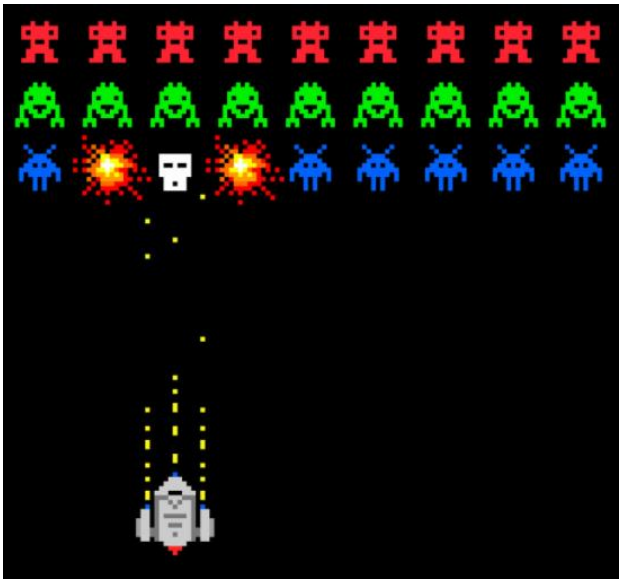
#Final Game นายชินพัฒน์ ลิ้มประธาน 64015030

Alien of typing เป็นเกม arcade คล้ายกับเกม space invaders คือเป็นเกมยิงเอเลียนที่จะเข้ามาโจมตียานอวกาศของเรา โดยเราต้องปกป้องตัวเองด้วยการยิงกระสุนเลเซอร์ใส่เอเลียนด้วยการ พิมพ์คำศัพท์ที่อยู่ด้านใต้ยานของเอเลียนแต่ละตัว

เหตุผลที่ทำเกมนี้เพราะว่า อยากทำเกมแนว space invaders แต่อยากเปลี่ยนจากกดยิง มาเป็นพิมพ์ดีด เพื่อนำมาเป็นเกมเอาไว้ฝึกพิมพ์คำภาษาอังกฤษ

ตัวอย่างเกม



Space Invader

Alien of typing

วิธีการเล่น

หน้าเมนู

หน้าเลือกความยาก

หน้าต่างในเกมเมื่อเล่น

หน้า pause game

หน้าจบเกม

การควบคุม

- Keyboard ใช้เพื่อพิมพ์คำที่แสดงขึ้นมาตรงศัตรูแต่ละตัว
- ปุ่ม Esc เพื่อ Pause เกมหรือ ออกจากการ Pause เกม

การเก็บคะแนน

เมื่อพิมพ์คำศัพท์ที่แสดงขึ้นมาตรงศัตรูแต่ละตัวจนครบคำ จะเป็นการจัดการกับศัตรูและจะได้คะแนนมา 1 คะแนน คูณกับ combo ที่เราสะสมมาได้ การเก็บ combo เกิดจากการพิมพ์ต่อเนื่องโดยไม่พิมพ์ผิด โดย combo จะเริ่มที่ 1 และบวกไปทีละ 0.25 เมื่อพิมพ์ต่อเนื่อง 5 ตัวอักษรโดยไม่พิมพ์ผิด ถ้าเกิดพิมพ์ผิดจะเป็นการ reset combo กลับมาเป็นค่าเริ่มต้นคือ 1

