รายงานประกอบ Project using principle of OOP

เรื่อง โปรเจกต์การสร้างเกมในรูปแบบ Survival Game 2 มิติจาก Pygame

จัดทำโดย

นายเกียรติพงษ์	ชูกรณ์	รหัสนักศึกษา 630910154
นายกฤตเมธ	แจ้งสนิท	รหัสนักศึกษา 630910305 **(ผู้จัดทำ)**
นายโชคชัย	สุขสำราญ	รหัสนักศึกษา 630910862
นายพีรณัฐ	ทุมนัส	รหัสนักศึกษา 630910872 **(ผู้จัดทำ)**
นายนฤกวินทร์	ประจันทร์นวล	รหัสนักศึกษา 630910951

เสนอ

อาจารย์ ดร.ศักดิ์ระพี ไพศาลนันทน์

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 618445 การออกแบบระบบเชิงวัตถุสำหรับวิศวกร

คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยศิลปากร

ภาคการศึกษาปลาย ปีการศึกษา 2566

ที่มาและความสำคัญเบื้องต้น

โปรเจกต์การสร้างเกมนี้ใช้เป็นส่วนหนึ่งของการประยุกต์ใช้ความรู้ด้านการออกแบบเชิงวัตถุ (OOP) สำหรับวิศวกรมาใช้ร่วมกับไลบรารี Pygame เพื่อใช้สร้างเป็นเกม 2 มิติขึ้นในรูปแบบ Survival Game โดยอ้างอิงจากเกม Stardew Valley, Terraria และ Rim world

หมายเหตุ

การพัฒนาโปรเจกต์เกมนี้ถูกพัฒนาอย่างต่อเนื่องเป็นระยะเวลายาว 2 เดือนตั้งแต่เดือนกุมภาพันธ์ (สังเกตได้ตาม commit) และยังคงพัฒนาอย่างต่อเนื่องเพื่อแก้ไขปัญหาที่พบในเกม แต่ตัวโปรเจกต์ขาดการดูแลช่วยเหลือจากสมาชิกภายในกลุ่มแม้ว่าสมาชิกภายในกลุ่มได้รับลิงค์ github และสิทธิ์ในการเข้าถึงแก้ไข จึงแก้ปัญหาโดยการแบ่งจัดสรรหน้าที่ใหม่ให้แก่สมาชิกภายในกลุ่มรับผิดชอบรายงานและสร้างในส่วนของแผนที่เกม แต่สมาชิกภายในกลุ่มที่ได้รับหน้าที่ใหม่ก็ไม่สามารถสร้างแผนที่ใหม่และรับผิดชอบในส่วนของรายงานได้ จึงเหลือเพียงสมาชิก 2 คนที่คอยดูแลในส่วนของรายงานและแก้ไขโค้ดการทำงาน วาดภาพองค์ประกอบภายในเกมและสร้างตัวละครขึ้นเอง พยายามปรับปรุงข้อบกพร่องที่พบเจออยู่อย่างสม่ำเสมอ หากพบเจอข้อผิดพลาดภายในรายงาน ข้อบกพร่องและความผิดพลาดในการเขียนรายงานร่วมกับโค้ดที่ใช้ภายในเกมต้องขออภัยเป็นอย่างสูง ตัวผู้เขียนรายงานและผู้พัฒนาตั้งใจพัฒนาทุกส่วนให้ออกมาใช้งานได้จริงและเรียบเรียงลงรายงานให้ครบถ้วนมากที่สุด

องค์ประกอบอุปกรณ์ที่ใช้

- PyGame ใช้กำหนดฟังก์ชันต่างๆในเกมพื้นฐาน ทั้ง sprite game, ขนาดหน้าต่าง, text บ่งบอกข้อความ, กำหนดสีภายในเกม และสร้างเวกเตอร์และขนาด magnitude พื้นฐานเพื่อใช้อ้างอิงทิศทาง
- PyTMX เพื่อใช้โหลดแม็พพื้นฐาน กำหนดขอบเขตอาณาเขตของพื้นที่ และกำหนด Event trigger สำหรับ cut scene maploading ต่างๆ
- SpriteSheet ผู้เล่น (วาดเอง)
- GameSprite รูปภาพประกอบแม็พแผนที่เกม

คู่มือการติดตั้งบนเว็บ Github

ภายใน repository ของโปรเจกต์มีคู่มือการติดตั้งให้ร่วมอยู่ภายใน สามารถติดตั้งตาม requirement ของโปรเจกต์ได้ตามสะดวก และควรเวอร์ชันของไลบรารีที่ใช้ภายในควรสัมพันธ์กันกับเวอร์ชันของ Python เพื่อให้ไม่เกิดปัญหาขัดข้องใดๆ

โครงสร้างการทำงานภายในเกม

ฟังก์ชันกำหนดการทำงานหลักของทุกไฟล์ในโปรเจกต์คือฟังก์ชันที่อยู่ในไฟล์ main.py ซึ่งเป็นฟังก์ชันหลักในการ initialize เริ่มต้นการทำงานและเป็นการเรียกใช้ Pygame Initialize เริ่มต้นการทำงาน และเรียกใช้ class ต่างๆภายในที่ import เข้ามาจากไฟล์อื่นแล้วเก็บในรูปของตัวแปรเรียกใช้งาน

อธิบายโค้ดการทำงานประกอบร่วมกับการประยุกต์ใช้การออกแบบเชิงวัตถุ (OOP)

```python
import pygame, sys
from level import Level
from settings import *

# Create an game class with init inside
You, 2 months ago | 1 author (You)
class Game:
    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
        pygame.display.set_caption('Prison Island')
        self.clock = pygame.time.Clock()
        self.level = Level()

    def run(self):
        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()

            dt = self.clock.tick() / 1000
            self.level.run(dt)
            pygame.display.update()

            keys = pygame.key.get_pressed()
            if keys[pygame.K_LCTRL] and keys[pygame.K_q]:
                pygame.quit()
                sys.exit()
```

```python
if __name__ == '__main__':
    game = Game()
    game.run()
```

