CRYSTAL LEGACY

REPORT

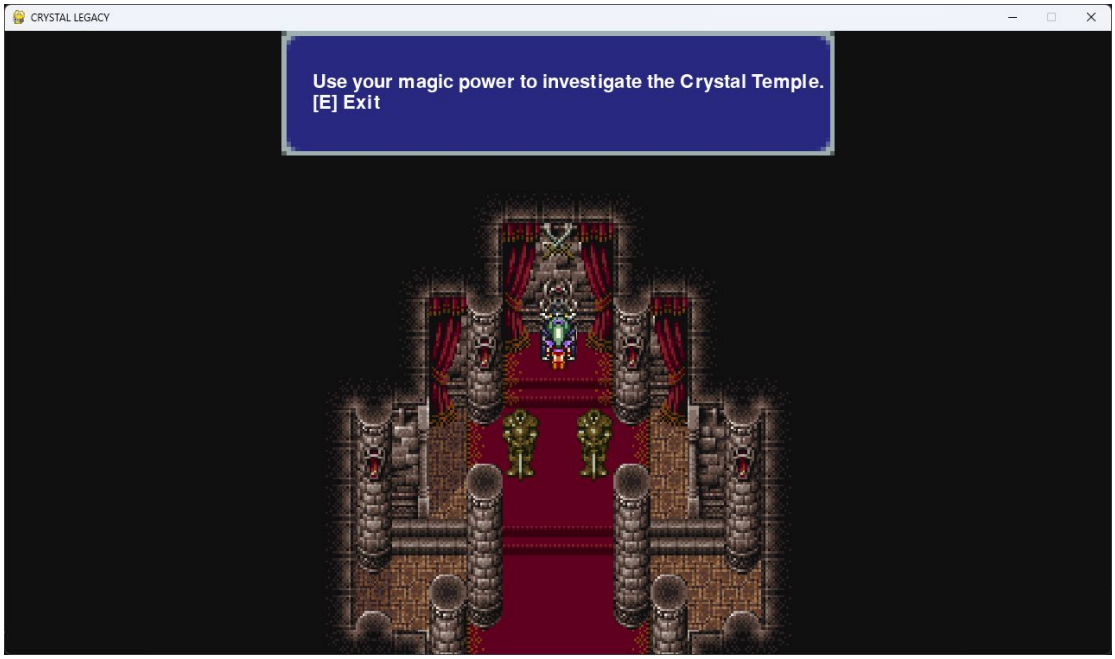
张恺昕

2023531029

zhangkx2023@shanghaitech.edu.cn

INTRODUCTION

拥有魔导之力的少女 **Sansan / 三三** 被国王 **Cid / 西德** 派遣调查位于西方的 **Crystal Temple / 水晶神殿**，并在 **Crystal Temple / 水晶神殿** 发现了 **Bahamut / 巴哈姆特** 的踪迹，他们之间会产生何种较量...



游戏采用 **ATB *(Action Time Battle)*** 战斗系统，战斗中玩家与怪物双方都具有随时间增长的 ATB 量表，ATB 量表满后可消耗 ATB 量表选择相应的战斗指令。