เงื่อไขการจบเกม

- ผู้เล่นจะจบเกมก็ต่อเมื่อกดออกจากเกมหรือ พลังชีวิตของผู้เล่นหมด

ศัตรู

ชื่อ: enemy-small

ความยาวตัวอักษร: <= 3 ตัวอักษร

ชื่อ enemy-medium

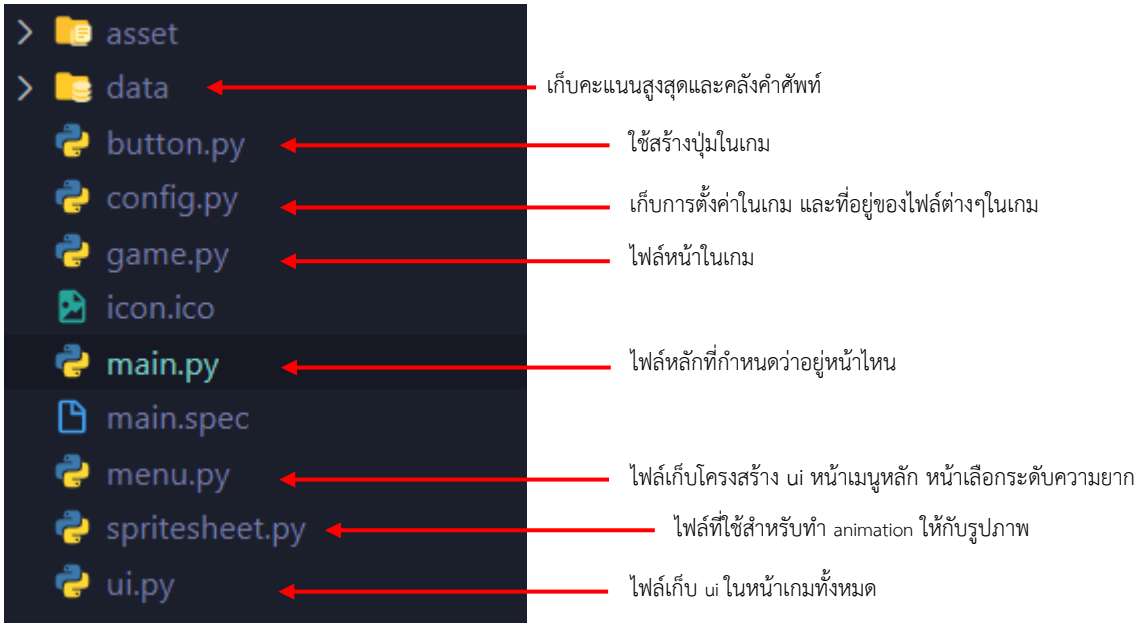
ความยาวตัวอักษร: <= 5 ตัวอักษร

ชื่อ enemy-big

ความยาวตัวอักษร: > 5 ตัวอักษร

เนื้อหาโค้ด

องค์ประกอบภายใน folder



Code ภายในไฟล์

main.py

```python
class MainGame:
    def __init__(self):
        self.state = 'main_menu'
        self.background = pygame.image.load(BACKGROUND_IMG).convert_alpha()
        self.background = pygame.transform.scale(self.background, (256*4, 608*4))
        self.background_rect = self.background.get_rect()
        self.bg_y = self.background_rect.y
        self.game_mode = "easy"
        self.bg_music = pygame.mixer.Sound(BACKGROUND_MUSIC)
        self.bg_music.set_volume(BG_VOLUME)

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

        game_menu.draw(screen)
        game_menu.set_higher_score(main_game.get_score()['high_score'])

        if game_menu.start_button.check_click():
            self.state = "select_level"
        if game_menu.exit_button.check_click():
            pygame.quit()
            sys.exit()

        pygame.display.update()
```

```python
def select_level(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                self.state = "main_menu"

    select_menu.draw(screen)
    if select_menu.easy_button.check_click():
        self.bg_music.play(-1)
        self.game_mode = "easy"
        self.state = "main_game"
    if select_menu.hard_button.check_click():
        self.game_mode = "hard"
        self.bg_music.play(-1)
        self.state = "main_game"
    if select_menu.back_botton.check_click():
        self.state = "main_menu"
    pygame.display.update()
```

```python
def main_game(self, mode):
    main_game.main(screen, mode)
    if main_game.is_game_end:
        self.state = "main_menu"
        self.bg_music.stop()
        main_game.reset()

def menu_state(self):
    self.draw_background(screen)
    if self.state == "main_menu":
        self.main_menu()
    if self.state == "select_level":
        self.select_level()
    if self.state == "main_game":
        self.main_game(self.game_mode)

def draw_background(self, screen):
    # return remain y
    rel_y = self.bg_y % self.background_rect.width
    screen.blit(self.background, (0, rel_y - self.background_rect.width))
    if rel_y < WINDOW_HEIGHT:
        screen.blit(self.background, (0, rel_y))
    # update screen
    self.bg_y += 1
```

```python
pygame.init()
clock = pygame.time.Clock()
screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption(GAME_CAPTION)
pygame.display.set_icon(pygame.image.load(ICON_IMG))

# obj create

game_menu = MainMenu()
select_menu = SelectMenu()
main_game = Game()
game = MainGame()

while True:
    game.menu_state()
    clock.tick(FPS)
```

เป็นไฟล์หลักที่ใช้จัดการ สถานะหน้าของเกมว่าตอนนี้กำลังรันหน้าไหนอยู่ โดยที่เริ่มต้นมาจะ run main_menu() คือหน้าหลัก และวาดปุ่มเริ่มเกมกับปุ่มออกเกม โดยที่ถ้ากดเริ่มเกมให้เปลี่ยน self.state = “select_level” คือให้ไปหน้าเลือก level และถ้าเลือก level ได้แล้วจะทำการเปลี่ยนเป็น “main_game” ซึ่งระบบจะเช็คว่าตอนนี้ต้อง run หน้าไหนโดย menu_state() ซึ่งจะ run อยู่ใน while loop ตลอด เป็น loop หลัก

