

Facultad de Ingenieria y Ciberseguridad

TEMA: TRABAJO FINAL DE LOGICA DE PROGRAMACION



Alumno:

PÉREZ BONILLA EDYAN STEVEN

Docente:

SALAZAR TAPIA MONICA PATRICIA

Pygame

Es una librería para el desarrollo de videojuegos en segunda dimensión 2D con el lenguaje de programación Python. Pygame está basada en SDL, que es una librería que nos provee acceso de bajo nivel al audio, teclado, ratón y al hardware gráfico de nuestro ordenador.





En el presente proyecto se plasma un lunk para que funciones, es mediante la función whill , colisiones , lista, coordenadas guardadas.



autoeficiencia

edyanperez Añadir archivos mediante carga

Código Culpa 268 lines (215 loc) · 10.4 KB

```
1  importar juego py # Importar la librería pygame
2  importar tiempo # Importar la librería time para pausar
3  importar aleatorio # Importar la librería random para la generación aleatoria de números
4
5  # Inicializar Pygame
6  juego py.Inicial()
7
8  # Definición de colores en RGB
9  BLANCO =(255,255,255) #Blanco
10 NEGRO =(0,0,0) #Negro
11 ROJO =(213,50,80) #Rojo
12 VERDE =(0,255,0) #Verde
13 AZUL =(50,153,213) # Azul
14 GRIS =(169,169,169) #Gris
15 AMARILLO =(255,255,0) #Amarillo para la cabeza de la serpiente
16
17 # Dimensiones de la pantalla
18 ANCHO,ALTURA = 800,600 # Aumentar tamaño para una mejor visualización
19 esto = juego py.mostrar.modo_establecer((ANCHO,ALTURA))
20 juego py.mostrar.establecer_caption('Juego de la Serpiente')
21
```



```
21
22     # Altura de la barra superior
23     ALTURA DE LA BARRA SUPERIOR = 50
24
25     # Reloj para controlar la velocidad del juego.
26     reloj = juego py.tiempo.Reloj()
27
28     # Tamaño y velocidad de la serpiente
29     bloque de serpiente = 15
30     velocidad de la serpiente = 15
31
32     # Fuentes de texto
33     estilo_fuente = juego py.fuente.Fuente del sistema("billete de banco",25) # Fuente más pequeña para mensajes de pérdida
34     puntuación_fuente = juego py.fuente.Fuente del sistema("comicsansms",35) # Fuente de puntuación
35
36     # Variables globales
37     nombre_jugador = "" # Nombre del jugador
38
39     # Función para mostrar la puntuación
40     definición tu_puntuación(puntaje):
41         juego py.dibujar.recto(esto,GRIS, [0,0,ANCHO,ALTURA DE LA BARRA SUPERIOR]) # Color de fondo para el encabezado
42         valor = puntuación_fuente.prestar(f"Jugador:{nombre_jugador} |Puntuación: " + cadena(puntaje),Verdadero,NEGRO) # Texto en negro
43         esto.blister(valor, [10,10])
44
```





```
44
45      # Función para dibujar la serpiente
46  ✓  definición nuestra_serpiente(bloque de serpiente,lista de serpientes):
47      para i,incógnita en enumerar(lista de serpientes):
48          si i == lente(lista de serpientes)- 1: # Si es la cabeza
49              juego py.dibujar.recto(esto,AMARILLO, [incógnita[0],incógnita[1],bloque de serpiente,bloque de serpiente]) # Cabeza en amarillo
50          demás:
51              juego py.dibujar.recto(esto,VERDE, [incógnita[0],incógnita[1],bloque de serpiente,bloque de serpiente]) #Cuerpo en verde
52
53      # Función para mostrar mensajes
54  definición mensaje(mensaje,color,y_desplazar=0):
55      mensaje = estilo_fuente.prestar(mensaje,Verdadero,color) #Crear el mensaje
56      mensaje_rect = mensaje.obtener_rect(centro=(ANCHO / 2,ALTURA / 2 + ALTURA DE LA BARRA SUPERIOR + y_desplazar)) # Centrando el mensaje
57      esto.blitar(mensaje,mensaje_rect) # Colocar el mensaje en la pantalla
58
```





