Proyecto 2:
Yoshi's Zones

Integrantes del Grupo:
Jean Paul Davalos - 1832375
Miguel Angel Escobar - 2159832
Juan Manuel Perea Coronado - 1926462
Yenny Margot Rivas Tello - 2182527

Profesor:
Oscar Bedoya

Universidad del Valle
Escuela de Ingeniería de Sistemas y Computación
Inteligencia Artificial
2025 - 1

Introducción

“Yoshi’s zones” es un juego entre dos adversarios en el que cada uno controla un Yoshi sobre un tablero de ajedrez. Cada Yoshi se mueve como un caballo y hay un Yoshi verde y otro rojo. En el tablero hay cuatro zonas especiales. Cada zona especial está formada por la celda que se encuentra en una de las esquinas y las cuatro casillas adyacentes indicadas en la imagen. Cuando un Yoshi alcanza cualquiera de las casillas que hacen parte de una zona especial, se pinta con su respectivo color dicha casilla. Las casillas pintadas no se pueden volver a usar por ningún jugador. Durante el juego, las casillas que no hacen parte de una zona especial no se pintan y por lo tanto pueden ser usadas por cualquier jugador.”

En este informe trataremos el desarrollo de este proyecto, centrándonos principalmente en el desarrollo del algoritmo para la toma de decisiones del Yoshi verde en Yoshi 's Zones las cuales son controladas por la inteligencia artificial utilizando el algoritmo Minimax. Y dado que el juego no permite explorar todos los estados posibles debido a la profundidad limitada del árbol de decisiones, se incorpora una función de utilidad heurística que permite estimar qué tan favorable es una posición del tablero para la IA.

Esta función asigna un valor numérico a cada estado del juego, guiando al Yoshi verde para elegir movimientos que incrementen sus probabilidades de ganar la mayoría de las zonas especiales. Estas son las heurísticas utilizadas para nuestro proyecto.

Control de Zonas Especiales

Aquí otorgamos puntos dependiendo de cuántas casillas ha pintado cada jugador dentro de cada zona especial:

```
# Bonificación por controlar zonas
if green_count > red_count:
    score += 10 * (green_count - red_count)
elif red_count > green_count:
    score -= 10 * (red_count - green_count)
```

Si el Yoshi verde controla más casillas que el rojo en una zona, se suma:

+10 × (diferencia de casillas)

Si el rojo lleva ventaja, se penaliza:

-10 × (diferencia de casillas)

Además, se bonifica estados cercanos a completar una zona:

```
# Bonificación por estar cerca de completar una zona
if green_count == 3:
    score += 25
elif green_count == 2:
    score += 15

# Penalización si el oponente está cerca de completar
if red_count == 3:
    score -= 30
elif red_count == 2:
    score -= 20
```

3 casillas verdes en una zona: +25 puntos

2 casillas verdes: +15 puntos

3 rojas: -30 puntos

2 rojas: -20 puntos

Priorización de Zona Inicial

En el nivel de dificultad "experto", se define una “zona inicial” cercana a la posición de inicio de la IA. Si dicha zona aún no está dominada por el oponente (esto de determina cuando tenemos menos de 3 casillas rojas tomadas en la zona), se da una bonificación adicional proporcional a las casillas verdes:

```python
# Priorizar zona inicial si no está completamente perdida
if (self.difficulty == Difficulty.EXPERT and
    self.initial_zone_index is not None and
    i == self.initial_zone_index):

    # Si aún podemos ganar esta zona, darle alta prioridad
    if red_count < 3:
        score += 15 * green_count
```

+15 × (cantidad de casillas verdes en esa zona)

Proximidad a Celdas No Pintadas

La heurística incentiva la cercanía del Yoshi verde a celdas especiales sin pintar, penalizando si el oponente está más cerca:

```python
# 2. Evaluar proximidad a zonas no controladas
for zone in self.special_zones:
    for cell in zone:
        if cell not in painted_cells:
            g_dist = abs(green_pos[0] - cell[0]) + abs(green_pos[1] - cell[1])
            r_dist = abs(red_pos[0] - cell[0]) + abs(red_pos[1] - cell[1])

            # Bonificación por estar más cerca de celdas no pintadas
            if g_dist < r_dist:
                score += 3
            elif r_dist < g_dist:
                score -= 2
```

Yoshi verde más cercano a la zona: +3 puntos

Yoshi rojo más cercano a al zona: -2 puntos

Posición Estratégica en el Tablero

Se valora la ubicación del Yoshi verde en relación con el centro del tablero, lo cual proporciona mayor libertad de movimiento.

La bonificación es inversamente proporcional a la distancia desde el centro (posición ideal: casillas cercanas a (3.5, 3.5)).

```
# 3. Evaluar posición estratégica
# Bonificación por posiciones centrales (más opciones de movimiento)
center_distance = abs(green_pos[0] - 3.5) + abs(green_pos[1] - 3.5)
score += (7 - center_distance) * 0.5
```

Prevención de Repetición de Movimientos

Si el movimiento actual ha sido repetido recientemente, se penaliza con:

-20 puntos

Con esto buscamos evitar bucles de movimiento y promover la progresión del juego.

```
# 4. Penalización por repetición de movimientos
if move_history and is_repetitive_move(green_pos, move_history):
    score -= 20
```

Relación con el Algoritmo Minimax

Dado que Minimax explora las posibles secuencias de jugadas hasta una profundidad definida por el nivel de dificultad (2, 4 o 6), la función heurística actúa como la evaluación final en los nodos hoja del árbol. La calidad de esta evaluación afecta directamente la eficacia de las decisiones de la IA. Donde cuanto más precisa y alineada esté la heurística con el objetivo del juego (básicamente se hallan gando zonas o casillas), mejor será el desempeño, incluso sin explorar todos los estados posibles.

Conclusión

En la función de utilidad heurística planteada se busca equilibrar múltiples criterios estratégicos relevantes, premiando el control de zonas, la cercanía táctica, el posicionamiento y la variedad de movimientos. Esta función permite a la IA tomar decisiones competitivas sin necesidad de explorar completamente el espacio de estados, adaptándose efectivamente a los distintos niveles de dificultad.

Para acceder al repositorio de GitHub con el proyecto, de click en el siguiente enlace:

Repositorio de GitHub