Metodología de las 6D

1.Descripción del problema

Se hará un juego estilo RPG en el que se crearan clases para representar personajes, con sus atributos como fuerza, salud, inteligencia, rapidez, etc.

2.Definición de la solución

Implementar sistemas de combate como por turnos o libremente, donde se pueda atacar, defender o usar habilidades especiales.

También un mapa con diferentes ubicaciones que el personaje pueda explorar.

Hacer enemigos los cuales pueden causar daños al personaje y ese enemigo sigue al personaje a un radio definido.

Animaciones del jugador al momento de moverse hacia arriba, abajo, izquierda y derecha.

Planificar el diseño del juego y la estructura del código. Esto incluye definir cómo se comportará el jugador, las animaciones, la lógica de colisiones, y cómo se implementará la cámara para seguir al jugador. Podrías crear diagramas o bocetos del flujo del juego y la disposición del HUD.

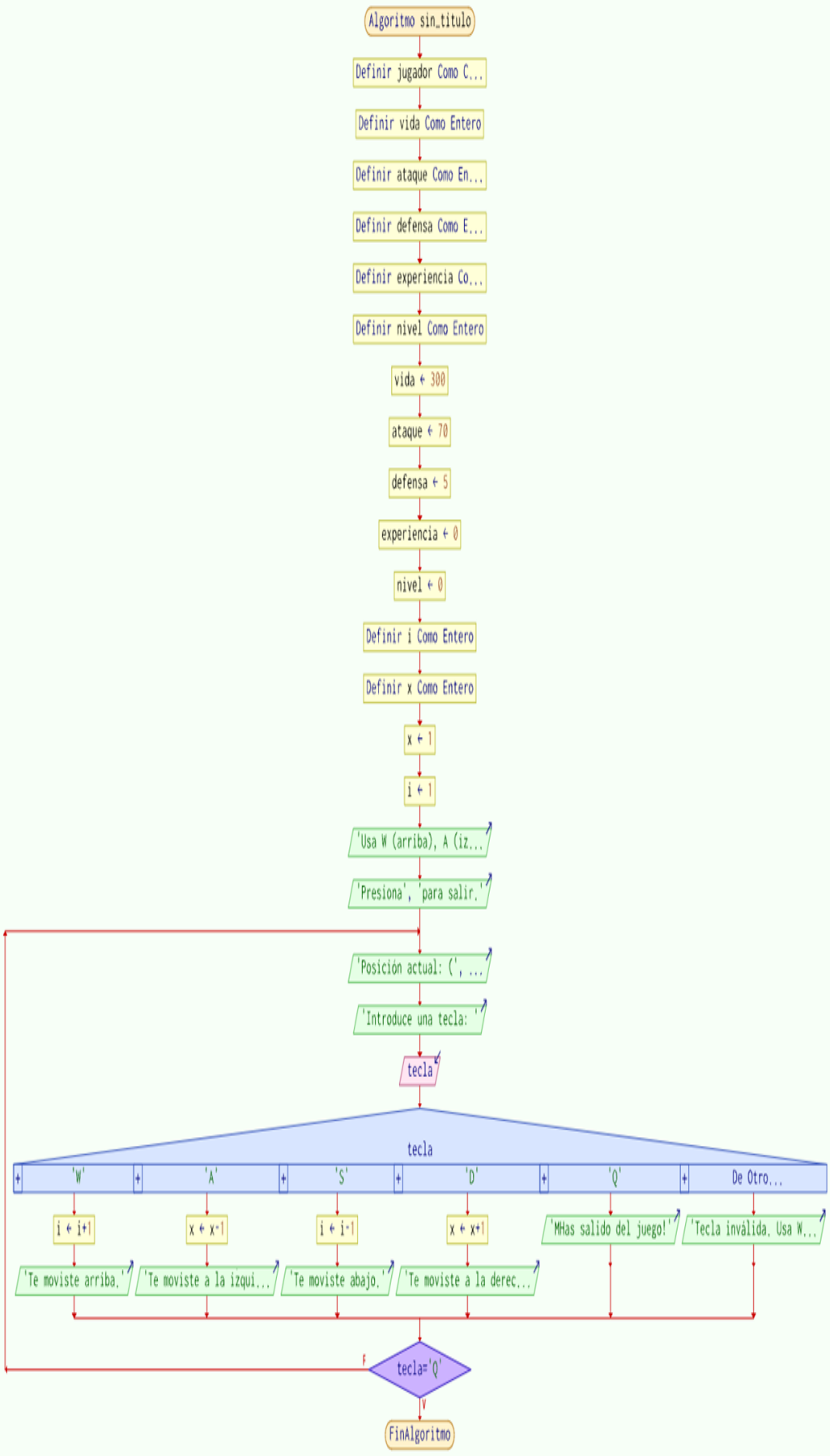
Implementar el juego en Pygame paso a paso. En esta fase, construirías las clases y funciones necesarias, como la clase Player, el sistema de colisiones, y la animación del personaje.

