
this.fireStrike.update();
this.strikePic.update();
this.fireStrikeCounter++;

if(this.fireStrikeCounter % 100 === 0){
  this.isUsingSkill = false;
}

if(this.player.currentHP > 0 && !this.isDead){
  if(this.player.isAttacked === false){
    this.touchPlayer();
  }

  if(this.closeToPlayer() && !this.isUsingSkill){
    this.usingFireStrike();
  }
}
```

```
  if(this.player.resistCounter === 0){
    this.player.isAttacked = false;
    this.isTouchingPlayer = false;
    this.player.resistCounter = this.player.resistTime;
  }
}

this.closeToPlayer = function(){
  var playerPos = this.player.playerPic.position;
  if(Math.abs(this.pic.position.x - playerPos.x) < 180 && Math.abs(this.pic.position.y - playerPos.y) < 180){
      return true;
  }
  return false;
}

this.usingFireStrike = function(){
  if(this.fireStrike.canAttack){
    this.isUsingSkill = true;
    this.fireStrike.isAttack = true;
    this.fireStrike.pic.position = {
      x: this.player.playerPic.position.x,
      y: this.player.playerPic.position.y
    };
    this.strikePic.position = {
      x: this.pic.position.x,
      y: this.pic.position.y
    }
    this.fireStrike.pic.start({from:0, to:11, loop:false});
    this.strikePic.start({from:0, to:11, loop:false});
    this.fireStrike.attack(this.direction);
  }
}

this.setPosition = function(pos){
  this.pic.position = {
    x: pos.x * 32 + 16,
```

```
    y: pos.y * 32 + 16
  };
}

this.tracePlayer = function(){
    var playerPos = this.player.playerPic.position;
    if(this.pic.position.x < playerPos.x && this.pic.position.y < playerPos.y){//Left top
        this.walkDir = {x: 0, y: 0.3};
        this.traceMove(2);
        this.walkDir = {x: 0.3, y: 0};
        this.traceMove(3);
        if(this.walkCounter < 3){
          if(Math.abs(this.pic.position.x - playerPos.x) <= 96){
            this.pic.start({from:0, to:2, loop:true});
          }
          else{
            this.pic.start({from:6, to:8, loop:true});
          }
        }
    }

    else if(this.pic.position.x > playerPos.x && this.pic.position.y < playerPos.y){//Right top
        this.walkDir = {x: 0, y: 0.3};
        this.traceMove(2);
        this.walkDir = {x: -0.3, y: 0};
        this.traceMove(1);
        if(this.walkCounter < 3){
          if(Math.abs(this.pic.position.x - playerPos.x) <= 96){
            this.pic.start({from:0, to:2, loop:true});
          }
          else{
            this.pic.start({from:3, to:5, loop:true});
          }
        }
    }
```

```
  else if(this.pic.position.x < playerPos.x && this.pic.position.y > playerPos.y){//Left down
      this.walkDir = {x: 0, y: -0.3};
      this.traceMove(0);
      this.walkDir = {x: 0.3, y: 0};
      this.traceMove(3);
      if(this.walkCounter < 3){
        if(Math.abs(this.pic.position.x - playerPos.x) <= 96){
          this.pic.start({from:9, to:11, loop:true});
        }
        else{
          this.pic.start({from:6, to:8, loop:true});
        }
      }
  }

  else if(this.pic.position.x > playerPos.x && this.pic.position.y > playerPos.y){//Right down
      this.walkDir = {x: 0, y: -0.3};
      this.traceMove(0);
      this.walkDir = {x: -0.3, y: 0};
      this.traceMove(1);
      if(this.walkCounter < 3){
        if(Math.abs(this.pic.position.x - playerPos.x) <= 96){
          this.pic.start({from:9, to:11, loop:true});
        }
        else{
          this.pic.start({from:3, to:5, loop:true});
        }
      }
  }
}
this.traceMove = function(dir){
  this.direction = dir;
  if(this.map.canMove(this.pic.position.x + this.walkDir.x, this.pic.position.y + this.walkDir.y, dir)){
      this.pic.position.x += this.walkDir.x;
      this.pic.position.y += this.walkDir.y;
  }
```

```
    }
}

Devil.prototype = Object.create(Monster.prototype);
Devil.prototype.constructor = Monster;

Devil.prototype.move = function(randomDir){
  this.direction = randomDir;
  switch (randomDir) {
    case 0://up
      this.walkDir = {x: 0, y: -0.3};
      this.pic.start({from:9, to:11, loop:true});
      break;
    case 1://left
      this.walkDir = {x: -0.3, y: 0};
      this.pic.start({from:3, to:5, loop:true});
      break;
    case 2://down
      this.walkDir = {x: 0, y: 0.3};
      this.pic.start({from:0, to:2, loop:true});
      break;
    case 3://right
      this.walkDir = {x: 0.3, y: 0};
      this.pic.start({from:6, to:8, loop:true});
      break;
  }
  if(this.map.canMove(this.pic.position.x + this.walkDir.x, this.pic.position.y + this.walkDir.y, randomDir)){
      this.pic.position.x += this.walkDir.x;
      this.pic.position.y += this.walkDir.y;
  }
};

Devil.prototype.revive = function(){
    if(this.isDead){
        this.reviveTimeCounter++;
    }
```

```
    if(this.reviveTimeCounter === 7201){
        this.boss.deadDevilCounter++;
    }

    if(this.reviveTimeCounter > 7200){
        this.isDead = false;
        this.currentHP = this.hp;
        this.currentAtk = this.atk;
        this.canGainExp = true;
        this.reviveTimeCounter = 0;
    }
};

Devil.prototype.touchPlayer = function(){
    var picPos = this.pic.position;
    var playerPos = this.player.playerPic.position;
    if(Math.abs(picPos.x - playerPos.x) < 50 && Math.abs(picPos.y - playerPos.y) < 50 && !this.isDead){
        if(this.player.currentHP <= this.currentAtk){
            this.player.currentHP = 0;
        }
        else{
            this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
        }
        this.player.isAttacked = true;
        this.isTouchingPlayer = true;
    }
};

Devil.prototype.attack = function(){

};
```

```
var Dragonfly = function(map,player){
  Monster.call(this, 5, 10, 1, player, map);//Call the parent's constructor(hp,atk,exp,player)
  this.load = function(){
    this.url = define.imagePathMonster + "dragonfly.png";
    this.pic = new Framework.AnimationSprite({url:this.url, col:7, row:4, loop: true, speed: 7});
    this.walkDir = {x:0, y:0};
    this.counter = 0;
    this.name = 'dragonfly';
  }

  this.draw = function(ctx){
    var picPos = this.pic.position;
    var screenPos = this.map.screenPosition;
    var playerPos = this.player.playerPic.position;

    ctx.fillStyle = 'rgb(176, 60, 87)';
    if(this.map.mapType === 'scroll'){
      ctx.fillRect(picPos.x-screenPos.x-30, picPos.y-screenPos.y-30, 60*(this.currentHP/this.hp), 10);
    }
    else{
      ctx.fillRect(picPos.x-30, picPos.y-30, 60*(this.currentHP/this.hp), 10);
    }
    if(this.isTouchingPlayer){
      this.printDamageReceived(ctx);
    }
  }

  this.update = function(){
    if(this.counter > 100){
      var randomDir = Math.floor(Math.random() * 4);
      this.counter = 0;
    }
    this.counter++;
    this.move(randomDir);
    this.checkMonsterDead();
    this.revive();
```

```
    this.pic.update();

    if(this.player.currentHP > 0 && this.player.isAttacked === false){
      this.touchPlayer();
    }

    if(this.player.resistCounter === 0){
      this.player.isAttacked = false;
      this.isTouchingPlayer = false;
      this.player.resistCounter = this.player.resistTime;
    }
  }

  this.setPosition = function(pos){
    this.pic.position = {
      x: pos.x * 32 + 16,
      y: pos.y * 32 + 16
    };
  }
}
Dragonfly.prototype = Object.create(Monster.prototype);
Dragonfly.prototype.constructor = Monster;

Dragonfly.prototype.move = function(randomDir){
 switch (randomDir) {
   case 0://up
     this.walkDir = {x: 0, y: -1};
     this.pic.start({from:0, to:6, loop:true});
     break;
   case 1://left
     this.walkDir = {x: -1, y: 0};
     this.pic.start({from:21, to:27, loop:true});
     break;
   case 2://down
     this.walkDir = {x: 0, y: 1};
     this.pic.start({from:14, to:20, loop:true});
```

```
      break;
    case 3://right
      this.walkDir = {x: 1, y: 0};
      this.pic.start({from:7, to:13, loop:true});
      break;
  }
  if(this.map.canMove(this.pic.position.x + this.walkDir.x, this.pic.position.y + this.walkDir.y, randomDir)){
      this.pic.position.x += this.walkDir.x;
      this.pic.position.y += this.walkDir.y;
  }
};

Dragonfly.prototype.touchPlayer = function(){
    var picPos = this.pic.position;
    var playerPos = this.player.playerPic.position;
    if(Math.abs(picPos.x - playerPos.x) < 33 && Math.abs(picPos.y - playerPos.y) < 33 && !this.isDead){
        if(this.player.currentHP <= this.currentAtk){
            this.player.currentHP = 0;
        }
        else{
            this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
        }
        this.player.isAttacked = true;
        this.isTouchingPlayer = true;
    }
};
```

18.Dragonfly.js：

```
var Dragonfly = function(map,player){
    Monster.call(this, 5, 10, 1, player, map);//Call the parent's constructor(hp,atk,exp,player)
    this.load = function(){
      this.url = define.imagePathMonster + "dragonfly.png";
      this.pic = new Framework.AnimationSprite({url:this.url, col:7, row:4, loop: true, speed: 7});
      this.walkDir = {x:0, y:0};
      this.counter = 0;
      this.name = 'dragonfly';
```

```
}

this.draw = function(ctx){
  var picPos = this.pic.position;
  var screenPos = this.map.screenPosition;
  var playerPos = this.player.playerPic.position;

  ctx.fillStyle = 'rgb(176, 60, 87)';
  if(this.map.mapType === 'scroll'){
    ctx.fillRect(picPos.x-screenPos.x-30, picPos.y-screenPos.y-30, 60*(this.currentHP/this.hp), 10);
  }
  else{
    ctx.fillRect(picPos.x-30, picPos.y-30, 60*(this.currentHP/this.hp), 10);
  }
  if(this.isTouchingPlayer){
    this.printDamageReceived(ctx);
  }
}

this.update = function(){
  if(this.counter > 100){
    var randomDir = Math.floor(Math.random() * 4);
    this.counter = 0;
  }
  this.counter++;
  this.move(randomDir);
  this.checkMonsterDead();
  this.revive();
  this.pic.update();

  if(this.player.currentHP > 0 && this.player.isAttacked === false){
    this.touchPlayer();
  }

  if(this.player.resistCounter === 0){
    this.player.isAttacked = false;
```

```
      this.isTouchingPlayer = false;
      this.player.resistCounter = this.player.resistTime;
    }
  }

  this.setPosition = function(pos){
    this.pic.position = {
      x: pos.x * 32 + 16,
      y: pos.y * 32 + 16
    };
  }
}
Dragonfly.prototype = Object.create(Monster.prototype);
Dragonfly.prototype.constructor = Monster;

Dragonfly.prototype.move = function(randomDir){
 switch (randomDir) {
  case 0://up
    this.walkDir = {x: 0, y: -1};
    this.pic.start({from:0, to:6, loop:true});
    break;
  case 1://left
    this.walkDir = {x: -1, y: 0};
    this.pic.start({from:21, to:27, loop:true});
    break;
  case 2://down
    this.walkDir = {x: 0, y: 1};
    this.pic.start({from:14, to:20, loop:true});
    break;
  case 3://right
    this.walkDir = {x: 1, y: 0};
    this.pic.start({from:7, to:13, loop:true});
    break;
 }
 if(this.map.canMove(this.pic.position.x + this.walkDir.x, this.pic.position.y + this.walkDir.y, randomDir)){
    this.pic.position.x += this.walkDir.x;
```

```
    this.pic.position.y += this.walkDir.y;
  }
};

Dragonfly.prototype.touchPlayer = function(){
   var picPos = this.pic.position;
   var playerPos = this.player.playerPic.position;
   if(Math.abs(picPos.x - playerPos.x) < 33 && Math.abs(picPos.y - playerPos.y) < 33 && !this.isDead){
      if(this.player.currentHP <= this.currentAtk){
         this.player.currentHP = 0;
      }
      else{
         this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
      }
      this.player.isAttacked = true;
      this.isTouchingPlayer = true;
   }
};
```

19.FireBall.js：

```
var FireBall = function(map,monster){
    Skill.call(this, 2, 0.2, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.monster = monster;
    this.load = function(){
      this.url = define.imagePathEffect + 'fireball2.png';
      this.pic = new Framework.Sprite(this.url);
      this.pic.scale = 1;
      this.counter = 0;
      this.isAttack = true;
      this.direction = 0;
    }
}
FireBall.prototype = Object.create(Skill.prototype);
FireBall.prototype.constructor = Skill;

FireBall.prototype.update = function(){
```

```
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;

    this.pic.update();
    this.move();

    if(this.canHitMonster()){
      if(this.player.currentHP <= this.damage){
          this.player.currentHP = 0;
      }
      else{
          this.player.currentHP -= Math.floor(this.damage * (1-this.player.def));
      }
      this.player.isAttacked = true;
    }

    if(this.isAttack && this.coolDownCounter > 0){
      this.coolDownCounter--;
    }

    if(this.coolDownCounter < this.coolDownTime * 100){
      this.canAttack = false;
    }
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
    }
};

FireBall.prototype.canHitMonster = function(){
    var playerPos = this.player.playerPic.position;
    var dirDifference = {
      x: playerPos.x - this.pic.position.x,
      y: playerPos.y - this.pic.position.y
    };
```

```
    if(Math.abs(dirDifference.x) < 65 && Math.abs(dirDifference.y) < 65){
        return true;
    }
    return false;
};

FireBall.prototype.move = function(){
    if(this.direction === 0){
      this.pic.position.y -= 5;
    }
    else{
      this.pic.position.y += 5;
    }
};

FireBall.prototype.attack = function(direction){
    var playerPos = this.player.playerPic.position;
    var screenRow = Math.floor(this.map.screenPosition.y/32), screenCol = Math.floor(this.map.screenPosition.x/32);

    this.direction = direction;
    if(this.direction === 0){
      this.pic.rotation = 180;
    }
};
```

20.FireStrike.js：

```
var FireStrike = function(map,monster){
     Skill.call(this, 500, 3, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
     this.monster = monster;
     this.load = function(){
       this.url = define.imagePathEffect + '010.png';
       this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:3, loop:true, speed:4});
       this.pic.scale = 1;
       this.pic.position = {
```

```
        x: -500,
        y: -500
      }
    }
}
FireStrike.prototype = Object.create(Skill.prototype);
FireStrike.prototype.constructor = Skill;

FireStrike.prototype.update = function(){
    var i = this.monsterNumber;
    var playerPos = this.player.playerPic.position;
    this.pic.update();

    if(this.isAttack && this.coolDownCounter > 0){
      this.coolDownCounter--;
    }

    if(this.coolDownCounter < this.coolDownTime * 100){
      this.canAttack = false;
    }
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
    }
};

FireStrike.prototype.canHitMonster = function(direction){
      var playerPos = this.player.playerPic.position;
      var monsterPos = this.monster.pic.position;
      var dirDifference = {
        x: playerPos.x - monsterPos.x,
        y: playerPos.y - monsterPos.y
      };

      if(Math.abs(dirDifference.x) < 180 && Math.abs(dirDifference.y) < 180){
```

```
        return true;
    }
    return false;
};

FireStrike.prototype.attack = function(direction){
   if(this.canHitMonster(direction)){
     if(this.player.currentHP <= this.damage){
        this.player.currentHP = 0;
     }
     else{
        this.player.currentHP -= Math.floor(this.damage * (1-this.player.def));
     }
     this.player.isAttacked = true;
   }
}
```

21.Frostbolt.js：

```
var Frostbolt = function(map,monster){
    Skill.call(this, 1000, 5, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.monster = monster;
    this.load = function(){
      this.url1 = define.imagePathEffect + '015.png';
      this.url2 = define.imagePathEffect + 'water_002.png';
      this.picArr = [];
      for(var i = 0; i < 10; i++){
        this.pic = new Framework.AnimationSprite({url:this.url1, col:5, row:3, loop:true, speed:3});
        this.pic.scale = 1;
        this.pic.position = {
          x: -100,
          y: -500
        }
        this.picArr.push(this.pic);
      }
      this.effectPic = new Framework.AnimationSprite({url:this.url2, col:5, row:2, loop:true, speed:3});
      this.counter = 0;
```

```
        this.isAttack = true;
    }
}
Frostbolt.prototype = Object.create(Skill.prototype);
Frostbolt.prototype.constructor = Skill;

Frostbolt.prototype.update = function(){
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    for(var i = 0; i < this.picArr.length; i++){
      this.picArr[i].update();
    }
    this.effectPic.update();

    if(this.isAttack && this.coolDownCounter > 0){
      this.coolDownCounter--;
    }
    if(this.coolDownCounter < this.coolDownTime * 100){
      this.canAttack = false;
    }
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
    }
};

Frostbolt.prototype.canHitMonster = function(i){
      var playerPos = this.player.playerPic.position;
      var dirDifference = {
        x: playerPos.x - this.picArr[i].position.x,
        y: playerPos.y - this.picArr[i].position.y
      };

      if(Math.abs(dirDifference.x) < 100 && Math.abs(dirDifference.y) < 100){
          return true;
```

```
    }
    return false;
};

Frostbolt.prototype.attack = function(){
   var playerPos = this.player.playerPic.position;
   var screenRow = Math.floor(this.map.screenPosition.y/32), screenCol = Math.floor(this.map.screenPosition.x/32);

   for(var i = 0; i < this.picArr.length; i++){
    if(this.map.mapType === 'scroll'){
      while(true){
        var row = Math.floor(Math.random() * 37), col = Math.floor(Math.random() * 85);
        if(row >= screenRow + 1 && row <= screenRow + 21 && col >= screenCol+1 && col <= screenCol + 42){
           break;
        }
      }
    }
    else{
        var row = Math.floor(Math.random() * 22), col = Math.floor(Math.random() * 43);
    }

    this.picArr[i].position = {
      x: col * 32 + 64,
      y: row * 32 - 64
    };

    this.picArr[i].start({from:0, to:14, loop:false});

    if(this.canHitMonster(i)){
      this.effectPic.position = {
        x: playerPos.x,
        y: playerPos.y-30
      };
      this.effectPic.start({from:0, to:9, loop:false});
      if(this.player.currentHP <= this.damage){
          this.player.currentHP = 0;
```

```
        }
        else{
            this.player.currentHP -= Math.floor(this.damage * (1-this.player.def));
        }
        this.player.isAttacked = true;
      }
    }
}
```

22.FrozenSword.js：

```
var FrozenSword = function(map){
    Skill.call(this, 2, 15, 0, 6, 3, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
      this.url = define.imagePathEffect + 'Warrior/005.png';
      this.url2 = define.imagePathEffect + '001.png';
      this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:3, loop:true, speed:5});
      this.effectPic = new Framework.AnimationSprite({url:this.url2, col:5, row:4, loop:true, speed:8});
      this.picArr = [];
      for(var i = 0; i < 150; i++){
        this.picArr.push(this.pic);
      }
    }

    this.init = function(){
      this.abilityList = [0, 6, 6.5, 7, 7.5, 8];
      this.mpCostList = [0, 60, 80, 100, 120, 140];
      this.pic.scale = 1.2;

      this.pic.position = {
        x: -500,
        y: -500
      }
      this.effectPic.position = {
        x: -500,
        y: -500
      }
```