[ภายในไฟล์ main.py]

```python
from pygame.math import Vector2
# screen
SCREEN_WIDTH = 1280
SCREEN_HEIGHT = 720
TILE_SIZE = 64

# Define some of the colors
BlackBG = (0, 0 ,0)
White = (255, 255, 255)

# overlay positions
OVERLAY_POSITIONS = {
    'tool' : (40, SCREEN_HEIGHT - 15),
    'seed': (70, SCREEN_HEIGHT - 5)}

PLAYER_TOOL_OFFSET = {
    'left': Vector2(-50,40),
    'right': Vector2(50,40),
    'up': Vector2(0,-10),
    'down': Vector2(0,50)
}

LAYERS = {
    'water': 0,
    'ground': 1,
    'soil': 2,
    'soil water': 3,
    'rain floor': 4,
    'house bottom': 5,
    'ground plant': 6,
    'main': 7,
```

1. สร้าง class Game() ขึ้นเพื่อจัดรูปโค้ดภายใน โดยเรียกใช้ pygame.init() เพื่อเริ่มต้นรันการทำงานของ pygame ขึ้นและกำหนดขนาดหน้าจอตามขนาดบนไฟล์ setting.py ซึ่งก็คือ 1280 x 720 และกำหนดชื่อเกมขึ้น พร้อมกับเรียกใช้ clock เพื่อจัดเก็บเวลานับจำนวนวินาทีเวลาในเกม เรียกใช้ class Level() จากไฟล์ level โดยกำหนดให้เป็น instance ของตัวแปร self.level กำหนดฟังก์ชัน run เพื่อให้ตัวเกมสามารถรันผลการทำงานอยู่ได้และกำหนดลูปให้เมื่อมี event ที่เป็นประเภท pygame.quit() จะทำการออกเกมทันทีเช่นเดียวกันกับเมื่อตรวจพบการกดปุ่ม CTRL ฝั่งซ้ายร่วมกับปุ่ม Q จะทำการออกเกมทันที และกำหนดตัวแปล dt ให้เป็น delta Time ของเวลาและใช้เป็นตัวแปรเพื่อรัน level และอัพเดตการทำงานของฟังก์ชัน

```python
import pygame
from settings import *
from player import Player
from overlay import Overlay
from sprite import Generic, Water, Tree, Interaction
from pytmx.util_pygame import load_pygame
from transition import Transition

You, 2 months ago | 1 author (You)
class Level:
    def __init__(self):
        # get the display surface
        self.display_surface = pygame.display.get_surface()

        # sprite groups
        self.all_sprites = CameraGroup()
        # Inherited
        self.collision_sprites = pygame.sprite.Group()
        self.tree_sprites = pygame.sprite.Group()
        self.interaction_sprites = pygame.sprite.Group()

        self.setup()
        self.overlay = Overlay(self.player)
        self.transition = Transition(self.reset, self.player)
```

```python
class Level:
    def setup(self):

        self.player = Player((640,360), self.all_sprites, self.collision_sprites)
        # we pass collision in player as parameter
        # but we pass collision in object as a container inside!
```

```python
class Level:
    def setup(self):

        # Instance of Generic
        Generic(pos = (0, 0),
                surf = pygame.image.load('./Maps/Island.png').convert_alpha(),
                groups = self.all_sprites,
                z = LAYERS['ground'])

    def player_add(self, item):
        self.player.item_inventory[item] += 1

    def reset(self):
        #  reset the world and create new apple and tree
        self.new = APPLE_POS

    def run(self, dt):
        self.display_surface.fill('white')
        # self.all_sprites.draw(self.display_surface)
        self.all_sprites.custom_draw(self.player)
        self.all_sprites.update(dt)

        self.overlay.display()
```

[ภายในไฟล์ level.py]

2. ภายใน class Level() เปรียบเสมือน class กำหนดแผนที่ในเกมที่เป็นทั้งด่านและเป็นตัวกำหนด surface ชั้นที่แสดงออกของเกม(นึกถึงรูปแบบ z-index ในแต่ละชั้น) ตัวรวมกลุ่ม sprite ภายในผ่าน class CameraGroup() แบ่งกลุ่มให้เป็นตัว collision กับวัตถุและภาพ sprite ต่างๆ และมีการเก็บไว้เป็นตัวแปร instance ภายในโดยเรียกใช้ผ่าน class ทั้ง class Overlay() และ class Transition() เมื่อเรียก method setup จะพบว่าตัวแปร player ถูกใช้งานผ่าน class Player() โดยกำหนดตำแหน่งเริ่มต้นไว้ที่จุด x=640, y=360 ส่งผ่าน instance สำหรับ sprite และ collision ผ่าน class นอกจากนั้นจะเรียก class Generic() กำหนดตำแหน่งของแผนที่พื้นหลังแสดงผล(map ของเกม) ร่วมกับกำหนดเลเยอร์และแบ่งกลุ่มชนิดเอาไว้
และภายใน class Level() ยังมี method อื่นๆ ได้แก่

- player_add ที่ทำหน้าที่เพิ่มปริมาณไอเทมที่ได้จากการตัดต้นไม้
- reset เพื่อใช้กำหนดตำแหน่งของแอปเปิลและต้นไม้ภายในเกม
- run ใช้ dt ในการอัพเดต frame ของ sprite แสดงผลผู้เล่นและถมพื้นหลังเป็นสีขาวในส่วนที่แผนที่ไม่ได้ครอบคลุมหรือถมแสดงพื้นหลังที่อยู่ฝั่งหลังสุดเป็นสีขาว

```python
class CameraGroup(pygame.sprite.Group):
    def __init__(self):
        super().__init__()
        self.display_surface = pygame.display.get_surface()
        self.offset = pygame.math.Vector2()


    def custom_draw(self, player):
        # Set player to be in middle
        self.offset.x = player.rect.centerx - SCREEN_WIDTH / 2
        self.offset.y = player.rect.centery - SCREEN_HEIGHT / 2

        # draw map opposite of moving
        for layer in LAYERS.values():
            for sprite in sorted(self.sprites(), key = lambda sprite: sprite.rect.centery):
                if sprite.z == layer:
                    offset_rect = sprite.rect.copy()
                    offset_rect.center -= self.offset
                    self.display_surface.blit(sprite.image, offset_rect)
```

