Entrega 1 : Interfaz del juego

Aldemar Ramirez - Camilo Moreno - Héctor Rodríguez

27 de abril de 2021

Resumen

En este documento se presenta el juego 2048. Se mostrará el análisis, el diseño y el pseudocódigo de la implementación.

Parte I
Análisis y diseño

1. Análisis

1.1. 2048

El juego 2048, informalmente, se describe como un rompecabezas deslizable cuyo objetivo es mover las baldosas numeradas para combinarlas hasta llegar al número 2048.

Se hace uso de una matriz M de tamaño 4x4 para su visualización.

Una baldosa hace referencia al valor contenido dentro de una posición $M[i][j]$.

Definición. Modo de juego:

1. Se inicia generando la matriz M, asignando valores a dos baldosas aleatoriamente escogidas.
2. El jugador elige en que dirección moverse y las baldosas se deslizarán en dicha dirección hasta que tengan contacto con otra baldosa o con el borde de la matriz.
3. En cada movimiento, se llena una baldosa desocupada con un valor de dos o de cuatro en la matriz.
4. Si al hacer movimiento la matriz no sufre ningún cambio no se agrega baldosa aleatoria.
5. Cuando una baldosa hace contacto con otra y el número contenido dentro de estas es el mismo, se convertirán en una sola baldosa cuyo valor es la suma de los valores de las dos previas baldosas. Esta baldosa creada no puede combinarse con otra baldosa en el mismo movimiento.

6. Cuando en un movimiento se deslizan tres baldosas consecutivas del mismo valor, solo se combinarán dos de ellas.

7. Dada la matriz M, si todas las baldosas tienen un valor, y no es posible realizar combinaciones en ninguna dirección, finaliza el juego.

8. El juego registra el puntaje del jugador. El puntaje comienza en cero y cuando dos baldosas se combinan, este se incrementa por el valor de la baldosa resultante.

9. Cuando el puntaje registrado en una baldosa es 2048, el jugador gana.

2. Diseño

Teniendo en cuenta las consideraciones del análisis, se puede definir el diseño del juego distinguiendo las entradas que permitan el correcto desarrollo y las salidas que generan el resultado.

Definición. Entradas:

- La tecla ingresada por el usuario es convertida en una cadena que representa la dirección en la cual se quiere realizar el movimiento. Las cadenas o teclas que se permite usar son:

  a: Movimiento hacia la derecha.

  w: Movimiento hacia arriba.

  s: Movimiento hacia abajo.

  d: Movimiento hacia la izquierda.

  n: Termina el juego.

- Una matriz M.

- El puntaje

Definición. Salidas:

- La matriz M transformada luego de aplicar el movimiento.

- El puntaje actual del jugador.

Parte II
Algoritmos

3. Algoritmos de Movimiento en Direcciones

3.1. Algoritmo de dirección - Arriba

Algorithm 1 Algoritmo Dirección Arriba

```
1:  procedure ARRIBA(M, puntaje)
2:      tam ← 4
3:      for i ← 1 to tam + 1 do
4:          j ← 1
5:          k ← j + 1
6:          while j < tam + 1 ∧ k < tam + 1 do
7:              if M[i][j] == 0 ∧ M[i][k] ≠ 0 then
8:                  M[i][j] ← M[i][k]
9:                  M[i][k] ← 0
10:             else
11:                 if M[i][j] == M[i][k] ∧ M[i][j] ≠ 0 then
12:                     M[i][j] ← M[i][j] × 2
13:                     puntaje ← puntaje + M[i][j]
14:                     M[i][k] ← 0
15:                     j ← j + 1
16:                     k ← j + 1
17:                 else
18:                     if M[i][j] ≠ 0 ∧ M[i][k] ≠ 0 then
19:                         j ← j + 1
20:                         k ← j + 1
21:                     else
22:                         if M[i][k] == 0 then
23:                             k ← k + 1
24:                         end if
25:                     end if
26:                 end if
27:             end if
28:         end while
29:     end for
30:     return M, puntaje
31: end procedure
```