```
    }

    this.draw = function(){
      var monsterArr = this.map.monsterArr;

      if(this.player.frozenSword.isDrawed){
        for(var i = 0; i < monsterArr.length; i++){
          if(!this.map.monsterArr[i].isDead){
            this.picArr[i].position = {
              x: monsterArr[i].pic.position.x,
              y: monsterArr[i].pic.position.y
            }
            if(this.map.mapType === 'scroll'){
              this.map.drawOtherObj(this.picArr[i]);
            }
            else{
              this.picArr[i].draw();
            }
          }
        }

        this.effectPic.position = this.player.playerPic.position;
        if(this.map.mapType === 'scroll'){
          this.map.drawOtherObj(this.effectPic);
        }
        else{
          this.effectPic.draw();
        }
      }
    }
}

FrozenSword.prototype = Object.create(Skill.prototype);
FrozenSword.prototype.constructor = Skill;

FrozenSword.prototype.levelUp = function(){
```

```
    this.level = 5;
};

FrozenSword.prototype.canHitMonster = function(direction,i){
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    var dirDifference = {
      x: playerPos.x - monsterArr[i].pic.position.x,
      y: playerPos.y - monsterArr[i].pic.position.y
    };

    switch (direction) {
      case 0:
        if(Math.abs(dirDifference.x) <= 96 && dirDifference.y <= 96 && dirDifference.y >= 0){
            return true;
        }
        return false;
      case 1:
        if(dirDifference.x <= 96 && dirDifference.x >= 0 && Math.abs(dirDifference.y) <= 96){
            return true;
          }
        return false;
      case 2:
        if(Math.abs(dirDifference.x) <= 96 && dirDifference.y >= -96 && dirDifference.y <= 0){
            return true;
          }
        return false;
      case 3:
        if(dirDifference.x >= -96 && dirDifference.x <= 0 && Math.abs(dirDifference.y) <= 96){
            return true;
          }
        return false;
    }
};

FrozenSword.prototype.attack = function(){
```

```
    var monsterArr = this.map.monsterArr;

    this.pic.start({from:0, to:13, loop:false});
    this.effectPic.start({from:0, to:16, loop:false});

    for(var i = 0; i < monsterArr.length; i++){//Determine if player hits the monster
      if(monsterArr[i].canBeAttacked){
          if(monsterArr[i].currentHP >= this.damage){
            monsterArr[i].currentHP -= this.damage;
          }
          else{
            monsterArr[i].currentHP = 0;
          }
      }
    }
};
```

23.GameLevel1.js：

```
var GLevel1 = Framework.Class(Framework.Level , {
    load: function(){
        this.allMap = new Map1();
        this.allMap.load();
        this.allMap.generateMonsters();

        this.audio = new Framework.Audio({
            song1:{
              mp3: define.musicPath + 'Normal.mp3'
            },
            song2:{
              mp3: define.musicPath + 'Boss.mp3'
            }
        });
        this.audio.play({name: 'song1', loop: true});
    },
    loadingProgress: function(context, requestInfo){
        context.fillRect(0, 0, 1360, 700);
```

```
},

initialize: function() {
    this.playSongCounter = 0;
    this.allMap.initialize();
},

update: function() {
     this.allMap.update();
    if(this.allMap.player.isChangeMap){
            this.allMap.generateMonsters();
    }
    if(this.allMap.player.currentMap === 4 && this.playSongCounter === 0){
            this.playSongCounter++;
            this.audio.stop('song1');
            this.audio.play({name: 'song2', loop: true});
    }
},

draw: function(parentCtx){
    this.allMap.draw(parentCtx);
},

keydown:function(e, list){
    this.allMap.player.keydown(e, list);

   if(e.key === 'F11') {
        if(!this.isFullScreen) {
            Framework.Game.fullScreen();
            this.isFullScreen = true;
        }
        else {
            Framework.Game.exitFullScreen();
            this.isFullScreen = false;
        }
    }
```

```
    },

    keyup:function(e, list){
        this.allMap.player.keyup(e, list);
    },

    mousemove:function(e){
        this.allMap.player.bag.mousemove(e);
        this.allMap.player.questUI.mousemove(e);
        this.allMap.player.skillUI.mousemove(e);
        this.allMap.treasureChest.mousemove(e);
        this.allMap.mousemove(e);
    },

    touchstart: function (e) {

    },

    click: function (e) {
        this.allMap.player.bag.click(e);
        this.allMap.player.questUI.click(e);
        this.allMap.player.skillUI.click(e);
        this.allMap.treasureChest.click(e);
        this.allMap.click(e);
    },
});
```

24.HeroSlash.js：

```
var HeroSlash = function(map){
    Skill.call(this, 2, 10, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
      this.url = define.imagePathEffect + 'Warrior/006.png';
      this.url2 = define.imagePathEffect + '023.png';
      this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:2, loop:true, speed:5});
      this.effectPic = new Framework.AnimationSprite({url:this.url2, col:5, row:5, loop:true, speed:8});
    }
```

```
    this.init = function(){
      this.abilityList = [0, 0.01, 0.02, 0.03, 0.04, 0.1];
      this.mpCostList = [0, 100, 150, 200, 250, 300];
      this.pic.scale = 1;

      this.durationTime = 300;
      this.timeCounter = 0;
      this.direction = 0;
      this.hit = false;
    }

    this.draw = function(){
      if(this.isDrawed && this.timeCounter !== this.durationTime && this.timeCounter > 1){
        if(this.map.mapType === 'scroll'){
          this.map.drawOtherObj(this.pic);
        }
        else{
          this.pic.draw();
        }
      }
    }

}

HeroSlash.prototype = Object.create(Skill.prototype);
HeroSlash.prototype.constructor = Skill;

HeroSlash.prototype.levelUp = function(){
   this.level = 5;
};

HeroSlash.prototype.move = function(){

  if(!this.hit && this.isAttack){
    if(this.direction === 0){//up
```

```
      this.pic.position.y-=2;
    }

    else if(this.direction === 1){//left
      this.pic.position.x-=2;
    }
    else if(this.direction === 2){
      this.pic.position.y+=2;
    }

    else if(this.direction === 3){
      this.pic.position.x+=2;
    }
  }
}

HeroSlash.prototype.update = function(){
    var i = this.monsterNumber;
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    this.pic.update();
    this.changeDamage();
    this.changeMpCost();
    this.levelUp();

    if(this.timeCounter < this.durationTime && this.isAttack){
      this.timeCounter++;
      this.attack();
      this.move();

    }

    if(this.timeCounter % 30 === 0){
      this.pic.start({from:0, to:7, loop:false});
    }
```

```
    if(this.isAttack && this.coolDownCounter > 0){
      this.coolDownCounter--;
    }

    if(this.coolDownCounter < this.coolDownTime * 100){
      this.canAttack = false;
    }
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
      this.timeCounter = 0;
    }
};

HeroSlash.prototype.canHitMonster = function(i){
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    var dirDifference = {
      x: this.pic.position.x - monsterArr[i].pic.position.x,
      y: this.pic.position.y - monsterArr[i].pic.position.y
    };

    if(Math.abs(dirDifference.x) <= 96 && Math.abs(dirDifference.y) <= 96 && !monsterArr[i].isDead){
      this.hit = true;
      return true;
    }
    return false;
};

HeroSlash.prototype.attack = function(){
    var monsterArr = this.map.monsterArr;

    for(var i = 0; i < monsterArr.length; i++){//Determine if player hits the monster
      if(monsterArr[i].canBeAttacked && this.canHitMonster(i)){
          if(monsterArr[i].currentHP >= this.damage){
```

```
            monsterArr[i].currentHP -= this.damage;
          }
          else{
            monsterArr[i].currentHP = 0;
          }
      }
    }
};
```

25.IceBall.js：

```
var IceBall = function(){
    this.load = function(){
      this.url = define.imagePathEffect + 'snowball2.png';
      this.pic = new Framework.Sprite(this.url);
    }

    this.init = function(){
      this.windowSize = {
        width: Framework.Game.getCanvasWidth(),
        height: Framework.Game.getCanvasHeight()
      };
      this.pic.position = {
        x: this.randomNumber(100, this.windowSize.width),
        y: this.randomNumber(-10, -300)
      };
      this.velocity = this.randomNumber(4, 8);
      this.hit = this.randomNumber(this.windowSize.height * 0.8, this.windowSize.height * 0.9);
    }

    this.draw = function(){
      this.pic.draw();
    }

    this.update = function(){

      if(this.pic.position.y < this.hit){
```

```
      this.pic.position.x -= this.velocity / 30;
      this.pic.position.y += this.velocity / 10;
    }
    else {
      this.init();
    }
  }

  this.randomNumber = function(min, max){
    return Math.random() * (max - min) + min;
  }
}
```

26.Light.js：

```
var Light = function(map){
    Skill.call(this, 0, 0, 0, 0, 0, map);
    this.load = function(){
      this.url = define.imagePathEffect + 'Warrior/003-1.png';
      this.pic = new Framework.Sprite(this.url);
      this.pic.scale = 0.7;
      this.direction = 0;
    }

    this.draw = function(){
      if(this.isDrawed && this.player.swordLight.isDrawed){
        this.map.drawOtherObj(this.pic);
      }
    }
}
Light.prototype = Object.create(Skill.prototype);
Light.prototype.constructor = Skill;

Light.prototype.move = function(){
  if(this.direction === 0){//up
    this.pic.position.y-=2;
    this.pic.rotation = 180;
  }
  else if(this.direction === 1){//left
    this.pic.position.x-=2;
    this.pic.rotation = 90;
  }
  else if(this.direction === 2){
    this.pic.position.y+=2;
    this.pic.rotation = 0;
  }

  else if(this.direction === 3){
    this.pic.position.x+=2;
```

```
    this.pic.rotation = 270;
  }
}

Light.prototype.changeDamage = function(){
    if(this.player.level >= 10){
      this.damage = this.player.atk;
    }
}

Light.prototype.update = function(){
    this.pic.update();
    this.changeDamage();
    if(this.player.swordLight.isAttack){
      this.move();
      this.isDrawed = true;
    }
    else{
      this.moveDistance = 0;
      this.isDrawed = false;
    }
    this.attack();
};

Light.prototype.attack = function(){
  var monsterArr = this.map.monsterArr;
  var index = this.map.toArrayIndex(this.pic.position.x, this.pic.position.y, -1);

  if(this.isAttack){
    for(var i = 0; i < monsterArr.length; i++){
      if(Math.abs(this.pic.position.x - monsterArr[i].pic.position.x) <= 32 &&
         Math.abs(this.pic.position.y - monsterArr[i].pic.position.y) <= 32 &&
         monsterArr[i].canBeAttacked){
           if(monsterArr[i].currentHP >= this.damage){
             monsterArr[i].currentHP -= this.damage;
           }
```

```
        else{
          monsterArr[i].currentHP = 0;
        }
        this.isAttack = false;
    }
  }
}

if(this.map.mapType === 'scroll'){
  if(index.row >= 0 && index.row < 38 && index.col >= 0 && index.col < 86){
    if(this.map.mapArr[this.player.currentMap][index.row][index.col] === 7) {
       this.map.mapArr[this.player.currentMap][index.row][index.col] = 1;
    }
    if(this.map.mapArr[this.player.currentMap][index.row][index.col] > 1 &&
      this.map.mapArr[this.player.currentMap][index.row][index.col] < 8){
       this.map.mapArr[this.player.currentMap][index.row][index.col] = 0;
    }
  }
}
else{
  if(index.row >= 0 && index.row < 22 && index.col >= 0 && index.col < 43){
    if(this.map.testMapArr[index.row][index.col] === 7) {
       this.map.testMapArr[index.row][index.col] = 1;
    }
    if(this.map.testMapArr[index.row][index.col] > 1 &&
      this.map.testMapArr[index.row][index.col] < 8){
       this.map.testMapArr[index.row][index.col] = 0;
    }
  }
}
}
```

27.LoadGame.js：

```
var loadGameEnd;
(function(){//anonymous function
   var  importJS = function(jsConf, src, lookFor) {
```

```
    var headID = document.getElementsByTagName("head")[0];
    var newJs = document.createElement('script');
    newJs.type = 'text/javascript';
    newJs.src= jsConf[0].src;
    headID.appendChild(newJs);
    wait_for_script_load(jsConf, function() {
        jsConf.splice(0, 1);
        if(jsConf.length > 0) {
            importJS(jsConf, lookFor);
        }else
        {
            loadGameEnd = true;
        }
    });
}

var wait_for_script_load = function(jsConf, callback) {
    var interval = setInterval(function() {
        if (typeof jsConf[0].lookFor === 'undefined') {
            jsConf[0].lookFor = '';
        }

        if (jsConf[0].lookFor === '') {
            clearInterval(interval);
            callback();
        } else if (eval("typeof " + jsConf[0].lookFor) !== 'undefined') {
                clearInterval(interval);
                callback();
            }
        }, 50);
}
var listScript =
[
    { src: 'js/Define.js', lookFor: 'define' },
    { src: 'js/MyMenu.js', lookFor: 'MyMenu'},
    { src: 'js/BaseMap.js', lookFor: 'BaseMap'},
```

```
{ src: 'js/TestMap.js', lookFor: 'TestMap' },
{ src: 'js/Map1.js', lookFor: 'Map1' },
{ src: 'js/Map2.js', lookFor: 'Map2' },
{ src: 'js/Map3.js', lookFor: 'Map3' },
{ src: 'js/Map4.js', lookFor: 'Map4' },
{ src: 'js/Map5.js', lookFor: 'Map5' },
{ src: 'js/GameLevel1.js', lookFor: 'GLevel1' },
{ src: 'js/Arrays.js', lookFor: 'Arrays'},
{ src: 'js/PlayerData.js', lookFor: 'PlayerData'},
{ src: 'js/WarriorData.js', lookFor: 'WarriorData'},
{ src: 'js/QuestInterface.js', lookFor: 'QuestInterface'},
{ src: 'js/Quest1.js', lookFor: 'Quest1'},
{ src: 'js/Quest2.js', lookFor: 'Quest2'},
{ src: 'js/Quest3.js', lookFor: 'Quest3'},
{ src: 'js/Quest4.js', lookFor: 'Quest4'},
{ src: 'js/Quest5.js', lookFor: 'Quest5'},
{ src: 'js/Player.js', lookFor: 'Player'},
{ src: 'js/SkillShowcase.js', lookFor: 'SkillShowcase'},
{ src: 'js/Skill.js', lookFor: 'Skill'},
{ src: 'js/SkillUI.js', lookFor: 'SkillUI'},
{ src: 'js/QuestUI.js', lookFor: 'QuestUI'},
{ src: 'js/BlackHole.js', lookFor: 'BlackHole'},
{ src: 'js/BasicAttack.js', lookFor: 'BasicAttack'},
{ src: 'js/VCut.js', lookFor: 'VCut'},
{ src: 'js/BurstCut.js', lookFor: 'BurstCut'},
{ src: 'js/CrossCut.js', lookFor: 'CrossCut'},
{ src: 'js/SwordLight.js', lookFor: 'SwordLight'},
{ src: 'js/Light.js', lookFor: 'Light'},
{ src: 'js/MicroShield.js', lookFor: 'MicroShield'},
{ src: 'js/DarkForce.js', lookFor: 'DarkForce'},
{ src: 'js/FrozenSword.js', lookFor: 'FrozenSword'},
{ src: 'js/HeroSlash.js', lookFor: 'HeroSlash'},
{ src: 'js/PotentialPower.js', lookFor: 'PotentialPower'},
{ src: 'js/Teleport.js', lookFor: 'Teleport'},
{ src: 'js/Thunder.js', lookFor: 'Thunder'},
{ src: 'js/BlackBall.js', lookFor: 'BlackBall'},
```

```
        { src: 'js/FireBall.js', lookFor: 'FireBall'},
        { src: 'js/Boomerang.js', lookFor: 'Boomerang'},
        { src: 'js/FireStrike.js', lookFor: 'FireStrike'},
        { src: 'js/Frostbolt.js', lookFor: 'Frostbolt'},
        { src: 'js/Bag.js', lookFor: 'Bag'},
        { src: 'js/Monster.js', lookFor:'Monster'},
        { src: 'js/Dragonfly.js', lookFor: 'Dragonfly'},
        { src: 'js/Ant.js', lookFor: 'Ant'},
        { src: 'js/Spider.js', lookFor: 'Spider'},
        { src: 'js/Calavera.js', lookFor: 'Calavera'},
        { src: 'js/Devil.js', lookFor: 'Devil'},
        { src: 'js/Defender.js', lookFor: 'Defender'},
        { src: 'js/Boss.js', lookFor: 'Boss'},
        { src: 'js/Property.js', lookFor: 'Property'},
        { src: 'js/QuestItem.js', lookFor: 'QuestItem'},
        { src: 'js/Cluster.js', lookFor: 'Cluster'},
        { src: 'js/Medicine.js', lookFor: 'Medicine'},
        { src: 'js/TreasureChest.js', lookFor: 'TreasureChest'},
        { src: 'js/RainDrop.js', lookFor: 'RainDrop'},
        { src: 'js/IceBall.js', lookFor: 'IceBall'},
        { src: 'js/MainGame.js'}
    ]
    importJS(listScript);
})();
```

28.MainGame.js：

```
Framework.Game.fps = 60;
Framework.Game.canvasWidth = 1600;
Framework.Game.canvasHeight = 900;

Framework.Game.addNewLevel({menu: new MyMenu()});
Framework.Game.addNewLevel({gameLevel1: new GLevel1()});
Framework.Game.addNewLevel({skillShowcase: new SkillShowcase()});
Framework.Game.start();
```

29.Map1.js：

```
var Map1 = function() {
    BaseMap.call(this,'scroll');
    this.screenPosition = {//on the map
        x: 600,
        y: 323
    };//x: 600, y: 323

    this.player = new Player(this);
    this.player.load();
    this.player.init();
    this.player.setPosition({x: 38, y: 18},this.screenPosition);//x: 38, y: 18

    this.alpha = 1;
    this.alpha2 = 1;

    this.propertyAmount = 70;
    this.monsterAmount = 50;

    this.map2 = new Map2(this);
    this.map3 = new Map3(this);
    this.map4 = new Map4(this);
    this.map5 = new Map5(this);

    this.load = function() {
        this.linkBtn = new Framework.Sprite(define.imagePath + "linkBtn.png");
        this.linkBtn_hover = new Framework.Sprite(define.imagePath + "linkBtn_hover.png");
        this.info = new Framework.Sprite(define.imagePathMap + "UI1-1.png");
        this.grassPic = new Framework.Sprite(define.imagePathMap + "grass.png");
        this.treePic = new Framework.Sprite(define.imagePathMap + "tree.png");
        this.stonePic = new Framework.Sprite(define.imagePathMap + "stone2.png");
        this.floorPic = new Framework.Sprite(define.imagePathMap + "bg5.png");
        this.flowerPic = new Framework.Sprite(define.imagePathMap + "flower2.png");
        this.cloudPic = new Framework.Sprite(define.imagePathMap + "sky.png");
```

```
    this.cluster = new Cluster(this);
    this.cluster.load();
    this.medicine = new Medicine(this);
    this.medicine.load();
    this.treasureChest = new TreasureChest(this);
    this.treasureChest.load();
    this.blackHole = new BlackHole(this);
    this.blackHole.load();

    this.rainArr = [];
    for(var i = 0; i < 180; i++){
      this.rainDrop = new RainDrop();
      this.rainDrop.init();
      this.rainArr.push(this.rainDrop);
    }
};

this.initialize = function() {
    this.player.setPlayerLevel(1);
    this.player.init();
    this.treasureChest.init();

    this.linkBtn.position = {
        x: 1280,
        y: 50
    };
    this.linkBtn_hover.position = this.linkBtn.position;

    this.floorPic.position = {x: 1370, y: 600};
    this.info.position = {x:700, y:785};

    this.floorPic.scale = 1.5;
    this.linkBtn.scale = 0.35;
    this.linkBtn_hover.scale = 0.35;
    this.isLinkBtnHoveer = false;
```

