Universidad del valle

Inteligencia artificial

1er Semestre 2024

Entrega proyecto 2

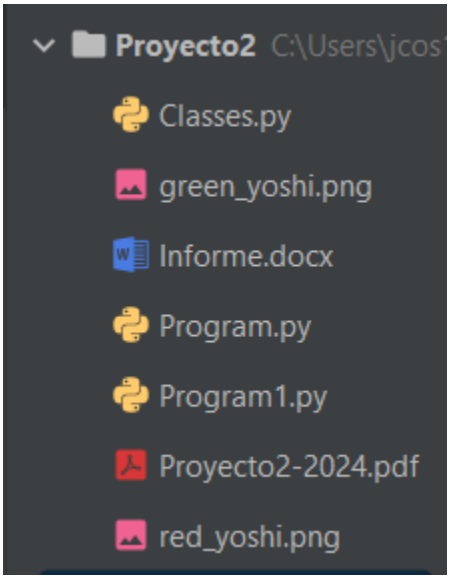
Juan Camilo Ortiz Sanchez 1810223

Fecha: 23 de mayo del 2024

Código

El archivo principal es el llamado Program.py que se encarga de mostrar la interfaz, recibir el evento y mostrar el avance del juego, dentro del cual se definen las constantes usadas durante el juego como lo son la cantidad de filas y columnas para el tablero.

Se creó el archivo Classes.py para representar los componentes del juego, Yoshi, el algoritmo minimax y la función heurística.



El acceso al código se encuentra en: [Github]

https://github.com/cmlooz/Yoshis_World

Función Heurística

Se definió la función heurística (Heuristic) como la una función que calcula el valor del movimiento siguiente manera:

- Recibe los parámetros:
  - Posición del Yoshi verde
  - Posicion del Yoshi rojo
  - Cantidad de filas del tablero
  - Cantidad de columnas del tablero
  - Lista de casillas pintadas por el Yoshi verde
  - Lista de casillas pintadas por el Yoshi rojo
- Se calculan los movimientos posibles para la posición de ambos Yoshis

- Se resta la cantidad de casillas pintadas por el Yoshi verde menos las pintadas por el Yoshi rojo
- Se reste la cantidad de movimientos posibles el Yoshi verde menos la cantidad de movimientos posibles para el Yoshi rojo

```
class Heuristic:
    def heuristic(self,green_position,red_position, ROWS, COLS,green_painted,red_painted,current_turn):
        green_possible_moves = MiniMax().get_valid_moves(green_position[0], green_position[1], ROWS, COLS, set(green_painted),
                                                         set(red_painted))
        red_possible_moves = MiniMax().get_valid_moves(red_position[0], red_position[1], ROWS, COLS, set(green_painted),
                                                       set(red_painted))
        return float((len(green_possible_moves)-len(red_possible_moves))+(len(green_painted)-len(red_painted)))
```

Algoritmo Minimax

La implementación del algoritmo miminax hace la poda del árbol y se ejecuta recursivamente dependiendo de la dificultad seleccionada, cuando alcanza la profundidad de las hojas se usa el valor de la heurística para cada nodo seleccionado y así completar el algoritmo.

Se define de la siguiente manera:

- Recibe los parámetros:
    - Posición del Yoshi verde
    - Posicion del Yoshi rojo
    - Profundidad
    - Si es nodo MAX o MIN
    - Valor alfa
    - Valor Beta
    - Turno
    - Cantidad de filas del tablero
    - Cantidad de columnas del tablero
    - Lista de casillas pintadas por el Yoshi verde
    - Lista de casillas pintadas por el Yoshi rojo
- Si en el nodo seleccionado ya se alcanzó la profundidad o no hay movimientos disponibles el algoritmo asigna al nodo hoja el valor de la función heurística
- Se calculan los movimientos posibles y en cada movimiento se simula la selección de la casilla, si es un nodo MAX se calcula el valor de alfa haciendo un llamado recursivo al algoritmo pasando la profundidad actual menos 1, la nueva posición y ahora como nodo MIN

```python
def minimax(self, green_yoshi_position, red_yoshi_position, depth, is_maximizing, alpha, beta, player, ROWS, COLS, green_painted, red_painted):
    moves = self.get_valid_moves(green_yoshi_position[0], green_yoshi_position[1], ROWS, COLS, green_painted, red_painted)
    if depth == 0 or not moves:
        return Heuristic().heuristic(green_yoshi_position, red_yoshi_position, ROWS, COLS,green_painted,red_painted)
    if is_maximizing:
        max_eval = float(-math.inf)
        for move in moves:
            green_painted.add(move)
            eval = self.minimax((move[0],move[1]), red_yoshi_position, depth - 1, False, alpha, beta, player, ROWS, COLS, green_painted, red_painted)
            green_painted.remove(move)
            max_eval = float(max(max_eval, eval))
            alpha = float(max(alpha, eval))
            if beta <= alpha:
                break
        return float(max_eval)
    else:
        min_eval = float(math.inf)
        for move in moves:
            red_painted.add(move)
            eval = self.minimax(green_yoshi_position,(move[0],move[1]), depth - 1, True, alpha, beta, player, ROWS, COLS, green_painted, red_painted)
            red_painted.remove(move)
            min_eval = float(min(min_eval, eval))
            beta = float(min(beta, eval))
            if beta <= alpha:
                break
        return float(min_eval)
```