Grupo H

El proyecto final de programación se trata de la codificación de un videojuego desde cero. Este constó con diferentes partes de entrega y en este momento nos encontramos en la cuarta y última entrega, en la cual detallaremos los objetivos, mecánicas y funcionamiento del código.

Objetivos:

1.- Al comenzar este proyecto nos pusimos como objetivo principal de que este videojuego tuviera la esencia de los míticos arcades games, pero con nuestra propia marca, la cual logramos con la mezcla de un juego de laberintos y los viajes espaciales, tomando como referencia los juegos Space invaders y The binding of isaac

2.- Durante la codificación del juego nos surgió un nuevo objetivo que como creadores teníamos, el cual es: presentar un juego fácil, pero con un leve grado de dificultad para que se necesaria un poco de atención para poder pasarse el juego.

Mecánicas:

Mecánicas espaciales (Nave espacial): El objetivo principal de este nivel es sobrevivir, esquivando los asteroides que se presentarán de manera aleatoria en nuestro viaje intergaláctico. Sus mecánicas de movilidad son:

- W. Movimiento hacia delante.
- D. Movimiento hacia la derecha.
- A. Movimiento hacia la izquierda.
- S. Movimiento hacia atrás.
- ESPACIO. Lanzar disparos de la nave

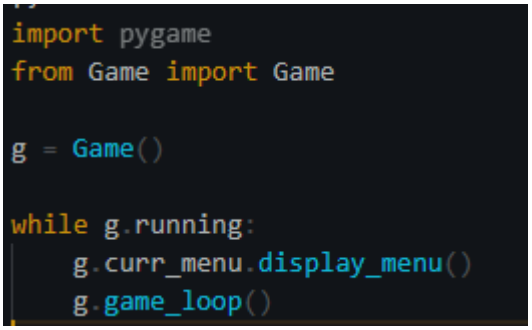
Mecánicas personaje principal: Estas mecánicas se utilizaran para el movimiento del personaje en los distintos escenarios que se presentarán a lo largo del juego.

- W. Movimiento hacia delante.
- D. Movimiento hacia la derecha.
- A. Movimiento hacia la izquierda.
- S. Movimiento hacia atrás.

- E. Para interactuar.

Funcionamiento del código:

Para el funcionamiento en general, creamos el archivo “main” donde se inicia el juego:



```
import pygame
from Game import Game

g = Game()

while g.running:
    g.curr_menu.display_menu()
    g.game_loop()
```

Como cerebro de esta operación tenemos el archivo “Game”, el cual se usará como una base de datos donde tendremos las funciones a utilizar al resto de archivos, como es en el caso de “main”. En este archivo también se presenta el loop principal, que abarca desde el menú hasta los niveles del juego, y utilizando la variable *self*, se conectan las variables, funciones y clases que tenemos en otros archivos, ya sean .py, o de imágenes.

```python
import pygame
from pygame.mixer import pause
from menu import *
from stage2 import *
from pygame import image as img
import sys

class Game():
    #weas basicas del propio juego
    def __init__(self):
        pygame.init()
        self.running = True
        self.playing = False
        self.stage = True
        self.W_KEY = False
        self.S_KEY = False
        self.START_KEY = False
        self.BACK_KEY = False
        self.QUIT_KEY = False
        self.DISPLAY_W = 1280
        self.DISPLAY_H = 720
        self.display = pygame.Surface((self.DISPLAY_W,self.DISPLAY_H))
        self.window = pygame.display.set_mode(((self.DISPLAY_W,self.DISPLAY_H)))
        #self.font_name = 'C:/Users/httpf/OneDrive/Escritorio/Universidad/programacion/Codigos/moscas be like' #xd
        self.font_name = 'Arial'
        #self.font_name = pygame.get_default_font() # por cambiar
        self.negro = (0,0,0)
        self.blanco = (255,255,255)
        self.main_menu = MainMenu(self)
        self.options = OptionsMenu(self)
        self.credits = CreditsMenu(self)
        self.fondo_menu = img.load('fondo_menu.jpeg')
        #self.control = Controles(self)
        #self.stage2 = juego(self)
        self.curr_menu = self.main_menu

    def game_loop(self):
        if self.playing: #condicion por si muere, resetea el juego y las vidas
            self.stage = True
        while self.playing:
            self.check_events()
            if self.START_KEY:
                self.playing = False
            if self.stage:
                self.stage2 = juego.correr(self.stage) #condicion para correr el juego, en este caso stage 2
            self.display.fill(self.negro)
            self.draw_text('Moriste', 20, self.DISPLAY_W/2 ,self.DISPLAY_H/2)
```