```
    this.generateProperties();
    this.isWin = false;
    this.drawEff = true;
    this.disPlayChar = false;
    this.isHeavyRain = false;

    this.littleRain = 30;
    this.heavyRain = 180;
    this.charIndex = 0;
    this.charIndex2 = 0;
    this.letterSpace = 30;
    this.counter = 0;
    this.rainDropCounter = 0;
    this.displayCharCounter = 0;
};

this.update = function() {
    this.player.update();
    this.treasureChest.update();
    this.blackHole.update();

    this.moveScreen();
    this.checkChangeMap();

    if(this.player.isChangeMap){
      this.monsterArr.splice(0, this.monsterArr.length);
    }
    for(var i = 0; i < this.monsterArr.length; i++){
      this.monsterArr[i].update();
      if(this.monsterArr[i].isTouchingPlayer){
        this.isDrawDamage = true;
      }
    }

    if(this.player.resistCounter === 270){
      this.isDrawDamage = false;
```

```
    }

    if(this.player.currentHP <= 0 && this.player.currentMap === 3){
      this.player.setPosition({x: 1, y: 2},this.screenPosition);
      this.screenPosition = {x:0, y: 0};
      this.player.currentHP = this.player.healthPoint;
      this.player.currentMP = this.player.manaPoint;
    }

    if(this.player.currentMap === 4){
      this.rainDropCounter++;
    }

    if(this.disPlayChar){
      this.displayCharCounter++;
    }

    if(this.displayCharCounter % 15 === 1){
      if(this.isWin && this.charIndex < this.charArr.length){
        this.charIndex++;
      }
      else if(this.charIndex2 < this.charArr2.length){
        this.charIndex2++;
      }
    }
};

this.draw = function(ctx) {
    var screenRow = Math.floor(this.screenPosition.y / 32), screenCol = Math.floor(this.screenPosition.x / 32);
    var rowNum = 0, colNum = 0;
    var playerPos = this.player.playerPic.position;
    var screenPos = this.screenPosition;

      this.floorPic.draw();
      if(this.player.currentMap === 4){
        ctx.fillStyle = 'rgba(0, 0, 0, 0.8)';
```

```
  ctx.fillRect(0, 0, 1360, 700);
}
for (var i = screenRow; i < screenRow+23; i++) {//38
    for (var j = screenCol; j < screenCol+44; j++) {//86
        if(i >= 0 && i < 38 && j >= 0 && j < 86){
          var picPosition = {
             x: 32 * j  - this.screenPosition.x + 16,
             y: 32 * i  - this.screenPosition.y + 16
          }
          switch(this.mapArr[this.player.currentMap][i][j]) {
             case 1 :
                this.grassPic.position = picPosition;
                this.grassPic.draw();
                break;
             case 6 :
                this.flowerPic.position = picPosition;
                this.flowerPic.draw();
                break;
             case 7 :
                this.treePic.position = picPosition;
                this.treePic.draw();
                break;
             case 8 :
                this.stonePic.position = picPosition;
                this.stonePic.draw();
                break;
             case 9 :
                this.cloudPic.position = picPosition;
                this.cloudPic.draw();
                break;
          }
          if(this.mapArr[this.player.currentMap][i][j] < 2){
             switch (this.propertyArr[i][j]) {
               case 1:
                this.cluster.blue_cluster.position = picPosition;
                this.cluster.blue_cluster.draw();
```

```
                break;
              case 2:
                this.cluster.red_cluster.position = picPosition;
                this.cluster.red_cluster.draw();
                break;
              case 3:
                this.cluster.green_cluster.position = picPosition;
                this.cluster.green_cluster.draw();
                break;
              case 4:
                this.medicine.hpPotion.position = picPosition;
                this.medicine.hpPotion.draw();
                break;
              case 5:
                this.medicine.mpPotoin.position = picPosition;
                this.medicine.mpPotoin.draw();
                break;
              case 10:
                this.cluster.tourmaline.position = picPosition;
                this.cluster.tourmaline.draw();
                break;
              case 11:
                this.cluster.zircon.position = picPosition;
                this.cluster.zircon.draw();
                break;
            }
        }
      }
    }
}
if(this.player.currentMap === 4 && this.rainDropCounter > 360){
  this.generateRain(ctx);
}

/*----------Draw Treasure Chest----------*/
if(this.player.currentMap === 3 && this.treasureChest.drawPic){
```

```
  if(this.treasureChest.isPicHover){
    this.drawOtherObj(this.treasureChest.picHover);
  }
  else{
    this.drawOtherObj(this.treasureChest.pic);
  }
}
this.treasureChest.draw(ctx);
/*-----------------------------------------------*/

if(this.player.currentMap === 3){
  this.blackHole.draw(ctx);
}

if(this.player.currentHP > 0){
  this.drawOtherObj(this.player.playerPic);
}
else{
  this.drawOtherObj(this.player.corpsePic);
}

for(var i = 0; i < this.monsterArr.length; i++){
  if(this.monsterArr[i].currentHP > 0){
    this.drawOtherObj(this.monsterArr[i].pic);
  }
  this.monsterArr[i].draw(ctx);
}
this.info.draw();
this.player.draw(ctx);

if(!this.isLinkBtnHoveer){
    this.linkBtn.draw();
}
else {
    this.linkBtn_hover.draw();
}
```

```
/*-------Transition scene-------*/
if(this.drawEff){
  this.alpha -= 0.002;
  ctx.globalAlpha = this.alpha;
  ctx.fillStyle = 'black';
  ctx.fillRect(0, 0, 1360, 700);
}
if(this.alpha <= 0){
  this.drawEff = false;
}

/*-------Player winning scene-------*/
if(this.isWin && this.player.currentHP > 0){
  this.heavyRain--;
  if(this.alpha < 0.8){
    this.alpha += 0.003;
    ctx.globalAlpha = this.alpha;
  }
  else{
    this.disPlayChar = true;
  }
  ctx.fillStyle = "black";
  ctx.fillRect(0, 0, 1360, 700);
}

if(this.displayCharCounter > 0 && this.charIndex <= this.charArr.length && this.isWin){
  ctx.font = 'bold 36px itcblkad ';
  ctx.fillStyle = 'white';
  for(var i = 0; i < this.charIndex; i++){
    ctx.fillText(this.charArr[i], 300 + i*this.letterSpace, 160);
  }
}

/*-------Player losing scene-------*/
if(this.player.currentHP <= 0){
```

```
        if(this.alpha < 0.8){
          this.alpha += 0.003;
          ctx.globalAlpha = this.alpha;
        }
        else{
          this.disPlayChar = true;
        }
        ctx.fillStyle = "black";
        ctx.fillRect(0, 0, 1360, 700);
      }

      if(this.displayCharCounter > 0 && this.charIndex2 <= this.charArr2.length && this.player.currentHP <= 0){
        ctx.font = 'bold 36px chiller ';
        ctx.fillStyle = 'red';
        for(var i = 0; i < this.charIndex2; i++){
          ctx.fillText(this.charArr2[i], 300 + i*this.letterSpace, 160);
        }
      }

};

this.generateRain = function(ctx){
    var width = this.rainDrop.windowSize.width
    var height = this.rainDrop.windowSize.height;

    ctx.fillStyle = this.rainDrop.clearColor;
    ctx.fillRect(0, 0, width, height);

    if(!this.isHeavyRain){
      for(var i = 0; i < this.littleRain; i++){
        this.rainArr[i].draw(ctx);
      }
    }
    else{
      for(var i = 0; i < this.heavyRain; i++){
        this.rainArr[i].draw(ctx);
```

```
        }
      }
    };

    this.mousemove = function(e){
      if(e.x >= 1250 && e.x <= 1300 && e.y >= 26 && e.y <= 69){
        this.isLinkBtnHoveer = true;
      }
      else{
        this.isLinkBtnHoveer = false;
      }
    };

    this.click = function(e){
      if(e.x >= 1250 && e.x <= 1300 && e.y >= 26 && e.y <= 69){
        window.open("Game_Info/MainPage/index.html");
      }
    };
}

Map1.prototype = Object.create(BaseMap.prototype);
Map1.prototype.constructor = BaseMap;

Map1.prototype.generateProperties = function(){
    var propertyType = 0;
    var row, col;
    for(var i = 0; i < this.propertyAmount; i++){
      while(true){
        row = Math.floor(Math.random() * 37);
        col = Math.floor(Math.random() * 85);
        if(this.mapArr[this.player.currentMap][row][col] > 5 &&
          this.mapArr[this.player.currentMap][row][col] < 8){
          break;
        }
      }
      if(i < 20){
```

```
        propertyType = 1;
      }
      else if(i < 40){
        propertyType = 2;
      }
      else if(i < 60){
        propertyType = 3;
      }
      else if(i < 65){
        propertyType = 4;
      }
      else if(i < 70){
        propertyType = 5;
      }
      this.propertyArr[row][col] = propertyType;
    }
};

Map1.prototype.moveScreen = function(){
   if(this.player.isWalking && this.player.isScreenMove){
     if(this.player.currentMap === 0){
       if(this.player.playerPic.position.x > 615 && this.player.playerPic.position.x < 2031){
          this.screenPosition.x += this.player.walkDir.x;
       }
       if(this.player.playerPic.position.y > 268 && this.player.playerPic.position.y < 900){
          this.screenPosition.y += this.player.walkDir.y;
       }
     }
   }
   if(this.player.currentMap === 1){
      this.map2.moveScreen();
   }
   if(this.player.currentMap === 2){
      this.map3.moveScreen();
   }
   if(this.player.currentMap === 3){
```

```
        this.map4.moveScreen();
    }
    if(this.player.currentMap === 4){
        this.map5.moveScreen();
    }
};

Map1.prototype.checkChangeMap = function(){
    var playerIndex = this.toArrayIndex(this.player.playerPic.position.x, this.player.playerPic.position.y, -1);
    var playerPos = this.player.playerPic.position;

    this.player.isChangeMap = false;
    if(this.player.currentMap === 0){
      if(playerPos.x >= 1184 && playerPos.x <= 1278 && playerPos.y <= 26){
          this.player.currentMap = 1;
          this.player.isChangeMap = true;
          this.player.drawChangeMapEff = true;
          this.screenPosition = {
              x: 732,
              y: 626
          };
          this.player.setPosition({x: 37, y: 35},this.screenPosition);
          this.clearProperties();
          this.map2.generateProperties();
      }
    }

    if(this.player.currentMap === 1){
        this.map2.checkChangeMap();
    }
    if(this.player.currentMap === 2){
        this.map3.checkChangeMap();
    }
    if(this.player.currentMap === 3){
        this.map4.checkChangeMap();
    }
```

```
};

Map1.prototype.generateMonsters = function(){
  /*-------------Normal Map-------------*/
  var calaveraNum = 0;
  for(var i = 0; i < this.monsterAmount; i++){
    while(true){
      var row = Math.floor(Math.random() * 37), col = Math.floor(Math.random() * 85);
      if(this.mapArr[this.player.currentMap][row][col] < 2){
        break;
      }
    }
    this.dragonfly = new Dragonfly(this, this.player);
    this.dragonfly.load();
    this.dragonfly.setPosition({x: col, y: row});

    this.ant = new Ant(this,this.player);
    this.ant.load();
    this.ant.setPosition({x: col, y: row});

    this.spider = new Spider(this,this.player);
    this.spider.load();
    this.spider.setPosition({x: col, y: row});

    this.calavera = new Calavera(this,this.player);
    this.calavera.load();

    if(i % 5 === 0){
      calaveraNum++;
      this.calavera.setPositionAndDirection({x: calaveraNum * 8, y: 35},0);
    }

    if((i-1) % 5 === 0){
      this.calavera.setPositionAndDirection({x: calaveraNum * 8, y: 2},2);
    }
```

```
  if(this.player.currentMap === 0){
    this.monsterArr.push(this.dragonfly);
  }
  else if(this.player.currentMap === 1){
    this.monsterArr.push(this.ant);
  }
  else if(this.player.currentMap === 2){
    this.monsterArr.push(this.spider);
  }
  else if(this.player.currentMap === 3 && (i % 5 === 0 || (i-1) % 5 === 0)){
    this.monsterArr.push(this.calavera);
  }
}
/*-------------Boss Map----------------*/
for(var i = 0; i < 5; i++){
  while(true){
    var row = Math.floor(Math.random() * 21), col = Math.floor(Math.random() * 42);
    if(this.mapArr[this.player.currentMap][row][col] < 2){
      break;
    }
  }
  this.devil = new Devil(this, this.player);
  this.devil.load();
  this.devil.setPosition({x: col, y: row});

  this.boss = new Boss(this, this.player);
  this.boss.load();
  this.boss.setPosition({x: 21, y: 10});
  this.boss.init();

  this.defender = new Defender(this, this.player);
  this.defender.load();

  if(this.player.currentMap === 4){
    if(i === 0){
      this.monsterArr.push(this.boss);
```

```
        }
        else if(i < 3){
          this.monsterArr.push(this.devil);
        }
        else if(i < 5){
          if(i === 3){
            this.defender.setPosition({x: 10, y: 10});
          }
          else{
            this.defender.setPosition({x: 32, y: 10});
          }
          this.monsterArr.push(this.defender);
        }
      }
    }
};
```

30.Map2.js：

```
var Map2 = function(map) {
    this.map = map;
    this.player = this.map.player;
    this.isChangeMap = false;
    this.monsterAmount = 30;
    this.propertyAmount = 50;

    this.setScreenPosition = function(screenPosition){
        this.screenPosition = {
          x: screenPosition.x,
          y: screenPosition.y
        };
    };
}

Map2.prototype = Object.create(BaseMap.prototype);
Map2.prototype.constructor = BaseMap;
```

```
Map2.prototype.generateProperties = function(){
    var propertyType = 0;
    var row, col;
    for(var i = 0; i < this.propertyAmount; i++){
      while(true){
        row = Math.floor(Math.random() * 37);
        col = Math.floor(Math.random() * 85);
        if(this.map.mapArr[this.player.currentMap][row][col] > 5 &&
          this.map.mapArr[this.player.currentMap][row][col] < 8){
          break;
        }
      }
      if(i < 10){
        propertyType = 1;
      }
      else if(i < 20){
        propertyType = 2;
      }
      else if(i < 30){
        propertyType = 3;
      }
      else if(i < 40){
        propertyType = 4;
      }
      else if(i < 50){
        propertyType = 5;
      }
      this.map.propertyArr[row][col] = propertyType;
    }
};


Map2.prototype.moveScreen = function(){
   if(this.player.isWalking && this.player.isScreenMove){
     if(this.player.playerPic.position.x > 430 && this.player.playerPic.position.x < 1891){
        this.map.screenPosition.x += this.player.walkDir.x;
     }
```

```
    if(this.player.playerPic.position.y > 268 && this.player.playerPic.position.y < 900){
        this.map.screenPosition.y += this.player.walkDir.y;
    }
  }
};

Map2.prototype.checkChangeMap = function(){
  var playerIndex = this.toArrayIndex(this.player.playerPic.position.x, this.player.playerPic.position.y, -1);
  var playerPos = this.player.playerPic.position;

  if(playerPos.x >= 1184 && playerPos.x <= 1278 && playerPos.y > 1174){
    this.player.currentMap = 0;
    this.player.isChangeMap = true;
    this.player.drawChangeMapEff = true;
    this.map.screenPosition = {
      x: 600,
      y: 0
    };
    this.player.setPosition({x: 37, y: 1}, this.map.screenPosition);
    this.map.clearProperties();
    this.map.generateProperties();
  }
  if(playerPos.x >= 1184 && playerPos.x <= 1278 && playerPos.y <= 26){
    this.player.currentMap = 2;
    this.player.isChangeMap = true;
    this.player.drawChangeMapEff = true;
    this.map.screenPosition = {
      x: 732,
      y: 626
    };
    this.player.setPosition({x: 37, y: 36},this.screenPosition);
    this.map.clearProperties();
    this.map.map3.generateProperties();
  }
};
```

```
Map2.prototype.chooseMonsterType = function(col, row){
   this.map.ant = new Ant(this.map,this.player);
   this.map.ant.load();
   this.map.ant.setPosition({x: col, y: row});
};
```

31.Map3.js：

```
var Map3 = function(map) {
   this.map = map;
   this.player = this.map.player;
   this.isChangeMap = false;
   this.monsterAmount = 30;
   this.propertyAmount = 60;

   this.setScreenPosition = function(screenPosition){
      this.screenPosition = {
        x: screenPosition.x,
        y: screenPosition.y
      };
   };
}

Map3.prototype = Object.create(BaseMap.prototype);
Map3.prototype.constructor = BaseMap;

Map3.prototype.generateProperties = function(){
     var propertyType = 0;
     var row, col;
     for(var i = 0; i < this.propertyAmount; i++){
       while(true){
         row = Math.floor(Math.random() * 37);
         col = Math.floor(Math.random() * 85);
         if(this.map.mapArr[this.player.currentMap][row][col] > 5 &&
           this.map.mapArr[this.player.currentMap][row][col] < 8){
           break;
         }
```

```
        }
        if(i < 20){
          propertyType = 10;
        }
        else if(i < 40){
          propertyType = 11;
        }
        else if(i < 50){
          propertyType = 4;
        }
        else if(i < 60){
          propertyType = 5;
        }
        this.map.propertyArr[row][col] = propertyType;
      }
      console.log(this.map.propertyArr);
};

Map3.prototype.moveScreen = function(){
    if(this.player.isWalking && this.player.isScreenMove){
      if(this.player.playerPic.position.x > 430 && this.player.playerPic.position.x < 1891){
          this.map.screenPosition.x += this.player.walkDir.x;
      }
      if(this.player.playerPic.position.y > 268 && this.player.playerPic.position.y < 900){
          this.map.screenPosition.y += this.player.walkDir.y;
      }
    }
};

Map3.prototype.checkChangeMap = function(){
    var playerIndex = this.toArrayIndex(this.player.playerPic.position.x, this.player.playerPic.position.y, -1);
    var playerPos = this.player.playerPic.position;

    if(playerPos.x >= 1184 && playerPos.x <= 1278 && playerPos.y > 1174){
        this.player.currentMap = 1;//to map2
        this.player.isChangeMap = true;
```

```
        this.player.drawChangeMapEff = true;
        this.map.screenPosition = {
            x: 732,
            y: 0
        };
        this.player.setPosition({x: 37, y: 1}, this.map.screenPosition);
        this.map.clearProperties();
        this.map.map2.generateProperties();
    }

    if(playerPos.x > 2730 && playerPos.y > 32 && playerPos.y < 128){
        this.player.currentMap = 3;
        this.player.isChangeMap = true;
        this.player.drawChangeMapEff = true;
        this.map.screenPosition = {
            x: 0,
            y: 0
        };
        this.player.setPosition({x: 1, y: 1}, this.map.screenPosition);
        this.map.clearProperties();
    }
};

Map3.prototype.chooseMonsterType = function(col, row){
    this.map.ant = new Ant(this.map,this.player);
    this.map.ant.load();
    this.map.ant.setPosition({x: col, y: row});
};
```

32.Map4.js：

```
var Map4 = function(map) {
    this.map = map;
    this.player = this.map.player;
    this.isChangeMap = false;
    this.monsterAmount = 30;
    this.propertyAmount = 50;
```

