UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ALGORITMOS Y ESTRUCTURAS DE DATOS 2- BMA20-N

TAREA 1:

APELLIDOS Y NOMBRES:

Flores Negreiros Margaly 20170426C

DOCENTE:

Ing. Uwe Rojas Villanueva

CICLO 2021-II

Lima - Perú

2021

JUEGO: DINO GAME

Dino Game es un juego que consiste en evitar los obstáculos usando las teclas “Flecha hacia arriba” (para saltar) y “Flecha hacia abajo” (para agacharse).

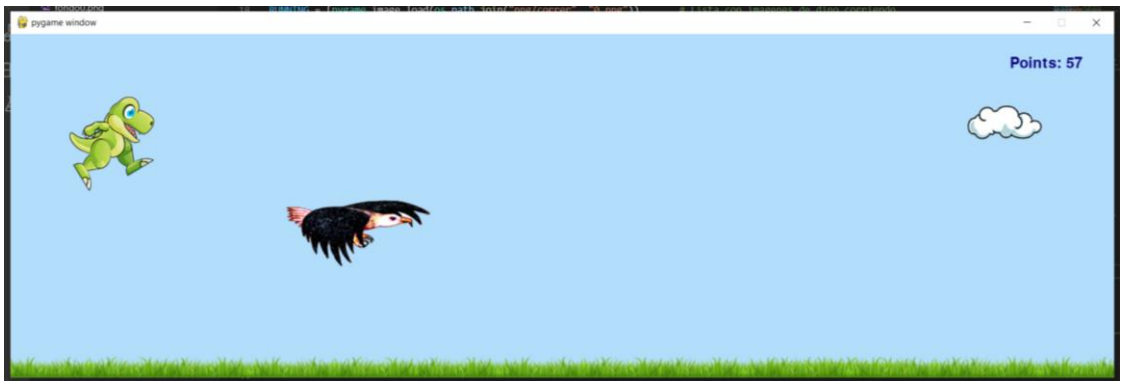
Cuenta con sonido para el salto, aumento de puntuación y cuando muere.

Inicio:

Presionar cualquier tecla para iniciar

Escenario:

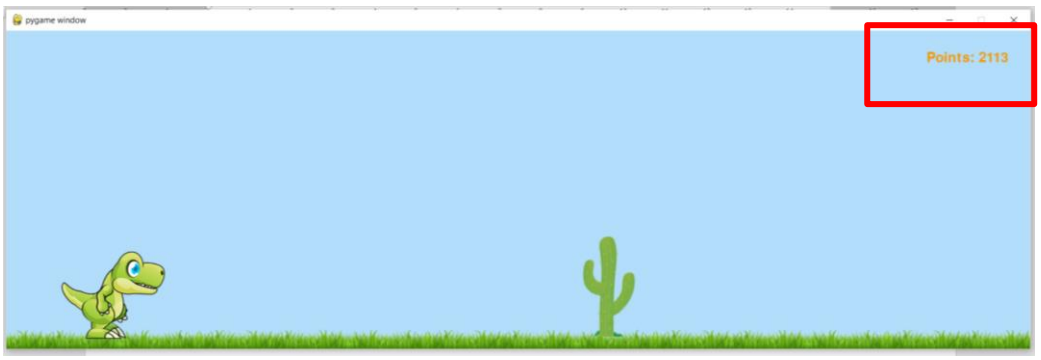
Salto: Tecla flecha hacia arriba



Agachado: Tecla flecha hacia abajo

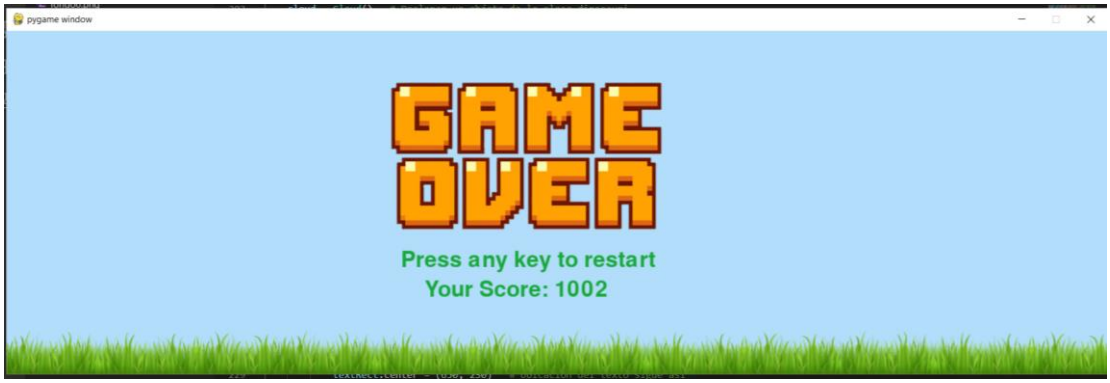
Cada 200 puntos muestra un mensaje: ¡Sigue Así! Y se escucha el sonido de aumento de puntaje.

Cuando el puntaje es mayor a 1000: Cambia de día a noche, y así sucesivamente cada 1000 puntos.



El juego termina cuando chocha con algún obstáculo, sea cactus o ave.

Perder el juego: Muestra mensaje para reiniciar y puntaje obtenido.



CODIGO

play.py > main > score

```python
#Importamos las librerias a usar

import os
import random
import threading

import pygame

pygame.init()  # Inicializar los modulos de pygame

# Declaramos constantes globales

SCREEN_HEIGHT = 464  # medida alto
SCREEN_WIDTH = 1492 # medida ancho
SCREEN = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))  # crear la pantalla


RUNNING = [pygame.image.load(os.path.join("png/correr", "0.png")),      # Lista con imagenes de dino corriendo
           pygame.image.load(os.path.join("png/correr", "1.png")),
           pygame.image.load(os.path.join("png/correr", "2.png")),
           pygame.image.load(os.path.join("png/correr", "3.png")),
           pygame.image.load(os.path.join("png/correr", "4.png"))]

JUMPING = pygame.image.load(os.path.join("png/saltar", "6.png"))      # Imagen dino saltando


DUCKING = [pygame.image.load(os.path.join("png/bajo","0.png")),      # Lista con imagenes dino agachado
           pygame.image.load(os.path.join("png/bajo","1.png"))]

SMALL_CACTUS = [pygame.image.load(os.path.join("png/cactus","4.png")),  # Lista con imagenes de cactus pequeños
                pygame.image.load(os.path.join("png/cactus","5.png")),
                pygame.image.load(os.path.join("png/cactus","6.png"))]

```

play.py > ...

```python
LARGE_CACTUS = [pygame.image.load(os.path.join("png/cactus","1.png")),    # Lista con imagenes de cactus grandes
                pygame.image.load(os.path.join("png/cactus","2.png")),
                pygame.image.load(os.path.join("png/cactus","3.png"))]

BIRD = [pygame.image.load(os.path.join("png/ave","0.png")),    # Lista con imagenes de aves
        pygame.image.load(os.path.join("png/ave","1.png"))]

CLOUD = pygame.image.load(os.path.join("png/nubes","0.png"))   # Imagen de nube

BG = pygame.image.load(os.path.join("png/cesped", "Track.png"))   # Imagen de pasto

fondo = pygame.image.load("fondoInicio.png")   # Cargar imagen de fondo mostrado al inicio
press_key = pygame.image.load("press.png")     # Cargar imagen de letras "Press Start"
game_over = pygame.image.load("game_over.png")  # Cargar imagen de letras Game Over
fondo_fin = pygame.image.load("fondo0.png")     # Cargar imagen de fondo final

jump_sound = pygame.mixer.Sound('sonido/jump.wav')   # Importar sonido de saltos
die_sound = pygame.mixer.Sound('sonido/die.wav')     # Importar sonido de muerte
checkPoint_sound = pygame.mixer.Sound('sonido/checkPoint.wav')  # Importar sonido de puntaje cada 200 puntos

class Dinosaur:  # Clase dinosaurio

    X_POS = 80    # Posicion X inicial del Dinosaurio
    Y_POS = 322     # Posicion y inical del Dinosaurio
    Y_POS_DUCK = 390 # Posición y inicial de imagen agachado
    JUMP_VEL = 10    # Velocidad de salto
    sonido = True
    # max_jump = True

    def __init__(self):   # Inicializar la clase
        self.duck_img = DUCKING  # Referencias
        self.run_img = RUNNING
        self.jump_img = JUMPING

        self.dino_duck = False   # Valores iniciales
        self.dino_run = True
        self.dino_jump = False
```