3.Diseño de la solución

```
Algoritmo sin_titulo
    definir jugador Como Caracter
    definir vida Como Entero
    definir ataque Como Entero
    definir defensa Como Entero
    definir experiencia Como Entero
```

```
    definir nivel Como Entero
    vida=300
    ataque=70
    defensa=5
    experiencia=0
    nivel=0

    Definir   i    Como Entero
    definir x Como Entero
  x <- 1
  i <- 1

  Escribir "Usa W (arriba), A (izquierda), S (abajo), D (derecha) para mover al jugador."
  Escribir "Presiona"  "para salir."

  Repetir
    Escribir "Posición actual: (", x, ",", i, ")"
    Escribir "Introduce una tecla: "
    Leer tecla

    Segun tecla Hacer
      "W":
        i <- i + 1
        Escribir "Te moviste arriba."
      "A":
```

```
            x <- x - 1
            Escribir "Te moviste a la izquierda."
        "S":
            i <- i - 1
            Escribir "Te moviste abajo."
        "D":
            x <- x + 1
            Escribir "Te moviste a la derecha."
        "Q":
            Escribir "¡Has salido del juego!"

        De Otro Modo:
            Escribir "Tecla inválida. Usa W, A, S, D o Q."
    FinSegun

Hasta Que tecla = "Q"


FinAlgoritmo
```


Algoritmo sin_titulo
Definir jugador Como C...
Definir vida Como Entero
Definir ataque Como En...
Definir defensa Como E...
Definir experiencia Co...
Definir nivel Como Entero
vida ← 300
ataque ← 70
defensa ← 5
experiencia ← 0
nivel ← 0
Definir i Como Entero
Definir x Como Entero
x ← 1
i ← 1
'Usa W (arriba), A (iz...
'Presiona', 'para salir.'
'Posición actual: (', ...
'Introduce una tecla: '
tecla
tecla
'W'
'A'
'S'
'D'
'Q'
De Otro...
i ← i+1
x ← x-1
i ← i-1
x ← x+1
'MHas salido del juego!'
'Tecla inválida. Usa W,...
'Te moviste arriba.'
'Te moviste a la izqui...
'Te moviste abajo.'
'Te moviste a la derec...
tecla='Q'
F
V
FinAlgoritmo