```
    this.setScreenPosition = function(screenPosition){
        this.screenPosition = {
          x: screenPosition.x,
          y: screenPosition.y
        };
    };

    this.checkChangeStage = function(){
        if(this.map.player.bag.getPropertyNumber(31) >= 1){
          return true;
        }
        return false;
    };
}

Map4.prototype = Object.create(BaseMap.prototype);
Map4.prototype.constructor = BaseMap;

Map4.prototype.moveScreen = function(){
    if(this.player.isWalking && this.player.isScreenMove){
      if(this.player.playerPic.position.x > 430 && this.player.playerPic.position.x < 1891){
          this.map.screenPosition.x += this.player.walkDir.x;
      }
      if(this.player.playerPic.position.y > 268 && this.player.playerPic.position.y < 900){
          this.map.screenPosition.y += this.player.walkDir.y;
      }
    }
};

Map4.prototype.checkChangeMap = function(){
    var playerIndex = this.toArrayIndex(this.player.playerPic.position.x, this.player.playerPic.position.y, -1);
    var playerPos = this.player.playerPic.position;

    if(playerPos.x < 26 && playerPos.y > 32 && playerPos.y < 128){
        this.player.currentMap = 2;//to map3_1
```

```
        this.player.isChangeMap = true;
        this.player.drawChangeMapEff = true;
        this.map.screenPosition = {
            x: 1418,
            y: 0
        };
        this.player.setPosition({x: 84, y: 1}, this.map.screenPosition);
        this.map.clearProperties();
        this.map.map3.generateProperties();
    }
    if(playerPos.x > 2634 && playerPos.x < 2730 && playerPos.y > 1174 && this.checkChangeStage()){
        this.player.currentMap = 4;
        this.player.isChangeMap = true;
        this.player.drawChangeMapEff = true;
        this.map.screenPosition = {
          x: 0,
          y: 0
        };
        this.player.setPosition({x: 21, y: 2},this.map.screenPosition);
        this.map.clearProperties();
        this.player.exp += 3586208;
    }
};
```

33.Map5.js：

```
var Map5 = function(map) {
    this.map = map;
    this.player = this.map.player;
    this.isChangeMap = false;
    this.monsterAmount = 30;
    this.propertyAmount = 50;
}
Map5.prototype = Object.create(BaseMap.prototype);
Map5.prototype.constructor = BaseMap;

Map5.prototype.moveScreen = function(){
```

```
};

Map5.prototype.generateProperties = function(){
    var propertyType = 0;
    var row, col;
    for(var i = 0; i < this.propertyAmount; i++){
      while(true){
        row = Math.floor(Math.random() * 37);
        col = Math.floor(Math.random() * 85);
        if(this.map.mapArr[this.player.currentMap][row][col] > 5 &&
          this.map.mapArr[this.player.currentMap][row][col] < 8){
          break;
        }
      }
      if(i < 25){
        propertyType = 4;
      }
      else if(i < 50){
        propertyType = 5;
      }

      this.map.propertyArr[row][col] = propertyType;
    }
    console.log(this.map.propertyArr);
};
```

34.Medicine.js：

```
var Medicine = function(map){
   Property.call(this, map);
   this.load = function(){
      this.url1 = define.imagePathItem + 'medicine/Hematic_tonic.png';
      this.url2 = define.imagePathItem + 'medicine/Mana_potion.png';
      this.hpPotion = new Framework.Sprite(this.url1);
      this.mpPotoin = new Framework.Sprite(this.url2);
   }
```

```
    this.update = function(){
        this.hpPotion.update();
        this.mpPotoin.update();
    }
}
Medicine.prototype = Object.create(Property.prototype);
Medicine.prototype.constructor = Property;
```

35.MicroShield.js：

```
var MicroShield = function(map){
    this.map = map;
    this.player = map.player;
    this.abilityList = [0, 0.05, 0.1, 0.15, 0.2, 0.25];
    this.level = 0;
}

MicroShield.prototype = Object.create(Skill.prototype);
MicroShield.prototype.constructor = Skill;

MicroShield.prototype.update = function(){
    this.levelUp();
    this.buff();
};

MicroShield.prototype.levelUp = function(){
    for(var level = 8; level <= 12; level++){
      if(this.player.level === level){
        this.level = level - 7;
      }
    }
    if(this.player.level > 13){
      this.level = 5;
    }
};
```

```
MicroShield.prototype.buff = function(){
    if(this.player.level >= 8){
      this.player.def = this.abilityList[this.level];
    }
}
```

36.Monster.js：

```
var Monster = function(hp, atk, exp, player,map){
    this.hp = hp;
    this.atk = atk;
    this.exp = exp;
    this.currentHP = this.hp;
    this.currentAtk = this.atk;
    this.map = map;
    this.player = player;
    this.canGainExp = true;
    this.isDead = false;
    this.canBeAttacked = true;
    this.reviveTimeCounter = 0;
    this.isTouchingPlayer = false;
}

Monster.prototype.move = function(){

};

Monster.prototype.checkMonsterDead = function(){
    if(this.currentHP <= 0){
      this.currentAtk = 0;
      this.isDead = true;
      if(this.canGainExp){
        this.player.exp += this.exp;
        this.canGainExp = false;
      }
    }
};
```

```
Monster.prototype.revive = function(){
    if(this.isDead){
        this.reviveTimeCounter++;
    }
    if(this.reviveTimeCounter > 2000){
        this.isDead = false;
        this.currentHP = this.hp;
        this.currentAtk = this.atk;
        this.canGainExp = true;
        this.reviveTimeCounter = 0;
    }
};

Monster.prototype.printDamageReceived = function(ctx){
    var playerPos = this.player.playerPic.position;
    var screenPos = this.map.screenPosition;
    if(this.player.resistCounter != 270 && this.isTouchingPlayer){
        ctx.font = '16pt Arial';
        ctx.fillStyle = 'rgb(176, 50, 42)';
        if(this.map.mapType == 'scroll'){
          ctx.fillText('-' + Math.floor(this.atk*(1-this.player.def)), playerPos.x-screenPos.x-13, playerPos.y-screenPos.y-30);
        }
        else{
          ctx.fillText('-' + Math.floor(this.atk*(1-this.player.def)), playerPos.x-13, playerPos.y-30);
        }
    }
};

Monster.prototype.touchPlayer = function(){

};

Monster.prototype.attack = function(){

};
```

37.MyMenu.js：

```
var MyMenu = Framework.exClass(Framework.GameMainMenu , {
  load: function(){
    this.menu = new Framework.Sprite(define.imagePathCanvas + "CoverPhoto2.png");
    this.startbtn = new Framework.Sprite(define.imagePathStartPage + "start_btn.png");
    this.startbtn_hover = new Framework.Sprite(define.imagePathStartPage + "start_btn_hover.png");
    this.startbtn2 = new Framework.Sprite(define.imagePathStartPage + "start_btn2.png");
    this.startbtn_hover2 = new Framework.Sprite(define.imagePathStartPage + "start_btn_hover2.png");
    this.linkBtn = new Framework.Sprite(define.imagePath + "linkBtn.png");
    this.linkBtn_hover = new Framework.Sprite(define.imagePath + "linkBtn_hover.png");
    this.ballPic = new Framework.Sprite(define.imagePathEffect + 'iceball.png');

    this.iceBallArr = [];
    for(var i = 0; i < 50; i++){
     this.iceBall = new IceBall();
     this.iceBall.load();
     this.iceBall.init();
     this.iceBallArr.push(this.iceBall);
    }

    this.audio = new Framework.Audio({
      initSong1:{
        mp3: define.musicPath + 'Begin.mp3'
      }
    });
    this.audio.play({name: 'initSong1', loop: true});
  },

  loadingProgress: function(context){
    context.globalAlpha = 0.8;
    context.fillRect(0, 0, 1360, 700);
  },

  initialize: function(){
    //position
```

```
this.startbtn.position = {
    x : 685,
    y : 300
};
this.startbtn_hover.position = this.startbtn.position;

this.startbtn2.position = {
    x: 680,
    y: 390
};
this.startbtn_hover2.position = this.startbtn2.position;

this.linkBtn.position = {
    x: 1280,
    y: 50
};
this.linkBtn_hover.position = this.linkBtn.position;

this.menu.position = {
  x: Framework.Game.getCanvasWidth() / 2,
  y: Framework.Game.getCanvasHeight() / 2
};
this.ballPic.position = {
  x: -10,
  y: -10
}
//scale
this.menu.scale = 0.68;
this.startbtn.scale = 0.5;
this.startbtn_hover.scale = 0.5;
this.startbtn2.scale = 0.5;
this.startbtn_hover2.scale = 0.5;
this.linkBtn.scale = 0.35;
this.linkBtn_hover.scale = 0.35;
//boolean
this.isStartbtnHover = false;
```

```
    this.isStartbtn2Hover = false;
    this.isLinkBtnHover = false;

    this.counter = 0;
},

update: function(){
    this.ballPic.draw();
    this.ballPic.position.y--;
    for(var i in this.iceBallArr){
      this.iceBallArr[i].update();
    }
},

draw: function(ctx){
    this.menu.draw();
    for(var i in this.iceBallArr){
        this.iceBallArr[i].draw();
    }
    if(!this.isStartbtnHover){
        this.startbtn.draw();
    }
    else{
        this.startbtn_hover.draw();
    }
    if(!this.isLinkBtnHover){
        this.linkBtn.draw();
    }
    else {
        this.linkBtn_hover.draw();
    }
    if(!this.isStartbtn2Hover){
        this.startbtn2.draw();
    }
    else{
        this.startbtn_hover2.draw();
```

```
    }
},

mousemove:function(e){
    if(e.x >= 602 && e.x <= 758 && e.y >= 264 && e.y <= 318){
        this.isStartbtnHover = true;
    }
    else{
        this.isStartbtnHover = false;
    }
    if(e.x >= 1250 && e.x <= 1300 && e.y >= 26 && e.y <= 69){
        this.isLinkBtnHover = true;
    }
    else{
        this.isLinkBtnHover = false;
    }
    if(e.x >= 565 && e.x <= 787 && e.y >= 343 && e.y <= 418){
        this.isStartbtn2Hover = true;
    }
    else{
        this.isStartbtn2Hover = false;
    }
},

click:function(e){
    if(e.x >= 602 && e.x <= 758 && e.y >= 264 && e.y <= 318){
      console.log("Go to next level!");
      this.audio.stop('initSong1');
      Framework.Game.goToNextLevel();
    }
    if(e.x >= 565 && e.x <= 787 && e.y >= 343 && e.y <= 418){
      console.log("Go to SkillShowcase Map!");
      Framework.Game.goToLevel('skillShowcase');
    }

    if(e.x >= 1250 && e.x <= 1300 && e.y >= 26 && e.y <= 69){
```

```
            window.open("Game_Info/MainPage/index.html");
        }
    },
});
```

38.Player.js：

```
var Player = function(map){
    this.map = map;
    this.alpha = 1;
    this.delta = 0.02;

    this.load = function(){
        this.url1 = define.imagePathCharacter + 'bodyRes.png';
        this.url2 = define.imagePathCharacter + 'corpse.png';
        this.url3 = define.imagePathEffect + '006.png';
        this.playerPic = new Framework.AnimationSprite({url:this.url1, col:9, row:8, loop:false, speed:1});
        this.corpsePic = new Framework.Sprite(this.url2);
        this.basicAttack = new BasicAttack(map);
        this.basicAttack.load();
        this.teleport = new Teleport(map);
        this.teleport.load();
        this.vCut = new VCut(map);
        this.vCut.load();
        this.burstCut = new BurstCut(map);
        this.burstCut.load();
        this.crossCut = new CrossCut(map);
        this.crossCut.load();
        this.swordLight = new SwordLight(map);
        this.swordLight.load();
        this.light = new Light(map);
        this.light.load();
        this.microShield = new MicroShield(map);
        this.darkForce = new DarkForce(map);
        this.darkForce.load();
        this.frozenSword = new FrozenSword(map);
        this.frozenSword.load();
```

```
    this.potentialPower = new PotentialPower(map);
    this.potentialPower.load();
    this.heroSlash = new HeroSlash(map);
    this.heroSlash.load();
    this.bag = new Bag(this);
    this.bag.load();
    this.questUI = new QuestUI(map);
    this.questUI.load();
    this.skillUI = new SkillUI(map);
    this.skillUI.load();
    this.data = new WarriorData();

    this.audio = new Framework.Audio({
        initSong1:{
          mp3: define.musicPath + 'slash.mp3'
        },
        initSong2:{
          mp3: define.musicPath + 'VCut.mp3'
        },
        initSong3:{
          mp3: define.musicPath + 'BurstCut.mp3'
        },
        initSong4:{
          mp3: define.musicPath + 'thunder.mp3'
        }
    });
}

this.init = function(){
    this.currentMap = 0;
    this.walkDir = {x:0,y:0};
    this.bag.init();
    this.questUI.init();
    this.skillUI.init();
    this.isWalking = false;
    this.isAttacked = false;
```

```
this.isScreenMove = false;
this.isChangeMap = false;
this.drawChangeMapEff = false;
this.isMovingScreen = false;
this.isInCheatMode = false;

this.direction = 0;//0:up, 1:left, 2:down, 3:right
this.screenPosOffset = 0;
this.resistTime = 270;
this.resistCounter = this.resistTime;
this.playerPic.scale = 0.5;
//player basic info
this.money = 0;
this.exp = 0;
this.atk = 1;
for(var i = 1; i < this.level; i++){
  this.atk += this.data.atkTable[i];
}
this.def = 0;
this.healthPoint = this.data.hpTable[this.level-1];
this.manaPoint = this.data.mpTable[this.level-1];
this.actionPoiot = this.data.apTable[this.level-1];
this.currentHP = this.healthPoint;
this.currentMP = this.manaPoint;
this.currentAP = this.actionPoiot;
this.playerSpeed = 0;
//Skills
this.basicAttack.init();
this.vCut.init();
this.burstCut.init();
this.crossCut.init();
this.swordLight.init();
this.darkForce.init();
this.frozenSword.init();
this.potentialPower.init();
this.heroSlash.init();
```

```
}

this.setPlayerLevel = function(level){
    this.level = level;
}

this.setPosition = function(pos,screenPos){
    this.position = pos;
    this.playerPic.position = {
      x:this.position.x * 32 + 16 ,
      y:this.position.y * 32 + 16
    };
}

this.update = function(){
    this.corpsePic.position = this.playerPic.position;
    if(this.isKeyPress &&
      this.map.canMove(this.playerPic.position.x + this.walkDir.x, this.playerPic.position.y +this.walkDir.y, this.direction) &&
      this.currentAP > 0 && this.currentHP > 0){
       this.isScreenMove = true;
       this.walk(this.walkDir);
    }
    else{
       this.isScreenMove = false;
    }

    if(this.currentAP < this.actionPoiot){
       this.currentAP += 1;
    }

    if(this.resistCounter > 0 && this.isAttacked){
       this.resistCounter--;
    }

    if(this.isKeyPress){
      if(this.resistCounter < 61){
```

```
      this.playerPic.start({from: (this.direction+4) * 9, to: (this.direction+4) * 9 + 8, loop: true});
    }
    else{
      this.playerPic.start({from: this.direction * 9, to: this.direction * 9 + 8, loop: true});
    }
  }
  else{
     this.playerPic.start({from: this.direction * 9, to: this.direction * 9, loop: true});
  }
  // if(this.isMovingScreen && this.playerPic.position.y > 300 && this.playerPic.position.y < 1100){
  //     this.screenPosOffset++;
  //     this.map.screenPosition.y += this.walkDir.y;
  // }
  //
  // if(this.screenPosOffset > 32){
  //     this.screenPosOffset = 0;
  //     this.isMovingScreen = false;
  // }
  this.basicAttack.update();
  this.vCut.update();
  this.burstCut.update();
  this.crossCut.update();
  this.swordLight.update();
  this.frozenSword.update();
  this.frozenSword.effectPic.update();
  this.light.update();
  this.microShield.update();
  this.darkForce.update();
  this.potentialPower.update();
  this.heroSlash.update();

  this.playerPic.update();
  this.teleport.update();
  this.bag.update();
  this.questUI.update();
  this.skillUI.update();
```

```
    this.checkLevelUp();

    if(this.isInCheatMode){
        this.def = 1;
    }
}
this.walk = function(movePos){
    if(this.isWalking){
      if(movePos.x > 0){//Move Right
          this.direction = 3;
      }
      else if(movePos.x < 0){//Move Left
          this.direction = 1;
      }
      if(movePos.y > 0){//Move Down
          this.direction = 2;
      }
      else if(movePos.y < 0){//Move Up
          this.direction = 0;
      }
      this.playerPic.position.x += movePos.x;
      this.playerPic.position.y += movePos.y;
      if(this.currentAP >= 0){
          this.currentAP -= 2;
      }
    }
    else{
      this.playerPic.stop();
    }
}
this.checkLevelUp = function(){
    if(this.exp >= this.data.expTable[this.level-1]){
        this.exp -= this.data.expTable[this.level-1];
        this.healthPoint = this.data.hpTable[this.level];
        this.manaPoint = this.data.mpTable[this.level];
```

```
        this.actionPoiot = this.data.apTable[this.level];
        this.currentHP = this.healthPoint;
        this.currentMP = this.manaPoint;
        this.currentAP = this.actionPoiot;
        this.atk += this.data.atkTable[this.level-1];
        this.level++;
    }
}

this.isKeyPress = false;
this.key = "";
this.keydown = function(e, list){
    var playerPos = this.playerPic.position;

    if(e.key === 'Right'){
      this.key = "Right";
      this.isKeyPress = true;
      this.walkDir = {x: 2 + this.playerSpeed, y: 0};
      this.direction = 3;
      this.isWalking = true;
    }
    if(e.key === 'Left'){
      this.key = "Left";
      this.isKeyPress = true;
      this.walkDir = {x: -2 - this.playerSpeed, y: 0};
      this.direction = 1;
      this.isWalking = true;
    }
    if(e.key === 'Up'){
      this.key = "Up";
      this.isKeyPress = true;
      this.walkDir = {x: 0, y: -2 - this.playerSpeed};
      this.direction = 0;
      this.isWalking = true;
    }
    if(e.key === 'Down'){
```

```
  this.key = "Down";
  this.isKeyPress = true;
  this.walkDir = {x: 0, y: 2 + this.playerSpeed};
  this.direction = 2;
  this.isWalking = true;
}
if(e.key === 'X'){
  this.key = 'X';
  this.isKeyPress = false;

  if(this.basicAttack.canAttack){
    this.basicAttack.isAttack = true;
    this.basicAttack.isDrawed = true;
    this.basicAttack.pic.position = {
       x: this.playerPic.position.x + this.walkDir.x * 15,
       y: this.playerPic.position.y + this.walkDir.y * 15
    };
    this.basicAttack.attack(playerPos.x + this.walkDir.x * 12, playerPos.y + this.walkDir.y * 12, this.direction);
    // this.audio.play({name: 'initSong1'});
  }

}

if(e.key === 'C'){
    this.key = 'C';
    this.isKeyPress = false;

    if(this.currentMP >= this.vCut.mpCost && this.vCut.canAttack && this.level >= 5){
      this.currentMP -= this.vCut.mpCost;
      this.vCut.isAttack = true;
      this.vCut.isDrawed = true;
      this.vCut.pic.position = {
        x: this.playerPic.position.x + this.walkDir.x * 32,
        y: this.playerPic.position.y + this.walkDir.y * 32
      };
      this.vCut.attack(playerPos.x, playerPos.y, this.direction);
```

