# Informe del Juego, Micael Cichello Hensen

## Explicación del código

Inicialmente el programa empieza importando las librerías necesarias, como pygame, numpy, time, etc. Luego se carga el archivo de la música y se empieza a reproducir.

Se declara el tamaño y resolución de los pixeles del tablero y cuantos contendrá el mismo, además de especificar el color y determinar cuándo una celda está muerta o viva.

Posteriormente declara las variables de posición (xpos, por ejemplo), las de velocidad (xvel, por ejemplo) y las de tiempo (xtiempo, por ejemplo). Inicializa un bucle while que nunca parará, ya que esta condicionado por la variable “stay”, la cual no vuelve a nombrarse en el programa. Este bucle corre las funciones necesarias para alterar la velocidad cada vez que se presiona una tecla.

Luego se presenta una seguidilla de bucles for, definiendo cada una de las acciones al presionar las teclas de control de la nave. Todo esto se acompaña por una porción de código que se encarga de pintar la celda que el mouse presiona.

Lo que sigue es la parte del código que establece y borra las celdas con forma de la nave, consta de unas líneas de código que cambian el estado de las celdas a viva o muerta, dependiendo de donde se encuentre la nave.

Procediendo, se encuentra el bucle for que se encarga de las físicas de los disparos, el color de los mismos y su velocidad y posición de aparición. Luego se define el movimiento de los escombros, determinando nuevamente su velocidad y si aleatoriedad en su posición de aparición. También se define la interacción entre los disparos y los escombros. El código finaliza contando cada una de las partículas destruidas por los disparos, y digita el numero en la ventana de comandos.

El primer inciso que decidí encarar, fue el de cambiar el color de los disparos. Realmente no me fue muy difícil averiguarlo

```
# Si la celda está "viva" pintamos un recuadro relleno de color
elif gameState[x, y] == 3:
    pygame.draw.polygon(screen, (200, 100, 0), poly, 0)
elif gameState[x, y] == 2:
    pygame.draw.polygon(screen, (200, 255, 255), poly, 0)
else:
    pygame.draw.polygon(screen, (200, 100, 100), poly, 0) #color
```

Simplemente era agregar un elif y cambiar la referencia de valor a 2, para luego declarar el color con código RGB

Luego decidí cambiarle la canción, esta línea de código se encuentra casi al principio

```
#Instantiate mixer
mixer.init()

#Load audio file
mixer.music.load('cancion.mp3') # musica

print("music started playing....")

#Set preferred volume
mixer.music.set_volume(0.2)

#Play the music
mixer.music.play()
```

En este caso hay que cambiar el nombre del archivo por el nombre del archivo de audio .mp3

Luego también agregué un efecto de sonido para los disparos:

```
# Detectamos si se presiona una tecla.
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_LEFT:
        xvel = xvel - 1
    elif event.key == pygame.K_RIGHT:
        xvel = xvel + 1
    elif event.key == pygame.K_UP:
        yvel = yvel - 1
    elif event.key == pygame.K_DOWN:
        yvel = yvel + 1
    elif event.key == pygame.K_SPACE:
        gameState[xpos_canon,ypos_canon] = 2
        mixer.music.load('piu.mp3')
        mixer.music.play()
    else:
        pauseExect = not pauseExect
```

En este caso había que primero cargar el archivo de audio y luego reproducirlo. Todo en la sección del código que se encarga de los eventos al presionar teclas.

Luego cambie levemente el diseño de la figura, basándome en las naves de starwars, cuyas partes se encuentran separadas pero sostenidas por una especie de fuerza invisible.

```
#H
gameState[bxpos+4,59+bypos] = 0
gameState[bxpos+4,58+bypos] = 0
gameState[bxpos+4,57+bypos] = 0
gameState[bxpos+4,56+bypos] = 0
gameState[bxpos+4,55+bypos] = 0
gameState[bxpos+5,57+bypos] = 0
gameState[bxpos+6,59+bypos] = 0
gameState[bxpos+6,58+bypos] = 0
gameState[bxpos+6,57+bypos] = 0
gameState[bxpos+6,56+bypos] = 0
gameState[bxpos+6,55+bypos] = 0
#Alas
gameState[bxpos+5,57+bypos] = 0
gameState[bxpos+9,57+bypos] = 0
gameState[bxpos+5,58+bypos] = 0
gameState[bxpos+9,58+bypos] = 0
gameState[bxpos+4,58+bypos] = 0
gameState[bxpos+10,58+bypos] = 0
gameState[bxpos+4,59+bypos] = 0
gameState[bxpos+10,59+bypos] = 0


#H
gameState[xpos+4,59+ypos] = 1
gameState[xpos+4,58+ypos] = 1
gameState[xpos+4,57+ypos] = 1
gameState[xpos+4,56+ypos] = 1
gameState[xpos+4,55+ypos] = 1
gameState[xpos+5,57+ypos] = 1
gameState[xpos+6,59+ypos] = 1
gameState[xpos+6,58+ypos] = 1
gameState[xpos+6,57+ypos] = 1
gameState[xpos+6,56+ypos] = 1
gameState[xpos+6,55+ypos] = 1
#Alas
gameState[xpos+5,57+ypos] = 1
gameState[xpos+9,57+ypos] = 1
gameState[xpos+5,58+ypos] = 1
gameState[xpos+9,58+ypos] = 1
gameState[xpos+4,58+ypos] = 1
gameState[xpos+10,58+ypos] = 1
gameState[xpos+4,59+ypos] = 1
gameState[xpos+10,59+ypos] = 1
```