menu.py

```python
class Menu:
    def __init__(self):
        self.font = pygame.font.Font(MINECRAFT_FONT, 120)
        self.small_font = pygame.font.Font(MINECRAFT_FONT, 20)
        self.mid_font = pygame.font.Font(MINECRAFT_FONT, 48)

        self.creator_text = self.small_font.render(AUTHOR, True, (255, 255, 255))
        self.creator_text_rect = self.creator_text.get_rect(bottomleft = (10, WINDOW_HEIGHT-10))

        self.title = GAME_TITLE
        self.title_font = self.font.render(self.title, True, (255, 255, 255))
        self.title_font_shadow = self.font.render(self.title, True, (0, 0, 0))
        self.title_font_rect = self.title_font.get_rect(center = (WINDOW_WIDTH / 2, WINDOW_HEIGHT * 0.2))
        self.title_font_shadow_rect = self.title_font_shadow.get_rect(center = (WINDOW_WIDTH / 2, (WINDOW_HEIGHT * 0.2)+10))
```

```python
class SelectMenu(Menu):
    def __init__(self):
        super().__init__()
        self.easy_button = Button("Easy", GREEN, 200, 50, (WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2), 6, 24)
        self.hard_button = Button("Hard", PURPLE, 200, 50, (WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2 + 100), 6, 24)
        self.back_botton = Button("Go back", RED, 180, 40, (WINDOW_WIDTH / 2, WINDOW_HEIGHT * 0.6 + 120), 6, 20)

    def draw(self, screen):
        #!! order the rendering
        screen.blit(self.title_font_shadow, self.title_font_shadow_rect)
        screen.blit(self.title_font, self.title_font_rect)
        screen.blit(self.creator_text, self.creator_text_rect)
        self.easy_button.draw(screen)
        self.hard_button.draw(screen)
        self.back_botton.draw(screen)
```

```python
class MainMenu(Menu):
    def __init__(self):
        super().__init__()
        self.start_button = Button("Start", GREEN, 200, 50, (WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2), 6)
        self.exit_button = Button("Exit", RED, 200, 50, (WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2 + 100), 6)
        self.high_score = 0


    def draw(self, screen):
        #!! order the rendering
        text_score = self.mid_font.render(f"highest Score {self.high_score}", True, (255, 255, 255))
        text_score_rect = text_score.get_rect(bottomright = (WINDOW_WIDTH - 10, WINDOW_HEIGHT - 10))
        screen.blit(self.title_font_shadow, self.title_font_shadow_rect)

        screen.blit(self.title_font, self.title_font_rect)
        screen.blit(self.creator_text, self.creator_text_rect)
        screen.blit(text_score, text_score_rect)
        self.start_button.draw(screen)
        self.exit_button.draw(screen)

    def set_higher_score(self, score):
        self.high_score = score
```

เป็นไฟล์ที่ใช้เก็บเมนูของหน้าหลัก ในเมนูนั้นก็จะเก็บปุ่มและพวก ui ต่างๆของหน้านั้น

ซึ่งจะถูกเรียกใช้โดย main.py

button.py

```python
class Button:
    def __init__(self, text, color, width, height, pos, elevation, font_size = 24):
        self.color = color
        self.pressed = False
        self.elevation = elevation
        self.dynamic_elev = elevation
        self.original_y_pos = pos[1]
        self.gui_font = pygame.font.Font(MINECRAFT_FONT, font_size)

        # top rect
        self.top_rect = pygame.Rect((pos), (width, height))
        self.top_rect.center = pos
        self.top_color = color

        #bottom rect
        self.bottom_rect = pygame.Rect((pos), (width, elevation))
        self.bottom_color = color

        # text
        self.text_surf = self.gui_font.render(text, True, WHITE)
        self.text_rect = self.text_surf.get_rect(center = self.top_rect.center)

    def draw(self, screen):
        # specify elev logic
        self.top_rect.y = self.original_y_pos - self.dynamic_elev
        self.text_rect.center = self.top_rect.center

        self.bottom_rect.midtop = self.top_rect.midtop
        self.bottom_rect.height = self.top_rect.height + self.dynamic_elev

        pygame.draw.rect(screen, self.bottom_color, self.bottom_rect, border_radius=12)
        pygame.draw.rect(screen, self.top_color, self.top_rect, border_radius=12)
        screen.blit(self.text_surf, self.text_rect)
```

```python
    def check_click(self):
        mouse_pos = pygame.mouse.get_pos()
        if self.top_rect.collidepoint(mouse_pos):
            self.top_color = GREY
            if pygame.mouse.get_pressed()[0]:
                self.dynamic_elev = 0
                self.pressed = True
            else:
                self.dynamic_elev = self.elevation
                if self.pressed:
                    self.pressed = False
                    return True
        else:
            self.dynamic_elev = self.elevation
            self.top_color = self.color
            self.pressed = False
```

