Tecnológico de Monterrey

Designing a programming language

Herramientas computacionales: el arte de la programación

(Gpo 201)

David Míreles Gutiérrez A00836010

Rodrigo Garza A01383556

Marcelo Marcos Flores A01722788

Emilio Vidal Cavazos Páez A00835995

Profesora:

Dra. Verónica Rodríguez Rodríguez

Mayo 10, 2024

SIMON DICE

Historial de versiones en la herramienta colaborativa:

https://replit.com/join/kjhrabzxgp-a00835995

https://web.whatsapp.com (herramienta colaborativa)

Descripción detallada del proyecto:

- **Planeación del proyecto:**

En la fase inicial del proyecto, exploramos diversas temáticas y juegos educativos apropiados para niños de entre 6 y 11 años y solucionar nuestro reto. Después de considerar e investigar varias opciones, decidimos desarrollar un juego centrado en la memoria y agilidad mental. Estábamos decidiendo si implementamos un juego de matemáticas donde si acertarás, un personaje o “sprite” podía animarse y saltar hacia una plataforma en forma ascendente. Nuestra idea evolucionó desde un juego de matemáticas hacia un juego de memorización de colores, inspirado en el clásico juego de “Simon Dice”. En este juego, los jugadores deben recordar y reproducir secuencias que se vuelven más desafiantes a medida que avanzan.

- **Desarrollo del juego:**

La etapa de desarrollo se centró en la implementación de la lógica del juego y la creación de la interfaz del usuario, que se trata de la facilidad de manejar el juego, y la facilidad de navegación y lo intuitivo que es. Utilizamos la librería Pygame, sys, time, y random para desarrollar el juego, explorando además herramientas de inteligencia artificial para mejorar la experiencia del usuario. Comenzamos con una versión básica que presentaba los colores rojo, amarillo, verde y azul en una pantalla negra, desafiando a los jugadores a recordar y repetir secuencias de colores cada vez más largas a medida que progresaban. Solo habíamos implementado la lógica, pero faltaba la interfaz visual que necesitaba nuestro público objetivo.

- **Diseño:**

Finalmente, para hacer el juego más atractivo y accesible para nuestro público objetivo, nos enfocamos en el diseño de personajes y estética visual. Introdujimos varios personajes reconocidos por colores específicos inspirados en figuras de entretenimiento infantil, cada uno asociado con un color específico del juego. Además, actualizamos la interfaz de usuario con colores vibrantes y elementos visuales atractivos, agregamos música, y efectos de sonido para los botones y creamos una experiencia más atractiva y amigable para los niños. Este enfoque en el diseño no solo lo hizo más atractivo para los niños, sino que facilitó su

jugabilidad y experiencia de recompensa. Los colores, personajes, y sonidos, son una manera para ayudar a los niños a asociar y memorizar la secuencia de colores.