2.1 class CameraGroup() ใช้การ Inherited มาจาก pygame.sprite.group เพื่อเรียกใช้ตัวแปร parent ภายในผ่านการ super Init และเพื่อเรียกใช้การดึงผิวแสดงผลผ่าน pygame.display.get_surface() และจัดเก็บจุด offset ผ่านในรูปแบบของ pygame.math.vector2 หรือเป็นเวกเตอร์ และมี custom_draw method ในการกำหนดค่ามุมมองกล้องจากด้านบนเพื่อให้มุมกล้องจากด้านบนสามารถยึดมุมมองไว้กับตัวผู้เล่นให้อยู่อยู่ตรงกลางของจอ อีกทั้งในขณะที่ผู้เล่นขยับไปในทิศทางไหนก็ตามการเคลื่อนไหวของมุมมองกล้องต้องไปในทิศตรงกันข้ามเพื่อสร้างมุมการเคลื่อนไหวในทิศนั้นๆ

```python
import pygame
from settings import *

You, 2 months ago | 1 author (You)
class Overlay:
    def __init__(self, player):

        # general setup
        self.display_surface = pygame.display.get_surface()
        self.player = player

        #set path for tool overlay in screen
        overlay_path = './Sprite/overlay/'
        self.tools_surf = {tool:pygame.image.load(f'{overlay_path}{tool}.png').convert_alpha() for tool in player.tools}

    def display(self):

        # import tools for displaying
        tool_surf = self.tools_surf[self.player.selected_tool]
        tool_surf = pygame.transform.scale(tool_surf, (50, 50))
        tool_rect = tool_surf.get_rect(midbottom = OVERLAY_POSITIONS['tool'])
        self.display_surface.blit(tool_surf, tool_rect)
```

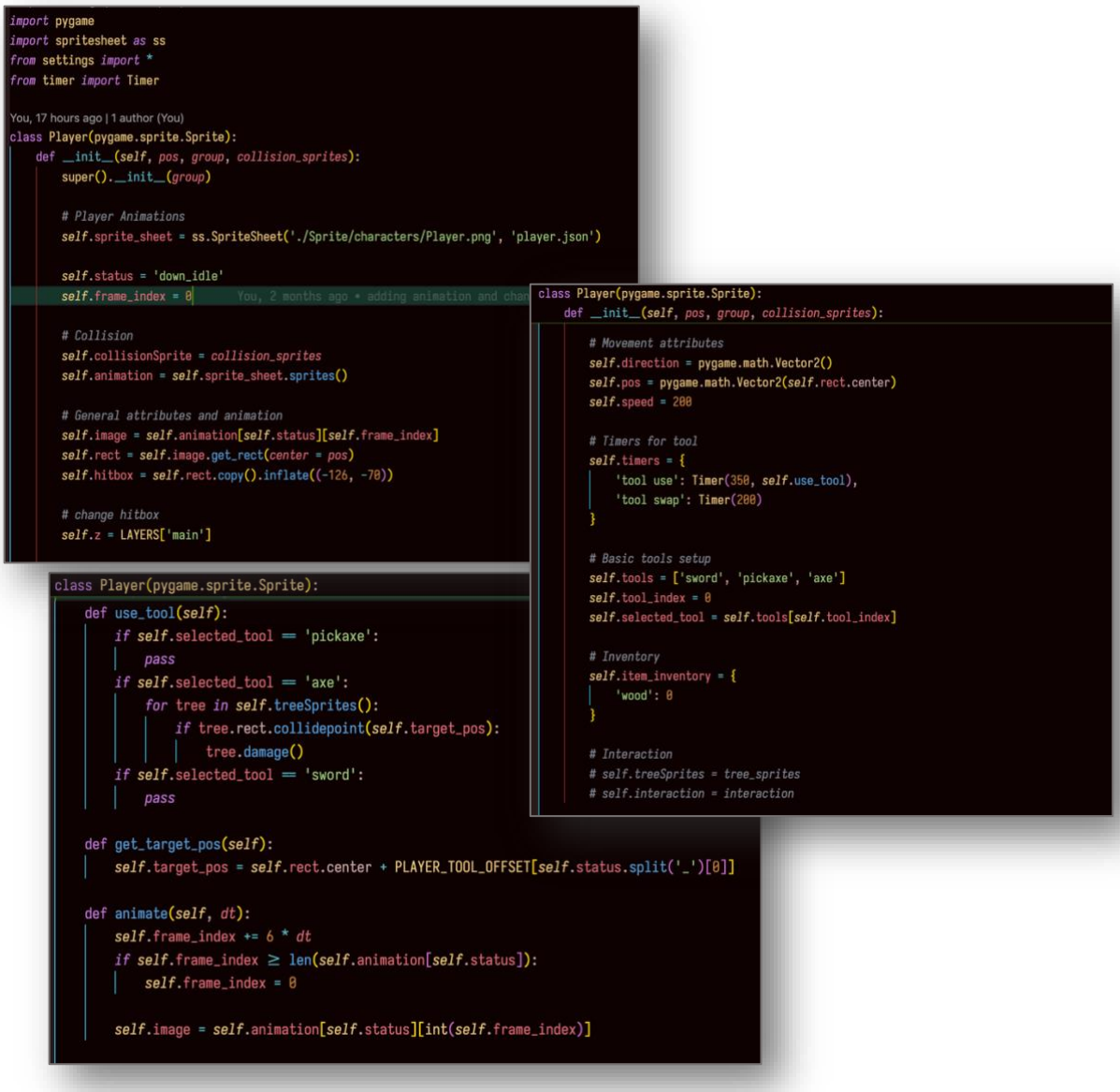
[ภายในไฟล์ overlay.py]