เป็นไฟล์ที่ไว้สำหรับทำปุ่มต่างๆ ของ ui ใน game และ เมนูต่าง

config.py

```python
AUTHOR = "Chinnapat Limprathan 64015030"
GAME_CAPTION = "Alien of typing V1.02b"
GAME_TITLE = "Alien of typing"

WINDOW_WIDTH = 1000
WINDOW_HEIGHT = 900
FPS = 60

MINECRAFT_FONT = './asset/Font/Minecraft.ttf'

IMG_SCALE = 4
BACKGROUND_IMG = "./asset/Graphics/backgrounds/desert-backgorund-looped.png"
ICON_IMG = "./asset/icon/icon.png"

HEALTH_BAR_BORDER_IMG = './asset/Hearts/health_bar/health_bar_decoration.png'
HEALTH_BAR = './asset/Hearts/health_bar/health_bar.png'

PLAYER_SHIP = './asset/Graphics/spritesheets/player_ship.png'
LASER_BULLET = './asset/Graphics/spritesheets/laser-bolts.png'
EXPLODE_EFFECT = './asset/Graphics/spritesheets/explosion.png'

ENEMY_BIG = './asset/Graphics/spritesheets/enemy-big.png'
ENEMY_MEDIUM = './asset/Graphics/spritesheets/enemy-medium.png'
ENEMY_SMALL = './asset/Graphics/spritesheets/enemy-small.png'

BACKGROUND_MUSIC = './asset/Sound/spaceship_shooter.wav'
LASER_SHOOT_SOUND = "./asset/Sound/laser-shoot.wav"
EXPLODE_SOUND = "./asset/Sound/explosion_sound.mp3"

BG_VOLUME = 0.08
EFFECT_VOLUME = 0.05

COLLISION_TOLERANCE = 22
DAMAGE = 49
```

```python
WHITE = "#FFFFFF"
ORANGE = "#EF7300"
GREEN = "#3CA47E"
PURPLE = "#B076C3"
RED = "#D42F3A"
BLACK = "#000000"
GREY = '#aEaFaF'
```

เป็นไฟล์ไว้สำหรับเก็บการตั้งค่าต่างๆของเกม เก็บที่อยู่ไฟล์ เพื่อความง่ายในการนำไปใช้ในแต่ละไฟล์

