Proyecto final: Parqués UN

Universidad Nacional de Colombia

Juan Camilo Pinto Trujillo

3 de Marzo del 2025
Programación De Computadores
Grupo 18
Profesor: Gustavo Adolfo Mojica Perdigon

Resumen Ejecutivo

1. **Descripción del proyecto**

   Introducción: Se ha pedido la elaboración de un juego de Parqués, el cual sea posible ser jugado en la consola de Python, además se debe tener en cuenta las reglas que se han pedido implementar con respecto a las fichas, movimento, bloqueos, comer fichas, cárcel, resultado en los dados para salir de la cárcel, seguros, dos dados, etc. También es importante recalcar que se ha pedido representar lo que pasa en la consola de una manera más gráfica y/o visual.

   El objetivo es que los jugadores puedan mover sus fichas alrededor del tablero para poder llegar a su base o camino final para ganar, teniendo en cuenta que para ganar, el jugador debe llevar todas las fichas a la meta.

2. **Desafíos y requerimientos técnicos:** El desafío más importante y que costó poder llegar a solucionar, fue en cuanto a la parte gráfica y visual porque es la primera vez que trabajaba de esa manera y tenía nula experiencia en este apartado.
   Primeramente investigué y probé de qué maneras podría llegar a implementar esto y la primera opción fue con matrices para hacer el tablero y hacer el movimiento de las fichas, sin embargo, fue demasiado complicado además de extenso para hacer todo el juego y sinceramente me costó entenderlo. La segunda opción fue con la librería pygame, librería que está enfocada para ayudar a hacer toda clase de juegos y tener una interfaz visual, por lo que arranqué a usar esta opción. De la segunda opción se derivaron 2, en la primera intenté usar imágenes para el tablero y para las fichas, pero fue complicado ajustar las resoluciones de las imágenes y que las fichas se ubicaran y movieran bien, caso por el que fui por la opción 2. El último intento, hice que python dibujara todo, tablero, fichas,dados,etc y también los colores pero cabe recalcar que, por falta de tiempo y que mis compañeros se fueron a otros grupos y quedé solo, simplemente pude hacer que las fichas hicieran un recorrido diferente al normal.

3. **Explicación de la implementación:**

```
# Colores
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (200, 0, 0)
GREEN = (0, 200, 0)
BLUE = (0, 0, 200)
YELLOW = (200, 200, 0)
COLORS = [RED, GREEN, BLUE, YELLOW]
LIGHT_COLORS = [
    (255, 150, 150),  # Rojo claro
    (150, 255, 150),  # Verde claro
    (150, 150, 255),  # Azul claro
    (255, 255, 150)   # Amarillo claro
]
```

   - Aquí se definen los colores que se usarán en el juego. Los colores principales son rojo, verde, azul y amarillo para cada jugador, además de versiones claras de estos colores. También se definen blanco y negro para el fondo y los bordes.

```
# Inicializar Pygame
pygame.init()
screen = pygame.display.set_mode((display_size, display_size))
pygame.display.set_caption("Parqués")
font = pygame.font.Font(None, 36)
small_font = pygame.font.Font(None, 24)
```

- Esta parte inicializa Pygame, crea la ventana del juego con el tamaño especificado, le da un título ("Parqués") y configura dos tamaños de fuente para mostrar texto.

```python
dice_values = [1, 1]  # Dos dados con valores iniciales
dice_rolled = False
turn = 0
selected_piece = 0
game_over = False
winner = -1
```

Estas variables controlan el estado del juego:

- dice_values: Valores actuales de los dos dados
- dice_rolled: Indica si ya se lanzaron los dados en el turno actual
- turn: El jugador actual (0-3)
- selected_piece: La ficha seleccionada del jugador actual
- game_over: Indica si el juego ha terminado
- winner: El jugador que ganó el juego (-1 significa que aún no hay ganador)

```python
# Definir casillas centrales del tablero (meta)
central_cell = (7, 7)

# Posiciones iniciales de las fichas (cárcel/casa)
initial_positions = {
    0: [(2, 2), (3, 2), (2, 3), (3, 3)],          # Rojo (esquina superior izquierda)
    1: [(11, 2), (12, 2), (11, 3), (12, 3)],      # Verde (esquina superior derecha)
    2: [(11, 11), (12, 11), (11, 12), (12, 12)],  # Azul (esquina inferior derecha)
    3: [(2, 11), (3, 11), (2, 12), (3, 12)]       # Amarillo (esquina inferior izquierda)
}
player_positions = {p: list(initial_positions[p]) for p in range(4)}

# Registro de vueltas dadas por cada ficha
piece_laps = {0: [0, 0, 0, 0], 1: [0, 0, 0, 0], 2: [0, 0, 0, 0], 3: [0, 0, 0, 0]}

# Casillas de salida para cada color - actualizadas según la imagen
starting_positions = [(6, 1), (13, 6), (8, 13), (1, 8)]
```