4.DESARROLLO DE LA SOLUCIÓN

```
#penultima prueba
import pygame
import random
# Definimos la ruta al directorio actual
path = "C:/Users/martinjr44/Desktop/"

# Inicializa Pygame
pygame.init()
clock = pygame.time.Clock()
# Define las dimensiones de la pantalla
screen_width = 1500
screen_height = 850
screen = pygame.display.set_mode((screen_width, screen_height))

# Función para cargar imágenes con manejo de errores
def load_image(path):
  try:
    return pygame.image.load(path).convert_alpha()
  except pygame.error as e:
    print(f"Error al cargar la imagen {path}: {e}")
    return None

player_width = 50
player_height = 50
player_x = 50
player_y = 50
# Clase para el Jugador
class Player:
  def __init__(self):
    # Carga las imágenes del jugador (animación)
    self.name = "master"
    self.player_speed= 0.5
    player = pygame.Rect(player_x, player_y, player_width, player_height)
    self.images = {
      "idle": [
        load_image("normal.png"),
        load_image("normal.png"),
        load_image("normal_2.png"),
        load_image("normal_2.png"),
      ],
      "walk_right": [
```

```
            load_image("correr_derecha_1.png"),
            load_image("correr_derecha_2.png"),
            load_image("correr_derecha_3.png"),
            load_image("correr_derecha_4.png"),
            load_image("correr_derecha_5.png"),
            load_image("correr_derecha_6.png"),
        ],
        "walk_left": [
            load_image("correr_izquierda_1.png"),
            load_image("correr_izquierda_2.png"),
            load_image("correr_izquierda_3.png"),
            load_image("correr_izquierda_4.png"),
            load_image("correr_izquierda_5.png"),
            load_image("correr_izquierda_6.png"),
        ],
        "walk_up": [
            load_image("correr_derecha_1.png"),
            load_image("correr_derecha_2.png"),
            load_image("correr_derecha_3.png"),
            load_image("correr_derecha_4.png"),
            load_image("correr_derecha_5.png"),
            load_image("correr_derecha_6.png"),
        ],
        "walk_down": [
            load_image("correr_izquierda_1.png"),
            load_image("correr_izquierda_2.png"),
            load_image("correr_izquierda_3.png"),
            load_image("correr_izquierda_4.png"),
            load_image("correr_izquierda_5.png"),
            load_image("correr_izquierda_6.png"),
        ],
        "attack_animacion": [
            load_image("correr_derecha_3.png"),
            load_image("correr_derecha_5.png"),
            load_image("atacar.png") # Añade más fotogramas si tienes
        ],
        "attack_animacion_projectile": [
            load_image("arco.png"),
            load_image("arco_2.png")  # Añade más fotogramas si tienes
        ]

    }

    # Escalar las imágenes del jugador
    self.player_size = (128, 128)
    for key, image_list in self.images.items():
```

```
      for i, image in enumerate(image_list):
        if image:  # Verifica que la imagen se haya cargado correctamente
          self.images[key][i] = pygame.transform.scale(image, self.player_size)
          # Agrandar el tamaño del jugador cuando este ataque
          if key == "attack_animacion":
            self.images[key][i] = pygame.transform.scale(image, (218 , 128))

    # Define la posición inicial del jugador
    self.pos = (150, 159)
    self.rect = self.images["idle"][0].get_rect(center=self.pos)

    # Estado actual del jugador
    self.state = "idle"
    self.current_frame = 0

    # Variables para la animación
    self.animation_speed = 10  # Velocidad de la animación (más pequeño, más rápido)
    self.animation_timer = 0
    self.projectiles = []
    # Estadísticas del jugador
    self.health = 300
    self.attack = 70
    self.defense = 5
    self.level = 1
    self.experience = 0

    # Control de la animación de ataque
    self.attack_timer = 0
    self.attack_cooldown = 1  # Enfriamiento entre ataques (en segundos)
    self.attack_animation_timer = 0  # Control del avance de la animación de ataque

  def update(self, delta_time):
    # Actualizar el temporizador de ataque usando delta_time
    if self.attack_timer > 0:
      self.attack_timer -= delta_time

    # Actualizar animaciones (ejemplo simplificado)
    if self.state == "attack_animacion":
      self.attack_animation_timer += delta_time * self.animation_speed
      if self.attack_animation_timer >= 1:
        self.attack_animation_timer = 0
        self.current_frame += 1
        if self.current_frame >= len(self.images["attack_animacion"]):
          self.state = "idle"  # Cambiar estado a idle
          self.current_frame = 0
    else:
      self.animation_timer += delta_time * self.animation_speed
```

```
      if self.animation_timer >= 1:
        self.animation_timer = 0
        self.current_frame = (self.current_frame + 1) % len(self.images[self.state])


  # Aquí se corrige la definición del método draw
  def draw(self, screen, camera_x, camera_y):
    # Verificar que el estado y el índice de la animación son válidos
    if self.state in self.images and 0 <= self.current_frame <
len(self.images[self.state]):
      # Resta las coordenadas de la cámara para hacer que el jugador se mueva en
función de la cámara
      screen.blit(self.images[self.state][self.current_frame],
            (self.rect.x - camera_x, self.rect.y - camera_y))

  def receive_damage(self, damage):
    # enfriamiento entre ataques
    if self.attack_timer < 0:
      self.attack_timer = 0.05
      # Si el enemigo ha sido derrotado, recibe experiencia
    self.attack_timer = 0.05