```
      // this.audio.play({name: 'initSong2'});
    }
}

if(e.key === 'V'){
    this.key = 'V';
    this.isKeyPress = false;

    if(this.currentMP >= this.burstCut.mpCost && this.burstCut.canAttack && this.level >= 10){
      this.burstCut.isDrawed = true;
      this.currentMP -= this.burstCut.mpCost;
      this.burstCut.isAttack = true;
      this.burstCut.pic.position = {
        x: this.playerPic.position.x + this.walkDir.x * 32,
        y: this.playerPic.position.y + this.walkDir.y * 32
      };
      this.burstCut.attack(playerPos.x, playerPos.y, this.direction);
      // this.audio.play({name: 'initSong3'});
    }
}

if(e.key === 'B'){
    this.key = 'B';
    this.isKeyPress = false;

    if(this.currentMP >= this.swordLight.mpCost && this.swordLight.canAttack && this.level >= 15){
      this.swordLight.isDrawed = true;
      this.currentMP -= this.swordLight.mpCost;
      this.swordLight.isAttack = true;
      this.light.isAttack = true;
      this.swordLight.pic.position = {
        x: this.playerPic.position.x + this.walkDir.x * 32,
        y: this.playerPic.position.y + this.walkDir.y * 32
      };
      this.light.pic.position = this.swordLight.pic.position;
      switch (this.direction) {
```

```
        case 0:
          this.light.direction = 0;
          break;
        case 1:
          this.light.direction = 1;
          break;
        case 2:
          this.light.direction = 2;
          break;
        case 3:
          this.light.direction = 3;
          break;
      }

      this.swordLight.attack(playerPos.x, playerPos.y, this.direction);
      // this.audio.play({name: 'initSong3'});
    }
  }

  if(e.key === 'Space'){
    this.key = 'Space';
    this.isKeyPress = false;

    if(this.currentHP >= this.crossCut.hpCost && this.crossCut.canAttack && this.level >= 18){
      this.crossCut.isDrawed = true;
      this.crossCut.isAttack = true;
      this.currentHP -= this.crossCut.hpCost;

      this.crossCut.attack(playerPos.x, playerPos.y, this.direction);
    }
  }

  if(e.key === 'ctrlKey'){
    this.key = 'ctrlKey';
    this.isKeyPress = false;
```

```
  if(this.currentMP >= this.frozenSword.mpCost && this.frozenSword.canAttack && this.level >= 25){
    this.frozenSword.isDrawed = true;
    this.currentMP -= this.frozenSword.mpCost;
    this.frozenSword.isAttack = true;

    this.frozenSword.attack();
  }
}

if(e.key === 'shiftKey'){
  this.key = 'shiftKey';
  this.isKeyPress = false;

  if(this.currentMP >= this.heroSlash.mpCost && this.heroSlash.canAttack &&
     this.potentialPower.isUsingSkill && this.level >= 20){
    this.currentMP -= this.heroSlash.mpCost;
    this.heroSlash.isDrawed = true;
    this.heroSlash.isAttack = true;
    this.heroSlash.pic.position = {
      x: this.playerPic.position.x + this.walkDir.x * 32,
      y: this.playerPic.position.y + this.walkDir.y * 32
    };
    switch (this.direction) {
      case 0:
        this.heroSlash.direction = 0;
        break;
      case 1:
        this.heroSlash.direction = 1;
        break;
      case 2:
        this.heroSlash.direction = 2;
        break;
      case 3:
        this.heroSlash.direction = 3;
        break;
    }
```

```
        }
      }

      if(e.key === 'A' && (this.map.mapType !== 'scroll' || this.currentMap === 4)){
        this.key = 'A';
        this.isKeyPress = false;
        if(this.map.canMove(this.playerPic.position.x + this.walkDir.x*32, this.playerPic.position.y + this.walkDir.y*32,
this.direction) &&
          this.teleport.canAttack){
          this.isMovingScreen = true;
          this.teleport.isDrawed = true;
          this.teleport.isAttack = true;

          this.teleport.move(this.walkDir);
        }
      }

      if(e.key === 'S'){
        this.key = 'S';
        this.isKeyPress = false;

        if(this.currentMP >= this.darkForce.mpCost && this.darkForce.canAttack && this.level >= 13){
          this.currentMP -= this.darkForce.mpCost;
          this.darkForce.buff();
          this.darkForce.isDrawed = true;
          this.darkForce.isAttack = true;
        }
      }

      if(e.key === 'D'){
        this.key = 'D';
        this.isKeyPress = false;

        if(this.currentMP >= Math.round(this.manaPoint / 2) && this.currentHP > Math.round(this.healthPoint / 2) &&
          this.potentialPower.canAttack && this.level >= 13){
          this.potentialPower.isDrawed = true;
```

```
      this.currentMP -= Math.round(this.manaPoint / 2);
      this.currentHP -= Math.round(this.healthPoint / 2);
      this.potentialPower.isAttack = true;

      this.potentialPower.buff();
    }
}

if(e.key === 'Z'){//pick up
    this.key = 'Z';
    this.isKeyPress = false;
    this.pickUpProperty();
}

if(e.key === 'Q'){//Using HP potion
    this.key = 'Q';
    this.isKeyPress = false;
    if(this.currentMap !== 3){
      this.useNormalHpPotion();
    }
}

if(e.key === 'W'){//Using MP potion
    this.key = 'W';
    this.isKeyPress = false;
    this.useNormalMpPotion();
}

if(e.key === '1'){
    if(this.isInCheatMode){
      this.isInCheatMode = false;
    }
    else{
      this.isInCheatMode = true;
    }
    this.atk = 99999;
```

```
    }

    if(e.key === '2'){
        this.currentHP = this.healthPoint;
        this.currentMP = this.manaPoint;
    }

    if(e.key === '3'){
        this.exp += this.data.expTable[this.level-1];
    }

    this.playerPic.start({from: this.direction * 9, to: this.direction * 9, loop: true});
    this.playerPic.update();

  }

  this.pickUpProperty = function(){
    var playerPos = this.map.toArrayIndex(this.playerPic.position.x, this.playerPic.position.y, this.direction);//Transform the player's position to array index
    var propIndex = this.map.propertyArr[playerPos.row][playerPos.col];

    if(propIndex !== 0){
      var propObj = {
        type: propIndex,
        number: 1
      };

      if(!this.bag.isPropertyExist(propObj)){
         this.bag.propertyArr.push(propObj);
      }
      else{
        this.bag.addProperty(propObj);
      }

      if(this.map.mapArr[this.currentMap][playerPos.row][playerPos.col] !== 1){
        this.map.mapArr[this.currentMap][playerPos.row][playerPos.col] = 0;
```

```
        }
        this.map.propertyArr[playerPos.row][playerPos.col] = 0;
    }
}

this.useNormalHpPotion = function(){
    if(this.map.mapType !== 'scroll'){
      this.currentHP = this.healthPoint;
    }
    for(var i = 0; i < this.bag.propertyArr.length; i++){
        if(this.bag.propertyArr[i].type === 4 && this.bag.propertyArr[i].number > 0){
            this.bag.propertyArr[i].number--;
            if(this.currentHP/this.healthPoint >= 0.75){
                this.currentHP = this.healthPoint;
            }
            else{
                this.currentHP += Math.round(this.healthPoint * 0.25);
            }
        }
    }
}

this.useNormalMpPotion = function(){
    if(this.map.mapType !== 'scroll'){
      this.currentMP = this.manaPoint;
    }
    for(var i = 0; i < this.bag.propertyArr.length; i++){
        if(this.bag.propertyArr[i].type === 5 && this.bag.propertyArr[i].number > 0){
            this.bag.propertyArr[i].number--;
            if(this.currentMP/this.manaPoint >= 0.75){
                this.currentMP = this.manaPoint;
            }
            else{
                this.currentMP += Math.round(this.manaPoint * 0.25);
            }
        }
```

```
    }
}

this.keyup = function(e, list){
    if(e.key){
        if(this.key === e.key)
        {
            this.isKeyPress = false;
            this.isWalking = false;
        }
    }
}

this.draw = function(ctx){
    ctx.fillStyle = 'black';
                                                ctx.font = 'bold 28px freeScript ';
                                                ctx.fillText('HP',23,620);
                                                ctx.fillText(this.currentHP + '/' + this.healthPoint,73,650);
    ctx.fillText('MP',250,620);
                                                ctx.fillText(this.currentMP + '/' + this.manaPoint,298,650);
    ctx.fillText('AP',480,620);
                                                ctx.fillText(this.currentAP + '/' + this.actionPoiot,523,650);
    ctx.fillText('Level  ' + this.level,700,620);
    ctx.fillText('Atk : ' + Math.round(this.atk), 650, 670);
    ctx.fillText('Def : ' + this.def * 100 + '%', 810, 670);

    ctx.fillText('EXP',23,670);
    ctx.fillText(this.exp + ' / ' + this.data.expTable[this.level-1], 280, 690);

    ctx.fillStyle = 'rgba(23, 24, 24, 0.50)';
    ctx.fillRect(63, 605, 150, 20);
    ctx.fillRect(290, 605, 150, 20);
    ctx.fillRect(520, 605, 150, 20);
    ctx.fillRect(73, 655, 550, 20);

    ctx.fillStyle = 'rgb(159, 29, 6)';
```

```
        ctx.fillRect(63, 605, 150*(this.currentHP/this.healthPoint), 20);
        ctx.fillStyle = 'rgb(114, 62, 159)';
        ctx.fillRect(290, 605, 150*(this.currentMP/this.manaPoint), 20);
        ctx.fillStyle = 'rgb(234, 236, 139)';
        ctx.fillRect(520, 605, 150*(this.currentAP/this.actionPoiot), 20);
        ctx.fillStyle = 'rgba(10, 213, 152, 0.86)';
        ctx.fillRect(73, 655, 550*(this.exp/this.data.expTable[this.level-1]), 20);

        this.basicAttack.draw();
        this.teleport.draw();
        this.vCut.draw();
        this.burstCut.draw();
        this.crossCut.draw();
        this.darkForce.draw();
        this.swordLight.draw();
        this.light.draw();
        this.frozenSword.draw();
        this.potentialPower.draw();
        this.heroSlash.draw();

        this.bag.draw(ctx);
        this.questUI.draw(ctx);
        this.skillUI.draw(ctx);

        if(this.drawChangeMapEff){
          this.alpha -= this.delta;
          ctx.globalAlpha = this.alpha;
          ctx.fillStyle = "black";
          ctx.fillRect(0, 0, 1360, 700);
        }
        if(this.alpha <= 0){
          this.drawChangeMapEff = false;
          this.alpha = 1;
        }
    }
};
```

39.PlayerData.js：

```
var PlayerData = function(){
    this.expTable = [];
    this.hpTable = [];
    this.mpTable = [];
    this.apTable = [];
    this.atkTable = [];
    this.createExpTable();
    this.createHpTable();
    this.createMpTable();
    this.createApTable();
    this.createAtkTable();
}

PlayerData.prototype.createExpTable = function(){
    for(var level = 1; level < 100; level++){
      var exp = 0;
      if(level < 21){
        exp = Math.round(Math.pow(level,3) + 9);
      }
      else if(level >= 21 && level < 41){
        exp = Math.round(Math.pow(level,3) * 1.5);
      }
      else if(level >= 41 && level < 61){
        exp = Math.round(Math.pow(level,3.3));
      }
      else if(level >= 61 && level < 81){
        exp = Math.round(Math.pow(level,3.5));
      }
      else{
        exp = Math.round(Math.pow(level,3.5) * 1.3);
      }
      this.expTable.push(exp);
    }
};
```

```
PlayerData.prototype.createHpTable = function(){

};

PlayerData.prototype.createMpTable = function(){

};

PlayerData.prototype.createApTable = function(){

};

PlayerData.prototype.createAtkTable = function(){

};
```

40.PotentialPower.js：

```
var PotentialPower = function(map){
    Skill.call(this, 0, 30, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
        this.url = define.imagePathEffect + 'Warrior/007.png';
        this.url2 = define.imagePathEffect + 'fire_003.png';
        this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:4, loop:true, speed:5});
        this.effectPic = new Framework.AnimationSprite({url:this.url2, col:5, row:8, loop:true, speed:15});
    }

    this.init = function(){
        this.effectPic.scale = 0.7;
        this.durationTime = 1000;
        this.timeCounter = 0;
        this.isUsingSkill = false;
    }

    this.draw = function(){
      if(this.isDrawed && this.timeCounter <= this.durationTime && this.timeCounter !== 0){
```

```
      this.isUsingSkill = true;
      this.pic.position = {
        x: this.player.playerPic.position.x,
        y: this.player.playerPic.position.y - 20
      };
      this.effectPic.position = {
        x: this.player.playerPic.position.x,
        y: this.player.playerPic.position.y - 20
      };
      if(this.map.mapType === 'scroll'){
        this.map.drawOtherObj(this.pic);
        this.map.drawOtherObj(this.effectPic);
      }
      else{
        this.pic.draw();
        this.effectPic.draw();
      }
    }
    else{
      this.isUsingSkill = false;
    }
  }
}
PotentialPower.prototype = Object.create(Skill.prototype);
PotentialPower.prototype.constructor = Skill;

PotentialPower.prototype.update = function(){
  var playerPos = this.player.playerPic.position;
  this.pic.update();
  this.effectPic.update();
  if(this.isAttack && this.coolDownCounter > 0){
    this.timeCounter++;
    this.coolDownCounter--;
  }

  if(this.timeCounter % 30 === 0 && this.timeCounter > 0){
```

```
    this.pic.start({from:1, to:16, loop:false});
  }

  if(this.coolDownCounter < this.coolDownTime * 100){//cool down
    this.canAttack = false;
  }

  if(this.coolDownCounter === 0){//skill ready
    this.isAttack = false;
    this.canAttack = true;
    this.coolDownCounter = this.coolDownTime * 100;
    this.timeCounter = 0;
  }

  if(this.timeCounter !== 0 && this.timeCounter <= this.durationTime / 2){
    this.player.def = 1;
  }

  if(this.timeCounter === this.durationTime){
    this.player.atk /= 1.5;
  }
};

PotentialPower.prototype.buff = function(){
  this.player.atk *= 1.5;

  this.effectPic.start({from:0, to:39, loop:false});
};
```

41.Property.js：

```
var Property = function(map){
  this.map = map;
}
Property.prototype.use = function(){

};
```

42.Quest1.js：

```
var Quest1 = function(map, player, questUI){
    QuestInterface.call(this, map, player, questUI);
    this.isTrigger = true;
    this.questNum = 0;

    this.draw = function(ctx){
      if(this.questUI.uiPic.scale >= 0.5){
        if(this.isHover === false){
          this.title.draw();
        }
        else{
          this.titleHover.draw();
        }
      }

      if(this.drawContent){
        this.content.draw();
        if(this.content.scale >= 0.37){
          if(!this.isSubmitBtnHover){
            this.submitBtn.draw();
          }
          else{
            this.submitBtnHover.draw();
          }
        }
      }
      else {
        this.content.scale = 0.1;
      }
    }

    this.update = function(){
      if(this.content.scale < 0.37){
        this.content.scale += 0.0045;
```

```
    }
    this.closeContentPos = {
      x: this.content.position.x + 159,
      y: this.content.position.y - 105
    };
  }
}

Quest1.prototype = Object.create(QuestInterface.prototype);
Quest1.prototype.constructor = QuestInterface;

Quest1.prototype.load = function(){
  this.title = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest1/title.png');
  this.titleHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest1/title_hover.png');
  this.content = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest1/content.png');
  this.rewardInfo = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest1/rewardInfo.png');
}

Quest1.prototype.init = function(){
  this.title.scale = 0.45;
  this.titleHover.scale = this.title.scale;
  this.content.scale = 0.05;//to 0.37
  this.submitBtn.scale = 0.35;
  this.submitBtnHover.scale = this.submitBtn.scale;
  this.rewardInfo.scale = 0.38;

  this.title.position = {
    x: 620,
    y: 150
  };
  this.titleHover.position = this.title.position;
  this.content.position = {
    x: 230,
    y: 304
  };
  this.submitBtn.position ={
```

```
    x: this.content.position.x - 20,
    y: this.content.position.y + 70
  };
  this.submitBtnHover.position = this.submitBtn.position;

  this.rewardInfo.position = {
    x: 630,
    y: 320
  };
};

Quest1.prototype.checkCompleted = function(){
  var bag = this.player.bag;

  if(bag.getPropertyNumber(1) >= 1 && bag.getPropertyNumber(2) >= 1 && bag.getPropertyNumber(3) >= 1){
    this.isCompleted = true;
  }
};

Quest1.prototype.getReward = function(){
  var propertyArr = this.player.bag.propertyArr;
  var propObj = {
    type: 4,
    number: 5
  };

  if(this.isCompleted && this.canReceiveReward){
    this.player.exp += 15;
    if(!this.player.bag.isPropertyExist(propObj)){
      this.player.bag.propertyArr.push(propObj);
    }
    else{
      this.player.bag.addProperty(propObj);
    }
    this.canReceiveReward = false;
    this.takeAwayProperties(propertyArr);
```

```
    }
};

Quest1.prototype.takeAwayProperties = function(propertyArr){
    for(var i = 0; i < propertyArr.length; i++){
        if(propertyArr[i].type === 1 || propertyArr[i].type === 2 || propertyArr[i].type === 3){
            this.player.bag.propertyArr[i].number--;
        }
    }
};
```

43.Quest2.js：

```
var Quest2 = function(map, player, questUI){
    QuestInterface.call(this, map, player, questUI);

    this.draw = function(ctx){
      if(this.questUI.uiPic.scale >= 0.5){
        if(this.isHover === false){
          this.title.draw();
        }
        else{
          this.titleHover.draw();
        }
      }

      if(this.drawContent){
        this.content.draw();
        if(this.content.scale >= 0.37){
          if(!this.isSubmitBtnHover){
            this.submitBtn.draw();
          }
          else{
            this.submitBtnHover.draw();
          }
        }
      }
```

```
    else {
      this.content.scale = 0.05;
    }

    if(this.isTrigger && this.alertCounter < 180){
      this.alertCounter++;
      this.alert.draw();
    }

  }

  this.update = function(){
    this.setTriggeringCondition();
    this.processingQuest();

    if(this.content.scale < 0.37){
      this.content.scale += 0.0045;
    }

    this.titleHover.position = this.title.position;
    this.content.position = {//width = 420
      x: 230,
      y: 304
    };
    this.submitBtn.position ={
      x: this.content.position.x - 20,
      y: this.content.position.y + 70
    };
    this.closeContentPos = {
      x: this.content.position.x + 159,
      y: this.content.position.y - 50
    };
  }

}
```

```
Quest2.prototype = Object.create(QuestInterface.prototype);
Quest2.prototype.constructor = QuestInterface;

Quest2.prototype.load = function(){
    this.title = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest2/title.png');
    this.titleHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest2/title_hover.png');
    this.content = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest2/content.png');
    this.rewardInfo = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest2/rewardInfo.png');
}

Quest2.prototype.init = function(){
    this.title.scale = 0.45;
    this.titleHover.scale = this.title.scale;
    this.content.scale = 0.05;
    this.submitBtn.scale = 0.35;
    this.rewardInfo.scale = 0.38;

    this.rewardInfo.position = {
      x: 630,
      y: 320
    };

    this.monsterKilledCounter = 0;
};

Quest2.prototype.drawInfo = function(ctx){
  if(!this.isCompleted && this.isTrigger && this.questUI.isHover){
      ctx.fillStyle = 'black';
      ctx.font = 'bold 22px freeScript ';
      ctx.fillText('Dragonfly: ' + this.monsterKilledCounter + ' / 10' ,1080, 278);
  }
};

Quest2.prototype.setTriggeringCondition = function(){
    if(this.player.level >= 2 && this.questUI.quest1.isCompleted){
        this.isTrigger = true;
```

```
    }
};