RUN

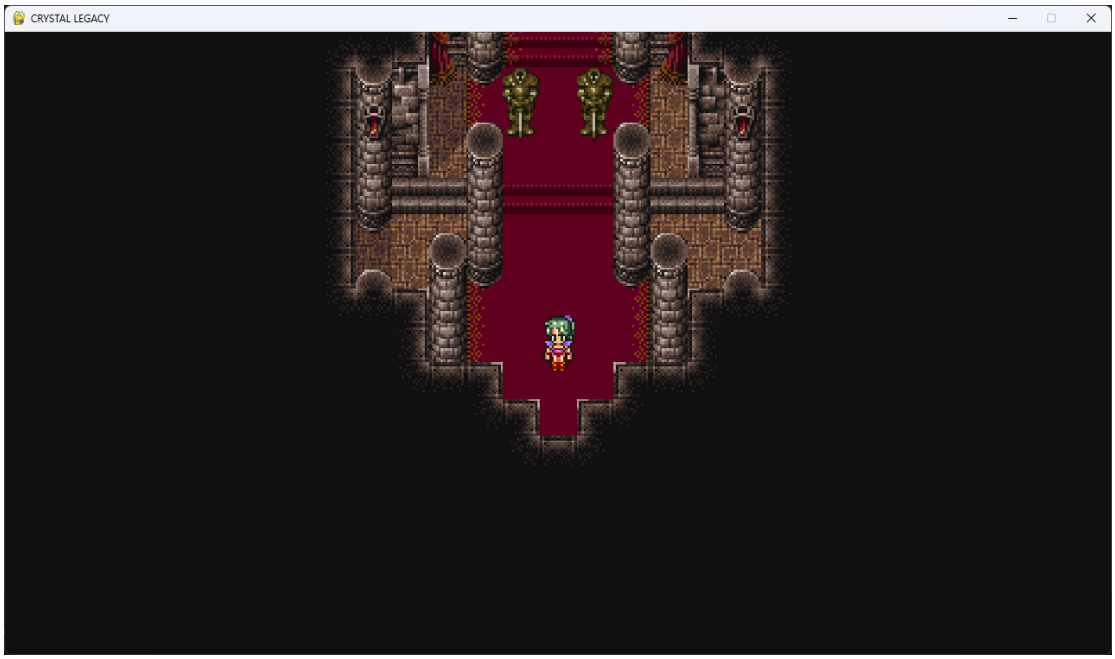
在电脑已安装 Python 与 Pygame 的状态下运行 Main.py。

游戏基于 1920 * 1080 分辨率制作，且在 1280 * 720 分辨率下运行，可在 Settings.py 中自行修改。

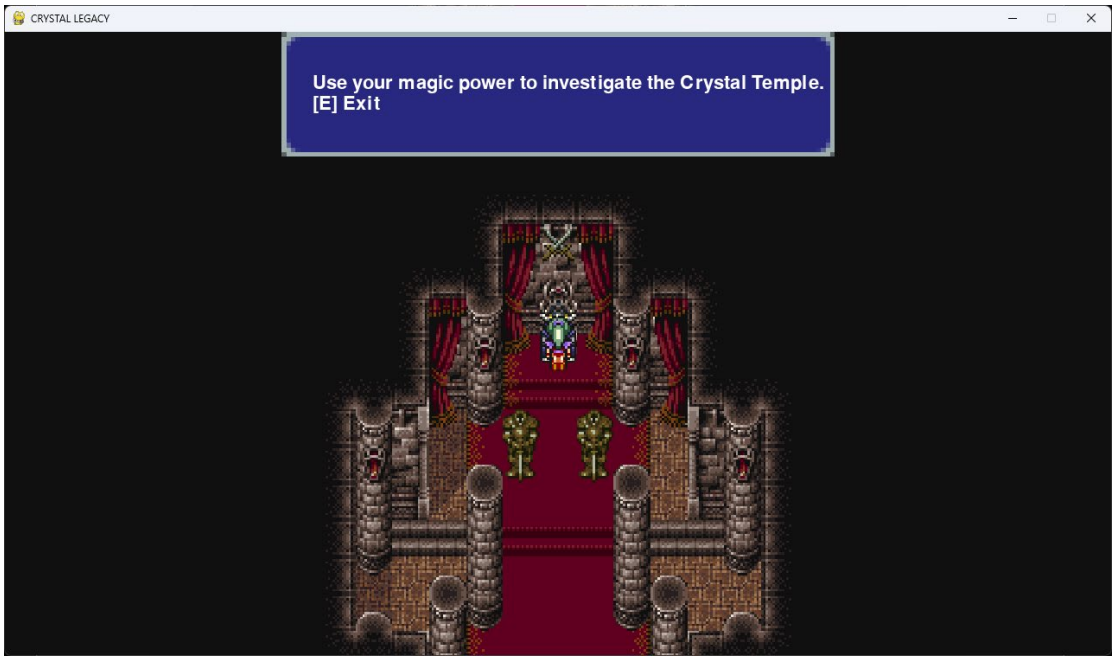
*※建议在 16:9 比例下运行，擅自改变分辨率可能会造成部分由于**四舍五入**的错误。*