    damage_taken = max(0, damage - self.defense)
    self.health -= damage_taken
    print(f"{self.health} - {damage_taken} (Daño recibido)")
    if self.health <= 0:
      self.health = 0  # Evitar que la salud sea negativa
      print("¡Has muerto!")
      return False  # El jugador ha muerto
    return True

  def attack_enemy(self, enemy):
    # Ataca solo si el cooldown ha terminado
    if self.attack_timer <= 0:
      damage = random.randint(self.attack - 2, self.attack + 2)
      print(f"{self.name} ataca a {enemy.name} con {damage} de daño.")
      enemy.receive_damage(damage)
      self.attack_timer = self.attack_cooldown  # Reinicia el cooldown
      self.state = "attack_animacion"
      return True
    return False

  # atacar al enemigo con proyectil a distancia
  def shoot_projectile(self):
    if self.attack_timer <= 0:
      direction = -1 if self.state == "walk_right" else 1
      projectile = Projectile(self.rect.centerx, self.rect.centery, direction)
      self.projectiles.append(projectile)
```

```
      self.attack_timer = 0.1  # Cooldown de 0.5 segundos

  def update_projectiles(self, enemies):
    for projectile in self.projectiles[:]:  # Iterar sobre una copia para evitar problemas al eliminar elementos
      projectile.move()
      for enemy in enemies:
        if projectile.rect.colliderect(enemy.rect):
          enemy.receive_damage(projectile.damage)
          if projectile in self.projectiles:
            self.projectiles.remove(projectile)  # Eliminar el proyectil tras el impacto
          break
      if projectile.rect.x < 0 or projectile.rect.x > screen_width:  # Si sale de la pantalla
        self.projectiles.remove(projectile)

  def draw_projectiles(self, screen, camera_x, camera_y):
    for projectile in self.projectiles:
      pygame.draw.rect(screen, (255, 255, 0),
              (projectile.rect.x - camera_x, projectile.rect.y - camera_y, projectile.rect.width, projectile.rect.height))

  def level_up(self):
    if self.experience >= 100 * self.level:
      self.level += 1
      self.attack += 5
      self.defense += 2
      self.health = 100 + self.level * 10
      self.experience = 0
      print(f"¡Nivel {self.level} alcanzado! Salud: {self.health}, Ataque: {self.attack}, Defensa: {self.defense}")

class Projectile:
  def __init__(self, x, y, direction, speed=10, damage=20):
    self.rect = pygame.Rect(x, y, 10, 10)  # Tamaño del proyectil
    self.direction = direction  # Direcciones posibles: (-1, 0, 1)
    self.speed = speed
    self.damage = damage

  def move(self):
    self.rect.x += self.speed * self.direction
```

```
  def draw(self, screen):
    pygame.draw.rect(screen, (255, 255, 0), self.rect)  # Amarillo para el proyectil

WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
class SimpleEnemy:
  def __init__(self, x, y, health, damage, sprite_paths, target_x, target_y,
name="enemigo"):
    self.health = health
    self.damage = damage
    self.name = name
    # Animación
    self.sprites = [pygame.image.load(path).convert_alpha() for path in sprite_paths]
    self.sprites = [pygame.transform.scale(sprite, (150, 150)) for sprite in
self.sprites]
    self.current_sprite_index = 0
    self.animation_timer = 0
    self.animation_speed = 10

    # Posición y movimiento
    self.rect = self.sprites[0].get_rect(center=(x, y))
    self.start_x, self.start_y = x, y
    self.target_x, self.target_y = target_x, target_y
    self.direction_x = 1
    self.direction_y = 1

    # Cooldown para atacar
    self.attack_cooldown = 60  # 1 segundo si el juego está a 60 FPS
    self.attack_timer = 0

  def move(self):
    if self.direction_x == 1 and self.rect.x >= self.target_x or self.direction_x == -
1 and self.rect.x <= self.start_x:
      self.direction_x *= -1
    self.rect.x += 2 * self.direction_x

    if self.direction_y == 1 and self.rect.y >= self.target_y or self.direction_y == -
1 and self.rect.y <= self.start_y:
      self.direction_y *= -1
    self.rect.y += 2 * self.direction_y

  def animate(self):
    self.animation_timer += 1
    if self.animation_timer >= self.animation_speed:
      self.animation_timer = 0
```

```
        self.current_sprite_index = (self.current_sprite_index + 1) % len(self.sprites)

    def draw(self, screen, camera_x, camera_y):
        screen.blit(self.sprites[self.current_sprite_index], (self.rect.x - camera_x,
self.rect.y - camera_y))
        pygame.draw.rect(screen, RED, (self.rect.x - camera_x, self.rect.y - 10 -
camera_y, 40, 5))
        pygame.draw.rect(screen, GREEN, (self.rect.x - camera_x, self.rect.y - 10 -
camera_y, 40 * (self.health / 100), 5))

    def deal_damage_to_player(self, player):
        if self.rect.colliderect(player.rect):
            if self.attack_timer <= 0:  # Verificar si el cooldown ha terminado
                player.receive_damage(self.damage)
                self.attack_timer = self.attack_cooldown  # Reinicia el cooldown

    def update_timer(self):
        if self.attack_timer > 0:
            self.attack_timer -= 1


    def receive_damage(self, damage):
        """Reduce la salud del enemigo y verifica si muere."""
        self.health -= damage
        print(f"Enemigo recibe {damage} de daño. Salud restante: {self.health}")
        if self.health <= 0:
            print("¡Enemigo derrotado!")
            return False
        return True

# Crea una lista de enemigos
enemies = [
    SimpleEnemy(800, 200, 500, 10, ["fantasma de sangre.png", "fantasma de sangre.gif",
"fantasma de sangre.png"], 200, 400),
    SimpleEnemy(200, 1500, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