Quest2.prototype.processingQuest = function(){
    var monsterArr = this.map.monsterArr;
    if(this.isTrigger){
      for(var i = 0; i < monsterArr.length; i++){
        if(monsterArr[i].name === 'dragonfly' && monsterArr[i].isDead){
          if(monsterArr[i].reviveTimeCounter > 0 && monsterArr[i].reviveTimeCounter < 2){
            this.monsterKilledCounter++;
          }
        }
      }
    }
};

Quest2.prototype.checkCompleted = function(){
    if(this.monsterKilledCounter >= 10){// >= 10
      this.isCompleted = true;
    }
}

Quest2.prototype.getReward = function(){
    var propertyArr = this.player.bag.propertyArr;
    var propObj = {
       type: 4,
       number: 5
    };

    if(this.isCompleted && this.canReceiveReward){
      this.player.exp += 30;
      this.player.money += 10;
      if(!this.player.bag.isPropertyExist(propObj)){
          this.player.bag.propertyArr.push(propObj);
      }
      else{
```

```
        this.player.bag.addProperty(propObj);
      }
      this.canReceiveReward = false;
    }
};

Quest2.prototype.takeAwayProperties = function(){

};
```

44.Quest3.js：

```
var Quest3 = function(map, player, questUI){
    QuestInterface.call(this, map, player, questUI);
    this.draw = function(ctx){
      if(this.questUI.uiPic.scale >= 0.5){
        if(this.isHover === false){
          this.title.draw();
        }
        else{
          this.titleHover.draw();
        }
      }

      if(this.drawContent){
        this.content.draw();
        if(this.content.scale >= 0.37){
          if(!this.isSubmitBtnHover){
            this.submitBtn.draw();
          }
          else{
            this.submitBtnHover.draw();
          }
        }
      }
      else {
        this.content.scale = 0.05;
```

```
    }

    if(this.isTrigger && this.alertCounter < 180){
      this.alertCounter++;
      this.alert.draw();
    }
  }

  this.update = function(){
    this.setTriggeringCondition();
    this.processingQuest();
    if(this.content.scale < 0.37){
      this.content.scale += 0.0045;
    }
    this.closeContentPos = {
      x: this.content.position.x + 159,
      y: this.content.position.y - 105
    };
  }
}

Quest3.prototype = Object.create(QuestInterface.prototype);
Quest3.prototype.constructor = QuestInterface;

Quest3.prototype.load = function(){
  this.title = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest3/title.png');
  this.titleHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest3/title_hover.png');
  this.content = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest3/content.png');
  this.rewardInfo = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest3/rewardInfo.png');
}

Quest3.prototype.init = function(){
  this.title.scale = 0.45;
  this.titleHover.scale = this.title.scale;
  this.content.scale = 0.05;//to 0.37
  this.submitBtn.scale = 0.35;
```

```
    this.rewardInfo.scale = 0.38;

    this.content.position = {//width = 420
      x: 230,
      y: 304
    };

    this.submitBtn.position ={
      x: this.content.position.x - 20,
      y: this.content.position.y + 70
    };

    this.rewardInfo.position = {
      x: 630,
      y: 320
    };

    this.monsterKilledCounter = 0;
};

Quest3.prototype.drawInfo = function(ctx){
  if(!this.isCompleted && this.isTrigger && this.questUI.isHover){
      ctx.fillStyle = 'black';
      ctx.font = 'bold 22px freeScript ';
      ctx.fillText('Ant: ' + this.monsterKilledCounter + ' / 3' ,1080, 278);
  }
};

Quest3.prototype.setTriggeringCondition = function(){
    if(this.player.level >= 4 && this.questUI.quest2.isCompleted){
        this.isTrigger = true;
    }
};

Quest3.prototype.processingQuest = function(){
    var monsterArr = this.map.monsterArr;
```

```
    if(this.isTrigger){
      for(var i = 0; i < monsterArr.length; i++){
        if(monsterArr[i].name === 'ant' && monsterArr[i].isDead){
          if(monsterArr[i].reviveTimeCounter > 0 && monsterArr[i].reviveTimeCounter < 2){
            this.monsterKilledCounter++;
            console.log(this.monsterKilledCounter);
          }
        }
      }
    }
};

Quest3.prototype.checkCompleted = function(){
    var propertyArr = this.player.bag.propertyArr;
    var conditionCount = 0;

    for(var i = 0; i < propertyArr.length; i++){
      if(propertyArr[i].type === 1 || propertyArr[i].type === 2 || propertyArr[i].type === 3){
        if(propertyArr[i].number >= 3){
          conditionCount++;
        }
      }
    }
    if(this.monsterKilledCounter >= 3 && conditionCount === 3){//>= 3 && === 3
      this.isCompleted = true;
    }
}

Quest3.prototype.getReward = function(){
    var propertyArr = this.player.bag.propertyArr;
    var propObj = {
      type: 5,
      number: 5
    };
    var questItem = {
      type: 30,
```

```
        number: 1
    };

    if(this.isCompleted && this.canReceiveReward){
      this.player.exp += 200;
      this.player.money += 30;
      this.player.bag.propertyArr.push(questItem);
      if(!this.player.bag.isPropertyExist(propObj)){
          this.player.bag.propertyArr.push(propObj);
      }
      else{
        this.player.bag.addProperty(propObj);
      }

      this.canReceiveReward = false;
      this.takeAwayProperties(propertyArr);
    }
}

Quest3.prototype.takeAwayProperties = function(propertyArr){
    for(var i = 0; i < propertyArr.length; i++){
        if(propertyArr[i].type === 1 || propertyArr[i].type === 2 || propertyArr[i].type === 3){
          this.player.bag.propertyArr[i].number-=3;
        }
    }
};
```

45.Quest4.js：

```
var Quest4 = function(map, player, questUI){
    QuestInterface.call(this, map, player, questUI);
    this.draw = function(ctx){
      if(this.questUI.uiPic.scale >= 0.5){
        if(this.isHover === false){
          this.title.draw();
        }
        else{
```

```
      this.titleHover.draw();
    }
  }

  if(this.drawContent){
    this.content.draw();
    if(this.content.scale >= 0.37){
      if(!this.isSubmitBtnHover){
        this.submitBtn.draw();
      }
      else{
        this.submitBtnHover.draw();
      }
    }
  }
  else {
    this.content.scale = 0.05;
  }

  if(this.isTrigger && this.alertCounter < 180){
    this.alertCounter++;
    this.alert.draw();
  }
}

this.update = function(){
  this.setTriggeringCondition();
  this.processingQuest();

  if(this.content.scale < 0.37){
    this.content.scale += 0.0045;
  }

  this.titleHover.position = this.title.position;
  this.content.position = {//width = 420
    x: 230,
```

```
      y: 304
    };
    this.submitBtn.position ={
      x: this.content.position.x - 20,
      y: this.content.position.y + 70
    };
    this.closeContentPos = {
      x: this.content.position.x + 159,
      y: this.content.position.y - 50
    };
  }
}

Quest4.prototype = Object.create(QuestInterface.prototype);
Quest4.prototype.constructor = QuestInterface;

Quest4.prototype.load = function(){
  this.title = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest4/title.png');
  this.titleHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest4/title_hover.png');
  this.content = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest4/content.png');
  this.rewardInfo = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest4/rewardInfo.png');
}

Quest4.prototype.init = function(){
  this.title.scale = 0.45;
  this.titleHover.scale = this.title.scale;
  this.content.scale = 0.05;
  this.submitBtn.scale = 0.35;
  this.rewardInfo.scale = 0.38;

  this.rewardInfo.position = {
    x: 630,
    y: 320
  };

  this.antKilledCounter = 0;
```

```
    this.spiderKilledCounter = 0;
};

Quest4.prototype.drawInfo = function(ctx){
  if(!this.isCompleted && this.isTrigger && this.questUI.isHover){
      ctx.fillStyle = 'black';
      ctx.font = 'bold 22px freeScript ';
      ctx.fillText('Ant: ' + this.antKilledCounter + ' / 10' ,1080, 278);
      ctx.fillText('Spider: ' + this.spiderKilledCounter + ' / 3' ,1080, 306);
  }
};

Quest4.prototype.setTriggeringCondition = function(){
    if(this.player.level >= 6 && this.questUI.quest3.isCompleted){
        this.isTrigger = true;
    }
};

Quest4.prototype.processingQuest = function(){
    var monsterArr = this.map.monsterArr;
    if(this.isTrigger){
      for(var i = 0; i < monsterArr.length; i++){
        if(monsterArr[i].name === 'ant' && monsterArr[i].isDead){
          if(monsterArr[i].reviveTimeCounter > 0 && monsterArr[i].reviveTimeCounter < 2){
            this.antKilledCounter++;
          }
        }
        else if(monsterArr[i].name === 'spider' && monsterArr[i].isDead){
          if(monsterArr[i].reviveTimeCounter > 0 && monsterArr[i].reviveTimeCounter < 2){
            this.spiderKilledCounter++;
          }
        }
      }
    }
};
```

```
Quest4.prototype.checkCompleted = function(){
    if(this.antKilledCounter >= 10 && this.spiderKilledCounter >= 3){// >= 10
      this.isCompleted = true;
    }
}

Quest4.prototype.getReward = function(){
    var propertyArr = this.player.bag.propertyArr;
    var hpPotion = {
       type: 4,
       number: 5
    };
    var mpPotion = {
      type: 5,
      number: 5
    };
    var zircon = {
      type: 11,
      number: 3
    };
    var itemArr = [];
    itemArr.push(hpPotion);
    itemArr.push(mpPotion);
    itemArr.push(zircon);

    if(this.isCompleted && this.canReceiveReward){
      this.player.exp += 600;
      this.player.money += 60;
      for(var i = 0; i < itemArr.length; i++){
        if(!this.player.bag.isPropertyExist(itemArr[i])){
            this.player.bag.propertyArr.push(itemArr[i]);
        }
        else{
           this.player.bag.addProperty(itemArr[i]);
        }
      }
```

```
    this.canReceiveReward = false;
  }
};
Quest4.prototype.takeAwayProperties = function(){

};
```

46.Quest5.js：

```
var Quest5 = function(map, player, questUI){
  QuestInterface.call(this, map, player, questUI);
  this.draw = function(ctx){
    if(this.questUI.uiPic.scale >= 0.5){
      if(this.isHover === false){
        this.title.draw();
      }
      else{
        this.titleHover.draw();
      }
    }

    if(this.drawContent){
      this.content.draw();
      if(this.content.scale >= 0.37){
        if(!this.isSubmitBtnHover){
          this.submitBtn.draw();
        }
        else{
          this.submitBtnHover.draw();
        }
      }
    }
    else {
      this.content.scale = 0.05;
    }

    if(this.isTrigger && this.alertCounter < 180){
```

```
        this.alertCounter++;
        this.alert.draw();
      }
    }

    this.update = function(){
      this.setTriggeringCondition();
      this.processingQuest();
      if(this.content.scale < 0.37){
        this.content.scale += 0.0045;
      }
      this.closeContentPos = {
        x: this.content.position.x + 159,
        y: this.content.position.y - 105
      };
    }
}

Quest5.prototype = Object.create(QuestInterface.prototype);
Quest5.prototype.constructor = QuestInterface;

Quest5.prototype.load = function(){
    this.title = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest5/title.png');
    this.titleHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest5/title_hover.png');
    this.content = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest5/content.png');
    this.rewardInfo = new Framework.Sprite(define.imagePathCharacter + 'Quest/Quest5/rewardInfo.png');
}

Quest5.prototype.init = function(){
    this.title.scale = 0.45;
    this.titleHover.scale = this.title.scale;
    this.content.scale = 0.05;//to 0.37
    this.submitBtn.scale = 0.35;
    this.rewardInfo.scale = 0.38;

    this.content.position = {//width = 420
```

```
    x: 230,
    y: 304
  };

  this.submitBtn.position ={
    x: this.content.position.x - 20,
    y: this.content.position.y + 70
  };

  this.rewardInfo.position = {
    x: 630,
    y: 320
  };

  this.monsterKilledCounter = 0;
};

Quest5.prototype.drawInfo = function(ctx){
 if(!this.isCompleted && this.isTrigger && this.questUI.isHover){
    ctx.fillStyle = 'black';
    ctx.font = 'bold freeScript ';
    ctx.fillText('Spider: ' + this.monsterKilledCounter + ' / 5' ,1080, 278);
 }
};

Quest5.prototype.setTriggeringCondition = function(){
  if(this.player.level >= 8 && this.questUI.quest4.isCompleted){
     this.isTrigger = true;
  }
};

Quest5.prototype.processingQuest = function(){
  var monsterArr = this.map.monsterArr;
  if(this.isTrigger){
    for(var i = 0; i < monsterArr.length; i++){
      if(monsterArr[i].name === 'spider' && monsterArr[i].isDead){
```

```
        if(monsterArr[i].reviveTimeCounter > 0 && monsterArr[i].reviveTimeCounter < 2){
          this.monsterKilledCounter++;
          console.log(this.monsterKilledCounter);
        }
       }
     }
   }
};

Quest5.prototype.checkCompleted = function(){
   var bag = this.player.bag;
   var propertyArr = this.player.bag.propertyArr;
   var conditionCount = 0;

   for(var i = 0; i < propertyArr.length; i++){
    if(propertyArr[i].type === 10 || propertyArr[i].type === 11){
      if(propertyArr[i].number >= 5){
        conditionCount++;
      }
    }
   }

   if(bag.isPropertyExist({type:30,number:1})){
    conditionCount++;
   }

   if(this.monsterKilledCounter >= 5 && conditionCount === 3){//>= 3 && === 3
    this.isCompleted = true;
   }
};

Quest5.prototype.getReward = function(){
   var propertyArr = this.player.bag.propertyArr;
   var hpPotion = {
     type: 4,
     number: 20
```

```
    };
    var mpPotion = {
      type: 5,
      number: 20
    };

    var itemArr = [];
    itemArr.push(hpPotion);
    itemArr.push(mpPotion);

    if(this.isCompleted && this.canReceiveReward){
      this.player.exp += 1300;
      this.player.money += 100;
      for(var i = 0; i < itemArr.length; i++){
        if(!this.player.bag.isPropertyExist(itemArr[i])){
            this.player.bag.propertyArr.push(itemArr[i]);
        }
        else{
          this.player.bag.addProperty(itemArr[i]);
        }
      }
      this.canReceiveReward = false;
      this.takeAwayProperties(propertyArr);
    }
}

Quest5.prototype.takeAwayProperties = function(propertyArr){
    for(var i = 0; i < propertyArr.length; i++){
        if(propertyArr[i].type === 10 || propertyArr[i].type === 11){
          this.player.bag.propertyArr[i].number -= 5;
        }
        if(propertyArr[i].type === 30){
          this.player.bag.propertyArr[i].number--;
        }
    }
};
```

47.QuestInterface.js：

```
var QuestInterface = function(map, player, questUI){
  this.map = map;
  this.player = player;
  this.questUI = questUI;
  this.isReceived = false;
  this.isTrigger = false;
  this.isCompleted = false;
  this.isHover = false;
  this.isSubmitBtnHover = false;
  this.canReceiveReward = true;
  this.alertCounter = 0;

  this.submitBtn = new Framework.Sprite(define.imagePathCharacter + 'Quest/SubmitBtn.png');
  this.submitBtnHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/SubmitBtnHover.png');
  this.alert = new Framework.Sprite(define.imagePathCharacter + 'Quest/alert.png');

  this.submitBtn.scale = 0.35;
  this.submitBtnHover.scale = this.submitBtn.scale;
  this.alert.scale = 0.5;

  this.submitBtn.position ={
    x: 210,
    y: 374
  };
  this.submitBtnHover.position = this.submitBtn.position;
  this.alert.position = {
    x: 1143,
    y: 630
  };

  this.load();
  this.init();
}
```

```
QuestInterface.prototype.init = function(){

};

QuestInterface.prototype.drawInfo = function(){

};

QuestInterface.prototype.setTriggeringCondition = function(){

};

QuestInterface.prototype.processingQuest = function(){

};

QuestInterface.prototype.checkCompleted = function(){

};

QuestInterface.prototype.getReward = function(){

};

QuestInterface.prototype.takeAwayProperties = function(){

};
```

48.QuestItem.js：

```
var QuestItem = function(){
   this.load = function(){
     this.url1 = define.imagePathItem + 'crafting_material/bone/Cloud_coral.png';
     this.url2 = define.imagePathItem + 'mission_item/Aged_scroll.png';
     this.url3 = define.imagePathItem + 'mission_item/Perfect_scroll.png';
     this.cloud_coral = new Framework.Sprite(this.url1);
     this.aged_scroll = new Framework.Sprite(this.url2);
```

```
    this.perfect_scroll = new Framework.Sprite(this.url3);
  }
}

QuestItem.prototype = Object.create(Property.prototype);
QuestItem.prototype.constructor = Property;
```

49.QuestUI.js：

```
var QuestUI = function(map){
  this.load = function(){
    this.bookPic = new Framework.Sprite(define.imagePathCharacter + 'Quest/Book.png');
    this.bookPicHover = new Framework.Sprite(define.imagePathCharacter + 'Quest/Book_hover.png');
    this.uiPic = new Framework.Sprite(define.imagePathCharacter + 'Quest/UI.png');
    this.alertPic = new Framework.Sprite(define.imagePathCharacter + 'Quest/alert.png');
    this.processingInfoPic = new Framework.Sprite(define.imagePathCharacter + 'Quest/processingInfo.png');
    this.map = map;
    this.player = this.map.player;
    this.questArr = [];
    this.quest1 = new Quest1(this.map, this.player, this);
    this.questArr.push(this.quest1);
    this.pushQuests();
  }

  this.init = function(){
    this.isHover = false;
    this.drawUI = false;
    this.isUIOpened = false;
    this.showCounter = 0;
    this.showRewardInfo = false;

    this.bookPic.scale = 0.2;
    this.bookPicHover.scale = 0.2;
    this.uiPic.scale = 0.1;
    this.alertPic.scale = 0.6;
    this.processingInfoPic.scale = 0.5;
```

```
    this.bookPic.position = {
      x: 1193,
      y: 630
    };
    this.bookPicHover.position = this.bookPic.position;
    this.processingInfoPic.position = {
      x: this.bookPic.position.x,
      y: this.bookPic.position.y - 250
    };

    this.alertPic.position = {
      x: this.bookPic.position.x - 50,
      y: this.bookPic.position.y
    };

    this.uiPic.position = {
      x: 660,
      y: 300
    };
  }

  this.update = function(){
    this.bookPic.update();
    this.bookPicHover.update();
    if(this.uiPic.scale <= 0.5){
      this.uiPic.scale += 0.0065;
    }

    for(var i = 0; i < this.questArr.length; i++){
      this.questArr[i].update();
      if(this.questArr[i].isCompleted && this.showCounter === 0){
          this.showRewardInfo = true;
      }
      if(this.showCounter > 180){
        this.showRewardInfo = false;
        this.showCounter = 0;
```

```
        this.questArr.splice(i, 1);
      }
    }

    if(this.showRewardInfo){
        this.showCounter++;
    }
}

this.draw = function(ctx){
    if(this.isHover === false){
        this.bookPic.draw();
    }
    else{
        this.bookPicHover.draw();
        this.processingInfoPic.draw();
        this.drawQuestInfo(ctx);
    }
    if(this.drawUI === true){
        this.uiPic.draw();
        this.drawQuests(ctx);
    }
    else{
      this.uiPic.scale = 0.2;
    }
}

this.drawQuests = function(ctx){
    for(var i = 0; i < this.questArr.length; i++){
      var picPosition = {
        x: 620,
        y: 150 + i * 60
      };
      if(this.questArr[i].isTrigger && this.questArr[i].isCompleted === false){
        this.questArr[i].title.position = picPosition;
        this.questArr[i].titleHover.position = picPosition;
```