game.py

```
class PlayerShip(pygame.sprite.Sprite):
    def __init__(self, pos_x, pos_y):
        super().__init__()
        self.asset = pygame.image.load(PLAYER_SHIP).convert_alpha()
        self.sheet = SpriteSheet(self.asset, 16, 24)
        self.sprites = [self.sheet.get_img(3, x) for x in range(2)]
        self.current_sp = 0
        self.image = self.sprites[self.current_sp]
        self.rect = self.image.get_rect(midtop = (pos_x, pos_y))

    def update(self, speed):
        self.current_sp += speed
        if self.current_sp ≥ len(self.sprites):
            self.current_sp = 0
        self.image = self.sprites[int(self.current_sp)]
```

```
class SpaceObj:
    def __init__(self, pos_x, pos_y):
        self.pos = Vector2(pos_x, pos_y)
        self.direction = Vector2(0, 0)

    def update(self, speed):
        self.current_sp += speed
        if self.current_sp ≥ len(self.sprites):
            self.current_sp = 0
        self.image = self.sprites[int(self.current_sp)]
```

```
class Enemy(SpaceObj):
    def __init__(self, pos_x, pos_y, world_list, enemy_sprites, spr_width, spr_height):
        super().__init__(pos_x, pos_y)
        self.asset = pygame.image.load(enemy_sprites).convert_alpha()
        self.sheet = SpriteSheet(self.asset, spr_width, spr_height)
        self.sprites = [self.sheet.get_img(x, 0) for x in range(2)]
        self.current_sp = 0
        self.image = self.sprites[self.current_sp]
        self.rect = self.image.get_rect(center = (pos_x, pos_y))
        self.velocity = 0.9
        self.original_word_list = world_list
        self.world_list = world_list
        self.user_input = ''
        self.word_font = pygame.font.Font(MINECRAFT_FONT, 24)
        self.get_lock = False
```

```
    def move(self):
        playerx, playery = WINDOW_WIDTH * 0.5, WINDOW_HEIGHT * 0.8
        dx = playerx - self.pos.x
        dy = playery - self.pos.y
        angle = math.atan2(dx, dy)
        velo_x = math.sin(angle)
        velo_y = math.cos(angle)
        self.pos += Vector2(velo_x * self.velocity, velo_y * self.velocity)
        self.rect = self.image.get_rect(center = self.pos)

    def draw(self, screen):
        self.update(0.2)
        self.move()
        if self.get_lock:
            color = ORANGE
        else:
            color = WHITE
        word_text = self.word_font.render(f"{self.world_list}", True, color)
        word_rect = word_text.get_rect(midtop = (self.rect.midbottom[0], self.rect.midbottom[1]+5))
        screen.blit(word_text, word_rect)
        screen.blit(self.image, self.rect)
```

```
class Bullet(SpaceObj):
    def __init__(self, pos_x, pos_y):
        super().__init__(pos_x, pos_y)
        self.asset = pygame.image.load(LASER_BULLET).convert_alpha()
        self.sheet = SpriteSheet(self.asset, 16, 16)
        self.sprites = [self.sheet.get_img(x, 0) for x in range(2)]
        self.current_sp = 0
        self.image = self.sprites[self.current_sp]
        self.rect = self.image.get_rect(center = (pos_x, pos_y))
        self.velocity = 50

    def draw(self, screen):
        self.update(0.2)
        screen.blit(self.image, self.rect)

    def shoot(self, pos):
        targetx, targety = pos.x, pos.y
        dx = targetx - self.pos.x
        dy = targety - self.pos.y
        angle = math.atan2(dx, dy)
        velo_x = math.sin(angle)
        velo_y = math.cos(angle)
        self.pos += Vector2(velo_x * self.velocity, velo_y * self.velocity)
        self.rect = self.image.get_rect(midtop = self.pos)
```

```python
class ShipExplode(SpaceObj):
    def __init__(self, pos_x, pos_y):
        super().__init__(pos_x, pos_y)
        self.asset = pygame.image.load(EXPLODE_EFFECT).convert_alpha()
        self.sheet = SpriteSheet(self.asset, 16, 16, 6)
        self.sprites = [self.sheet.get_img(x, 0) for x in range(5)]
        self.current_sp = 0
        self.image = self.sprites[self.current_sp]
        self.rect = self.image.get_rect(center = (pos_x, pos_y))

    def draw(self, screen):
        if self.update(0.15):
            return True
        screen.blit(self.image, self.rect)

    def update(self, speed):
        self.current_sp += speed
        if self.current_sp ≥ len(self.sprites):
            return True
        self.image = self.sprites[int(self.current_sp)]
```

```python
class Game():
    def __init__(self):
        self.is_game_end = False
        self.background = pygame.image.load(BACKGROUND_IMG).convert_alpha()
        self.background = pygame.transform.scale(self.background, (256*4, 608*4))
        self.background_rect = self.background.get_rect()
        self.bg_y = self.background_rect.y
        self.option_menu = OptionMenu()
        self.game_ui = GameUI()
        self.is_pause = False
        self.heal = True
        self.current_time = pygame.time.get_ticks()
        self.word_dict = self.open_word_json()

        self.enemys = []
        self.enemy_amount = 0
        self.max_enemys = 4

        self.player_sp = pygame.sprite.Group()
        self.player = PlayerShip(WINDOW_WIDTH * 0.5, WINDOW_HEIGHT * 0.8)
        self.player_sp.add(self.player)

        self.lock_enemy = 0
        self.is_lock_on = False
        self.bullets = []
        self.shoot_music = pygame.mixer.Sound(LASER_SHOOT_SOUND)
        self.shoot_music.set_volume(EFFECT_VOLUME)
        self.high_score = self.get_score()

        self.count_kill = 0
        self.steak = 0
        self.level = 1

        self.explosion = []
        self.explode_sound = pygame.mixer.Sound(EXPLODE_SOUND)
        self.explode_sound.set_volume(EFFECT_VOLUME)
```

```python
def draw_background(self, screen):
    # return remain y
    rel_y = self.bg_y % self.background_rect.width
    screen.blit(self.background, (0, rel_y - self.background_rect.width))
    if rel_y < WINDOW_HEIGHT:
        screen.blit(self.background, (0, rel_y))
    # update screen
    self.bg_y += 1

def game_pause(self, screen):
    while self.is_pause:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.is_pause = False

        self.option_menu.draw(screen, self.game_ui.score)

        if self.option_menu.botton_resume.check_click():
            self.is_pause = False
        if self.option_menu.botton_exit.check_click():
            self.is_game_end = True
            self.is_pause = False

        pygame.display.update()