HOW TO PLAY

在进入游戏后可通过按下[W] [A] [S] [D]按键移动，并可按下[Space]加速移动。



与 NPC 接触可触发对话，当与对话 NPC 对话时可按下[E]关闭对话框。



当与商人 NPC 对话时可使用[W] [S]上下移动选择条目，并按下[E]消耗金钱购买条目，按下[Q]关闭对话框。

与怪物接触可触发战斗，战斗中玩家 ATB 量表将随时间上涨。

当 ATB 量表满后，可使用[E]开启 Command 面板。

开启 Command 面板后，可使用[F]攻击，使用[R]开启魔法面板，使用[Q]关闭面板。

开启魔法面板后，可按下[Z]消耗 MP 使用 Fire 魔法，按下[X]消耗 MP 使用 Thunder 魔法，按下[Q]关闭面板。

击败怪物后，将回到地图并获得金币，怪物将消失。

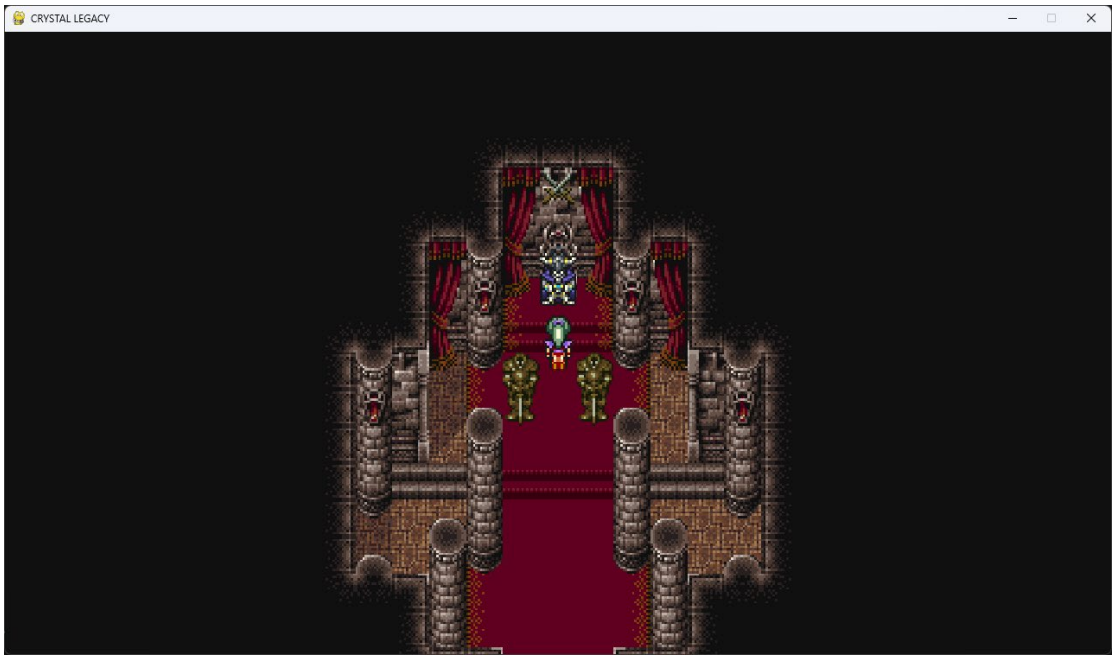
在玩家被击败后，将回到标题界面。

在击败 Boss 后，将回到标题界面并通关。

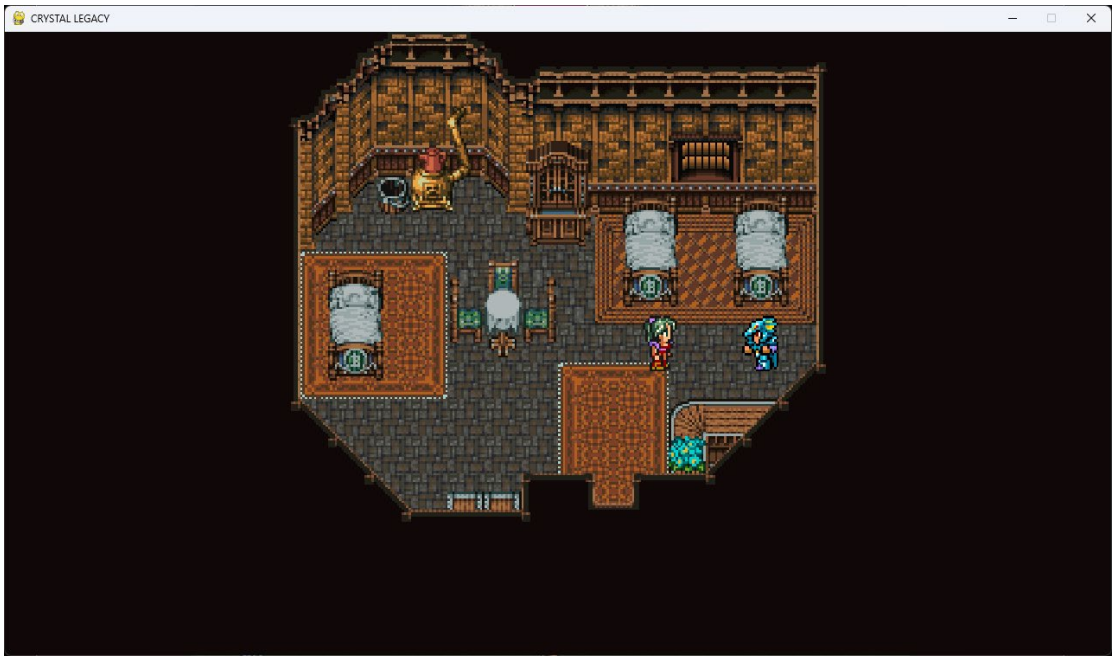
GAMEPLAY

游戏中存在四个场景。

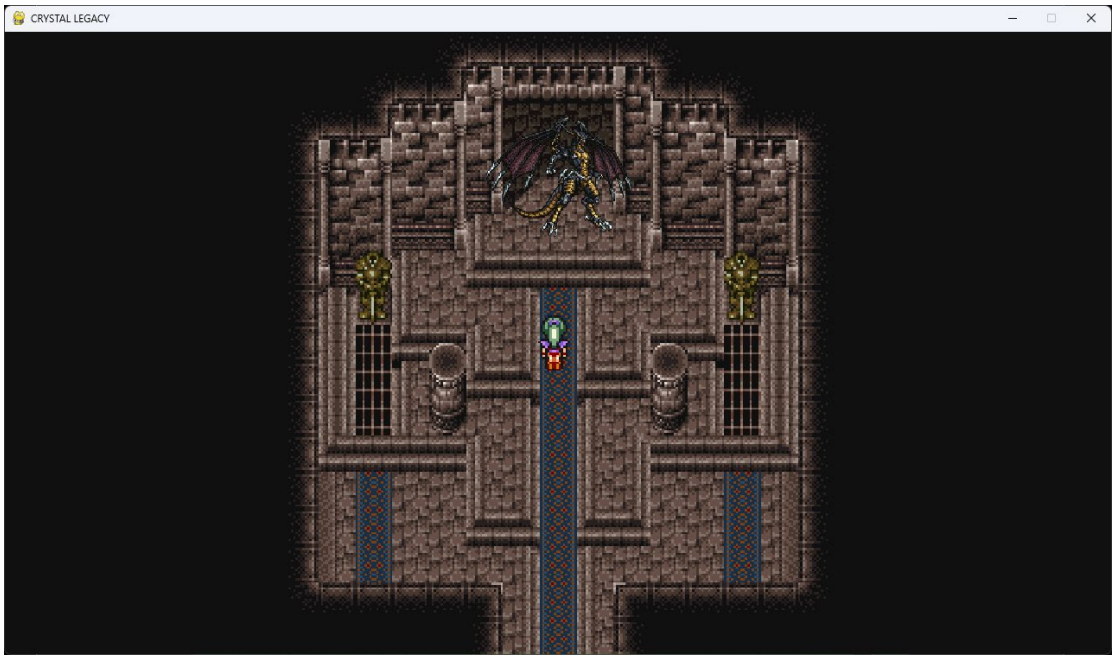
位于东方的场景 **Castle / 城堡** 中有可对话 NPC **Cid / 西德** 。



位于南方的场景 **Hut / 小屋** 中有可对话/交易 NPC **Knight / 骑士** 。



位于西方的场景 **Crystal Temple / 水晶神殿** 中有Boss级魔物 **Bahamut / 巴哈姆特** 。



Wild / 荒野 中有随机生成的障碍物与游荡的怪物。

CODE

※此处代码不包含全部内容

场景：

1. 场景镜头跟随

```
def update_camera(self, player):
        #设置 Camera 位置
        if player.x < WindowSettings.width // 2: #如果人物距左侧边框小于窗口大小一半
```

```
            self.cameraX = 0
        elif player.x > self.cameramaxX - WindowSettings.width // 2:
#如果人物距上侧边框小于窗口大小一半
            self.cameraX = self.cameramaxX - WindowSettings.width
        else:
            self.cameraX = player.x - WindowSettings.width // 2
        if player.y < WindowSettings.height // 2: #如果人物距上侧边框
小于窗口大小一半
            self.cameraY = 0
        elif player.y > self.cameramaxY - WindowSettings.height // 2:
#如果人物距下侧边框小于窗口大小一半
            self.cameraY = self.cameramaxY - WindowSettings.height
        else:
            self.cameraY = player.y - WindowSettings.height // 2
```

通过判断玩家与窗口位置，如果玩家坐标距地图边界超过窗口大小的一半，则设置摄像机位置与玩家位置相同，如果小于窗口大小一半，设置摄像机位置在不超过窗口边界的位置。地图中所有人物，物品渲染均在其原本坐标位置减去摄像机偏移位置。