3. class Overlay() เป็นตัวทำหน้าที่แสดงอุปกรณ์ไอเทมที่ผู้เล่นกำลังสวมใสอยู่ ซึ่งสามารถเห็นได้จากมุมซ้ายล่างของผู้เล่น โดยที่เราจำกำหนดตัว display_surface ให้ใช้ประโยชน์จากภายใน pygame.display.surface สำหรับดึงเลเยอร์แสดงผล และกำหนดตัวแปร player ให้เป็น parameter ไว้ใช้งานหลังจากนั้นทำการลูปผ่านไฟล์ภาพ sprite ภายในโฟลเดอร์ overlay เพื่อใช้แสดงผลอุปกรณ์ไอเทมที่ผู้เล่นถืออยู่โดยอาศัย method display ในการแสดงผลบนชั้นเลเยอร์ที่กำหนดบน surface พร้อมทั้งปรับขนาดของสเกลรูปและจัดการรูปแบบการ overlay

```python
import pygame
from settings import *

...
class Transition:
    def __init__(self, reset, player):
        #  basic setup
        self.display_surface = pygame.display.get_surface()
        self.reset = reset
        self.player = player

        # Overlay image
        self.image = pygame.Surface((SCREEN_WIDTH, SCREEN_HEIGHT))
        self.color = 255
        self.speed = -2

    def play(self):
        self.color += self.speed
        if self.color <= 0:
            self.speed *= -1
            self.color = 0
            self.reset()
        if self.color > 255:
            self.color = 0
            self.player.sleep = False
            self.speed = -2

        self.image.fill((self.color, self.color, self.color))
        self.display_surface.blit(self.image, (0,0), special_flags = pygame.BLEND_RGBA_MULT)
```

[ภายในไฟล์ transition.py]

4. class Transition() เป็น class ที่ใช้แสดงผลการตัด cutscene, การเริ่มต้นใหม่ของเกมเมื่อผู้เล่นแพ้ และใช้สำหรับคั่นหน้าเกมเพื่อให้เหมือนการตัดช่วงเกม โดยอาศัยการถมหน้าต่างเป็นสีขาวชั่วเวลาขณะหนึ่งเพื่อเป็นการดีเลย์สำหรับผู้เล่น โดยอาศัยหลักการแสดงผลบน surface ที่อยู่ด้านหน้าเพื่อบังหน้าจอชั่วขณะหนึ่งผ่านตัวแปร surface และกำหนดการ reset ในส่วนของ map และตัวผู้เล่นเองด้วยจากนั้นกำหนดสีพื้นหลัง ความเร็วในการ transition สร้างเป็น method play เมื่อเกิดการ transition จะเปลี่ยนทิศทางของสีสลับกันและตรวจสอบสีปัจจุบันให้อยู่ในสีที่ตรงกันข้ามกับเฉดสีปัจจุบันพร้อมทั้งเรียกการรีเซ็ตเริ่มต้นใหม่ ทำการถมหน้าจอเป็นสีที่ color ปัจจุบันและแสดงผลทับบน surface



```python
import pygame
import spritesheet as ss
from settings import *
from timer import Timer

You, 17 hours ago | 1 author (You)
class Player(pygame.sprite.Sprite):
    def __init__(self, pos, group, collision_sprites):
        super().__init__(group)

        # Player Animations
        self.sprite_sheet = ss.SpriteSheet('./Sprite/characters/Player.png', 'player.json')

        self.status = 'down_idle'
        self.frame_index = 0        You, 2 months ago • adding animation and char

        # Collision
        self.collisionSprite = collision_sprites
        self.animation = self.sprite_sheet.sprites()

        # General attributes and animation
        self.image = self.animation[self.status][self.frame_index]
        self.rect = self.image.get_rect(center = pos)
        self.hitbox = self.rect.copy().inflate((-126, -70))

        # change hitbox
        self.z = LAYERS['main']
```

```python
class Player(pygame.sprite.Sprite):
    def __init__(self, pos, group, collision_sprites):

        # Movement attributes
        self.direction = pygame.math.Vector2()
        self.pos = pygame.math.Vector2(self.rect.center)
        self.speed = 200

        # Timers for tool
        self.timers = {
            'tool use': Timer(350, self.use_tool),
            'tool swap': Timer(200)
        }

        # Basic tools setup
        self.tools = ['sword', 'pickaxe', 'axe']
        self.tool_index = 0
        self.selected_tool = self.tools[self.tool_index]

        # Inventory
        self.item_inventory = {
            'wood': 0
        }

        # Interaction
        # self.treeSprites = tree_sprites
        # self.interaction = interaction
```

```python
class Player(pygame.sprite.Sprite):
    def use_tool(self):
        if self.selected_tool == 'pickaxe':
            pass
        if self.selected_tool == 'axe':
            for tree in self.treeSprites():
                if tree.rect.collidepoint(self.target_pos):
                    tree.damage()
        if self.selected_tool == 'sword':
            pass

    def get_target_pos(self):
        self.target_pos = self.rect.center + PLAYER_TOOL_OFFSET[self.status.split('_')[0]]

    def animate(self, dt):
        self.frame_index += 6 * dt
        if self.frame_index ≥ len(self.animation[self.status]):
            self.frame_index = 0

        self.image = self.animation[self.status][int(self.frame_index)]
```

[ภายในไฟล์ player.py]