```

```python
def game_over(self, screen):
    while self.is_pause:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        self.option_menu.draw_game_over(screen, self.game_ui.score)
        if self.option_menu.botton_exit.check_click():
            if self.game_ui.score > self.high_score['high_score']:
                self.high_score['high_score'] = self.game_ui.score
                self.write_score(self.high_score)
            self.is_game_end = True
            self.is_pause = False

        pygame.display.update()
```

```
def main(self, screen, mode):
    self.draw_background(screen)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            for index, enemy in enumerate(self.enemys):
                if enemy.world_list:
                    if event.unicode == enemy.world_list[0] and not self.is_lock_on:
                        self.lock_enemy = index
                        self.is_lock_on = True
                        enemy.get_lock = True
                        break
            if self.is_lock_on:
                char_typed = event.unicode
                if self.enemys[self.lock_enemy].world_list:
                    if char_typed == self.enemys[self.lock_enemy].world_list[0]:
                        self.steak += 1
                        # shoot
                        self.shoot_music.play()
                        new_bullet = Bullet(WINDOW_WIDTH * 0.5, WINDOW_HEIGHT * 0.8)
                        self.bullets.append(new_bullet)
                        self.enemys[self.lock_enemy].user_input += char_typed
                    else:
                        self.steak = 0
                        self.game_ui.multiplier_score = 1

            if event.key == pygame.K_ESCAPE:
                self.is_pause = True
```

```
    if self.enemy_amount < self.max_enemys and pygame.time.get_ticks() - self.current_time > 1500:
        self.current_time = pygame.time.get_ticks()
        self.enemy_amount += 1
        self.spawn_enemy(mode)
    if self.count_kill ≥ (6 * self.level):
        for index, enemy in enumerate(self.enemys):
            enemy.velocity += 0.2
        self.max_enemys += 1
        self.count_kill = 0
        self.level += 1
    if self.steak ≥ 5:
        self.game_ui.multiplier_score += 0.25
        self.steak = 0
```

```
    if self.heal:
        self.game_ui.health_amount += 2
    if self.game_ui.health_amount == self.game_ui.max_health:
        self.heal = False
```

```
for index, enemy in enumerate(self.enemys):
    if self.player.rect.colliderect(enemy.rect):
        self.explosion.append(ShipExplode(self.player.rect.x, self.player.rect.y))
        self.explode_sound.play()
        self.steak = 0
        self.game_ui.multiplier_score = 1
        self.is_lock_on = False
        enemy.get_lock = False
        self.enemy_amount -= 1
        self.game_ui.health_amount -= DAMAGE
        self.enemys.pop(index)
    enemy.draw(screen)
    for index, bullet in enumerate(self.bullets):
        bullet.shoot(self.enemys[self.lock_enemy].pos)
        bullet.draw(screen)
        if bullet.rect.colliderect(enemy.rect) or (abs(bullet.rect.midtop[0] - enemy.rect.midbottom[0]) ≤ COLLISION_TOLERANCE and abs
        (bullet.rect.midtop[1] - enemy.rect.midbottom[1]) ≤ COLLISION_TOLERANCE):
            self.enemys[self.lock_enemy].world_list = self.enemys[self.lock_enemy].world_list[1:]
            if (self.enemys[self.lock_enemy].user_input == self.enemys[self.lock_enemy].original_word_list) or not enemy.world_list:
                self.explosion.append(ShipExplode(self.enemys[self.lock_enemy].pos.x, self.enemys[self.lock_enemy].pos.y))
                self.explode_sound.play()
                self.enemy_amount -= 1
                self.game_ui.score += 1 * self.game_ui.multiplier_score
                self.count_kill += 1
                self.enemys.pop(self.lock_enemy)
                self.bullets.pop(index)
                self.is_lock_on = False
                self.enemys[self.lock_enemy].get_lock = False
                self.lock_enemy = 0
            else:
                self.bullets.pop(index)
            break
```

```
for index, effect in enumerate(self.explosion):
    if effect.draw(screen):
        self.explosion.pop(index)

self.player_sp.draw(screen)
self.player_sp.update(0.2)
self.game_ui.draw(screen)

# pause game
if self.game_ui.popup_botton.check_click():
    self.is_pause = True
if self.game_ui.health_amount ≤ 0:
    self.is_pause = True
    self.game_over(screen)
if self.is_pause:
    self.game_pause(screen)

pygame.display.update()
```

```python
def spawn_enemy(self, mode):
    rand_word = self.random_word(self.word_dict, mode)
    rand_word_len = len(rand_word)
    if rand_word_len ≤ 3:
        spawn_enemy = ENEMY_SMALL
        enemy_spr_width = 16
        enemy_spr_height = 16
    elif rand_word_len ≤ 5:
        spawn_enemy = ENEMY_MEDIUM
        enemy_spr_width = 32
        enemy_spr_height = 16
    else:
        spawn_enemy = ENEMY_BIG
        enemy_spr_width = 32
        enemy_spr_height = 32
    enemy = Enemy(random.randrange(0, WINDOW_WIDTH), -50, rand_word, spawn_enemy, enemy_spr_width, enemy_spr_height)
    self.enemys.append(enemy)
```

```python
def reset(self):
    self.is_game_end = False
    self.heal = True
    self.enemys = []
    self.enemy_amount = 0
    self.max_enemys = 3
    self.steak = 0
    self.game_ui.health_amount = 0
    self.game_ui.multiplier_score = 1
    self.game_ui.score = 0
    self.is_lock_on = False
    self.explosion.clear()
    self.bullets.clear()
    self.enemys.clear()
```

```python
def open_word_json(self):
    with open('data/words.json') as json_file:
        json_data = json.load(json_file)
    return json_data['data']
```

```python
def random_word(self, word_dict, level):
    type_word = ''
    while True:
        type_word = random.choice(word_dict)
        if (level == 'easy' and len(type_word) ≤ 5) or level == 'hard':
            break
    return type_word
```

```
    def get_score(self):
        try:
            with open('./data/score_board.json', 'r') as score:
                return json.load(score)
        except:
            return { "high_score" : 0}

    def write_score(self, data):
        with open('./data/score_board.json', 'w') as score:
            json.dump(data, score)
```