    SimpleEnemy(100, 1500, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
    SimpleEnemy(50, 1500, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
    SimpleEnemy(10, 1500, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
    SimpleEnemy(200, 1900, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
    SimpleEnemy(100, 1900, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
```

```
  SimpleEnemy(50, 1900, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
  SimpleEnemy(10, 1900, 1000, 30, ["Attack.png", "Attack_2.png", "attack_3.png"],
1200, 2),
]
```

```
# Función para dibujar el HUD
def draw_hud(screen, player):
  font = pygame.font.SysFont("Arial", 20)
  health_text = font.render(f"Salud: {player.health}", True, (255, 1, 1))
  attack_text = font.render(f"Ataque: {player.attack}", True, (1, 255, 1))
  defense_text = font.render(f"Defensa: {player.defense}", True, (1, 1, 255))
  level_text = font.render(f"Nivel: {player.level}", True, (200, 200, 100))
  experience_text = font.render(f"Experiencia: {player.experience}", True, (255, 1,
255))

  screen.blit(health_text, (10, 10))
  screen.blit(attack_text, (10, 40))
  screen.blit(defense_text, (10, 70))
  screen.blit(level_text, (10, 100))
  screen.blit(experience_text, (10, 130))
```

```
background_image = load_image("experimento_grande.png")
background_rect = background_image.get_rect() if background_image else None

# Crea una superficie para la sombra
shadow_surface = pygame.Surface((screen_width, screen_height), pygame.SRCALPHA)

# Crea una instancia del jugador
player = Player()
```

```
# Bucle principal del juego
clock = pygame.time.Clock()

# Definir los límites del mapa (puedes ajustar estos valores dependiendo del tamaño
del mapa)
map_width = 3000  # Ejemplo de un mapa más grande que la pantalla
map_height = 3000
```

```
# Coordenadas iniciales de la cámara
camera_x = 0
camera_y = 0
camera_speed = 5


# Lista de rectángulos
rects = [
  pygame.Rect(220, 10, 1400, 200),
  pygame.Rect(50, 1, 120, 80),
  pygame.Rect(10, 50, 20, 2000),
  pygame.Rect(50, 700, 500, 650),
  pygame.Rect(400, 1780, 950, 230),
  pygame.Rect(1260, 700, 1000, 400),
  pygame.Rect(1590, 200, 700, 250),
  pygame.Rect(10, 2000, 2300, 30),
  pygame.Rect(2300, 200, 50, 2000),
  pygame.Rect(950, 860, 50, 85)
]


running = True
while running:

  delta_time = clock.tick(120) / 1000.0

  screen.fill((0, 0, 0))  # Limpiar la pantalla

  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False


  # Obtén las teclas presionadas
  keys = pygame.key.get_pressed()

  # Movimiento del jugador
  if keys[pygame.K_LEFT]:
    player.rect.x -= 5
    player.state = "walk_left"
    if any(player.rect.colliderect(rect) for rect in rects):
      player.rect.x += 5
      player.state = "idle"

  elif keys[pygame.K_RIGHT]:
    player.rect.x += 5
```

```
    player.state = "walk_right"
    if any(player.rect.colliderect(rect) for rect in rects):
      player.rect.x -= 5
      player.state = "idle"

  elif keys[pygame.K_UP]:
    player.rect.y -= 5
    player.state = "walk_up"
    if any(player.rect.colliderect(rect) for rect in rects):
      player.rect.y += 5
      player.state = "idle"

  elif keys[pygame.K_DOWN]:
    player.rect.y += 5
    player.state = "walk_down"
    if any(player.rect.colliderect(rect) for rect in rects):
      player.rect.y -= 5
      player.state = "idle"

  elif keys[pygame.K_SPACE]:  # Atacar
    player.state = "attack_animacion"
    for enemy in enemies:
      if player.rect.colliderect(enemy.rect):
        player.attack_enemy(enemy)

  elif keys[pygame.K_w]:
    player.shoot_projectile()
    player.state = "attack_animacion_projectile"


  else:
    player.state = "idle"  # Si no se presionan teclas
  # Actualizar proyectiles y enemigos

  # Actualiza la animación del jugador
  player.update(delta_time)


  # Variables de la cámara
  camera_x = max(0, min(player.rect.centerx - screen_width // 2, map_width -
screen_width))
  camera_y = max(0, min(player.rect.centery - screen_height // 2, map_height -
screen_height))
```

```
  # Limita el movimiento de la cámara
  camera_x = max(0, min(camera_x, map_width - screen_width))
  camera_y = max(0, min(camera_y, map_height - screen_height))

  # Dibuja el fondo
  if background_image:
    screen.blit(background_image, (-camera_x, -camera_y))

  # Dibuja la sombra
  screen.blit(shadow_surface, (0, 0))

  # Dibujar los rectángulos (colisiones o obstáculos)
  for rect in rects:
    rect_screen_x = rect.x - camera_x
    rect_screen_y = rect.y - camera_y
    #pygame.draw.rect(screen, (255, 0, 0), (rect_screen_x, rect_screen_y, rect.width,
rect.height))


  # Dibujar al jugador
  player.draw(screen, camera_x, camera_y)
  # si la salud de player es menor a 0 el juego termina
  if player.health <= 0:
    print("Game Over!")
    running = False

# Actualizar y dibujar cada enemigo
  for enemy in enemies:
    enemy.move()
    enemy.animate()
    enemy.deal_damage_to_player(player)  # Verifica colisiones y daño
    enemy.update_timer()  # Actualiza el temporizador del ataque
    enemy.draw(screen, camera_x, camera_y)
    # Detener el bucle cuando todos los enemigos han muerto
    if enemy.health <= 0:
      enemies.remove(enemy)
      if not enemies:
        print("��Todos los enemigos han muerto!")

  player.update_projectiles(enemies)  # Mover y verificar colisiones de proyectiles
  player.draw_projectiles(screen, camera_x, camera_y)  # Dibujar proyectiles
  draw_hud(screen, player)

  pygame.display.update()
  clock.tick(60)  # 60 FPS