Aquí se define la estructura del tablero, incluidas:

- La casilla central (meta)
- Las posiciones iniciales de las fichas (cárcel)
- Las posiciones actuales de las fichas
- Un registro de las vueltas completadas por cada ficha
- Las casillas de salida para cada color

```python
main_path = []

# Desde salida roja hacia la derecha
for i in range(6, 14):
    main_path.append((i, 1))

# Bajando por el lado derecho
for i in range(2, 14):
    main_path.append((13, i))

# Hacia la izquierda por abajo
for i in range(12, 0, -1):
    main_path.append((i, 13))

# Subiendo por el lado izquierdo
for i in range(12, 0, -1):
    main_path.append((1, i))

# Hacia la derecha arriba
for i in range(2, 6):
    main_path.append((i, 1))

# Caminos de llegada a meta (según la imagen)
home_paths = {
    0: [(i, 7) for i in range(1, 7)],          # Rojo: hacia la derecha en fila 7
```

```python
safe_spots = set([
    (6, 1),    # Salida roja
    (13, 6),   # Salida verde
    (8, 13),   # Salida azul
    (1, 8),    # Salida amarilla
    (10, 1),   # Seguro en camino común superior
    (13, 10),  # Seguro en camino común derecho
    (4, 13),   # Seguro en camino común inferior
    (1, 4)     # Seguro en camino común izquierdo
])
```

Estos bloques definen:

- El camino principal que rodea el tablero (por donde se mueven todas las fichas)
- Los caminos finales para cada color (por donde las fichas entran a la meta)
- Las casillas seguras donde las fichas no pueden ser capturadas

```python
def draw_board():
```

Esta función dibuja el tablero completo, incluyendo:

- Las casas o cárceles en las esquinas
- La cuadrícula
- El camino común
- Los caminos finales
- La meta central
- Las casillas de salida
- Las casillas seguras

- Opcionalmente, el recorrido del jugador amarillo (implentado únicamente para que el profesor vea el tipo de recorrido que se hace en el tablero)

```
def draw_pieces():
```

Esta función dibuja todas las fichas en el tablero, mostrando:

- El color de cada jugador
- Un número para identificar cada ficha
- Un borde para la ficha seleccionada
- El número de vueltas que ha dado cada ficha

```
def draw_dice():
```

Esta función dibuja los dos dados y sus valores actuales, así como la suma total.

```
def roll_dice():
```

Esta función genera valores aleatorios para los dos dados cuando el jugador presiona la barra espaciadora.

```
def can_leave_jail(dice_values):
    """
```

Comprueba si los valores de los dados permiten que una ficha salga de la cárcel (si un dado muestra 5 o la suma es 5).

```
def get_next_position(player, piece_idx, steps):
```

Esta compleja función calcula la nueva posición de una ficha después de moverse un número determinado de pasos, considerando:

- Si está en la cárcel, meta o camino final
- Si completa una vuelta
- Si debe entrar en el camino final
- Si llega exactamente a la meta

```
def move_piece():
```

Esta función maneja el movimiento de la ficha seleccionada según los valores de los dados, incluyendo:

- Salir de la cárcel
- Moverse por el camino principal
- Capturar fichas
- Verificar si el jugador ha ganado

```
def check_capture(new_pos):
```

Esta función verifica si una ficha ha capturado a otras al caer en la misma casilla, enviando las fichas capturadas de vuelta a la cárcel.

```
def draw_game_info():
    # Mostrar turno actual, ficha seleccionada, instrucciones, gan
    # ...

def draw_instructions():
    # Mostrar las instrucciones del juego...
    # ...
```

Estas funciones muestran información relevante para el jugador, como:

- El turno actual
- La ficha seleccionada
- Instrucciones para jugar
- El ganador (si el juego ha terminado)

Conclusión

Este proyecto puso a prueba todas mis habilidades de programación aprendidas en clase, me enfrenté a varios retos a la hora de tener que ejecutar un código tan extenso, además del reto a la hora de hacer una representación visual de lo que pasa en la consola. Sinceramente, como reflexión, faltó una mayor dedicación al proyecto para pulir los detalles en la parte gráfica y el movimiento para llegar a una perfecta resolución del pedido.

Un proyecto bastante lograble, pero con un gran trabajo atrás (tiempo) y quizás una mejor colaboración y comunicación con mis antiguos compañeros para no terminar separados.

Repositorio y video
https://youtu.be/oTn__khOIRk
https://github.com/Sunmigod/ParquesUN-Juan-Camilo-Pinto
Bibliografía

Pygame Development Team. (s.f.). *Pygame Front Page*. Recuperado de https://www.pygame.org/docs/