เป็นไฟล์ที่เกี่ยวข้องกับระบบของเกมทั้งหมด โดยที่จะมี class Game() เป็น class หลักในการจัดการเกม ซึ่งระบบของเกมจะรันอยู่ในฟังก์ชั่น main() โดยการทำงานของฟังก์ชั่น main คือ

- ดักจับ event ของ keyboard
- ตรวจสอบ หรือ ทำเงื่อนไขของเกม
- การ draw component ต่างๆ ลงใน game

1. เกมจะดักจับการกดปุ่มของ keyboard โดยที่จะมีอยู่ 3 ระบบย่อยคือ
   1.1 ตรวจหาเป้าหมายว่าตัวอักษรแรกที่พิมพ์ตรงกับศัตรูตัวไหนที่อยู่ไกล้ไหม โดยถ้าเจอ ให้ทำการ lock เป้าหมายไว้
   1.2 ระบบจะตรวจสอบว่ามีการ lock เป้าหมายไว้ไหม ถ้ามีการ lock จะตรวจสอบว่าตัวอักษรที่พิมพ์มาตรงกับตัวอักษรตัวแรกของศัตรูที่ lock ไว้ไหม ถ้าตรงให้ทำการยิงใส่ศัตรูและขยับไปตัวอักษรถัดไป
   1.3 ดักจับปุ่ม Esc เพื่อทำการ pause เกม
2. เกมทำการตรวจสอบเงื่อนไขดังนี้
   2.1 ถ้าศัตรูที่มีอยู่น้อยกว่าจำนวนสูงสุดที่ศัตรูมีได้ และเวลาผ่านไป 1500 ms นับจากการเกิดตัวแรกของศัตรู ให้ทำการ spawn_enemy() หรือก็คือเรียกศัตรูให้เกิดใหม่
   2.2 ถ้า kill ศัตรูเกิน 6 ตัวขึ้นไป ให้ทำการเพิ่มความเร็วของศัตรู 0.2 และเพิ่มจำนวนสูงสุดของศัตรู และทำการขยับ kill ที่ใช้เพิ่มทั้งสองอย่างไป 2 เท่า

2.3 ถ้าพิมพ์ติดกัน 5 ตัวอักษรโดยไม่ผิดให้ทำการเพิ่มแต้มคูณ combo ไป 0.25

2.4 ถ้าเริ่มเกมใหม่ระบบเพิ่มเลือดจะ run ครั้งเดียวเพื่อเป็นการเพิ่มเลือดจาก 0 ไป 4 หน่วย และถ้าเพิ่มเต็มให้หยุดเพิ่ม

2.5 ทำการวน loop ในศัตรูแต่ละตัวเพื่อให้ศัตรูตรวจหาดังนี้

ทำการตรวจสอบว่าศัตรูได้ชนกับตัวของ player หรือยัง ถ้าชนให้เกิดการระเบิดเกิดขึ้นและ reset ค่า combo เป็น 1 และ ตรวจสอบว่า กระสุนที่ player ยิงมานั้น ถึงตัวของศัตรูหรือยัง ถ้าถึงแล้วจะเช็คอีกว่า กระสุนที่ยิงเข้ามานั้นตรงกับตัวสุดท้ายที่พิมพ์หรือไม่ ถ้าตรงให้ทำลายศัตรูตัวนั้นและเกิดการระเบิดขึ้น แต่ถ้าไม่ ให้กระสุนนั้นหายไปแทน

2.6 ทำการเช็คว่ามีการกดปุ่ม pause เกมหรือไม่ ถ้ามีให้ทำการ pause เกม

2.7 ทำการเช็คว่าเลือดนั้นเหลือ 0 หรือยัง ถ้าเหลือ 0 ให้ gameover ทันที

3. ส่วนของการ draw component จะ rendering order ดังนี้

3.1 วาด background

3.2 วาดศัตรูที่มี

3.3 วาดกระสุนที่มีการยิง

3.4 วาด effect ระเบิดที่มีการเกิดขึ้น

3.5 วาด player

3.6 วาด game ui หน้าต่าง หลอดเลือด score ปุ่มในเกมต่างๆ

spritesheet.py

```
class SpriteSheet():
    def __init__(self, image, width, height, scale = 3):
        self.sheet = image
        self.scale = scale
        self.width = width
        self.height = height

    def get_img(self, frame, row, deg = 0):
        image = pygame.Surface((self.width, self.height)).convert_alpha()
        image.blit(self.sheet, (0, 0), ((frame * self.width), (row * self.height), self.width, self.height))
        image = pygame.transform.rotate(image, deg)
        image = pygame.transform.scale(image, (self.width * self.scale, self.height * self.scale))
        image.set_colorkey((0, 0, 0))
        return image
```