play.py > Dinosaur > __init__

```python
        self.step_index = 0   # Inicializar
        self.jump_vel = self.JUMP_VEL  # Referenciar velocidad de salto
        self.image = self.run_img[0]   # Referenciar primera imagen de la lista corriendo
        self.dino_rect = self.image.get_rect()  # Posicionar la imagen
        self.dino_rect.x = self.X_POS   # Coodenada x de la posicion
        self.dino_rect.y = self.Y_POS   #Coodenada y de la posicion

    def update(self, userInput):
        if self.dino_duck:   # Llamado de funciones
            self.duck()
        if self.dino_run:
            self.run()
        if self.dino_jump : #and self.max_jump
            self.jump()
            if self.sonido == True:
                jump_sound.play()   # Sonido de salto
                self.sonido = False

        if self.step_index >= 10:  # Indice de paso de las imagenes
            self.step_index = 0

        if userInput[pygame.K_UP] and not self.dino_jump:  # Condicion si se presiona tecla flecha arriba
            self.dino_duck = False
            self.dino_run = False
            self.dino_jump = True
        elif userInput[pygame.K_DOWN] and not self.dino_jump:  # Condicion si se presiona techa flecha abajo
            self.dino_duck = True
            self.dino_run = False
            self.dino_jump = False
        elif not (self.dino_jump or userInput[pygame.K_DOWN]): #Condicion si no se presiona alguna de las teclas
            self.dino_duck = False
            self.dino_run = True
            self.dino_jump = False

    def duck(self):            # Funcion agachado
        self.image = self.duck_img[self.step_index // 5]  # Para repetir las imagenes cada 2 veces
        self.dino_rect = self.image.get_rect()       # Posicionar la imagen
```

play.py > Dinosaur > duck

```
109         self.dino_rect.x = self.X_POS
110         self.dino_rect.y = self.Y_POS_DUCK
111         self.step_index += 1
112
113     def run(self):     #Funcion correr
114         self.sonido = True     # auxiliar
115         self.image = self.run_img[self.step_index // 2]  # Repite imagenes cada dos veces
116         self.dino_rect = self.image.get_rect()   # Metodo para ingresar ubicacion
117         self.dino_rect.x = self.X_POS  # Posicion en x del dino
118         self.dino_rect.y = self.Y_POS  # Posicion en y del dino
119         self.step_index += 1
120
121     def jump(self):    #Funcion Salto
122         # if self.jump_vel == 0.4:
123         #     self.max_jump = False
124         self.image = self.jump_img  # Referenciar imagen salto
125         if self.dino_jump:
126             self.dino_rect.y -= self.jump_vel * 4  # Varia la posición en y
127             self.jump_vel -= 0.8     # Disminuye la velocidad
128         if self.jump_vel < -self.JUMP_VEL:
129             # self.max_jump = True
130             self.dino_jump = False
131             self.jump_vel = self.JUMP_VEL
132
133     def draw(self, SCREEN):   # Poner imagen en pantalla
134         SCREEN.blit(self.image, (self.dino_rect.x, self.dino_rect.y))
135
136
137 class Cloud:    # Clase nube
138     def __init__(self):
139         self.x = SCREEN_WIDTH + random.randint(600, 1000)  # Posicion aleatoria en x mayor al ancho
140         self.y = random.randint(50, 100)     # Ubicar la posicion en y aleatoria
141         self.image = CLOUD          # Llamar imagen Nube
142         self.width = self.image.get_width()
143
144     def update(self):
145         self.x -= game_speed   # desplazamiento de la nube
```