2. 大场景

```
 def gen_wild_map(self):
        self.map = Tile(pygame.image.load(GamePath.groundTiles)) #读
取地面
        self.portal.add(Portal(SceneSettings.tileXnum *
SceneSettings.tileWidth * 61 // 64, SceneSettings.tileYnum *
SceneSettings.tileHeight // 2, 0))
        self.portal.add(Portal(SceneSettings.tileXnum *
SceneSettings.tileWidth * 3 // 64, SceneSettings.tileYnum *
SceneSettings.tileHeight // 2, 1))
        self.portal.add(Portal(SceneSettings.tileXnum *
SceneSettings.tileWidth // 2, SceneSettings.tileYnum *
SceneSettings.tileHeight * 27 // 32, 2))

    def gen_wild_obstacle(self):
        midx = SceneSettings.tileXnum // 2
        midy = SceneSettings.tileYnum // 2
        for i in range(SceneSettings.tileXnum):
            for j in range(SceneSettings.tileYnum):
                if random() < 0.05 and (not i in range(midx - 3, midx
+ 3) and not j in range (midy - 3, midy + 3)):
                    self.obstacles.add(Tile(pygame.image.load(GamePat
h.tree[randint(0,1)]), SceneSettings.tileWidth * i,
SceneSettings.tileHeight * j)) #在(i * tile width, j * tile height)处
添加障碍物

    def gen_WILD(self):
```

```
        self.gen_wild_map()
        self.gen_wild_obstacle()
        self.gen_monsters()
        self.music.update(SceneType.WILD)

    def gen_monsters(self, num = 10):
        while num > 0:
            self.monsters.add(Monster(randint(WindowSettings.width //
30, WindowSettings.width * 29 // 15), randint(WindowSettings.height
* 8 // 135, WindowSettings.height * 56 // 15), 1000, 25, 15,
randint(WindowSettings.width // 15, WindowSettings.width // 6)))
            num -= 1
```

将地面读取进入 Tile 内，并随机根据坐标生成障碍物，随机生成怪物位置。

3. 其他场景

```
 def gen_castle_obstacle(self):
        for i in [
            (WindowSettings.width * 5 // 12, WindowSettings.height *
388 // 45),
            (WindowSettings.width * 5 // 12, WindowSettings.height *
1172 // 135),
            (WindowSettings.width * 5 // 12, WindowSettings.height *
1196 // 135),
            (WindowSettings.width * 5 // 12, WindowSettings.height *
1204 // 135),
            (WindowSettings.width * 5 // 12, WindowSettings.height *
404 // 45),
            (WindowSettings.width * 5 // 12, WindowSettings.height *
1244 // 135),
            (WindowSettings.width * 9 // 20, WindowSettings.height *
236 // 27),
            (WindowSettings.width * 9 // 20, WindowSettings.height *
1252 // 135),
            (WindowSettings.width * 31 // 60, WindowSettings.height *
236 // 27),
            (WindowSettings.width * 31 // 60, WindowSettings.height *
1252 // 135),
            (WindowSettings.width * 11 // 20, WindowSettings.height *
388 // 45),
            (WindowSettings.width * 11 // 20, WindowSettings.height *
1172 // 135),
            (WindowSettings.width * 11 // 20, WindowSettings.height *
1196 // 135),
            (WindowSettings.width * 11 // 20, WindowSettings.height *
1204 // 135),
```

```
            (WindowSettings.width * 11 // 20, WindowSettings.height *
404 // 45),
            (WindowSettings.width * 11 // 20, WindowSettings.height *
1244 // 135)
            ]:
            self.obstacles.add(Tile(pygame.image.load(GamePath.emptyo
bstacles), i[0], i[1]))

    def gen_castle(self):
        self.gen_castle_obstacle()
        self.npcs.add(Cid(WindowSettings.width // 2,
WindowSettings.height * 388 // 45, "Cid", "Use your magic power to
investigate the Crystal Temple."))
        self.portal.add(Portal(WindowSettings.width // 2,
WindowSettings.height * 1256 // 135, 3))
        self.music.update(SceneType.CASTLE)
```

在限定整体地图坐标范围后根据坐标手动加入透明障碍物位置，再添加 NPC，Boss，传送门，音乐等内容。

4. 互动

```
def trigger_dialog(self, npc):
        if npc.talkCD == 0:
            self.istalking = True
            self.popupbox = DialogBox(self.window, npc)

    def end_dialog(self):
        self.istalking = False

    def trigger_shop(self, npc, player):
        if npc.talkCD == 0:
            self.istalking = True
            self.popupbox = ShoppingBox(self.window, npc, player)

    def end_shop(self):
        self.istalking = False
```

开启，关闭对话框，商店界面。

角色：

1. 玩家角色

```
def attr_update(self, addCoins = 0, addHP = 0, addMP = 0):
        if self.money + addCoins < 0:
            return
        elif self.HP < PlayerSettings.playerHP and self.HP + addHP >
PlayerSettings.playerHP:
            self.money += addCoins
```

```
            self.HP = PlayerSettings.playerHP
            return
        elif self.HP == PlayerSettings.playerHP and self.HP + addHP >
PlayerSettings.playerHP:
            return
        elif self.MP < PlayerSettings.playerMP and self.MP + addMP >
PlayerSettings.playerMP:
            self.money += addCoins
            self.MP = PlayerSettings.playerMP
            return
        elif self.MP == PlayerSettings.playerMP and self.MP + addMP >
PlayerSettings.playerMP:
            return
        elif self.MP + addMP < 0:
            return
        self.money += addCoins
        self.HP += addHP
        self.MP += addMP
```