```
        this.questArr[i].draw(ctx);
      }
      if(this.showRewardInfo){
        this.questArr[0].rewardInfo.draw();
      }
    }
}

this.drawQuestInfo = function(ctx){
    for(var i = 0; i < this.questArr.length; i++){
      this.questArr[i].drawInfo(ctx);
    }
}

this.pushQuests = function(){
    this.quest2 = new Quest2(this.map, this.player, this);
    this.questArr.push(this.quest2);
    this.quest3 = new Quest3(this.map, this.player, this);
    this.questArr.push(this.quest3);
    this.quest4 = new Quest4(this.map, this.player, this);
    this.questArr.push(this.quest4);
    this.quest5 = new Quest5(this.map, this.player, this);
    this.questArr.push(this.quest5);
}

this.mousemove = function(e){
    if(e.x > 1150 && e.x < 1233 && e.y > 595 && e.y < 655){
        this.isHover = true;
    }
    else {
        this.isHover = false;
    }

    for(var i = 0; i < this.questArr.length; i++){
      var submitBtnPos = this.questArr[i].submitBtn.position;
      if(e.x > 474 && e.x < 764 && e.y > 127 + i * 60 && e.y < 153 + i * 60){
```

```
            this.questArr[i].isHover = true;
        }
        else {
            this.questArr[i].isHover = false;
        }
        if(this.questArr[i].drawContent === true){//Check if the submit button is pressed
          if(e.x > submitBtnPos.x - 50 && e.x < submitBtnPos.x + 50 &&
             e.y > submitBtnPos.y - 20 && e.y < submitBtnPos.y + 20){
            this.questArr[i].isSubmitBtnHover = true;
          }
          else {
            this.questArr[i].isSubmitBtnHover = false;
          }
        }
      }
    }

    this.click = function(e){
      if(e.x > 1150 && e.x < 1233 && e.y > 595 && e.y < 655){
          if(this.drawUI === false){
              this.drawUI = true;
          }
          else{
              this.drawUI = false;
          }
          if(this.isUIOpened === false){
              this.isUIOpened = true;
          }
          else{
              this.isUIOpened = false;
          }
      }
      for(var i = 0; i < this.questArr.length; i++){
        var closeContentPos = this.questArr[i].content.position;
        var submitBtnPos = this.questArr[i].submitBtn.position;
        if(e.x > 474 && e.x < 764 && e.y > 127 + i * 60 && e.y < 153 + i * 60){
```

```
            this.questArr[i].drawContent = true;
        }
        else if(e.x > closeContentPos.x + 210 || e.x < closeContentPos.x - 210 ||
              e.y > closeContentPos.y + 125 ||  e.y < closeContentPos.y - 125){
            this.questArr[i].drawContent = false;
        }

        if(this.questArr[i].drawContent === true){//Check if the submit button is pressed
          if(e.x > submitBtnPos.x - 50 && e.x < submitBtnPos.x + 50 &&
             e.y > submitBtnPos.y - 20 && e.y < submitBtnPos.y + 20){
            this.questArr[i].checkCompleted();
            this.questArr[i].getReward();
          }
        }
      }
    }
}
```

50.RainDrop.js：

```
var RainDrop = function(){
    this.clearColor = 'rgba(0, 0, 0, 0.1)';
    this.init = function(){
      this.windowSize = {
        width: Framework.Game.getCanvasWidth(),
        height: Framework.Game.getCanvasHeight()
      };
      this.position = {
        x: this.randomNumber(0, this.windowSize.width),
        y: this.randomNumber(-10, -300)
      };
      this.color = 'rgba(162, 221, 221, 0.36)';
      this.velocity = this.randomNumber(4, 8);
      this.hit = this.randomNumber(this.windowSize.height * 0.1, this.windowSize.height * 0.9);
      this.size = 2;
      this.circleWidth = 2.5;
      this.circleHeight = 1;
```

```
  this.a = 1;
  this.va = 0.96;
  this.vw = 3;
  this.vh = 1;
}

this.randomNumber = function(min, max){
  return Math.random() * (max - min) + min;
}

this.draw = function(ctx){
  if(this.position.y < this.hit){
    ctx.fillStyle = this.color;
    ctx.fillRect(this.position.x, this.position.y, this.size, this.size * 15);
  }
  else{
    ctx.beginPath();
    ctx.moveTo(this.position.x, this.position.y - this.circleHeight / 2);

    ctx.bezierCurveTo(
          this.position.x + this.circleWidth / 2, this.position.y - this.circleHeight / 2,
          this.position.x + this.circleWidth / 2, this.position.y + this.circleHeight / 2,
          this.position.x, this.position.y + this.circleHeight / 2);

      ctx.bezierCurveTo(
          this.position.x - this.circleWidth / 2, this.position.y + this.circleHeight / 2,
          this.position.x - this.circleWidth / 2, this.position.y - this.circleHeight / 2,
          this.position.x, this.position.y - this.circleHeight / 2);

      ctx.strokeStyle = 'rgba(162, 221, 221, '+this.a+')';
      ctx.stroke();
      ctx.closePath();
  }
  this.changeRainPosition();
}
```

```
    this.changeRainPosition = function(){
      if(this.position.y < this.hit){
        this.position.x -= 1;
        this.position.y += this.velocity;
      }
      else{
        if(this.a > 0.03){
          this.circleWidth += this.vw;
          this.circleHeight += this.vh;
          if(this.circleWidth > 10){
            this.a *= this.va;
            this.vw *= 0.98;
            this.vh *= 0.98;
          }
        }
        else{
          this.init();
        }
      }
    }
}
```

51.Skill.js：

```
var Skill = function(damage, coolDownTime, hpCost, mpCost, mmH, map){
    this.damage = damage;
    this.abilityList = [];
    this.mpCostList = [];
    this.hpCostList = [];
    this.coolDownTime = coolDownTime;
    this.hpCost = hpCost;
    this.mpCost = mpCost;
    this.maxMonsterHit = mmH;
    this.canAttack = true;
    this.map = map;
    this.player = map.player;
    this.level = 0;
```

```
    this.monsterNumber = 0;
    this.coolDownCounter = this.coolDownTime * 100;
    this.isAttack = false;
    this.isDrawed = false;
}

Skill.prototype.update = function(){
    var i = this.monsterNumber;
    var playerPos = this.player.playerPic.position;
    var monsterArr = this.map.monsterArr;
    this.pic.update();
    this.changeDamage();
    this.changeMpCost();
    this.levelUp();

    if(this.isAttack && this.coolDownCounter > 0){
      this.coolDownCounter--;
    }

    if(this.coolDownCounter < this.coolDownTime * 100){
      this.canAttack = false;
    }
    if(this.coolDownCounter === 0){
      this.isAttack = false;
      this.canAttack = true;
      this.coolDownCounter = this.coolDownTime * 100;
    }
};

Skill.prototype.changeDamage = function(){
    this.damage = Math.round(this.player.atk * this.abilityList[this.level]);
};

Skill.prototype.changeMpCost = function(){
    this.mpCost = this.mpCostList[this.level];
};
```

```
Skill.prototype.changeHpCost = function(){
    this.hpCost = this.hpCostList[this.level];
};

Skill.prototype.levelUp = function(){

};

Skill.prototype.move = function(){

};

Skill.prototype.attack = function(){

};

Skill.prototype.removeObstacle = function(){

};

Skill.prototype.canHitMonster = function(){

};

Skill.prototype.recover = function(){

};

Skill.prototype.buff = function(){

};
```

52.SkillShowcase.js：

```
var SkillShowcase = Framework.Class(Framework.Level , {
    load: function(){
        this.allMap = new TestMap();
        this.allMap.load();
        this.allMap.generateMonsters();
    },
    loadingProgress: function(context, requestInfo){
        context.fillRect(0, 0, 1360, 700);
    },

    initialize: function() {
        this.allMap.initialize();
    },

    update: function() {
        this.allMap.update();
    },

    draw: function(parentCtx){
        this.allMap.draw(parentCtx);
    },

    keydown:function(e, list){
        this.allMap.player.keydown(e, list);

        if(e.key === 'F11') {
            if(!this.isFullScreen) {
                Framework.Game.fullScreen();
                this.isFullScreen = true;
            } else {
                Framework.Game.exitFullScreen();
                this.isFullScreen = false;
            }
        }
```

```
    },

    keyup:function(e, list){
        this.allMap.player.keyup(e, list);
    },

    mousemove:function(e){
        this.allMap.mousemove(e);
    },

    click: function (e) {
        this.allMap.click(e);
    },
});
```

53.SkillUI.js：

```
var SkillUI = function(map){
    this.map = map;
    this.player = map.player;
    this.load = function(){
        this.pic = new Framework.Sprite(define.imagePathCharacter + 'Skill/skill_icon.png');
        this.picHover = new Framework.Sprite(define.imagePathCharacter + 'Skill/skill_iconHover.png');
        this.uiPic = new Framework.Sprite(define.imagePathCharacter + 'Skill/skillUI.png');
        this.frozenSwordPic = new Framework.Sprite(define.imagePathCharacter + 'Skill/FrozenSword.png');
        this.heroSlashPic = new Framework.Sprite(define.imagePathCharacter + 'Skill/HeroSlash.png');
        this.potentialPowerPic = new Framework.Sprite(define.imagePathCharacter + 'Skill/PotentialPower.png');
        this.infoPic = new Framework.Sprite(define.imagePathCharacter + 'Skill/Skill_info.png');
    }

    this.init = function(){
        this.skillArr = [];
        this.isHover = false;
        this.isUIOpened = false;

        this.pic.scale = 0.45;
        this.picHover.scale = this.pic.scale;
```

```
    this.uiPic.scale = 0.5;
    this.frozenSwordPic.scale = 0.6;
    this.heroSlashPic.scale = 0.6;
    this.potentialPowerPic.scale = 0.6;

    this.pic.position = {
      x: 1090,
      y: 630
    }
    this.picHover.position = this.pic.position;
    this.uiPic.position = {
      x: 260,
      y: 300
    }
    this.frozenSwordPic.position = {
      x: 840,
      y: 545
    }
    this.heroSlashPic.position = {
      x: 710,
      y: 545
    }
    this.potentialPowerPic.position = {
      x: 580,
      y: 541
    }
    this.infoPic.position = {
      x: 830,
      y: 250
    }
}

this.update = function(){
    this.pic.update();
}
```

```
this.draw = function(ctx){
    if(this.isHover === false){
        this.pic.draw();
    }
    else{
        this.picHover.draw();
    }
    if(this.isUIOpened){
        this.uiPic.draw();
        this.infoPic.draw();
        this.drawEachSkill(ctx);
    }
    this.frozenSwordPic.draw();
    this.heroSlashPic.draw();
    this.potentialPowerPic.draw();
    this.drawCoolDownTime(ctx);
}

this.drawEachSkill = function(ctx){
    ctx.fillStyle = 'black';
    ctx.font = 'bold 26px freeScript ';
    if(this.player.vCut.level === 0){
      ctx.fillStyle = 'red';
    }
    ctx.fillText('V-Cut  Level: ' + this.player.vCut.level, 126, 142);
    ctx.fillText('Damage: ' + Math.floor(this.player.vCut.abilityList[this.player.vCut.level] * 100 )+ '%', 126, 168);

    if(this.player.microShield.level === 0){
      ctx.fillStyle = 'red';
    }
    ctx.fillText('Micro Shield  Level: ' + this.player.microShield.level, 126, 208);

    if(this.player.burstCut.level === 0){
      ctx.fillStyle = 'red';
    }
    ctx.fillText('Burst Cut  Level: ' + this.player.burstCut.level, 126, 274);
```

```
    ctx.fillText('Damage: ' + Math.floor(this.player.burstCut.abilityList[this.player.burstCut.level] * 100 )+ '%', 126, 300);

    if(this.player.swordLight.level === 0){
      ctx.fillStyle = 'red';
    }
    ctx.fillText('Sword Light  Level: ' + this.player.swordLight.level, 126, 340);
    ctx.fillText('Damage: ' + Math.floor(this.player.swordLight.abilityList[this.player.swordLight.level] * 100 )+ '%', 126, 366);
}

this.drawCoolDownTime = function(ctx){
    ctx.fillStyle = 'black';
    ctx.fillText('Cooldown Time', 380, 550);
    if(this.player.potentialPower.canAttack){
      ctx.fillText('0', 620, 541);
    }
    else{
      ctx.fillText(Math.round(this.player.potentialPower.coolDownCounter/100), 620, 541);
    }

    if(this.player.heroSlash.canAttack){
      ctx.fillText('0', 750, 541);
    }
    else{
      ctx.fillText(Math.round(this.player.heroSlash.coolDownCounter/100), 750, 541);
    }

    if(this.player.frozenSword.canAttack){
      ctx.fillText('0', 880, 541);
    }
    else{
      ctx.fillText(Math.round(this.player.frozenSword.coolDownCounter/100), 880, 541);
    }
}

this.mousemove = function(e){
    if(e.x > 1055 && e.x < 1112 && e.y > 599 && e.y < 655){
```

```
            this.isHover = true;
        }
        else {
            this.isHover = false;
        }
    }

    this.click = function(e){
        if(e.x > 1055 && e.x < 1112 && e.y > 599 && e.y < 655){
            if(this.isUIOpened === false){
                this.isUIOpened = true;
            }
            else{
                this.isUIOpened = false;
            }
        }
    }
}
```

54.Spider.js：

```
var Spider = function(map,player){
    Monster.call(this, 120, 23, 7, player, map);//Call the parent's constructor(hp,atk,exp,player)
    this.load = function(){
      this.url = define.imagePathMonster + "spider.png";
      this.pic = new Framework.AnimationSprite({url:this.url, col:6, row:4, loop: true, speed: 6});
      this.pic.scale = 1;
      this.walkDir = {x:0, y:0};
      this.counter = 0;
      this.name = 'spider';
      this.map = map;
    }

    this.draw = function(ctx){
      var picPos = this.pic.position;
      var screenPos = this.map.screenPosition;
      var playerPos = this.player.playerPic.position;
```

```
  ctx.fillStyle = 'rgb(9, 4, 23)';
  ctx.fillRect(picPos.x-screenPos.x-30, picPos.y-screenPos.y-30, 60*(this.currentHP/this.hp), 10);
  if(this.isTouchingPlayer){
    this.printDamageReceived(ctx);
  }
}

this.update = function(){
  if(this.counter > 90){
    var randomDir = Math.floor(Math.random() * 4);
    this.counter = 0;
  }
  this.counter++;
  this.move(randomDir);
  this.checkMonsterDead();
  this.revive();
  this.pic.update();

  if(this.player.currentHP > 0 && this.player.isAttacked === false){
    this.touchPlayer();
  }

  if(this.player.resistCounter === 0){
    this.player.isAttacked = false;
    this.isTouchingPlayer = false;
    this.player.resistCounter = this.player.resistTime;
  }
}

this.setPosition = function(pos){
  this.pic.position = {
    x: pos.x * 32 + 16,
    y: pos.y * 32 + 16
  };
}
```

```
}

Spider.prototype = Object.create(Monster.prototype);
Spider.prototype.constructor = Monster;

Spider.prototype.move = function(randomDir){
 switch (randomDir) {
   case 0://up
     this.walkDir = {x: 0, y: -1};
     this.pic.start({from:18, to:23, loop:true});
     break;
   case 1://left
     this.walkDir = {x: -1, y: 0};
     this.pic.start({from:6, to:11, loop:true});
     break;
   case 2://down
     this.walkDir = {x: 0, y: 1};
     this.pic.start({from:0, to:5, loop:true});
     break;
   case 3://right
     this.walkDir = {x: 1, y: 0};
     this.pic.start({from:12, to:17, loop:true});
     break;
 }
 if(this.map.canMove(this.pic.position.x + this.walkDir.x, this.pic.position.y + this.walkDir.y, randomDir)){
     this.pic.position.x += this.walkDir.x;
     this.pic.position.y += this.walkDir.y;
 }
};

Spider.prototype.touchPlayer = function(){
   var picPos = this.pic.position;
   var playerPos = this.player.playerPic.position;
   if(Math.abs(picPos.x - playerPos.x) < 33 && Math.abs(picPos.y - playerPos.y) < 33 && !this.isDead){
       if(this.player.currentHP <= this.currentAtk){
           this.player.currentHP = 0;
```

```
        }
        else{
            this.player.currentHP -= Math.floor(this.currentAtk * (1-this.player.def));
        }
        this.player.isAttacked = true;
        this.isTouchingPlayer = true;
    }
};
```

55.SwordLight.js：

```
var SwordLight = function(map){
    Skill.call(this, 2, 1, 0, 6, 3, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
      this.url = define.imagePathEffect + 'Warrior/003.png';
      this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:6, loop:true, speed:10});
    }

    this.init = function(){
      this.abilityList = [0, 2.1, 2.2, 2.3, 2.4, 2.5];
      this.mpCostList = [0, 11, 13, 15, 17, 19];
      this.pic.scale = 0.8;
    }

    this.draw = function(){
      if(this.isDrawed){
        this.map.drawOtherObj(this.pic);
      }
    }
}

SwordLight.prototype = Object.create(Skill.prototype);
SwordLight.prototype.constructor = Skill;

SwordLight.prototype.levelUp = function(){
    for(var level = 15; level <= 19; level++){
      if(this.player.level === level){
```

```
        this.level = level - 14;
      }
    }
    if(this.player.level > 19 || this.map.mapType !== 'scroll'){//>19
      this.level = 5;
    }
};

SwordLight.prototype.removeObstacle = function(index,direction){
    var mapArr = this.map.mapArr;
    var currentMap = this.player.currentMap;

    if(direction === 0){//up
      for(var row = index.row - 1; row >= index.row - 3; row--){
        for(var col = index.col - 1; col <= index.col + 1; col++){
          if(this.map.mapType === 'scroll'){
            if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
              this.map.mapArr[this.player.currentMap][row][col] = 0;
            }
          }
          else{
            if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
              this.map.testMapArr[row][col] = 0;
            }
          }
        }
      }
    }
    else if(direction === 1){//left
      for(var row = index.row - 1; row <= index.row + 1; row++){
        for(var col = index.col - 3; col <= index.col - 1; col++){
          if(this.map.mapType === 'scroll'){
            if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
              this.map.mapArr[this.player.currentMap][row][col] = 0;
            }
```

```
        }
        else{
          if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
            this.map.testMapArr[row][col] = 0;
          }
        }
      }
    }
  }
  else if(direction === 2){
    for(var row = index.row + 1; row <= index.row + 3; row++){
      for(var col = index.col - 1; col <= index.col + 1; col++){
        if(this.map.mapType === 'scroll'){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
            this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
        }
        else{
          if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
            this.map.testMapArr[row][col] = 0;
          }
        }
      }
    }
  }
  else if(direction === 3){
    for(var row = index.row + 1; row >= index.row - 1; row--){
      for(var col = index.col + 1; col <= index.col + 3; col++){
        if(this.map.mapType === 'scroll'){
          if(this.map.mapArr[this.player.currentMap][row][col] > 1 && this.map.mapArr[this.player.currentMap][row][col] < 8 &&
row >= 0 && col >= 0){
            this.map.mapArr[this.player.currentMap][row][col] = 0;
          }
        }
        else{
          if(this.map.testMapArr[row][col] > 1 && this.map.testMapArr[row][col] < 8 && row >= 0 && col >= 0){
```

```
            this.map.testMapArr[row][col] = 0;
          }
        }
      }
    }
  }
};