play.py > Cloud > update

```
146         if self.x < -self.width:  # Cuando la nube termina de mostrarse
147             self.x = SCREEN_WIDTH + random.randint(2500, 3000)   #posicion en x
148             self.y = random.randint(50, 100)   # Posicion aleatoria en y
149
150     def draw(self, SCREEN):   # Mostrar imagen en pantalla
151         SCREEN.blit(self.image, (self.x, self.y))
152
153
154 class Obstacle:    # Clase obstaculo
155     def __init__(self, image, type):  # Inicializamos la clase
156         self.image = image  # Referenciamos
157         self.type = type
158         self.rect = self.image[self.type].get_rect()  ### Tipo de obstaculo
159         self.rect.x = SCREEN_WIDTH
160
161     def update(self):
162         self.rect.x -= game_speed   # para el desplazamiento
163         if self.rect.x < -self.rect.width:
164             obstacles.pop()   # Elimini al último elemento
165
166     def draw(self, SCREEN):  # Mostrar imagen en pantalla
167         SCREEN.blit(self.image[self.type], self.rect)
168
169
170 class SmallCactus(Obstacle):   # Clase obstaculos pequeños
171     def __init__(self, image):
172         self.type = random.randint(0, 2)   # Generar un tipo de cactus
173         super().__init__(image, self.type)
174         self.rect.y = 325
175
176
177 class LargeCactus(Obstacle):   # Clase obstaculo cactus
178     def __init__(self, image):
179         self.type = random.randint(0, 2)
180         super().__init__(image, self.type)
181         self.rect.y = 300
182
```

play.py > ...

```python

class Bird(Obstacle):   # Clase ave
    def __init__(self, image):
        self.type = 0
        super().__init__(image, self.type)
        self.rect.y = random.randint(10,240)   # Posicion de y aleatoria
        self.index = 0

    def draw(self, SCREEN):
        if self.index >= 9:
            self.index = 0
        SCREEN.blit(self.image[self.index // 5], self.rect)  # para cambiar entre las imagenes de aves
        self.index += 1


def main():
    global game_speed, x_pos_bg, y_pos_bg, points, obstacles  # declarar variables globales
    run = True    # Auxiliar del while
    clock = pygame.time.Clock()   # Clock
    player = Dinosaur()   # Declarar un objeto de la clase dinosaurio
    cloud = Cloud()   # Declarar un objeto de la slase dinosauri
    game_speed = 20   #Velocidad del juego
    x_pos_bg = 0   # Posición del cesped en x
    y_pos_bg = 436 # Posición del cesped en y
    points = 0   # Puntaje inicia en 0
    font = pygame.font.Font("freesansbold.ttf", 20)  ## Tipo de letra
    obstacles = []   # obstaculos
    death_count = 0 # Conteo de muertes
    dia = True

    def score():   # Definimos la funcion de puntaje
        global points, game_speed   # Declarar variables locales
        points += 1  # Aumentar el puntaje
        if points % 100 == 0:
            game_speed += 1   # La velocidad aumenta cada 100 puntos

        text = font.render("Points: " + str(points), True, (246, 151,1))   # Imprimir el puntaje
```

play.py > main > score

```python
        textRect = text.get_rect()  # Metodo para ingresar ubicacion
        textRect.center = (1400, 40)   # Posicion del puntaje
        SCREEN.blit(text, textRect)   # Mostrar en pantalla

        if  0 < points % 200 < 100  and points > 200:   # Muestra mensaje por cada 200 puntos conseguidos
            text = pygame.font.Font("freesansbold.ttf", 30).render("¡Sigue así!", True, (246, 151,1))
            if points % 200 == 1 :
                checkPoint_sound.play()    # Sonido de aumentar puntaje en 200 puntos
            textRect = text.get_rect()
            textRect.center = (650, 230)   # Ubicacion del texto sigue asi
            SCREEN.blit(text, textRect)


    def background():   # Funcion que genera ilusion de movimiento del cesped
        global x_pos_bg, y_pos_bg    # Declaramos variables para la posicion incial del cesped
        image_width = BG.get_width() # Guarda el valor del ancho del cesped
        SCREEN.blit(BG, (x_pos_bg, y_pos_bg))   # Muestra el cesped en pantalla
        SCREEN.blit(BG, (image_width + x_pos_bg, y_pos_bg))  # Muestra cesped fuera de pantalla visual
        if x_pos_bg <= -image_width:   # Genera ilusion de movimiento
            SCREEN.blit(BG, (image_width + x_pos_bg, y_pos_bg))
            x_pos_bg = 0
        x_pos_bg -= game_speed

    while run:
        for event in pygame.event.get():   # Evento para salir
            if event.type == pygame.QUIT:
                run = False

        # Cambia de día a noche cuando supera el puntaje de 1000 y asi sucesivamente
        if dia:
            SCREEN.fill((178, 221, 251))   #fondo celeste
            if (points % 1000 == 0 and points > 999):
                dia = not dia
        else:
            SCREEN.fill((21, 19, 99))   # fondo oscuro
            if (points % 1000 == 0 and points > 999):
                dia = not dia
```

