**Integrantes:**

- Mauricio Carrillo - 2024092
- Juan Esteban Mazuera - 2043008
- Sheilly Ortega - 2040051

**Docente:**

Oscar Fernando Bedoya Leiva

## Proyecto de Inteligencia Artificial 2

La heurística utilizada se basa en calcular la distancia euclidiana entre la posición actual del personaje y las posiciones de los elementos que deben ser recolectados. Para este proyecto tomamos los puntos que se encuentran en el campo de juego y los priorizamos en orden de descendencia. Desde 7 hasta 1. Se escoge como objetivo para la heurística el campo de mayor puntaje que se tenga.

La distancia euclidiana (ver Ec. 1) es una medida de la distancia recta entre dos puntos en un plano. En este contexto, se utiliza para estimar la distancia real que se necesita recorrer para llegar a los elementos restantes.

$$distancia = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Ec 1. Distancia euclidiana.

La heurística para el algoritmo minimax, entonces, toma la casilla de mayor puntaje que se tenga en el campo de juego y obtiene la distancia euclidiana hacia ella. Por ejemplo, si existen dos 7, se toma el último que encuentre como la base para la heurística. Esto se debe a que a propósito se colocó una debilidad en la inteligencia artificial para darle ventaja al jugador. El 7 puede estar en una posición muy lejana por lo cual el jugador humano tendrá la capacidad de ir a por otras casillas con puntos más cercanos. Por defecto las casillas se encuentran en -1, lo cual es un valor imposible y por ende lo utilizamos para definir que ese punto no existe.

```
def setHeuristica(self):
    punto7X, punto7Y = -1, -1
    punto6X, punto6Y = -1, -1
    punto5X, punto5Y = -1, -1
    punto4X, punto4Y = -1, -1
    punto3X, punto3Y = -1, -1
    punto2X, punto2Y = -1, -1
    punto1X, punto1Y = -1, -1
```

**Figura 1. Proceso de inicialización de variables para los puntos válidos**

Universidad del Valle

```
for i in range(len(self.mapa)):
    for j in range(len(self.mapa[i])):
        if self.mapa[i][j] == 1:
            punto1Y = i
            punto1X = j
        if self.mapa[i][j] == 2:
            punto2Y = i
            punto2X = j
        if self.mapa[i][j] == 3:
            punto3Y = i
            punto3X = j
        if self.mapa[i][j] == 4:
            punto4Y = i
            punto4X = j
        if self.mapa[i][j] == 5:
            punto5Y = i
            punto5X = j
        if self.mapa[i][j] == 6:
            punto6Y = i
            punto6X = j
        if self.mapa[i][j] == 7:
            punto7Y = i
            punto7X = j
```

**Figura 2. Proceso de identificación de casillas con puntos**

```
if(punto7X != -1 and punto7Y != -1):
    hx = punto7X
    hy = punto7Y
elif(punto6X != -1 and punto6Y != -1):
    hx = punto6X
    hy = punto6Y
elif(punto5X != -1 and punto5Y != -1):
    hx = punto5X
    hy = punto5Y
elif(punto4X != -1 and punto4Y != -1):
    hx = punto4X
    hy = punto4Y
elif(punto3X != -1 and punto3Y != -1):
    hx = punto3X
    hy = punto3Y
elif(punto2X != -1 and punto2Y != -1):
    hx = punto2X
    hy = punto2Y
elif(punto1X != -1 and punto1Y != -1):
    hx = punto1X
    hy = punto1Y

self.heuristica = np.sqrt((hx - self.posicionX)**2 + (hy - self.posicionY) ** 2)
```

**Figura 3. Toma de decisiones para casilla a enfocar**