Documento post mortem Individual

Miguel Angel Cardenas Cardenas - 202010998

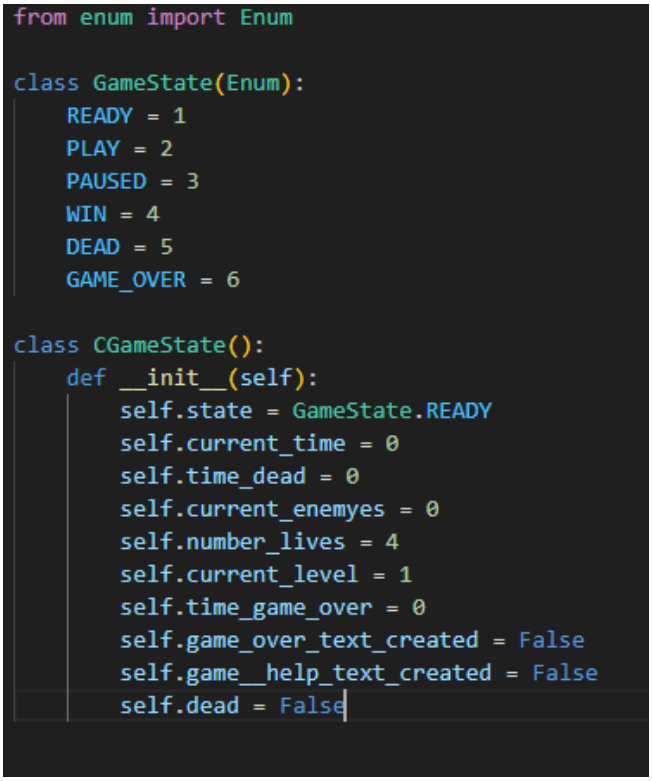
Rol y trabajo dentro del proyecto: mi principal rol y trabajo dentro del proyecto fue el desarrollo de tres features principales que mencionaré a continuación:

- Implementación de pause en el juego.
- Implementación de la escena ready al comenzar el juego.
- Implementación de la escena game over al finalizar el juego.

Adicionalmente, también trabajé features adicionales y algunos estéticos, que son los siguientes:

- Implementación de vidas, tanto en su funcionamiento como en su forma visual, al pintar en pantalla las 3 vidas del jugador.
- Implementación del reinicio de nivel, tanto en su funcionamiento como en su forma visual, al pintar la banderita de nivel y el número de nivel actual.

Para cumplir con los requerimientos anteriores, desarrollé mediante el patrón State, un sistema de estados que controla todos los estados del juego, siguiendo las recomendaciones del grupo docente. Se definió una clase CGameState que se encarga de guardar el estado y las variables de estado del juego, como se ve a continuación:



```python
from enum import Enum

class GameState(Enum):
    READY = 1
    PLAY = 2
    PAUSED = 3
    WIN = 4
    DEAD = 5
    GAME_OVER = 6

class CGameState():
    def __init__(self):
        self.state = GameState.READY
        self.current_time = 0
        self.time_dead = 0
        self.current_enemyes = 0
        self.number_lives = 4
        self.current_level = 1
        self.time_game_over = 0
        self.game_over_text_created = False
        self.game__help_text_created = False
        self.dead = False
```

Como se puede observar, este componente permite almacenar el estado, y usarlo para gestionar de forma global todo lo que sucede en el juego. Para esto, se definieron 6 estados principales:

READY, es la introducción del juego; PLAY, el jugador está jugando; PAUSE, el jugador pausa el juego; WIN, el jugador elimina a todos los enemigos y pasa de nivel, DEAD, el jugador muere y dura unos segundos en reaparecer y GAME_OVER, el jugador pierde todas sus vidas. Así mismo, se definió un sistema que se encarga de gestionar estos estados del juego y los cambios según sea el caso. Por ejemplo, cuando el tiempo actual una vez se inicia el juego es mayor o igual a 3 segundos, la introducción a terminado, y por ende el texto ready debe desaparecer, los enemigos deben ser creados y pintados en el canvas y el estado del juego (controlado mediante el componente CGameState) debe cambiar de READY a PLAY. Este proceso es similar para los demás estados en caso de que se cumplan ciertas condiciones para pasar de un estado a otro, para ejemplificar, se puede mencionar que se lleva un contador que registra el número de enemigos, al ser 0, el jugador gana, por lo que su estado cambia (PLAY a WIN), y ese cambio es manejado por el sistema, el cual se encarga de aumentar el nivel, crear el texto que indica haber ganado y crear el nuevo nivel.

```python
def system_game_manager(world : esper.World, delta_time: float, level_cfg, enemies_cfg, game_manager_entity, interface_cfg, player_entity, player_info):
    manager_component = world.component_for_entity(game_manager_entity, CGameState)
    component = world.get_component(CTagReady)
    player_surface = world.component_for_entity(player_entity, CSurface)
    player_pos = world.component_for_entity(player_entity, CTransform)
    life_component = world.get_component(CTagLife)
    level_components = world.get_components(CSurface, CTagLevel)

    if manager_component.state != GameState.PAUSED:
            manager_component.current_time += delta_time

    if manager_component.current_time >= 3:
        if  manager_component.state == GameState.READY:
            for entity, (c_t) in component:
                world.delete_entity(entity)
            enemies = create_level(world, level_cfg, enemies_cfg, interface_cfg)
            manager_component.state = GameState.PLAY
            manager_component.current_enemyes = enemies

        elif manager_component.current_enemyes == 0 and manager_component.state == GameState.PLAY:

            win =create_text(world,
                        interface_cfg["win"])
            ready_win = create_text(world,
                        interface_cfg["ready_win"])
            world.add_component(win, CTagReady())
            world.add_component(ready_win, CTagReady())
            manager_component.state = GameState.WIN
            ServiceLocator.sounds_service.play(interface_cfg["ready"]["sound"])
            manager_component.current_time = 0
            update_high_score_value(world)

        elif manager_component.state == GameState.WIN:
            manager_component.current_level += 1
            for entity, (c_t) in component:
                world.delete_entity(entity)
            for entity, (c_s, c_lev) in level_components:
                font = ServiceLocator.fonts_service.get_font(interface_cfg["level_text"]["font"], interface_cfg["level_text"]["size"])
                text_surface = font.render(str("0")+str(manager_component.current_level), True, pygame.Color(interface_cfg["level_text"]["color"]["r"],
                                                                                                              interface_cfg["level_text"]["color"]["g"],
                                                                                                              interface_cfg["level_text"]["color"]["b"]))
```

Qué salió bien, mal y que cambiarían para un nuevo proyecto a título personal: personalmente, creo que nada salió mal en este proyecto, siendo completamente sincero, el grupo con el que estuve trabajando fue uno de los grupos más responsables en los que he estado. Todos estuvieron pendientes del proyecto, cumplían con ciertas fechas específicas para la realización de sus requerimientos, y ayudaban a los demás en caso de ser necesario. Quizá me hubiera gustado estar mucho más involucrado en el proyecto aunque trabajé bastante en el control de estados del juego. No estuve tan activo debido a ciertos problemas de salud y mi carga académica por la realización de la tesis, sin embargo, logré los objetivos del proyecto. Creo que el hecho de intentar replicar un juego hizo mucho más fácil el trabajo, y tal vez hubiera sido interesante y retador implementar features adicionales distintas a las ya existentes de forma obligatoria. Creo que lo una de las cosas que más rescato en un proyecto, especialmente de videojuegos es un buen equipo de trabajo, si tuviera la oportunidad de trabajar

nuevamente con mis compañeros lo haría, pues esto facilitó mucho las tareas a la hora de escribir código y de probarlo, muchas veces se me pasó algún error que fue encontrado por mis compañeros, y por ende pude corregirlo.

Resumen de lo aprendido y análisis de lo aprendido en relación con la creación de videojuegos: la verdad aprendí mucho de este curso, desde cómo funciona un juego simple, hasta como se diseña un juego complejo y para diferentes plataformas. Una de las cosas que más me sorprendió es todo este mundo de ECS y la programación orientada a datos. Es algo que anteriormente no conocía, y que me puede ser útil en algún momento, especialmente porque uno de mis hobbies es los videojuegos. También aprendí demasiado sobre los motores en los videojuegos, especialmente en la realización de los talleres y el proyecto del curso, poniendo en práctica todo lo aprendido. Creo que la industria de los videojuegos y su desarrollo es algo muy complejo, no solamente de programar, sino el concepto del juego en sí mismo, especialmente desde el desarrollo y diseño, pues crear un juego "divertido" no es nada facil, y crear una experiencia de juego que sea agradable y sostenible es muy difícil. A pesar de eso, es un mundo muy interesante que con sus avances permite llevar felicidad a muchas personas al rededor del mundo. Finalmente, lo que más rescato del curso es que la industria de los videojuegos está en crecimiento constantemente, y que gracias a ello existen muchas herramientas que permiten llevar nuestra imaginación a lo más alto, y dar a conocer nuestras ideas al mundo.