5. class Player() จะเป็น class ที่ได้รับการ Inherited จาก pygame.sprite.Sprite เพื่อรับเอาในส่วนของการ sprite ในการแสดงผลรูปและจัดกลุ่มกันภายใน ซึ่ง class นี้จะมีองค์ประกอบเยอะและจำเป็นต้องสร้าง method ในการใช้งานให้สัมพันธ์กันกับผู้เล่น โดยเริ่มต้นที่การกำหนด spriteSheet ของผู้เล่นผ่าน class SpriteSheet() เพื่อโหลดเอารูปภาพแสดงผลตัวละครด้วยการ map สถานะเข้าด้วยกันกับไฟล์ Json จากนั้นกำหนดสถานะเริ่มต้นเป็น ‘การยืนนิ่งปกติในทิศทางหันหน้าลงด้านล่าง’ กำหนด frame_index เพื่อใช้เริ่มต้นและกำหนด collision, animation สถานะแสดงผลรูปปัจจุบัน, hitbox ของผู้เล่น, เลเยอร์ปัจจุบัน, ทิศทาง ตำแหน่ง และความเร็วของผู้เล่น, ระยะเวลาใช้อุปกรณ์ไอเทม, อุปกรณ์ไอเทมที่ผู้เล่นสามารถใช้ได้และคลังเก็บของของผู้เล่น

```python
class Player(pygame.sprite.Sprite):
    def input(self):
        keys = pygame.key.get_pressed()

        if not self.timers['tool use'].active:
            # and not self.sleep

            # Create direction
            if keys[pygame.K_UP] or keys[pygame.K_w]:
                self.direction.y = -1
                self.status = 'up'
            elif keys[pygame.K_DOWN] or keys[pygame.K_s]:
                self.direction.y = 1
                self.status = 'down'
            else:
                self.direction.y = 0

            if keys[pygame.K_LEFT] or keys[pygame.K_a]:
                self.direction.x = -1
                self.status = 'left'
            elif keys[pygame.K_RIGHT] or keys[pygame.K_d]:
                self.direction.x = 1
                self.status = 'right'
            else:
                self.direction.x = 0

            # Equipment usage
            if keys[pygame.K_SPACE]:
                self.timers['tool use'].activate()
                self.direction = pygame.math.Vector2()
                self.frame_index = 0

            # Change current equipment
            if keys[pygame.K_q] and not self.timers['tool swap'].active:
                self.timers['tool swap'].activate()
                self.tool_index += 1
                # Ternary in python
                self.tool_index = self.tool_index if self.tool_index < len(self.tools) else 0
                self.selected_tool = self.tools[self.tool_index]

            # Set and interaction button
            if keys[pygame.K_f]:
                collided_interaction_sprite = pygame.sprite.spritecollide(self, self.interaction, False)
                if collided_interaction_sprite:
                    if collided_interaction_sprite[0].name == 'Boat':
                        self.status = 'left_idle'
```

```python
class Player(pygame.sprite.Sprite):
    def get_status(self):

        # idle status
        if self.direction.magnitude() == 0:
            self.status = self.status.split('_')[0] + '_idle'

        if self.timers['tool use'].active:
            self.status = self.status.split('_')[0] + '_' + self.selected_tool

    def update_timers(self):
        for timer in self.timers.values():
            timer.update()

    def collision(self, direction):
        for sprite in self.collisionSprite.sprites():
            if hasattr(sprite, 'hitbox'):
                if sprite.hitbox.colliderect(self.hitbox):
                    pass
```

```python
class Player(pygame.sprite.Sprite):
    def move(self, dt):
        if self.direction.magnitude() > 0:
            self.direction = self.direction.normalize()

        # horizontal movement
        self.pos.x += self.direction.x * self.speed * dt
        self.hitbox.centerx = round(self.pos.x)
        self.rect.centerx = self.hitbox.centerx

        #vertical movement
        self.pos.y += self.direction.y * self.speed * dt
        self.hitbox.centery = round(self.pos.y)
        self.rect.centery = self.hitbox.centery

    def update(self, dt):
        self.input()
        self.get_status()
        self.update_timers()
        self.get_target_pos()

        self.move(dt)
        self.animate(dt)
```

```python
class Player(pygame.sprite.Sprite):
    def move(self, dt):
        if self.direction.magnitude() > 0:
            self.direction = self.direction.normalize()

        # horizontal movement
        self.pos.x += self.direction.x * self.speed * dt
        self.hitbox.centerx = round(self.pos.x)
        self.rect.centerx = self.hitbox.centerx

        #vertical movement
        self.pos.y += self.direction.y * self.speed * dt
        self.hitbox.centery = round(self.pos.y)
        self.rect.centery = self.hitbox.centery

    def update(self, dt):
        self.input()
        self.get_status()
        self.update_timers()
        self.get_target_pos()

        self.move(dt)
        self.animate(dt)
```

และในส่วนของ method ต่างๆจะเป็นการสรุปการทำงานภายในคร่าวแทนๆเนื่องจากรายละเอียดค่อนข้างเยอะ

- use_tool: เป็น method กำหนดความสามารถของไอเทมอุปกรณ์และเงื่อนไขของอุปกรณ์ เช่น ขวานจำเป็นต้อง collide กับต้นไม้เพื่อตัดต้นไม้
- get_target_pos: ใช้สำหรับเป็นตำแหน่งบ่งบอกระยะและความกว้างในการ collide ของอุปกรณ์ไอเทม ดังเช่น ความยาวที่ดาบสามารถฟันได้ 4 บล็อค เป็นต้น
- animate: เป็นตัวกำหนดภาพให้เคลื่อนไหว โดยที่ผู้เล่นจะมี sprite animation อยู่ 6 รูปและทำการวนลูป frame_index ผ่าน 6 รูปเพื่อให้เกิดเป็น animation เคลื่อนไหวขึ้น
- get_status: เป็น method บ่งบอกสถานะปัจจุบันว่ามีการเคลื่อนไหวไปทิศทางใดหรือไม่กำลังใช้อุปกรณ์ใดๆอยู่เช่นเดียวกันด้วยหรือไม่
- update_timers: สร้างขึ้นเพื่ออัพเดตการทำงานภายในระยะเวลาที่เราใช้ในแต่ละอุปกรณ์ (ไม่ถึงวินาที)
- collision: ใช้เป็นตัวกำหนดการ collide กันระหว่าง sprite ที่มีสถานะเลเยอร์ hitbox จะสามารถที่จะเกิด collision ร่วมกันได้
- input: เป็น method รับค่าปุ่มต่างๆเพื่อใช้ในการเคลื่อนไหวพร้อมทั้งอัพเดตทิศทางและการ interact ระหว่างกัน เช่น ปุ่ม E ใช้สำหรับสลับอุปกรณ์ไอเทมอาวุธ, ปุ่ม SpaceBar ใช้สำหรับใช้อุปกรณ์