Universitat
Valenciana

```
58     # Función de pantalla de inicio
59
60     ✓ definición introducción del juego():
61         global nombre_jugador
62         Introducción = Verdadero
63         nombre_ingresado = FALSO # Variable para controlar si el nombre fue ingresado
64
65         mientras Introducción:
66             esto.llenar(BLANCO) # Fondo blanco
67
68             si no nombre_ingresado:
69                 # Campo para ingresar el nombre del jugador
70                 nombre_jugador = nombre_de_entrada()
71                 nombre_ingresado = Verdadero # Marcar que el nombre fue ingresado
72
73             # Mensaje de bienvenida
74             dibujar_texto("Bienvenido al Juego de la Serpiente",puntuación_fuente,AZUL,esto,ANCHO // 2,ALTURA // 4)
75             dibujar_texto("Selecciona el modo de juego",estilo_fuente,NEGRO,esto,ANCHO // 2,ALTURA // 3)
76
77             # Botones para seleccionar modo
78             botón("Modo normal",150,400,150,50,VERDE,AZUL,"modo_normal")
79             botón("Modo Desafío",500,400,150,50,VERDE,AZUL,"modo_desafío")
80
81             juego py.mostrar.actualizar() # Actualizar pantalla
82
83             para evento en juego py.evento.conseguir():
84                 si evento.tipo == juego py.ABANDONAR:
85                     juego py.abandonar()
86                     abandonar()
```

```
87
88     # Función para ingresar el nombre del jugador
89     ✓ definición nombre_de_entrada():
90         nombre = ""
91         entrada_activa = Verdadero
92         mientras entrada_activa:
93             esto.llenar(BLANCO)
94             dibujar_texto("Ingresa tu nombre:",estilo_fuente,NEGRO,esto,ANCHO // 2,ALTURA // 2)
95             dibujar_texto(nombre,estilo_fuente,AZUL,esto,ANCHO // 2,ALTURA // 2 + 40)
96             juego py.mostrar.actualizar()
97
98             para evento en juego py.evento.conseguir():
99                 si evento.tipo == juego py.ABANDONAR:
100                     juego py.abandonar()
101                     abandonar()
102                     si evento.tipo == juego py.TECLA_ABAJO:
103                         si evento.llave == juego py.K_RETORNO: #Presionar Enter para confirmar
104                             entrada_activa = FALSO
105                         Elif evento.llave == juego py.K_RETROCESO: # Borrar el último personaje
106                             nombre = nombre[:-1]
107                         demás:
108                             nombre += evento.Unicode # Agregar el carácter a la cadena
109
110             devolver nombre
111
112             # Función principal del juego
113     ✓ definición bucle_de_juego(modo='normal'):
114         juego_terminado = FALSO # Indica si el juego ha terminado
115         juego_cerrado = FALSO # Indica si el jugador ha perdido
```





```
112     # Función principal del juego
113 ~  definición bucle de juego(modo='normal'):
114     juego_terminado = FALSO # Indica si el juego ha terminado
115     juego_cerrado = FALSO # Indica si el jugador ha perdido
116
117     # Variables de posición de la serpiente
118     x1 = ANCHO / 2
119     y1 =(ALTURA / 2)+(ALTURA DE LA BARRA SUPERIOR / 2) # Ajustar posición inicial debajo de la barra
120
121     # Cambios de posición
122     x1_cambio = 0
123     y1_cambio = 0
124
125     Lista de serpientes =[] # Lista para las partes de la serpiente
126     Longitud de la serpiente = 1 # Longitud inicial de la serpiente
127
128     # Generación de la fruta en una posición aleatoria
129     comidax = redondo(aleatorio.rango de rand(0,ANCHO - bloque de serpiente)/ bloque de serpiente)* bloque de serpiente
130     Comestible = redondo(aleatorio.rango de rand(ALTURA DE LA BARRA SUPERIOR,ALTURA - bloque de serpiente)/ bloque de serpiente)* bloque de serpiente
131
132     # Variables de modo de juego
133     desafío_seleccionado = FALSO
134
135     #Desafío diario
136     desafío_activo = FALSO
137     desafío_objetivo = 5 # Número de frutas para completar el desafío
138     frutas_comidas = 0 # Frutas comidas
139     desafío_completado = FALSO # Estado del desafío
140
```



```
141     # Desafíos específicos
142     si modo == 'desafío_1':
143         desafío_objetivo = 10 # Ejemplo: meta del desafío 1
144     Elif modo == 'desafío_aleatorio':
145         desafío_objetivo = aleatorio.Randint(5,15) # Ejemplo: meta aleatoria
146
147     mientras no juego terminado: # Bucle principal del juego
148
149         mientras juego_cerrado: # Cuando el jugador pierde
150             esto.llenar(BLANCO) # Limpiar la pantalla
151             tu_puntuación(Longitud de la serpiente - 1) # Mostrar puntuación final
152             mensaje("¡Has perdido! Presiona C para reiniciar",ROJO,-50)
153             mensaje("Q para salir o M para menú",ROJO,0)
154             juego py.mostrar.actualizar() # Actualizar la pantalla
155
156         para evento en juego py.evento.conseguir():
157             si evento.tipo == juego py.TECLA ABAJO:
158                 si evento.llave == juego py.K_q: # Opción de salida
159                     juego terminado = Verdadero
160                     juego_cerrado = FALSO
161                 si evento.llave == juego py.K_c: # Reiniciar el juego
162                     bucle de juego(modo)
163                 si evento.llave == juego py.K_m: # Volver al menú principal
164                     introducción del juego()
165
166         para evento en juego py.evento.conseguir():
167             si evento.tipo == juego py.ABANDONAR:
168                 juego terminado = Verdadero
169             si evento.tipo == juego py.TECLA ABAJO: # Detectar teclas presionadas
```