更新玩家状态，并防止玩家数据超过上限。

```
def try_move(self, events, maxX = WindowSettings.width, maxY =
WindowSettings.height, minX = 0, minY = 0):
        self.dx = self.dy = 0
        #尝试移动
        if events[pygame.K_w]:
            self.move[0] = True
            if self.y - PlayerSettings.playerYSpeed >= minY +
PlayerSettings.playerHeight // 2:
                self.dy -= PlayerSettings.playerYSpeed
        if events[pygame.K_s]:
            self.move[1] = True
            if self.y + PlayerSettings.playerYSpeed <= maxY -
PlayerSettings.playerHeight //2:
                self.dy += PlayerSettings.playerYSpeed
        if events[pygame.K_a]:
            self.move[2] = True
            if self.x - PlayerSettings.playerXSpeed >= minX +
PlayerSettings.playerWidth // 2:
                self.dx -= PlayerSettings.playerXSpeed
        if events[pygame.K_d]:
            self.move[3] = True
            if self.x + PlayerSettings.playerXSpeed <= maxX -
PlayerSettings.playerWidth // 2:
                self.dx += PlayerSettings.playerXSpeed
        if events[pygame.K_SPACE]: #加速
```

```
            self.dx = self.dx * 2
            self.dy = self.dy * 2
        self.x += self.dx
        self.y += self.dy
        self.rect.center = (self.x, self.y)
```

根据输入按键在地图边界内移动，并载入移动方向数据以便于改变玩家图像。

```
def update(self, scene):
        #设置人物图像
        if self.move == [True, False, False, False] or self.move ==
[True, False, True, True]:
            self.index = (self.index + 1) % 12
            self.images = self.imageback
        elif self.move == [False, True, False, False] or self.move ==
[False, True, True, True]:
            self.index = (self.index + 1) % 12
            self.images = self.imagefront
        elif self.move == [False, False, True, False] or self.move ==
[True, False, True, False] or self.move == [False, True, True,
False] or self.move == [True, True, True, False]:
            self.index = (self.index + 1) % 12
            self.images = self.imageleft
        elif self.move == [False, False, False, True] or self.move ==
[True, False, False, True] or self.move == [False, True, False,
True] or self.move == [True, True, False, True]:
            self.index = (self.index + 1) % 12
            self.images = self.imageright
        else:
            self.index = 0
        self.image = self.images[self.index]
        self.move = [False, False, False, False]
        #设置人物位置
        testx = Player(self.x - self.dx, self.y)
        testy = Player(self.x, self.y - self.dy)
        if self.collide.is_colliding():
            if self.collide.collidingWith["obstacle"]:
                if pygame.sprite.spritecollide(testy,
scene.obstacles, False): #人物移动后碰撞但取消 X 方向移动后不碰撞
                    self.x -= self.dx
                if pygame.sprite.spritecollide(testx,
scene.obstacles, False): #人物移动后碰撞但取消 Y 方向移动后不碰撞
                    self.y -= self.dy
                self.rect.center = (self.x, self.y)
            if self.collide.collidingWith["portal"]:
                if self.collide.collidingObject["portal"] == 0:
```

```
                    self.event = GameEvent.EVENT_SWITCH
                    self.tpto = SceneType.CASTLE
                elif self.collide.collidingObject["portal"] == 1:
                    self.event = GameEvent.EVENT_SWITCH
                    self.tpto = SceneType.TEMPLE
                elif self.collide.collidingObject["portal"] == 2:
                    self.event = GameEvent.EVENT_SWITCH
                    self.tpto = SceneType.HUT
                elif self.collide.collidingObject["portal"] == 3 or
self.collide.collidingObject["portal"] == 4:
                    self.event = GameEvent.EVENT_SWITCH
                    self.tpto = SceneType.WILD
            if self.collide.collidingWith["npc"]:
                if self.collide.collidingObject["npc"].name == "Cid":
                    self.event = GameEvent.EVENT_DIALOG
                elif self.collide.collidingObject["npc"].name ==
"Knight":
                    self.event = GameEvent.EVENT_SHOP
            if self.collide.collidingWith["monster"]:
                self.event = GameEvent.EVENT_BATTLE
        if self.HP <=0:
            self.event = GameEvent.EVENT_RESTART
```

更新人物图像，并根据与障碍物碰撞取消移动（同时分别设置取消 X 移动与取消 Y 移动后的测试以便于分别取消 X 方向 Y 方向移动，达到对障碍物可以斜向擦边移动）根据碰撞到的传送门，NPC，怪物向 GameManager 发送相应的事件。

2. NPC

```
class Cid(NPC):
    def __init__(self, x, y, name, dialog):
        super().__init__(x, y, name)
        self.image =
pygame.transform.scale(pygame.image.load(GamePath.cid),
(NPCSettings.npcWidth, NPCSettings.npcHeight))
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)
        self.dialog = dialog

    def update(self):
        if self.talkCD != 0:
            self.talkCD -=1

class ShopNPC(NPC):
    def __init__(self, x, y, name):
        super().__init__(x, y, name)
```

```
        self.image =
pygame.transform.scale(pygame.image.load(GamePath.knight),
(NPCSettings.npcWidth, NPCSettings.npcHeight))
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)

    def update(self):
        if self.talkCD != 0:
            self.talkCD -=1
```

生成 NPC 图像，位置，并更新对话冷却。

3. 敌人

```
class Monster(pygame.sprite.Sprite):
    def __init__(self, x, y, HP = 10, Attack = 3, Money = 15, range =
WindowSettings.width // 15):
        super().__init__()
        self.name = "Sweeper"
        self.image =
pygame.transform.scale(pygame.image.load(GamePath.monster),
(MonsterSettings.monsterWidth, MonsterSettings.monsterHeight))
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)
        self.skillname = "Machine Gun"
        self.x = self.initialX = x
        self.y = y
        self.speed = MonsterSettings.monsterSpeed
        self.direction = 1
        self.patrollingRange = range
        self.HP = HP
        self.attack = Attack
        self.money = Money

    def update(self):
        self.x += self.speed * self.direction
        if abs(self.x - self.initialX) > self.patrollingRange:
            self.direction *= -1  # 反转方向
            self.image = pygame.transform.flip(self.image, True,
False)
        self.rect.center = (self.x, self.y)

    def attr_update(self, HP):
        self.HP += HP

    def draw(self, window, dx=0, dy=0):
        window.blit(self.image, self.rect.move(dx, dy))
```

```
class Boss(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.name = "Bahamut"
        self.image =
pygame.transform.scale(pygame.image.load(GamePath.boss),
(WindowSettings.width * 2 // 15, WindowSettings.height * 7 // 36))
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)
        self.skillname = "Megaflare"
        self.x = x
        self.y = y
        self.HP = 9999
        self.attack = 50
        self.money = 0

    def attr_update(self, HP):
        self.HP += HP

    def draw(self, window, dx=0, dy=0):
        window.blit(self.image, self.rect.move(dx, dy))
```