- move: ใช้เป็น method ที่ใช้ร่วมกับ deltatime เพื่อตรวจหาขนาด magnitude ว่ามากกว่า 0 หรือไม่ เพื่ออัพเดตตำแหน่งการเคลื่อนไหวบนทิศทางนั้นๆร่วมกับความเร็วของผู้เล่นและ deltatime และกำหนดจุดศูนย์กลางทิศทางที่เคลื่อนที่
- update: เป็น method รวมการเรียกใช้ในแต่ละ method เพื่ออัพเดตตัวผู้เล่นและใช้เป็น method หลักที่เรียก method อื่นๆ

```python
import pygame
from settings import *
from settings import LAYERS
from random import randint, choice
from timer import Timer
```

```
You, 2 months ago | 1 author (You)
```

```python
class Generic(pygame.sprite.Sprite):
    def __init__(self, pos, surf, groups, z = LAYERS['main']):
        super().__init__(groups)
        self.image = surf
        self.rect = self.image.get_rect(topleft = pos)
        self.z = z
        self.hitbox = self.rect.copy().inflate(-self.rect.width * 0.2, -self.rect.height * 0.75)
        # change hitbox
```

```
You, 2 months ago | 1 author (You)
```

```python
class Interaction(Generic):
    def __init__(self, pos, surf, size, groups, name):
        surf = pygame.Surface(size)
        super().__init__(pos, surf, groups)
        self.name = name
```

```
You, 18 hours ago | 1 author (You)
```

```python
class Tree(Generic):
    def __init__(self, pos, surf, groups, name, player_add):
        super().__init__(pos, surf, groups)

        # Tree attributes
        self.health = 5
        self.alive = True
        # need tree stump
        self.stump_surf = pygame.image.load(f'../graphic/treestump').convert_alpha()
        self.invul_timer = Timer(200)

        # Wood and apple drop
        self.apple_surf = pygame.image.load('../Sprite/item/apple.png')
        self.apple_pos = APPLE_POS[name]
        self.apple_sprites = pygame.sprite.Group()

        self.player_add = player_add
```

```
You, 2 months ago | 1 author (You)
```

```python
class Particle(Generic):
    def __init__(self, pos, surf, groups, z, duration = 200):
        super().__init__(pos, surf, groups, z)
        self.star_time = pygame.time.get_ticks()
        self.duration = duration

        # white surface particle
        mask_surf = pygame.mask.from_surface(self.image)
        new_surf = mask_surf.to_surface()
        new_surf.set_colorkey((255, 255, 255))
        self.image = new_surf

    def update(self, dt):
        current_time = pygame.time.get_ticks()
        if current_time - self.start_time > self.duration:
            self.kill()
```

```python
# Inheritance from generic
```

```
You, 2 months ago | 1 author (You)
```

```python
class Water(Generic):
    def __init__(self, pos, frames, groups):

        # Animation setup
        self.frames = frames
        self.frame_index = 0

        # sprite setup
        super().__init__(pos = pos,
                         surf = self.frames[self.frame_index],
                         groups = groups,
                         z = LAYERS['water'])

    def animate(self, dt):
        self.frame_index  += 3 * dt
        if self.frame_index >= len(self.frames):
            self.frame_index = 0
        self.image = self.frames[int(self.frame_index)]

    def update(self, dt):
        self.animate(dt)
```

```
You, 2 months ago | 1 author (You)
```

```python
class Seashell(Generic):
    def __init__(self, pos, surf, groups):
        super().__init__(pos, surf, groups)
        self.hitbox = self.rect.copy().inflate(-20, -self.height * 0.9)
```

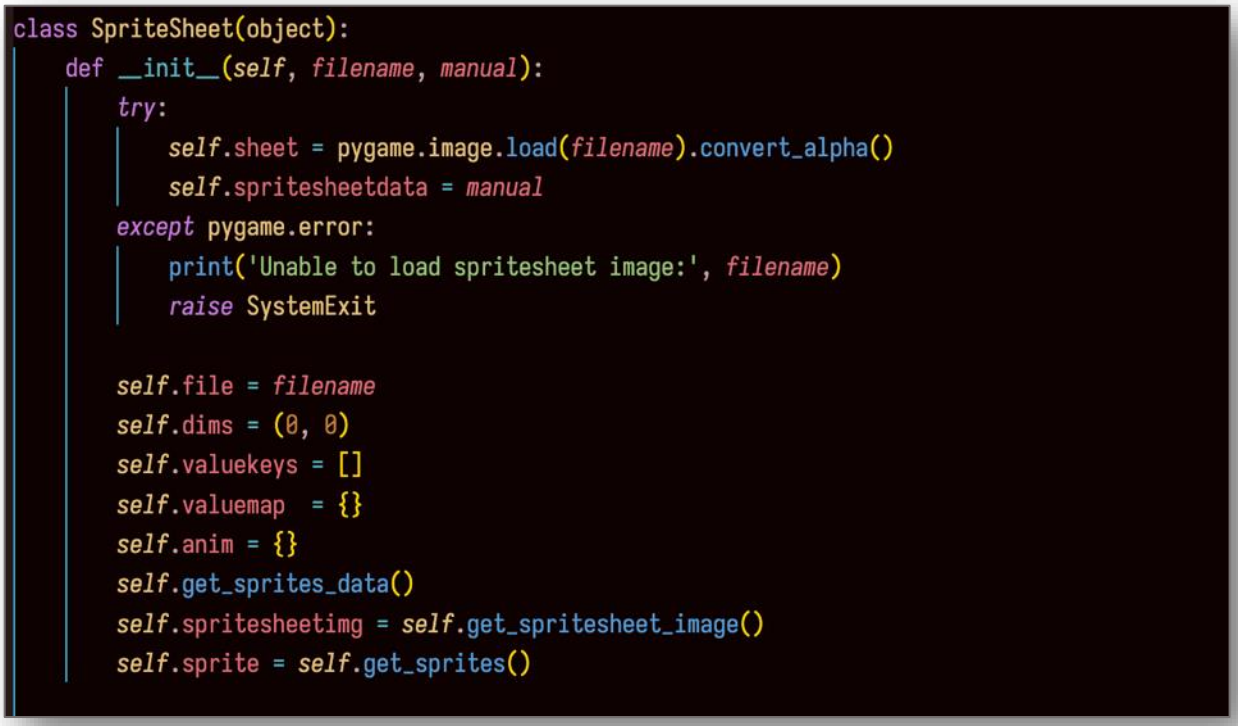
[ภายในไฟล์ sprite.py]