```

5. DEPURACIÓN Y PRUEBAS

6. Documentación

Se creo un personaje que responda a movimientos como ir a la izquierda, derecha, arriba y abajo.

Se uso una imagen para el personaje estilo mediaval como un caballero y se le agrego una espada la cual permite atacar.

Al personaje se le agrego salud, fuerza y ataque para poder golpear al enemigo.

El enemigo se creo para que siga al personaje queriendo golpear, a este también se le agrego daño, salud, rapidez de ataque y se uso una imagen para el enemigo.

Se creo un mapa donde puedes ir a cualquier parte en el y te permite tener libertad al momento de moverte.

Se define lo que son los bordes del mapa o sea los limites del mapa para que el personaje y el enemigo no se puedan salir de el.

Se le puso al personaje una velocidad un poco mas rápida que al enemigo para que sea mas fácil atacar y moverte.

Se usaron librerías de Pygame para la aclaración de dudas y métodos para que el juego tenga un mejor desarrollo.

Hay directorios donde contienen imágenes para el mapa, las animaciones del personaje, el personaje, el enemigo y las animaciones del enemigo.

Agregamos las estadísticas del personaje en la parte superior izquierda que es la Salud, energía y otras mas.

librerias: pygame 2.6.1, (SDL 2.28.4, Python 3.12.6)

import pygame

import Random