FUNCION PRINCIPAL DEL JUEGO

```
166 para evento en juego py.evento.conseguir():
167     si evento.tipo == juego py.ABANDONAR:
168         juego terminado = Verdadero
169     si evento.tipo == juego py.TECLA ABAJO: # Detectar teclas presionadas
170         si evento.llave == juego py.K_IZQUIERDA y x1_cambio == 0: # Movimiento a la izquierda
171             x1_cambio = -bloque de serpiente
172             y1_cambio = 0
173         Elif evento.llave == juego py.K_DERECHO y x1_cambio == 0: # Movimiento a la derecha
174             x1_cambio = bloque de serpiente
175             y1_cambio = 0
176         Elif evento.llave == juego py.K_UP y y1_cambio == 0: # Movimiento hacia arriba
177             y1_cambio = -bloque de serpiente
178             x1_cambio = 0
179         Elif evento.llave == juego py.K_ABAJO y y1_cambio == 0: # Movimiento hacia abajo
180             y1_cambio = bloque de serpiente
181             x1_cambio = 0
182
183     # Verificación de colisiones con los bordes
184     si x1 >= ANCHO o x1 < 0 o y1 >= ALTURA o y1 < ALTURA DE LA BARRA SUPERIOR:
185         juego cerrado = Verdadero
186
187     # Actualizar la posición de la serpiente
188     x1 += x1_cambio
189     y1 += y1_cambio
190     esto.llenar(BLANCO) # Limpiar la pantalla
191
192     # Dibujar el borde del área de juego
193     juego py.dibujar.recto(esto,NEGRO, [0,ALTURA DE LA BARRA SUPERIOR,ANCHO,ALTURA - ALTURA DE LA BARRA SUPERIOR],2)
194
```

Es donde se puede realizar funciones como primer lugar si el juego a terminado o no , la posicionamiento de la serpiente , anchos mediante de coordenadas , lista , generación de la frutas de forma aleatoria