6. class Generic() เป็น class ที่ใช้ส่งต่อการสืบทอดในหลาย class ที่สร้างขึ้นเพื่อให้เกิดการ inheritance อ้างอิงต่อๆจาก class Generic() และตัว class Generic() เองก็ inherited มาจาก pygame.sprite.Sprite และถูก Init ผ่าน groups โดย class Generic() มี attribute ภายในหลายตัว ได้แก่ image ที่มาจาก surface, rectangular หรือ rect กำหนดขอบเขตขนาดของกรอบจากรูปภาพ sprite, เลเยอร์ชั้นของภาพบน z-index และ hitbox ที่คำนวณจากขนาดกรอบของ sprite

จากตัวอย่างภายในโค้ดจะเห็นได้ชัดว่าในไฟล์ sprite.py มีการดึง class Generic() มาใช้ด้วยหลักการ Inheritance แทบทุก class เพื่อใช้จำแนกชั้นเลเยอร์ต่างๆ ทั้งขอบเขตของแผนที่เกม ต้นไม้ที่สามารถตัดได้ น้ำทะเลในเกมและรวมถึง Particle หลังจากการตัดต้นไม้

ตัวอย่างการใช้ Encapsulation ภายในโค้ด

การ Encapsulate ตัว class สามารถพบเจอในทุกไฟล์ เนื่องจากภายในตัวเกมต้องการ reuse ตัวแปรต่างๆที่กักเก็บไว้เพื่อดึงไปใช้ต่อใน class อื่นๆ และจำเป็นต้องส่งค่าตัวแปรนั้นผ่าน class อื่นอีก จึงพบเห็นได้ในทุกไฟล์ว่ามีการเรียกใช้ตัวแปรผ่าน method ที่ผ่านการ encapsulate มาเรียบร้อยแล้ว ยกตัวอย่างจากไฟล์ spritesheet.py ที่มี class SpriteSheet() รับเอา filename หรือ “filepath” ของตัวแปรมาใช้ใน self.sheet แปลงค่าไฟล์รูปที่โหลดเข้ามาเป็นรูปของ sheet_image และ manual ที่รับเข้ามาเป็นในส่วนของไฟล์ Json ประกอบการ mapping ใช้บ่งบอกสถานะของตัวละครและท่าทางที่ตัวละครกำลังทำอยู่ในปัจจุบัน



```python
class SpriteSheet(object):
    def __init__(self, filename, manual):
        try:
            self.sheet = pygame.image.load(filename).convert_alpha()
            self.spritesheetdata = manual
        except pygame.error:
            print('Unable to load spritesheet image:', filename)
            raise SystemExit

        self.file = filename
        self.dims = (0, 0)
        self.valuekeys = []
        self.valuemap  = {}
        self.anim = {}
        self.get_sprites_data()
        self.spritesheetimg = self.get_spritesheet_image()
        self.sprite = self.get_sprites()
```

ตัวอย่างการใช้ Inheritance ภายในโค้ด

เช่นเดียวกันกับหลักการ Encapsulate เมื่อสังเกตจากโค้ดภายในโปรเจกต์จะพบว่าแทบทุก class ของการใช้งานมีการ Inherit จาก class หลักภายใน pygame และ Init เพื่อใช้ class ดังเป็น class ภายใน pygame ที่เพิ่ม attribute และ data ต่างๆใส่เข้ามากักเก็บภายใน class ได้ สังเกตได้จากไฟล์ sprite.py ที่มีการ Inherited จาก class Generic() แทบทุก class เพื่อใช้ในรูปแบบของ sprite รูปภาพ

```python
import pygame
from settings import *
from settings import LAYERS
from random import randint, choice
from timer import Timer
You, 2 months ago | 1 author (You)
class Generic(pygame.sprite.Sprite):
    def __init__(self, pos, surf, groups, z = LAYERS['main']):
        super().__init__(groups)
        self.image = surf
        self.rect = self.image.get_rect(topleft = pos)
        self.z = z
        self.hitbox = self.rect.copy().inflate(-self.rect.width * 0.2, -self.rect.height * 0.75)
        # change hitbox


You, 2 months ago | 1 author (You)
class Interaction(Generic):
    def __init__(self, pos, surf, size, groups, name):
        surf = pygame.Surface(size)
        super().__init__(pos, surf, groups)
        self.name = name
```

เห็นได้ชัดว่า class Interaction() มีการ Inherited ต่อจาก class Generic() และส่งต่อตัวแปร position, surface, groups และเพิ่มในส่วนของตัวแปรชื่อและตัวแปร size เข้ามาเพื่อใช้ต่อในการกักเก็บตัวแปร surf อ้างอิงจากขนาดตัวแปร size