- **Código Comentado:**

```python
"""
Evidencia de proyecto
David Míreles Gutiérrez A00836010
Rodrigo Garza A01383556
Marcelo Marcos Flores A01722788
Emilio Vidal Cavazos Páez A00835995
"""
import pygame #Cargamos las librerias necesarias
import sys
import time
import random
from pygame.locals import *

# Configuración de la pantalla
windowWidth = 1280
windowHeight = 720
fps = 30
buttonSize = 150  # Tamaño del botón
gap = 10  # Espacio entre botones

# Tiempo del flash
flashDelay = 100
timeout = 5

# Configuración de los colores (los colores 'bright' se activan cuando se
pulsa o se debe pulsar el color)
white = (255, 255, 255)
black = (0, 0, 0)
red = (255, 0, 0)
brightRed = (200, 0, 0)
green = (0, 155, 0)
brightGreen = (0, 255, 0)
blue = (0, 100, 255)
brightBlue = (0, 0, 155)
yellow = (224, 224, 0)
brightYellow = (255, 255, 0)

# Márgenes por defecto
xMargin = (windowWidth - 2 * buttonSize - gap) // 2
yMargin = (windowHeight - 2 * buttonSize - gap) // 2

# Carga de imágenes de estrellas
```

```
star_image = pygame.image.load('star.png')
star_image2 = pygame.image.load('star2.png')
star_image3 = pygame.image.load('star3.png')
star_image4 = pygame.image.load('star4.png')
star_image5 = pygame.image.load('star5.png')

# Inicialización de Pygame
pygame.init()
screen = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption('Simon Says')
startScreenImage = pygame.image.load('HOMESCREAN.png')  # Imagen de inicio
del juego
startScreenRect = startScreenImage.get_rect()
font = pygame.font.Font("freesansbold.ttf", 20)
displaySurf = pygame.display.set_mode((windowWidth, windowHeight))  #
Pantalla del juego
background_image = pygame.image.load('BACKNORMAL.png')  # Fondo de pantalla
normal


# Función para obtener el puntaje máximo guardado
def getMaxScore():
    try:
        with open("max_score.txt", "r") as file:
            maxScore = int(file.read())
    except FileNotFoundError:
        maxScore = 0
    return maxScore


# Función para actualizar el puntaje máximo
def updateMaxScore(score):
    maxScore = getMaxScore()
    if score > maxScore:
        with open("max_score.txt", "w") as file:
            file.write(str(score))


# Pantalla de inicio del juego
#Incluye un mensaje para presionar "Enter" Espera hasta que este se presione
para salir del bucle y continuar el juego
#Maneja los eventos del teclado para detecta cuando se presiona la tecla
Enter ('K_RETUR') y detiene la musica de inicio
('pygame.mixer.music.stop()')
def startScreen():
    maxScore = getMaxScore()

    # Música de inicio
```

```
    pygame.mixer.music.load("menu.mp3")
    # Reproducción en bucle de la música
    pygame.mixer.music.play(loops=-1)

    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == KEYDOWN:
                if event.key == K_RETURN:
                    # Detiene la música al presionar la tecla Enter
                    pygame.mixer.music.stop()
                    return

        # Dibuja la imagen de inicio
        screen.blit(startScreenImage, startScreenRect)

        # Muestra el texto que indica que se presione Enter para empezar
        textSurface = font.render('Presione Enter para empezar', True,
black)
        textRect = textSurface.get_rect(center=(windowWidth // 2,
windowHeight - 50))
        screen.blit(textSurface, textRect)

        # Muestra el puntaje máximo obtenido anteriormente
        maxScoreText = font.render('Max Score: ' + str(maxScore), True,
black)
        maxScoreRect = maxScoreText.get_rect(topright=(windowWidth - 20,
20))
        screen.blit(maxScoreText, maxScoreRect)

        # Despliega estrellas según el puntaje máximo
        if maxScore >= 5 and maxScore < 10:
            displaySurf.blit(star_image, (800, 40))
        elif maxScore >= 10 and maxScore < 15:
            displaySurf.blit(star_image, (800, 40))
            displaySurf.blit(star_image2, (850, 40))
        elif maxScore >= 15 and maxScore < 20:
            displaySurf.blit(star_image, (800, 40))
            displaySurf.blit(star_image2, (850, 40))
            displaySurf.blit(star_image3, (900, 40))
        elif maxScore >= 20 and maxScore < 25:
            displaySurf.blit(star_image, (800, 40))
            displaySurf.blit(star_image2, (850, 40))
            displaySurf.blit(star_image3, (900, 40))
            displaySurf.blit(star_image4, (950, 40))
        elif maxScore >= 25:
```

```
            displaySurf.blit(star_image, (800, 40))
            displaySurf.blit(star_image2, (850, 40))
            displaySurf.blit(star_image3, (900, 40))
            displaySurf.blit(star_image4, (950, 40))
            displaySurf.blit(star_image5, (1000, 40))

        pygame.display.flip()

        # Limpia la pantalla para evitar mostrar elementos no deseados
        displaySurf.fill(black)


# Dibuja los botones
def drawButtons():
    pygame.draw.rect(displaySurf, red, redRect)
    pygame.draw.rect(displaySurf, blue, blueRect)
    pygame.draw.rect(displaySurf, green, greenRect)
    pygame.draw.rect(displaySurf, yellow, yellowRect)


# Obtiene el botón clickeado
def getClickedButton(left, top):
    if redRect.collidepoint(left, top):
        return red
    if blueRect.collidepoint(left, top):
        return blue
    if greenRect.collidepoint(left, top):
        return green
    if yellowRect.collidepoint(left, top):
        return yellow


# Función para mostrar el color o la animación del patrón
#Realizza la animacion de destello cuando un boton se ilumina
#Superficie 'flashSurf' para crear un efecto de destello
#Dos bucles 'for' para modificar el valor principal del color y crear el
efecto
#finalmente restaura el color original
def flashButtonAnimation(color, animationSpeed=50):
    if color == red:
        rect = redRect
        flashColor = brightRed
        background_image = pygame.image.load('BACKRED.png')  # Cambia el
fondo igual que el botón
        displaySurf.blit(background_image, (0, 0))
        sound = beep1
    if color == green:
        rect = greenRect
```

```
        flashColor = brightGreen
        background_image = pygame.image.load('BACKGREEN.png')
        displaySurf.blit(background_image, (0, 0))
        sound = beep2
    if color == blue:
        rect = blueRect
        flashColor = brightBlue
        background_image = pygame.image.load('BACKBLUE.png')
        displaySurf.blit(background_image, (0, 0))
        sound = beep3
    if color == yellow:
        rect = yellowRect
        flashColor = brightYellow
        background_image = pygame.image.load('BACKYELLOW.png')
        displaySurf.blit(background_image, (0, 0))
        sound = beep4

    sound.play()
    originalSurf = displaySurf.copy()
    flashSurf = pygame.Surface((buttonSize, buttonSize))
    flashSurf = flashSurf.convert_alpha()
    r, g, b = flashColor

    for start, end, step in ((0, 255, 1), (255, 0, -1)):
        for alpha in range(start, end, animationSpeed * step):
            flashSurf.fill((r, g, b, alpha))
            displaySurf.blit(flashSurf, rect.topleft)
            pygame.display.update()
            clock.tick(fps)

    displaySurf.blit(originalSurf, (0, 0))


# Animación de final de juego
def gameOverAnimation():
    color = white
    animationSpeed = 50
    originalSurf = displaySurf.copy()
    flashSurf = pygame.Surface((windowWidth, windowHeight))
    flashSurf = flashSurf.convert_alpha()
    r, g, b = color

    wrong.play()

    for i in range(3):
        for start, end, step in ((0, 255, 1), (255, 0, -1)):
            for alpha in range(start, end, animationSpeed * step):
                flashSurf.fill((r, g, b, alpha))
```

```
                displaySurf.blit(originalSurf, (0, 0))
                displaySurf.blit(flashSurf, (0, 0))
                drawButtons()
                pygame.display.update()
                clock.tick(fps)

#se aumenta ligeramente el tiempo para hacer el patron
def aumentarTime():
    global timeout
    timeout += 0.25
    return


#El nucleo del juego. Controla el bucle principal del juego donde se
actualiza la pantalla, se maneja los eventos y se ejecuta la logia del juego
#Utiliza un bucle "while True" para mantener el juego en funcionamiento
hasta que se cierre el juego donde ademas entra al StarScreen
#Actualiza la pantalla, muestra el puntaje maximo, maneja eventos de cierre