3.2. Algoritmo de dirección - Derecha

Algorithm 2 Algoritmo Dirección Derecha

```
1: procedure DERECHA(M, puntaje)
2:     tam ← 4
3:     for j ← 1 to tam + 1 do
4:         i ← tam − 1
5:         k ← i − 1
6:         while j > 0 ∧ k > 0 do
7:             if M[i][j] == 0 ∧ M[k][j] ≠ 0 then
8:                 M[i][j] ← M[k][j]
9:                 M[k][j] ← 0
10:            else
11:                if M[i][j] == M[k][j] ∧ M[i][j] ≠ 0 then
12:                    M[i][j] ← M[i][j] × 2
13:                    puntaje ← puntaje + M[i][j]
14:                    M[k][j] ← 0
15:                    i ← i − 1
16:                    k ← k − 1
17:                else
18:                    if M[i][j] ≠ 0 ∧ M[k][j] ≠ 0 ∧ M[i][j] ≠ M[k][j] then
19:                        i ← i − 1
20:                        k ← i − 1
21:                    else
22:                        if M[k][j] == 0 then
23:                            k ← k − 1
24:                        end if
25:                    end if
26:                end if
27:            end if
28:        end while
29:    end for
30:    return M, puntaje
31: end procedure
```

3.3. Algoritmo de dirección - Abajo

Algorithm 3 Algoritmo Dirección Abajo

```
1: procedure ABAJO(M, puntaje)
2:     tam ← 4
3:     for i ← 1 to tam + 1 do
4:         j ← tam − 1
5:         k ← j − 1
6:         while j > 0 ∧ k > 0 do
7:             if M[i][j] == 0 ∧ M[i][k] ≠ 0 then
8:                 M[i][j] ← M[i][k]
9:                 M[i][k] ← 0
10:            else
11:                if M[i][j] == M[i][k] ∧ M[i][j] ≠ 0 then
12:                    M[i][j] ← M[i][j] × 2
13:                    puntaje ← puntaje + M[i][j]
14:                    M[i][k] ← 0
15:                    j ← j − 1
16:                    k ← j − 1
17:                else
18:                    if M[i][j] ≠ 0 ∧ M[i][k] ≠ 0 then
19:                        j ← j − 1
20:                        k ← j − 1
21:                    else
22:                        if M[i][k] == 0 then
23:                            k ← k − 1
24:                        end if
25:                    end if
26:                end if
27:            end if
28:        end while
29:    end for
30:    return M, puntaje
31: end procedure
```

3.4. Algoritmo de dirección - Izquierda

Algorithm 4 Algoritmo Dirección Izquierda

```
1: procedure IZQUIERDA(M, puntaje)
2:     tam ← 4
3:     for j ← 1 to tam + 1 do
4:         i ← 1
5:         k ← i + 1
6:         while j < tam + 1 ∧ k < tam + 1 do
7:             if M[i][j] == 0 ∧ M[k][j] ≠ 0 then
8:                 M[i][j] ← M[k][j]
9:                 M[k][j] ← 0
10:            else
11:                if M[i][j] == M[k][j] ∧ M[i][j] ≠ 0 then
12:                    M[i][j] ← M[i][j] × 2
13:                    puntaje ← puntaje + M[i][j]
14:                    M[k][j] ← 0
15:                    i ← i + 1
16:                    k ← i + 1
17:                else
18:                    if M[i][j] ≠ 0 ∧ M[k][j] ≠ 0 ∧ M[i][j] ≠ M[k][j] then
19:                        i ← i + 1
20:                        k ← i + 1
21:                    else
22:                        if M[k][j] == 0 then
23:                            k ← k + 1
24:                        end if
25:                    end if
26:                end if
27:            end if
28:        end while
29:    end for
30:    return M, puntaje
31: end procedure
```

Parte III
Entregables

Para la primera entrega correspondiente a la interfaz, se tienen los siguientes entregables:

- Logica2048.py : Tiene la lógica de movimiento, las validaciones y la creación de la matriz.Contiene la clase arcade.

- GUI2.py: Tiene la creación de la interfaz gráfica.Es el ejecutable.Contiene la clase interface.
- Colores.py:Enum en donde se establecen los colores para las celdas y números.