play.py > main

```
        userInput = pygame.key.get_pressed()  # Modulo que dectecta si se presiono una tecla

        background()   # Funcion del movimiento del cesped
        player.draw(SCREEN)  # Mostrar el pantalla al dinosaurio
        player.update(userInput) # Llamada a la funcion update de la clase dinosaurio o player


        if len(obstacles) == 0:
            rand = random.randint(0,2)   # Genera numeros aleatorios para mostrar obstaculos (cactus/ave) de forma aleatoria
            if rand == 0:                # Compara el valor obtenido
                obstacles.append(SmallCactus(SMALL_CACTUS))  # Añade cactus pequeños
            elif rand == 1:
                obstacles.append(LargeCactus(LARGE_CACTUS)) # Añade cactus grandes
            elif rand == 2:
                obstacles.append(Bird(BIRD))  # Añade aves

        for obstacle in obstacles:
            obstacle.draw(SCREEN)  # Muestra obstaculos en pantalla
            obstacle.update()
            if player.dino_rect.colliderect(obstacle.rect):   # Para detectar si el dino choca con un obstaculo
                die_sound.play()     # Para reproducir sonido de muerte
                player.draw(SCREEN)
                pygame.time.delay(2000)
                death_count += 1
                menu(death_count) # Llamar a la funcion menu/ (contabiliza la muerte )


        cloud.draw(SCREEN) # Muestra nubes en pantalla
        cloud.update()
        score()

        clock.tick(30)
        pygame.display.update()


def menu(death_count):
    global points   # declarar variables globales
```

play.py > main

```
    global points   # declarar variables globales
    cont = 0
    tiempo = True
    run = True


    while run:

        SCREEN.fill((178, 221, 251))       ###  Fondo celeste
        font = pygame.font.Font("freesansbold.ttf", 30)   # Tipo de letra

        if death_count == 0:
            SCREEN.blit(fondo,(0,0))   # Ubicamos el fondo incial
            if tiempo:
                pygame.time.delay(1000)
                tiempo = False

            # Genera ilusion de parpadeo en la imagen "press start"
            if 0 <= cont < 15 :     # primeros 15 sg
                SCREEN.blit(press_key,(600,320))  # Muestra imagen " Press Start "
            if 15<= cont < 30:      # Siguientes 15 sg
                SCREEN.blit(fondo,(0,0))  # Muestra fondo
            cont += 1
            if cont >= 30:
                cont = 0

        elif death_count > 0:    # Cuando muere
            SCREEN.blit(fondo_fin,(0,0))      # fondo final
            SCREEN.blit(game_over,(520,70))   # Imagen over again
            text = font.render("Press any key to restart ", True, (24, 168, 58))  # Escribimos las letras en color verde
            score = font.render("Your Score: " + str(points), True, (24, 168, 58)) # Escribimos el puntaje en color verde
            scoreRect = score.get_rect()  # Metodo para poder establecer coodenadas
            scoreRect.center = (690, 350)  # ubica el puntaje en dicha posicion
            SCREEN.blit(score, scoreRect)  # Imprime en pantalla
            textRect = text.get_rect()    # Metodo para poder establecer coodenadas
            textRect.center = (710, 310)  # Ubica el texto en dicha posicion
            SCREEN.blit(text, textRect)  # Imprime en pantalla
```

```
        pygame.display.update()
        for event in pygame.event.get():     # evento para cerrar pantalla
            if event.type == pygame.QUIT:
                run = False
                pygame.display.quit()
                pygame.quit()
                exit()
            if event.type == pygame.KEYDOWN:
                main()

# Para poder ejecutar tareas "simultaneas" (menu/main)
T = threading.Thread(target=menu(death_count=0), daemon=True)  #(Genera una ilusion de simultaneidad)
T.start()   # Ejecutar threading
```