ไฟล์นี้จะมี class SpritesSheet เพื่อเอาไว้ใช้สร้าง animation จากรูปภาพดังตัวอย่างนี้

โดยทำการ crop รูปภาพให้ได้ size พอเหมาะกับ frame เดียวแล้วไล่ทำไปจนครบรูปที่มีแล้ว สร้างเป็น list เอาไว้ใช้ทำ animation ต่อไป

ui.py

```
class HealthBorder(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.health_border = pygame.image.load(HEALTH_BAR_BORDER_IMG).convert_alpha()
        self.image = pygame.transform.scale(self.health_border, (self.health_border.get_width() * IMG_SCALE,
        self.health_border.get_height() * IMG_SCALE))
        self.rect = self.health_border.get_rect(bottomleft = (10, WINDOW_HEIGHT-55))
```

```
class GameUI(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.health_img = pygame.image.load(HEALTH_BAR).convert_alpha()
        self.health_img = pygame.transform.scale(self.health_img, (self.health_img.get_width() * IMG_SCALE,
        self.health_img.get_height() * IMG_SCALE))
        self.health_img_rect = self.health_img.get_rect(bottomleft = (66, WINDOW_HEIGHT-4))
        self.max_health = self.health_img.get_width()
        self.health_amount = 0
        self.multiplier_score = 1
        self.score = 0

        self.popup_botton = Button("menu", GREEN, 100, 30, (55, 10), 6, 16)
        self.health_pack = pygame.sprite.Group()
        self.health_border = HealthBorder()
        self.health_pack.add(self.health_border)
        self.score_font = pygame.font.Font(MINECRAFT_FONT, 32)
        self.mulitple_font = pygame.font.Font(MINECRAFT_FONT, 28)
        self.lives_font = pygame.font.Font(MINECRAFT_FONT, 24)
```

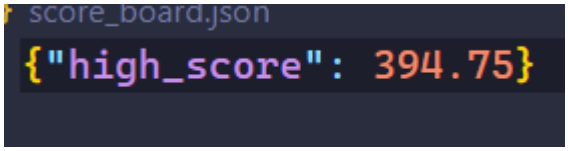
```python
def draw(self, screen):
    self.popup_botton.draw(screen)
    self.health_pack.draw(screen)
    score_text = self.score_font.render(f"Score {self.score}", True, WHITE)
    score_text_rect = score_text.get_rect(bottomright = (WINDOW_WIDTH - 10, WINDOW_HEIGHT - 10))
    multi_score_text = self.mulitple_font.render(f"{self.multiplier_score} X ", True, WHITE)
    multi_score_rect = multi_score_text.get_rect(midright = score_text_rect.midleft)
    lives_text = self.lives_font.render(f"{int(self.health_amount/DAMAGE)}", True, WHITE)
    lives_rect = lives_text.get_rect(center = (self.health_img_rect.midleft[0]-21, self.health_img_rect.
    midleft[1]-3))

    screen.blit(score_text, score_text_rect)
    screen.blit(multi_score_text, multi_score_rect)
    screen.blit(self.health_img, self.health_img_rect, (0, 0, self.health_amount, 100))
    screen.blit(lives_text, lives_rect)
```

```python
class OptionMenu:
    def __init__(self):
        self.box = pygame.Rect(0, 0, 300, 230)
        self.box.center = (WINDOW_WIDTH / 2, WINDOW_HEIGHT / 2)
        self.botton_resume = Button("Continue", GREEN, 130, 40, (self.box.centerx, self.box.centery - 10), 6,
        16)
        self.botton_exit = Button("Exit", RED, 130, 40, (self.box.centerx, self.box.centery + 50), 6, 16)
        self.score_font = pygame.font.Font(MINECRAFT_FONT, 26)

    def draw(self, screen, score):
        pygame.draw.rect(screen, WHITE, self.box, border_radius=12)
        self.draw_score(screen, score, "SCORE")
        self.botton_resume.draw(screen)
        self.botton_exit.draw(screen)

    def draw_game_over(self, screen, score):
        pygame.draw.rect(screen, WHITE, self.box, border_radius=12)
        self.draw_score(screen, score, "FINAL SCORE")
        self.botton_exit.draw(screen)

    def draw_score(self, screen, score, text):
        score_text = self.score_font.render(text+f" {score}", True, BLACK)
        score_text_rect = score_text.get_rect(center = (self.box.center[0], self.box.center[1] - 55))
        screen.blit(score_text, score_text_rect)
```

ไฟล์นี้เก็บหน้า ui component โครงสร้างทั้งหมด เช่นหลอดเลือด score หน้าต่าง pause ไว้ในไฟล์นี้ เพื่อให้ file game.py ไปเรียกใช้ต่อ

data/score_board.json

ใช้เก็บคะแนน high score



data/words.json

ใช้เก็บคลังคำศัพท์ที่เอาไว้นำไปใช้ในเกม