生成敌人图像，位置，设置游荡，更新状态。

游戏机制：

1. 战斗系统

```
def render(self):
        #UI
        self.window.blit(self.battlestatus,
(BattleSettings.statusStartX, BattleSettings.statusStartY)) #显示战斗
UI
        self.window.blit(self.monstername,
(BattleSettings.textMonsterStartX, BattleSettings.textStartY)) #显示
怪物名称
        self.window.blit(self.playername,
(BattleSettings.textPlayerStartX, BattleSettings.textStartY)) #显示玩
家名称
        HP = self.font.render(f"HP {self.player.HP} /
{PlayerSettings.playerHP}", True, self.fontcolor)
        MP = self.font.render(f"MP {self.player.MP} /
{PlayerSettings.playerMP}", True, self.fontcolor)
        self.window.blit(HP, (BattleSettings.textPlayerStatusStartX,
BattleSettings.textStartY)) #显示玩家 HP
```

```
        self.window.blit(MP, (BattleSettings.textPlayerStatusStartX,
BattleSettings.textStartY + BattleSettings.textVerticalDist)) #显示玩
家 MP
        self.window.blit(self.ATBbackground,
(BattleSettings.ATBStartX, BattleSettings.textStartY)) #显示玩家 ATB
条背景
        self.window.blit(pygame.transform.scale(self.ATBbar,
(WindowSettings.width * self.ATB // 2250, WindowSettings.height //
50)), (BattleSettings.ATBStartX, BattleSettings.textStartY))
        if self.ATB == 150 and not self.iscommanding:
            self.window.blit(self.CommandText,
(BattleSettings.ATBStartX, BattleSettings.textStartY +
BattleSettings.textVerticalDist)) #显示 Command 按键指引
        elif self.iscommanding:
            self.window.blit(self.commandbackground,
(BattleSettings.statusStartX, BattleSettings.statusStartY)) #显示
Command 面板 UI
            self.window.blit(self.CancelText,
(BattleSettings.ATBStartX, BattleSettings.textStartY +
BattleSettings.textVerticalDist)) #显示 Cancel 按键指引
            self.window.blit(self.AttackText,
(BattleSettings.textMonsterStartX, BattleSettings.textStartY)) #显示
Attack 按键指引
            self.window.blit(self.MagicText,
(BattleSettings.textMonsterStartX, BattleSettings.textStartY +
BattleSettings.textVerticalDist)) #显示 Magic 按键指引
            if self.ismagic:
                self.window.blit(self.commandbackground,
(BattleSettings.statusStartX, BattleSettings.statusStartY)) #显示
Magic 面板 UI
                self.window.blit(self.FireText,
(BattleSettings.textMonsterStartX, BattleSettings.textStartY)) #显示
Fire 魔法按键指引
                self.window.blit(self.ThunderText,
(BattleSettings.textMonsterStartX, BattleSettings.textStartY +
BattleSettings.textVerticalDist)) #显示 Thunder 魔法按键指引
        #怪物动画
        if self.monsterattackCD == 0:
            self.monsterrect.center = (BattleSettings.monsterCoordX,
BattleSettings.monsterCoordY)
            self.window.blit(self.monsterimage, self.monsterrect)
        else:
            self.monsterrect.center =
(BattleSettings.mosterattackCoordX, BattleSettings.monsterCoordY)
```

```
            self.window.blit(self.monsterimage, self.monsterrect)
            self.window.blit(self.monsterskillbackground,
(BattleSettings.skillStartX, BattleSettings.skillStartY))
            self.window.blit(self.monsterskill,
self.monsterskillrect)
        #玩家动画
        if not self.ismagic and self.attackCD == 0 and self.magicCD
== 0: #站立动画
            self.playerimage =
pygame.transform.scale(self.player.battlestandimage,
(BattleSettings.playerWidth, BattleSettings.playerHeight))
            self.playerimagerect = self.playerimage.get_rect(center =
(BattleSettings.playerCoordX, BattleSettings.playerCoordY))
            self.window.blit(self.playerimage, self.playerimagerect)
        elif self.ismagic: #咏唱魔法动画
            self.playerimage =
pygame.transform.scale(self.player.battlemagic(),
(BattleSettings.playerWidth, BattleSettings.playerHeight))
            self.playerimagerect = self.playerimage.get_rect(center =
(BattleSettings.playerCoordX, BattleSettings.playerCoordY))
            self.window.blit(self.playerimage, self.playerimagerect)
        elif not self.ismagic and self.attackCD != 0:
            self.playerimage =
pygame.transform.scale(self.player.battleattackimage,
(BattleSettings.playerWidth, BattleSettings.playerHeight))
            self.playerimagerect = self.playerimage.get_rect(center =
(BattleSettings.playerattackCoordX, BattleSettings.playerCoordY))
            self.window.blit(self.playerimage, self.playerimagerect)
        elif not self.ismagic and self.magicCD != 0: #使用魔法动画
            self.playerimage =
pygame.transform.scale(self.player.battleusemagicimage,
(BattleSettings.playerWidth, BattleSettings.playerHeight))
            self.playerimagerect = self.playerimage.get_rect(center =
(BattleSettings.playerCoordX, BattleSettings.playerCoordY))
            self.window.blit(self.playerimage, self.playerimagerect)
            self.window.blit(self.effect, self.effectrect)
```