ตัวอย่างการใช้ Polymorphism ภายในโค้ด

ตัวอย่างที่เห็นชัดเจนที่สุดคือ method update หรือฟังก์ชัน update ที่แต่ละ class ใช้และจำเป็นต้องใช้ตัวแปร dt ในหลายๆกรณีเพื่อส่งต่อ delta time ในการอัพเดตการทำงานภายในบน method และเรียกใช้ method อื่นๆ ตัวอย่างเช่น move, create component, animate และ method check ฟังก์ชันต่างๆ

```python
import pygame
from settings import *
from support import import_folder
from sprite import Generic

You, 41 seconds ago | 1 author (You)
class Drop(Generic):
    def __init__(self, surf, pos, moving, group, z):
        self.surf = surf

You, 41 seconds ago | 1 author (You)
class Rain:
    def __init__(self, all_sprites):
        self.all_sprites = all_sprites
        self.rain_drops = import_folder()
        self.rain_floor = import_folder()
        self.floor_w, self.floor_h = pygame.image.load('../Sprite/water/').get_size()

    def create_floor(self):
        pass

    def create_drop(self):
        pass

    def update(self):
        self.create_floor()
        self.create_drop()
```

```python
class Particle(Generic):
    def __init__(self, pos, surf, groups, z, duration = 200):
        super().__init__(pos, surf, groups, z)
        self.star_time = pygame.time.get_ticks()
        self.duration = duration

        # white surface particle
        mask_surf = pygame.mask.from_surface(self.image)
        new_surf = mask_surf.to_surface()
        new_surf.set_colorkey((255, 255, 255))
        self.image = new_surf

    def update(self, dt):
        current_time = pygame.time.get_ticks()
        if current_time - self.start_time > self.duration:
            self.kill()
```

```python
class Water(Generic):
    def __init__(self, pos, frames, groups):

        # Animation setup
        self.frames = frames
        self.frame_index = 0

        # sprite setup
        super().__init__(pos = pos,
                         surf = self.frames[self.frame_index],
                         groups = groups,
                         z = LAYERS['water'])

    def animate(self, dt):
        self.frame_index  += 3 * dt
        if self.frame_index >= len(self.frames):
            self.frame_index = 0
        self.image = self.frames[int(self.frame_index)]

    def update(self, dt):
        self.animate(dt)
```

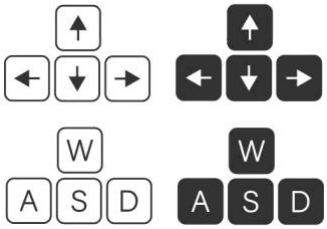
```python
class Tree(Generic):

    def check_death(self):
        if self.health <= 0:
            Particle(self.rect.topleft, self.image, self.groups()[0], LAYERS['fruits'], 300)
            self.image = self.stump_surf
            self.rect = self.image.get_rect(midbottom = self.rect.midbottom)
            self.hitbox = self.rect.copy().inflate(-10, -self.rect.height * 0.6)
            self.alive = False


    def create_fruit(self):
        # create apple randomly on tree
        for pos in self.apple_pos:
            if randint(0, 10) < 2:
                Generic()

        # add wood drop and apple to tree

    def update(self, dt):
        if self.alive:
            self.check_death()
```

แสดงให้เห็นว่าแต่ละ class จากตัวอย่างมี method update ที่คล้ายกันอยู่ แต่องค์ประกอบของ method อื่นภายใน class อาจจะแตกต่างกันออกไปแล้วแต่การประยุกต์ใช้บน class นั้นๆ

ตัวอย่างการใช้ Abstraction ภายในโค้ด

ภายในโปรเจกต์เกมนี้ในส่วนของ Abstraction principle พบว่าไม่ค่อยมีการนำ principle ดังกล่าวมาใช้มากนัก โดยส่วนมากจะกำหนดให้ class เกือบทุก class เป็น Abstract class ที่มีการทำงานเชิง process abstraction และจะไม่เน้นในส่วนของ data abstraction มากเท่าใด ซึ่งสามารถสังเกตได้จาก class จากไฟล์ต่างๆ

คู่มือการใช้งาน (วิธีการเล่น)



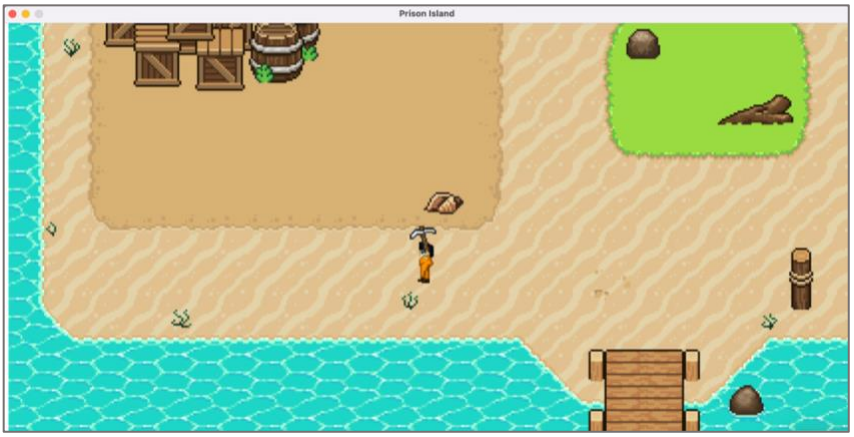
สามารถกดปุ่ม arrow keys หรือปุ่มลูกศร และปุ่ม W A S D เพื่อบังคับทิศทางของตัวละครได้ตามรูป

ปุ่ม SpaceBar มีไว้เพื่อใช้อุปกรณ์ต่างๆที่ถืออยู่ปัจจุบัน

ปุ่ม E มีไว้เพื่อสลับอุปกรณ์ที่ใช้อยู่ปัจจุบันให้เป็นไอเทมอื่นแทน

Gameplay

Sword swing



Using pickaxe

Using axe

การพัฒนาต่อยอด

เมื่อโปรเจกต์การสร้างเกมสำเร็จ ผู้พัฒนาต้องการนำโปรเจกต์ไปใช้งานพัฒนาร่วมด้วยกับเว็บไซต์เพื่อสร้างเป็นเกมสไตล์ผ่อนคลายความเครียดที่สามารถเล่นบนเว็บไซต์หรือเล่นผ่านบนเบราว์เซอร์คล้ายกับเกมบนเว็บ itch.io ประเภทของ HTML5 web base game ที่สามารถเล่นเกมบนเว็บได้โดยไม่ต้องติดตั้งตัวเกมบนอุปกรณ์จากผู้ใช้งานเข้าถึง

เอกสารอ้างอิง

https://github.com/clear-code-projects/PyDew-Valley (Game Reference)

https://github.com/DarthData410/PyGames-Mars (Game Reference)

https://www.pygame.org/docs/ (Pygame documentation)

https://cupnooble.itch.io/sprout-lands-asset-pack (Sprite)

https://game-endeavor.itch.io/mystic-woods (Sprite)

https://limezu.itch.io/serenevillagerevamped (Sprite)

https://raou.itch.io/island (แผนที่เสริม)