SwordLight.prototype.canHitMonster = function(direction,i){
  var playerPos = this.player.playerPic.position;
  var monsterArr = this.map.monsterArr;
  var dirDifference = {
    x: playerPos.x - monsterArr[i].pic.position.x,
    y: playerPos.y - monsterArr[i].pic.position.y
  };

  switch (direction) {
    case 0:
      if(Math.abs(dirDifference.x) <= 96 && dirDifference.y <= 96 && dirDifference.y >= 0){
        return true;
      }
      return false;
    case 1:
      if(dirDifference.x <= 96 && dirDifference.x >= 0 && Math.abs(dirDifference.y) <= 96){
        return true;
      }
      return false;
    case 2:
      if(Math.abs(dirDifference.x) <= 96 && dirDifference.y >= -96 && dirDifference.y <= 0){
        return true;
      }
      return false;
    case 3:
      if(dirDifference.x >= -96 && dirDifference.x <= 0 && Math.abs(dirDifference.y) <= 96){
        return true;
      }
```

```
        return false;
    }
};

SwordLight.prototype.attack = function(i, j, direction){
    var index = this.map.toArrayIndex(i, j, direction);
    var monsterArr = this.map.monsterArr;

    this.pic.start({from:0, to:29, loop:false});
    for(var i = 0; i < monsterArr.length; i++){//Determine if player hits the monster
      if(this.canHitMonster(direction,i) && monsterArr[i].canBeAttacked){
          if(monsterArr[i].currentHP >= this.damage){
            monsterArr[i].currentHP -= this.damage;
          }
          else{
            monsterArr[i].currentHP = 0;
          }
      }
    }
    this.removeObstacle(index,direction);
};
```

56.Teleport.js：

```
var Teleport = function(map){
    Skill.call(this, 0, 0.5, 0, 0, 0, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
        this.url = define.imagePathEffect + '006.png';
        this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:5, loop:true, speed:12});
    }

    this.draw = function(){
      if(this.isDrawed){
        this.isUsingSkill = true;
        this.pic.position = {
          x: this.player.playerPic.position.x,
          y: this.player.playerPic.position.y
```

```
        };
        this.map.drawOtherObj(this.pic);
      }
    }
}

Teleport.prototype = Object.create(Skill.prototype);
Teleport.prototype.constructor = Skill;

Teleport.prototype.changeDamage = function(){

};

Teleport.prototype.changeMpCost = function(){

};

Teleport.prototype.levelUp = function(){

};

Teleport.prototype.move = function(walkDir){
    this.player.playerPic.position.x += walkDir.x * 32;
    this.player.playerPic.position.y += walkDir.y * 32;
    this.pic.start({from:0, to:24, loop:false});
}
```

57.TestMap.js：

```
var TestMap = function() {
    BaseMap.call(this,'normal');
    this.screenPosition = {//on the map
        x: 0,
        y: 0
    };

    this.testMapArr = this.arrays.testMapArray;
```

```
this.propertyArr = [];
this.rainArr = [];

this.player = new Player(this);
this.player.load();
this.player.init();
this.player.setPosition({x: 21, y: 10},this.screenPosition);

this.alpha = 1;

this.monsterAmount = 10;
this.counter = 0;

this.load = function() {
    this.linkBtn = new Framework.Sprite(define.imagePath + "linkBtn.png");
    this.linkBtn_hover = new Framework.Sprite(define.imagePath + "linkBtn_hover.png");
    this.info = new Framework.Sprite(define.imagePathMap + "UI1-1.png");
    this.grassPic = new Framework.Sprite(define.imagePathMap + "grass.png");
    this.treePic = new Framework.Sprite(define.imagePathMap + "tree.png");
    this.stonePic = new Framework.Sprite(define.imagePathMap + "stone2.png");
    this.floorPic = new Framework.Sprite(define.imagePathMap + "bg.png");
    this.flowerPic = new Framework.Sprite(define.imagePathMap + "flower2.png");
    this.cloudPic = new Framework.Sprite(define.imagePathMap + "sky.png");
    this.treeCut = new Framework.Sprite(define.imagePathMap + "tree_cut.png");

    this.cluster = new Cluster(this);
    this.cluster.load();
    this.medicine = new Medicine(this);
    this.medicine.load();

    for(var i = 0; i < 200; i++){
      this.rainDrop = new RainDrop();
      this.rainDrop.init();
      this.rainArr.push(this.rainDrop);
    }
```

```
};

this.initialize = function() {
    this.player.setPlayerLevel(99);
    this.player.init();
    this.player.currentMap = 5;

    this.floorPic.position = this.screenPosition;
    this.linkBtn.position = {
        x: 1280,
        y: 50
    };
    this.linkBtn_hover.position = this.linkBtn.position;

    this.floorPic.scale = 1;
    this.linkBtn.scale = 0.35;
    this.linkBtn_hover.scale = 0.35;
    this.isLinkBtnHoveer = false;
    this.littleRain = 30;
    this.heavyRain = 180;

    this.info.position = {x:700, y:785};
    this.generateProperties();
    this.drawEff = true;
    this.isHeavyRain = false;
    this.isWin = false;
};

this.update = function() {
    this.player.update();
    this.checkChangeMap();
    this.counter++;

    if(this.player.isChangeMap){
      this.monsterArr.splice(0, this.monsterArr.length);
      console.log(this.monsterArr);
```

```
    }
    for(var i = 0; i < this.monsterArr.length; i++){
      this.monsterArr[i].update();
    }
};

this.draw = function(ctx) {
      ctx.fillStyle = 'black';
      ctx.fillRect(0,0,1360,700);
      for (var i = 0; i < 22; i++) {
         for (var j = 0; j < 43; j++) {
              var picPosition = {
                 x: 32 * j  + 16,
                 y: 32 * i  + 16
              }
              switch(this.testMapArr[i][j]) {
                 case 0 :
                    break;
                 case 1 :
                    this.grassPic.position = picPosition;
                    this.grassPic.draw();
                    break;
                 case 6 :
                    this.flowerPic.position = picPosition;
                    this.flowerPic.draw();
                    break;
                 case 7 :
                    this.treePic.position = picPosition;
                    this.treePic.draw();
                    break;
                 case 8 :
                    this.stonePic.position = picPosition;
                    this.stonePic.draw();
                    break;
                 case 9 :
                    this.cloudPic.position = picPosition;
```

```
                this.cloudPic.draw();
                break;
        }
    }
}
if(this.counter > 120){
  this.generateRain(ctx);
}

this.player.playerPic.draw();
for(var i = 0; i < this.monsterArr.length; i++){
  if(this.monsterArr[i].currentHP > 0){
    this.monsterArr[i].pic.draw();
  }
  this.monsterArr[i].draw(ctx);
}
this.info.draw();
this.player.draw(ctx);
if(!this.isLinkBtnHoveer){
    this.linkBtn.draw();
}
else {
    this.linkBtn_hover.draw();
}

if(this.drawEff){
  this.alpha -= 0.002;
  ctx.globalAlpha = this.alpha;
  ctx.fillStyle = "black";
  ctx.fillRect(0, 0, 1360, 700);
}

if(this.isWin){
  this.heavyRain--;
  if(this.alpha < 0.8){
    this.alpha += 0.003;
```

```
        ctx.globalAlpha = this.alpha;
      }
      else{
        this.disPlayChar = true;
      }
      ctx.fillStyle = "black";
      ctx.fillRect(0, 0, 1360, 700);
    }

    if(this.alpha <= 0){
      this.drawEff = false;
    }
};

this.generateRain = function(ctx){
  var width = this.rainDrop.windowSize.width
  var height = this.rainDrop.windowSize.height;

  ctx.fillStyle = this.rainDrop.clearColor;
  ctx.fillRect(0, 0, width, height);
  if(!this.isHeavyRain){
    for(var i = 0; i < this.littleRain; i++){
      this.rainArr[i].draw(ctx);
    }
  }
  else{
    for(var i = 0; i < this.heavyRain; i++){
      this.rainArr[i].draw(ctx);
    }
  }
};



this.mousemove = function(e){
  if(e.x >= 1250 && e.x <= 1300 && e.y >= 26 && e.y <= 69){
      this.isLinkBtnHoveer = true;
```

```
        }
        else{
            this.isLinkBtnHoveer = false;
        }
    };

    this.click = function(e){
        if(e.x >= 1250 && e.x <= 1300 && e.y >= 26 && e.y <= 69){
            window.open("Game_Info/MainPage/index.html");
        }
    };
}

TestMap.prototype = Object.create(BaseMap.prototype);
TestMap.prototype.constructor = BaseMap;


TestMap.prototype.canMove = function(i, j, direction) {
    this.toArrayIndex(i, j, direction);

    if(i < 16 || i > 1336){ return false; }
    if(j < 16 || j > 686){ return false; }
    if (this.testMapArr[this.arrRow][this.arrCol] > 1) {
        return false;
    }
    else {
        return true;
    }
};

TestMap.prototype.generateMonsters = function(){
    for(var i = 0; i < 5; i++){
      while(true){
        var row = Math.floor(Math.random() * 22), col = Math.floor(Math.random() * 43);
        if(this.testMapArr[row][col] < 2){
          break;
```

```
      }
    }
    this.devil = new Devil(this, this.player);
    this.devil.load();
    this.devil.setPosition({x: col, y: row});

    this.boss = new Boss(this, this.player);
    this.boss.load();
    this.boss.setPosition({x: 21, y: 10});
    this.boss.init();

    this.defender = new Defender(this, this.player);
    this.defender.load();
  }
};
```

58.Thunder.js：

```
var Thunder = function(map,monster){
  Skill.call(this, 10, 1, 0, 0, 5, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
  this.monster = monster;
  this.load = function(){
    this.url = define.imagePathEffect + '008.png';
    this.url2 = define.imagePathEffect + 'fire_001.png';
    this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:3, loop:true, speed:5});
    this.effectPic = new Framework.AnimationSprite({url:this.url2, col:5, row:4, loop:true, speed:5});
    this.effectPic.scale = 0.65;
    this.pic.position = {
      x: -20,
      y: -20
    };
    this.effectPic.position = {
      x: -500,
      y: -500
    }
    this.drawEff = false;
  }
```

```
    this.draw = function(){
      if(this.isDrawed){
        this.map.drawOtherObj(this.pic);
        this.effectPic.position = {
          x: this.player.playerPic.position.x,
          y: this.player.playerPic.position.y - 10
        }
        if(this.drawEff){
          this.map.drawOtherObj(this.effectPic);
        }
      }
    }
}

Thunder.prototype = Object.create(Skill.prototype);
Thunder.prototype.constructor = Skill;

Thunder.prototype.canHitMonster = function(i){
      var playerPos = this.player.playerPic.position;
      var dirDifference = {
        x: playerPos.x - this.pic.position.x,
        y: playerPos.y - this.pic.position.y
      };

      if(Math.abs(dirDifference.x) < 33 && Math.abs(dirDifference.y) < 33){
          return true;
      }
      return false;
};

Thunder.prototype.attack = function(){
    var playerPos = this.player.playerPic.position;
    var row = Math.floor(Math.random() * 22), col = Math.floor(Math.random() * 43);

    this.pic.position = {
```

```
    x: col * 32,
    y: row * 32
   };

   this.pic.start({from:0, to:12, loop:false});
   if(this.canHitMonster(i)){
    this.effectPic.start({from:0, to:18, loop:false})
    if(this.player.currentHP > Math.floor(this.player.healthPoint * 0.8)){
     this.player.currentHP -= Math.floor(this.player.healthPoint * 0.8);
    }
    else{
     this.player.currentHP = 0;
    }
    this.player.isAttacked = true;
    this.drawEff = true;
   }
};
```

59.TreasureChest.js：

```
var TreasureChest = function(map){
   this.map = map;
   this.player = map.player;
   this.load = function(){
    this.url1 = define.imagePathCharacter + 'treasure_chest.png';
    this.url2 = define.imagePathCharacter + 'treasure_chest_hover.png';
    this.url3 = define.imagePathItem + 'mission_item/Perfect_scroll.png';
    this.pic = new Framework.Sprite(this.url1);
    this.picHover = new Framework.Sprite(this.url2);
    this.perfect_scroll = new Framework.Sprite(this.url3);
   };

   this.init = function(){
    this.pic.position = {
     x: 2680,
     y: 1132
    };
```

```
  this.picHover.position = this.pic.position;
  this.perfect_scroll.position = this.pic.position;
  this.pic.scale = 0.2;
  this.picHover.scale = 0.2;
  this.isPicHover = false;
  this.drawPic = true;
  this.picMoveOffset = 0;
  this.picMoveCounter = 0;
};

this.draw = function(ctx){
  if(!this.drawPic && this.map.propertyArr[35][83] !== 0){
    this.map.drawOtherObj(this.perfect_scroll);
    this.picMoveCounter++;
    if(this.picMoveCounter < 30){
      this.picMovingUp();
      this.perfect_scroll.rotation += 18;
    }
    else if(this.picMoveCounter >= 30 && this.picMoveCounter <= 60){
      this.picMovingDown();
      this.perfect_scroll.rotation += 18;
    }
  }
};

this.update = function(){
  this.perfect_scroll.update();
};

this.picMovingUp = function(){
  this.perfect_scroll.position.y-=2;
};

this.picMovingDown = function(){
  this.perfect_scroll.position.y+=2;
};
```

```
    this.mousemove = function(e){
        if(e.x + this.map.screenPosition.x >= 2643 && e.x + this.map.screenPosition.x <= 2725 &&
          e.y + this.map.screenPosition.y >= 1093 && e.y + this.map.screenPosition.y <= 1177){
            this.isPicHover = true;
        }
        else{
            this.isPicHover = false;
        }
    };

    this.click = function(e){
      if(e.x + this.map.screenPosition.x >= 2643 && e.x + this.map.screenPosition.x <= 2725 &&
        e.y + this.map.screenPosition.y >= 1093 && e.y + this.map.screenPosition.y <= 1177){
          this.getReward();
      }
    };

    this.getReward = function(){
      if(this.player.level >= 10){
          this.drawPic = false;
          this.map.propertyArr[35][83] = 31;
      }
    };
}
```

60.VCut.js：

```
var VCut = function(map){
    Skill.call(this, 2, 0.8, 0, 5, 3, map);//ctor(damage, coolDownTime, hpCost, mpCost, maxMonsterHit)
    this.load = function(){
      this.url = define.imagePathEffect + 'Warrior/001.png';
      this.pic = new Framework.AnimationSprite({url:this.url, col:5, row:4, loop:true, speed:10});
    }

    this.init = function(){
      this.abilityList = [0, 1.1, 1.2, 1.3, 1.4, 1.5];
```

```
    this.mpCostList = [0, 5, 6, 7, 8, 9];
    this.pic.scale = 0.5;
  }

  this.picDirection = function(direction){
    switch (direction) {
      case 0:
        this.pic.rotation = 180;
        break;
      case 1:
        this.pic.rotation = 90;
        break;
      case 2:
        this.pic.rotation = 0;
        break;
      case 3:
        this.pic.rotation = 270;
        break;
    }
  }

  this.draw = function(){
    if(this.isDrawed){
      this.map.drawOtherObj(this.pic);
    }
  }
}

VCut.prototype = Object.create(Skill.prototype);
VCut.prototype.constructor = Skill;

VCut.prototype.levelUp = function(){
  for(var level = 5; level <= 9; level++){
    if(this.player.level === level){
      this.level = level - 4;
    }
```

```
    }
    if(this.player.level > 9 || this.map.mapType !== 'scroll'){
      this.level = 5;
    }
};

VCut.prototype.removeObstacle = function(index,direction){
    var mapArr = this.map.mapArr;
    var currentMap = this.player.currentMap;

    if(direction === 0){//up
      for(var row = index.row - 1; row >= index.row - 3; row--){
        if(mapArr[currentMap][row][index.col] > 0 &&
          mapArr[currentMap][row][index.col] < 8 && row >= 0){
          mapArr[currentMap][row][index.col] = 0;
        }
      }
    }
    else if(direction === 1){//left
      for(var col = index.col - 1; col >= index.col - 3; col--){
        if(mapArr[currentMap][index.row][col] > 0 &&
          mapArr[currentMap][index.row][col] < 8 && col >= 0){
          mapArr[currentMap][index.row][col] = 0;
        }
      }
    }
    else if(direction === 2){
      for(var row = index.row + 1; row <= index.row + 3; row++){
        if(mapArr[currentMap][row][index.col] > 0 &&
          mapArr[currentMap][row][index.col] < 8 && row >= 0){
          mapArr[currentMap][row][index.col] = 0;
        }
      }
    }
    else if(direction === 3){
      for(var col = index.col + 1; col <= index.col + 3; col++){
```

```
        if(mapArr[currentMap][index.row][col] > 0 &&
          mapArr[currentMap][index.row][col] < 8 && col >= 0){
          mapArr[currentMap][index.row][col] = 0;
        }
      }
    }
};

VCut.prototype.canHitMonster = function(direction,i){
      var playerPos = this.player.playerPic.position;
      var monsterArr = this.map.monsterArr;
      var dirDifference = {
        x: playerPos.x - monsterArr[i].pic.position.x,
        y: playerPos.y - monsterArr[i].pic.position.y
      };

      switch (direction) {
        case 0:
          if(Math.abs(dirDifference.x) <= 32 && dirDifference.y <= 96 && dirDifference.y >= 0){
              return true;
          }
          return false;
        case 1:
          if(dirDifference.x <= 96 && dirDifference.x >= 0 && Math.abs(dirDifference.y) <= 32){
              return true;
            }
          return false;
        case 2:
          if(Math.abs(dirDifference.x) <= 32 && dirDifference.y >= -96 && dirDifference.y <= 0){
              return true;
            }
          return false;
        case 3:
          if(dirDifference.x >= -96 && dirDifference.x <= 0 && Math.abs(dirDifference.y) <= 32){
              return true;
            }
```

```
        return false;
    }
};

VCut.prototype.attack = function(i, j, direction){
    var index = this.map.toArrayIndex(i, j, direction);
    var monsterArr = this.map.monsterArr;

    this.picDirection(direction);
    this.pic.start({from:0, to:16, loop:false});

    for(var i = 0; i < monsterArr.length; i++){//Determine if player hits the monster
      if(this.canHitMonster(direction,i) && monsterArr[i].canBeAttacked){
          if(monsterArr[i].currentHP >= this.damage){
            monsterArr[i].currentHP -= this.damage;
          }
          else{
            monsterArr[i].currentHP = 0;
          }
      }
    }
    this.removeObstacle(index,direction);
}
```

61.WarriorData.js：

```
var WarriorData = function(){
    PlayerData.call(this);
}
WarriorData.prototype = Object.create(PlayerData.prototype);
WarriorData.prototype.constructor = PlayerData;

WarriorData.prototype.createHpTable = function(){
    this.hpTable.push(100);
    for(var level = 2; level <= 100; level++){
      var previousHp = this.hpTable[level-2];
      var currentHp = 0;
```

```
    if(level < 51){
      currentHp = Math.round(previousHp * 1.1);
    }
    else{
      currentHp = Math.round(previousHp * 1.03);
    }
    this.hpTable.push(currentHp);
  }
};

WarriorData.prototype.createMpTable = function(){
  this.mpTable.push(100);
  for(var level = 2; level <= 100; level++){
    var previousMp = this.mpTable[level-2];
    var currentMp = Math.round(previousMp * 1.05);
    this.mpTable.push(currentMp);
  }
};

WarriorData.prototype.createApTable = function(){
  this.apTable.push(1000);
  for(var level = 2; level <= 100; level++){
    var currentAp = this.apTable[level-2] + 50;
    this.apTable.push(currentAp);
  }
};

WarriorData.prototype.createAtkTable = function(){
  for(var level = 1; level <= 50; level++){
    var currentAtk = level;
    if(level >= 6){
      currentAtk = Math.round(this.atkTable[level-2] * 1.1);
    }
    this.atkTable.push(currentAtk);
  }
}
```