根据战斗状态渲染 UI，人物，人物动画，怪物，特效。

```
def ATBmanage(self):
        if self.ATB != 150:
            self.ATB +=1
        if self.attackCD != 0:
            self.attackCD -=1
        if self.magicCD != 0:
            self.magicCD -=1
```

```
    def Update(self):
        if self.monsterCD > 0:
            self.monsterCD -= 1
        else:
            self.MonsterAttack()
            self.monsterCD = 300
        if self.monsterattackCD != 0:
            self.monsterattackCD -= 1
        if self.monster.HP <= 0:
            self.player.event = GameEvent.EVENT_END_BATTLE
            self.player.attr_update(self.monster.money)
            self.music.update(SceneType.WILD)
        if self.player.HP <= 0:
            self.player.event = GameEvent.EVENT_RESTART
```

更新人物 ATB 数据与动画计时，更新怪物 ATB，若怪物 HP 减少至 0 则向 GameManager 发送结束战斗指令，玩家 HP 减少至 0 则发送游戏重置指令。当玩家 ATB 小于 ATB 总量，每帧增加 ATB。

```
    def Attack(self):
        self.ATB = 0
        self.attackCD = 30
        self.monster.attr_update(- self.player.attack)

    def MagicFire(self):
        if self.player.MP > 0:
            self.ATB = 0
            self.magicCD = 60
            self.player.attr_update(0, 0, -5)
            self.monster.attr_update(- self.player.attack * 2)
            self.effect = self.effects[0]
            self.effectrect = self.effect.get_rect(center =
(BattleSettings.monsterCoordX, BattleSettings.monsterCoordY))

    def MagicThunder(self):
        if self.player.MP > 0:
            self.ATB = 0
            self.magicCD = 60
            self.player.attr_update(0, 0, -5)
            self.monster.attr_update(- self.player.attack * 2)
            self.effect = self.effects[1]
            self.effectrect = self.effect.get_rect(center =
(BattleSettings.monsterCoordX, BattleSettings.monsterCoordY))

    def MonsterAttack(self):
```

```
        self.player.attr_update(0, - self.monster.attack)
        self.monsterattackCD = 30
```

当玩家攻击与使用魔法时消耗相应的 ATB 与 MP，并完成玩家，怪物的状态更新，设置玩家动画时长，当怪物攻击时完成玩家状态更新，设置怪物动画时长

2. 碰撞系统

```
    def update_collide(self):
        #Obstacles
        if pygame.sprite.spritecollide(self.player,
self.scene.obstacles, False):
            self.player.collide.collidingWith["obstacle"] = True
        else:
            self.player.collide.collidingWith["obstacle"] = False

        #Portal
        if pygame.sprite.spritecollide(self.player,
self.scene.portal, False):
            self.player.collide.collidingWith["portal"] = True
            self.player.collide.collidingObject["portal"] =
pygame.sprite.spritecollide(self.player, self.scene.portal,
False)[0].index
        else:
            self.player.collide.collidingWith["portal"] = False
            self.player.collide.collidingObject["portal"] = None

        # Player -> NPCs; if multiple NPCs collided, only first is
accepted and dealt with.
        if pygame.sprite.spritecollide(self.player, self.scene.npcs,
False):
            self.player.collide.collidingWith["npc"] = True
            self.player.collide.collidingObject["npc"] =
pygame.sprite.spritecollide(self.player, self.scene.npcs, False)[0]
        else:
            self.player.collide.collidingWith["npc"] = False
            self.player.collide.collidingObject["npc"] = None

        # Player -> Monsters
        if pygame.sprite.spritecollide(self.player,
self.scene.monsters, False):
            self.player.collide.collidingWith["monster"] = True
            self.player.collide.collidingObject["monster"] =
pygame.sprite.spritecollide(self.player, self.scene.monsters,
True)[0]
        else:
            self.player.collide.collidingWith["monster"] = False
```

```
            self.player.collide.collidingObject["monster"] = None
```

根据人物碰撞设置目前碰撞状态与碰撞对象，在 Player 中更新并发送相应指令。

3. 资源系统

```
    def buy(self):
        if self.index == 0:
            self.player.attr_update(- 10, 100)
        elif self.index == 1:
            self.player.attr_update(- 5, 0, 10)

    def draw(self):
        self.window.blit(self.background, (ShopSettings.boxStartX,
ShopSettings.boxStartY))
        if self.index == 0:
            text0 = self.font.render("[E] " + self.text[0], True,
self.fontcolor)
            text1 = self.font.render(self.text[1], True,
self.fontcolor)
        elif self.index == 1:
            text0 = self.font.render(self.text[0], True,
self.fontcolor)
            text1 = self.font.render("[E] " + self.text[1], True,
self.fontcolor)
        quit = self.font.render("[Q] Quit", True, self.fontcolor)
        player = self.font.render("HP: " + str(self.player.HP) + "
MP: " + str(self.player.MP) + " Money: " + str(self.player.money),
True, self.fontcolor)
        self.window.blit(text0, (ShopSettings.textStartX,
ShopSettings.textStartY))
        self.window.blit(text1, (ShopSettings.textStartX,
ShopSettings.textStartY + ShopSettings.textVerticalDist))
        self.window.blit(quit, (ShopSettings.textStartX,
ShopSettings.textStartY + 2 * ShopSettings.textVerticalDist))
        self.window.blit(player, (ShopSettings.textStartX,
ShopSettings.textStartY + 3 * ShopSettings.textVerticalDist))
```