Tanto para los sub-loops que manejan los menús y los niveles a jugar, son llamados y utilizados en el archivo Game.py de la manera que se mencionó anteriormente, esto se usa primordialmente para volver a utilizar estas clases o funciones las veces que queramos y en los archivos que queramos, con el código *self*.game.nombre(haciendo referencia a lo que queremos), como se muestra a continuación:

```python
class MainMenu(menu):
    def __init__(self, game):
        menu.__init__(self, game)
        self.state = 'Start' #empezar
        self.startx = self.mid_w +25
        self.starty = self.mid_h +80 # estos datos son posicion en la pantalla
        self.optionsx = self.mid_w +25
        self.optionsy = self.mid_h +100
        self.creditsx= self.mid_w +25
        self.creditsy = self.mid_h + 120
        self.cursor_rect.midtop = (self.startx + self.offset, self.starty)

    def display_menu(self):
        self.run_display = True
        while self.run_display:
            self.game.check_events()
            self.check_input()
            self.game.display.fill(self.game.negro)
            self.game.display.blit(self.game.fondo_menu, (0,0))
            self.game.draw_text('Moscas be like', 50, self.game.DISPLAY_W /2, self.game.DISPLAY_H/4) #titulo
            self.game.draw_text('Jugar', 30, self.startx,self.starty)
            self.game.draw_text('Opciones', 25,self.optionsx, self.optionsy)
            self.game.draw_text('Creditos', 25, self.creditsx, self.creditsy)
            self.draw_cursor2()
            self.blit_screen()
```

Sin embargo, no todas nuestras partes funcionan de esta manera, ya que tambien necesitabamos variables o funciones independientes que se valgan por sí mismas, como lo hicimos con la pausa:

```python
def pausa():

    pausa = True
    while pausa:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN: #Quitar la Pausa
                    pausa = False
                if event.key == pygame.K_ESCAPE:
                    pausa = False
                elif event.key == pygame.K_q: #Salir del juego
                    pygame.quit()
                    quit()


        win.fill((0,0,0))
        dibujar_texto('Pausa', 60, blanco, win, 1280/2, (720/2) -50)
        dibujar_texto('Presione Escape o Enter para volver al juego', 40, blanco, win, 1280/2, (720/2) +20)
        dibujar_texto('Presione "Q" para cerrar el juego', 40, blanco, win, 1280/2, (720/2) +70)
        pygame.display.update()
```

La idea aquí, es llamar a la pausa en cada nivel por separado, importando y llamando a esta función, funciona las veces que se aplique, y no afecta al resto del código.

Objetivos no logrados:

Este proyecto fue un sueño hecho realidad, pero nunca pensamos lo difícil que esto podía llegar a ser y es por eso que algunas cosas que nos propusimos no se concluyeron, estas son:

- En primer lugar: El enemigo. Nosotros habíamos diseñado toda una historia de cómo el protagonista caía sobre la torta de cumpleaños del hijo del enemigo, por lo que este lo empezaba a perseguir para así poder vengarse, esto se puede ver en el informe anterior, donde incluso agregamos el escenario del planeta enemigo. Por lo mismo, finalmente tuvimos que optar por hacer un nivel estilo laberinto.
- En segundo lugar: Fue la incorporación de las moscas. Es irónico pero en nuestro juego nunca hay moscas, esto ocurrió debido a algunos errores que el código nos arrojaba y es por eso que no las incluimos, a causa de lo antes mencionado tuvimos que modificar algunos escenarios del juego, en conclusión, el juego se tratara del viaje que tendrá nuestro personaje para encontrar el “elemento” y así salvar su planeta. Por lo mismo se eliminó el escenario del planeta tierra.(esta fue una decisión tomada a último minuto y es por lo mismo que no se puede apreciar en el video la tercera etapa.)