CICLOS DE CONDICIONES

Para verificar si colisiona o no la serpiente

```
195     # Dibujar la fruta
196     juego py.dibujar.recto(esto,ROJO, [comidax,Comestible,bloque de serpiente,bloque de serpiente])
197
198     # Actualizar la lista de la serpiente
199     cabeza de serpiente = []
200     cabeza de serpiente.append(x1)
201     cabeza de serpiente.append(y1)
202     Lista de serpientes.append(cabeza de serpiente)
203
204     # Mantener la longitud de la serpiente
205     si lente(Lista de serpientes) > Longitud de la serpiente:
206         del Lista de serpientes[0]
207
208     # Verificar colisión con el cuerpo de la serpiente
209     para incógnita en Lista de serpientes[:-1]:
210         si incógnita == cabeza de serpiente:
211             juego_cerrado = Verdadero
212
213     # Dibujar la serpiente
214     nuestra_serpiente(bloque de serpiente,Lista de serpientes)
215
216     # Mostrar la puntuación
217     tu_puntuación(Longitud de la serpiente - 1)
218
219     juego py.mostrar.actualizar() # Actualizar la pantalla
220
```



```
221     # Verificar si la serpiente ha comido la fruta
222     si abdominales(x1 - comidas)< bloque de serpiente y abdominales(y1 - Comestible)< bloque de serpiente:
223         comidas = redondo(aleatorio.rango de rand(0,ANCHO - bloque de serpiente)/ bloque de serpiente)* bloque de serpiente
224         Comestible = redondo(aleatorio.rango de rand(ALTURA DE LA BARRA SUPERIOR,ALTURA - bloque de serpiente)/ bloque de serpiente)* bloque de serpiente
225         longitud de la serpiente += 1 #Aumentar la longitud de la serpiente
226         frutas comidas += 1 # Aumentar el contador de frutas comidas
227
228     # Verificar si se ha completado un desafío
229     si frutas comidas >= desafio_objetivo:
230         desafio_completado = Verdadero # Marcar el desafío como completado
231
232     # Terminar el juego si el desafío está completo y la serpiente ha chocado
233     si desafio_completado y juego_cerrado:
234         juego_cerrado = Verdadero
235         juego terminado = Verdadero
236
237         reloj.garrapata(velocidad de la serpiente) # Controlar la velocidad del juego
238
239         juego py.abandonar() # Salir de pygame
240         abandonar() # Salir del script
241
242     # Función para dibujar texto centrado
243     definición dibujar_texto(texto,fuente,color,superficie,incógnita,y):
244         textobj = fuente.prestar(texto,Verdadero,color)
245         texto recto = textobj.obtener_rect(centro=(incógnita,y))
246         superficie.blit(textobj, texto recto)
247
```



```
239 juego py.abandonar() # Salir de pygame
240 abandonar() # Salir del script
241
242 # Función para dibujar texto centrado
243 definición dibujar_texto(texto,fuente,color,superficie,incógnita,y):
244     textobj = fuente.prestar(texto,Verdadero,color)
245     texto recto = textobj.obtener_rect(centro=(incógnita,y))
246     superficie.blíster(textobj, texto recto)
247
248 # Función para los botones del menú
249 ✓ definición botón(mensaje,incógnita,y,el,yo,yo,C.A,acción=Ninguno):
250     ratón = juego py.ratón.obtener_pos()
251     hacer clic = juego py.ratón.Obtener_presionado()
252
253     si incógnita + el > ratón[0]> incógnita y y + yo > ratón[1]> y:
254         juego py.dibujar.recto(estoyo, (incógnita,y,el,yo))
255         si hacer clic[0]== 1 y acción != Ninguno:
256             si acción == "modo_normal":
257                 bucle de juego('normal')
258             Elif acción == "modo_desafío":
259                 bucle de juego('desafío')
260         demás:
261             juego py.dibujar.recto(estoyo, (incógnita,y,el,yo))
262
263         textoSurf = estilo_fuente.prestar(mensaje,Verdadero,NEGRO)
264         textoRect = textoSurf.obtener_rect(centro=((incógnita +(el / 2)), (y +(yo / 2))))
265         esto.blíster(textoSuf, textoRect)
266
```

CONCLUSIÓN



Se puede llegar a la conclusión que para la utilización y la implementación de variables se tiene que previsualizar el producto que se va a entregar como consideraciones se debe tomar las coordenadas , variables a utilizar , colisiones y funcionalidad de cada objeto como a su vez la estructura lógica que se va a plasmar en nuestro trabajo es por ello que es necesariamente importante realizar un consolidado y un listado de que realmente necesitamos para que nuestro código sea lo mas simplificado y funcional posible

GRACIAS POR SU ATENCIÓN