设置对话框，并根据选择与购买指令在 Player 中更新相应状态。

4. 其他

```
    def update_wild(self, events):
        # Deal with EventQueue First
        for event in events:
            if event.type == pygame.QUIT: #退出游戲
                pygame.quit()
                sys.exit()
        self.player.try_move(pygame.key.get_pressed(),
self.scene.maxX, self.scene.maxY) #嘗試移動
```

```
        if self.player.event == GameEvent.EVENT_SWITCH:
            self.flush_scene(self.player.tpto)
        elif self.player.event == GameEvent.EVENT_BATTLE:
            self.state = GameState.GAME_PLAY_BATTLE
            self.popupscene = MonsterBattle(self.window, self.player,
self.player.collide.collidingObject["monster"])
        self.player.event = None

        # Then deal with regular updates
        self.update_collide()
        self.player.update(self.scene)
        for monster in self.scene.monsters:
            monster.update()

    def update_battle(self, events):
        # Deal with EventQueue First
        for event in events:
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_e:
                    if self.popupscene.ATB == 150 and not
self.popupscene.iscommanding:
                        self.popupscene.iscommanding = True #开启
Command 面板
                elif event.key == pygame.K_q:
                    if self.popupscene.iscommanding and not
self.popupscene.ismagic:
                        self.popupscene.iscommanding = False #关闭
Command 面板
                    elif self.popupscene.iscommanding and
self.popupscene.ismagic:
                        self.popupscene.ismagic = False #关闭 Magic 面
板
                elif event.key == pygame.K_f:
                    if self.popupscene.iscommanding and not
self.popupscene.ismagic:
                        self.popupscene.Attack() #Attack
                        self.popupscene.iscommanding = False
                elif event.key == pygame.K_r:
                    if self.popupscene.iscommanding and not
self.popupscene.ismagic:
                        self.popupscene.ismagic = True #开启 Magic 面板
```

```
                elif event.key == pygame.K_z:
                    if self.popupscene.iscommanding and
self.popupscene.ismagic:
                        self.popupscene.MagicFire() #使用 Fire 魔法
                        self.popupscene.ismagic = False
                        self.popupscene.iscommanding = False
                elif event.key == pygame.K_x:
                    if self.popupscene.iscommanding and
self.popupscene.ismagic:
                        self.popupscene.MagicThunder() #使用 Thunder 魔
法
                        self.popupscene.ismagic = False
                        self.popupscene.iscommanding = False
        if self.player.event == GameEvent.EVENT_END_BATTLE:
            self.state = GameState.GAME_PLAY_WILD
            return
        elif self.player.event == GameEvent.EVENT_RESTART:
            self.game_reset()
            return
        self.player.event = None

        # Then deal with regular updates
        self.popupscene.ATBmanage() #管理 ATB
        self.popupscene.Update()
```

根据玩家按键发出相应指令，当有事件指令时执行相应事件内容，完成每帧更新内容。

5. 渲染

   所有渲染位置都根据渲染对象原本坐标位置减去摄像机位置以完成画面的偏移。

游玩过程：

1. 主菜单

```
    def __init__(self, window):
        self.window = window
        self.background =
pygame.transform.scale(pygame.image.load(GamePath.menu),
(WindowSettings.width, WindowSettings.height))
        self.font = pygame.font.Font(None,
int(pow(WindowSettings.width * WindowSettings.height, 0.5) // 30)) #
主界面文字大小
        self.text = self.font.render("PRESS ANY BUTTON", True, (0, 0,
0)) #主界面文字内容 1
        self.textRect =
self.text.get_rect(center=(WindowSettings.width // 2,
WindowSettings.height * 3 // 4)) #主界面文字位置
        self.music = BgmPlayer()
```

```
        self.music.update(SceneType.TITLE)

    def render(self):
        self.window.blit(self.background, (0, 0))
        self.window.blit(self.text, self.textRect) #显示主界面文字 1
```

读取主菜单画面，文字，音乐。

2. 音乐

```
    def play(self, name, loop=-1):
        pygame.mixer.music.load(self.music[name])
        pygame.mixer.music.play(loop)

    def stop(self):
        pygame.mixer.music.stop()

    def update(self, GOTO):
        self.stop()
        if GOTO == SceneType.CASTLE:
            self.play("castle")
        elif GOTO == SceneType.WILD:
            self.play("wild")
        elif GOTO == SceneType.TEMPLE:
            self.play("temple")
        elif GOTO == SceneType.HUT:
            self.play("hut")
        elif GOTO == SceneType.TITLE:
            self.play("prelude")
        elif GOTO == SceneType.BATTLE:
            self.play("battle")
        elif GOTO == SceneType.BOSS:
            self.play("boss")
```

读取，暂停，播放音乐。

CREDITS

游戏中音乐由 **Nobuo Uematsu / 植松伸夫**，**Masashi Hamauzu / 滨涡正志**，**Yoshitaka Suzuki / 铃木克崇**, **Ryo Yamazaki / 山崎良**, **Shotaro Shima / 岛翔太朗**制作，LOGO 插画与人物设计为 **Yoshitaka Amano / 天野喜孝**，怪物设计为 **Tetsuya Nomura / 野村哲也**，原案设计，剧本，编程以及其他所有内容制作为 **Kaixin "Nontown_Notch" Zhang / 张恺昕**。

STAFF

Director / 导演
Kaixin "Nontown_Notch" Zhang / 张恺昕

Concept Design / 原案设计
Kaixin "Nontown_Notch" Zhang / 张恺昕

Scenario / 剧本
Kaixin "Nontown_Notch" Zhang / 张恺昕

Art Director / 艺术指导
Kaixin "Nontown_Notch" Zhang / 张恺昕

Programming / 编程
Kaixin "Nontown_Notch" Zhang / 张恺昕

Character Design / 角色设计
Yoshitaka Amano / 天野喜孝（*Yoshitaka Amano Office*）

Logo Illustration / Logo 插画
Yoshitaka Amano / 天野喜孝（*Yoshitaka Amano Office*）

Monster Design / 怪物设计
Tetsuya Nomura / 野村哲也（*Square Enix Co., Ltd*）

Battle Director / 战斗导演
Kaixin "Nontown_Notch" Zhang / 张恺昕

Environment Director / 环境导演
Kaixin "Nontown_Notch" Zhang / 张恺昕

Graphics Director / 图形导演
Kaixin "Nontown_Notch" Zhang / 张恺昕

Music / 音乐
Nobuo Uematsu / 植松伸夫（*Dog Ear Records Co., Ltd*）
Masashi Hamauzu / 滨涡正志（*Monomusik Inc.*）
Yoshitaka Suzuki / 铃木克崇（*Square Enix Co., Ltd*）
Ryo Yamazaki / 山崎良（*Square Enix Co., Ltd*）
Shotaro Shima / 岛翔太朗（*Square Enix Co., Ltd*）

Producer / 制作人
Kaixin "Nontown_Notch" Zhang / 张恺昕