de ventana y esperar al jugador a que ingrese una entrada
#Logica principal del juego como la generacion del patron de colores , la
validacion de la entrada del jugador y la actualizacion del puntaje
def main():
    startScreen()
    global displaySurf, redRect, blueRect, greenRect, yellowRect, clock,
beep1, beep2, beep3, beep4, wrong
    displaySurf = pygame.display.set_mode((windowWidth, windowHeight))
    pygame.display.set_caption("Simon Says")
    clock = pygame.time.Clock()

    # Establece los sonidos de cada color y cuando se pierde
    beep1 = pygame.mixer.Sound("blue.mp3")
    beep2 = pygame.mixer.Sound("green.mp3")
    beep3 = pygame.mixer.Sound("red.mp3")
    beep4 = pygame.mixer.Sound("yellow.mp3")
    wrong = pygame.mixer.Sound("wrong.mp3")

    # Posiciones de los cuadrados
    redRect = pygame.Rect((xMargin - 250) - gap, (yMargin - 48) - gap,
buttonSize, buttonSize)
    yellowRect = pygame.Rect((xMargin + 250) + buttonSize + gap, (yMargin -
48) - gap, buttonSize, buttonSize)
    greenRect = pygame.Rect((xMargin - 250) - gap, (yMargin + 150) +
buttonSize + gap, buttonSize, buttonSize)
    blueRect = pygame.Rect((xMargin + 250) + buttonSize + gap, (yMargin +
150) + buttonSize + gap, buttonSize, buttonSize)
    font = pygame.font.Font("freesansbold.ttf", 20)
    pattern = []  # Guarda el patrón de colores
    currentStep = 0  # Variable para seguir el paso actual
```

```
    waitingForInput = False
    score = 0
    lastTimeClick = 0

    while True:
        scoreSurf = font.render("Score: " + str(score), False, white)
        scoreRect = scoreSurf.get_rect()
        scoreRect.topleft = (windowWidth - 100, 45)
        clickedButton = None
        displaySurf.blit(background_image, (0, 0))  # Fondo del juego
        displaySurf.blit(scoreSurf, scoreRect)
        drawButtons()

        # Obtiene el puntaje máximo y lo muestra en pantalla
        maxScore = getMaxScore()
        maxScoreText = font.render('Max Score: ' + str(maxScore), True,
white)
        maxScoreRect = maxScoreText.get_rect(topright=(windowWidth - 20,
20))
        displaySurf.blit(maxScoreText, maxScoreRect)

        # Revisa si se cierra el juego
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()
            if event.type == MOUSEBUTTONUP:
                clickedButton = getClickedButton(event.pos[0], event.pos[1])

        # Espera un input del jugador
        if not waitingForInput:
            pygame.display.update()
            pygame.time.wait(2000)
            pattern.append(random.choice((blue, red, yellow, green)))  #
Elige aleatoriamente el siguiente color del patrón
            for button in pattern:  # Muestra el patrón
                flashButtonAnimation(button)
                pygame.time.wait(flashDelay)
            waitingForInput = True

        else:  # Comprueba si el patrón coincide con la entrada del jugador
            if clickedButton and pattern[currentStep] == clickedButton:
                flashButtonAnimation(clickedButton)
                lastTimeClick = time.time()
                currentStep += 1
                if currentStep == len(pattern):
                    currentStep = 0
                    score += 1
```

```
                    aumentarTime()#funcion para que mientras que son mas
patrones tambien aumente el tiempo para hacerlo
                    updateMaxScore(score)
                    waitingForInput = False
            # Si el jugador elige incorrectamente, muestra la animación de
final de juego
            elif (clickedButton and pattern[currentStep] != clickedButton)
or (
                    time.time() - lastTimeClick > timeout and currentStep !=
0):
                gameOverAnimation()
                startScreen()
            #se reinician estas variables cuando se reinicia el juego para
que empieze de nuevo
                pattern = []
                currentStep = 0
                waitingForInput = False
                score = 0
                lastTimeClick = 0
                pygame.time.wait(2000)
        pygame.display.update()
        clock.tick(fps)


main()
```