

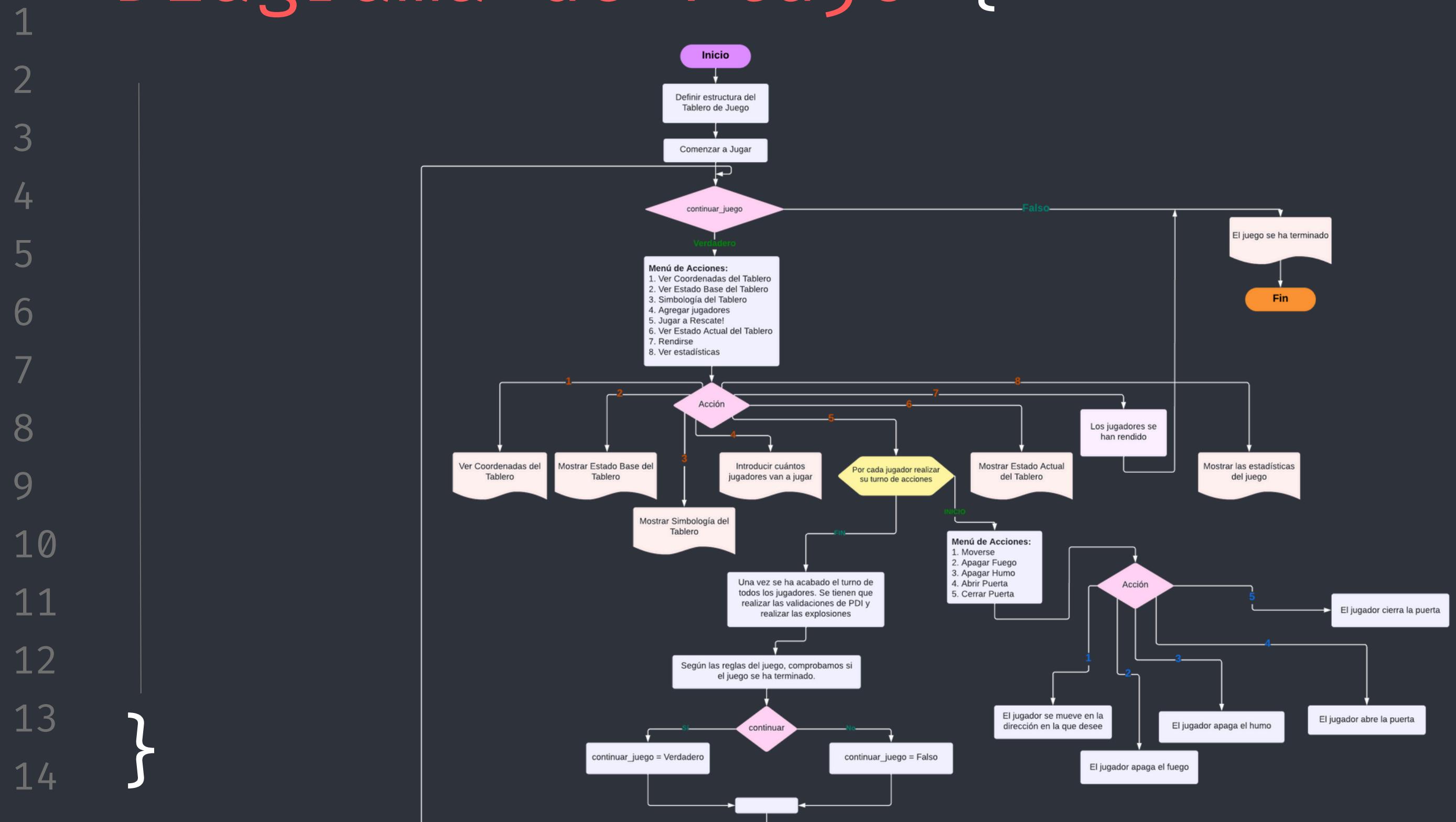
1 PROYECTO FINAL ‘Algoritmos’ {  
2  
3 [Rescate! Juego de Mesa]  
4  
5 <li>  
6 **Integrantes:**  
7 • Del Carmen Arteaga Jorge Eduardo  
8 • Herrera Correa Luis Axel  
9 • López Ordoñes David Esaú  
10 • Luna Gómez Guadalupe de Jesús  
11 • Rodríguez Medina Dennis Rogelio  
12 <li/>  
13 }  
14 }



# Contenido;

```
1  
2  
3 <ol>  
4   <li> Diagramas de Flujo  
5   <li> Características  
6   <li> Limitaciones  
7   <li> Glitches  
8   <li> Porteo hacia otros lenguajes  
9   <li> Nivel de Complejidad  
10 <ol/>  
11  
12  
13  
14
```

# Diagrama de Flujo



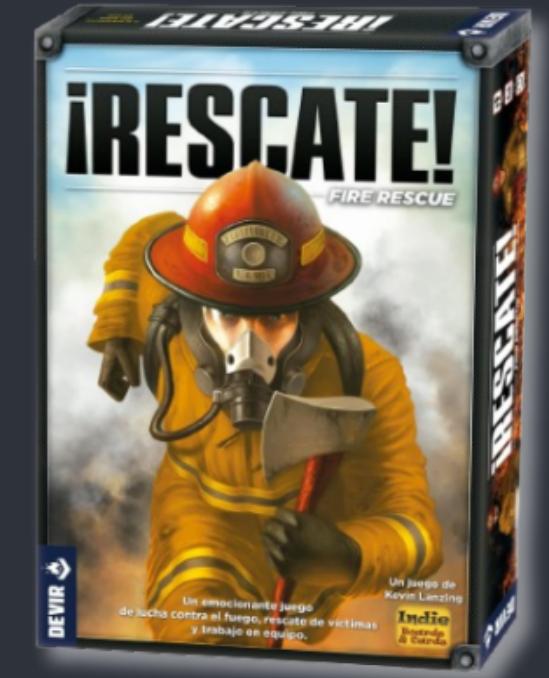
# Características {

1  
2     | ¡Rescate! es un juego cooperativo. En el que tienes que salvar a  
3     | personas de una casa en llamas.  
4  
5

## 6     | **Modificaciones:** 7 8

- 9     | • Dificultad fácil  
10    | • No hay ambulancias, focos de calor, primeros auxilios,  
11    |     materias peligrosas y habilidades de jugador  
12    | • Al momento de llegar a una casilla donde hay un PDI, en  
13    |     caso de ser una persona el equipo de rescate llegará a  
14    |     salvarla.

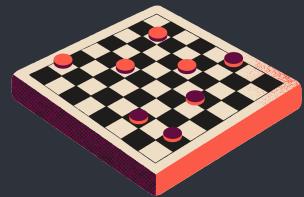
11    | Por lo demás, el juego sigue siendo el mismo.  
12  
13    | }



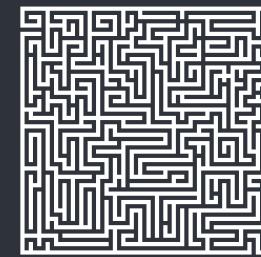
# Limitaciones {



Conocimientos básicos de programación.



Programación del tablero de juego puede provocar errores.



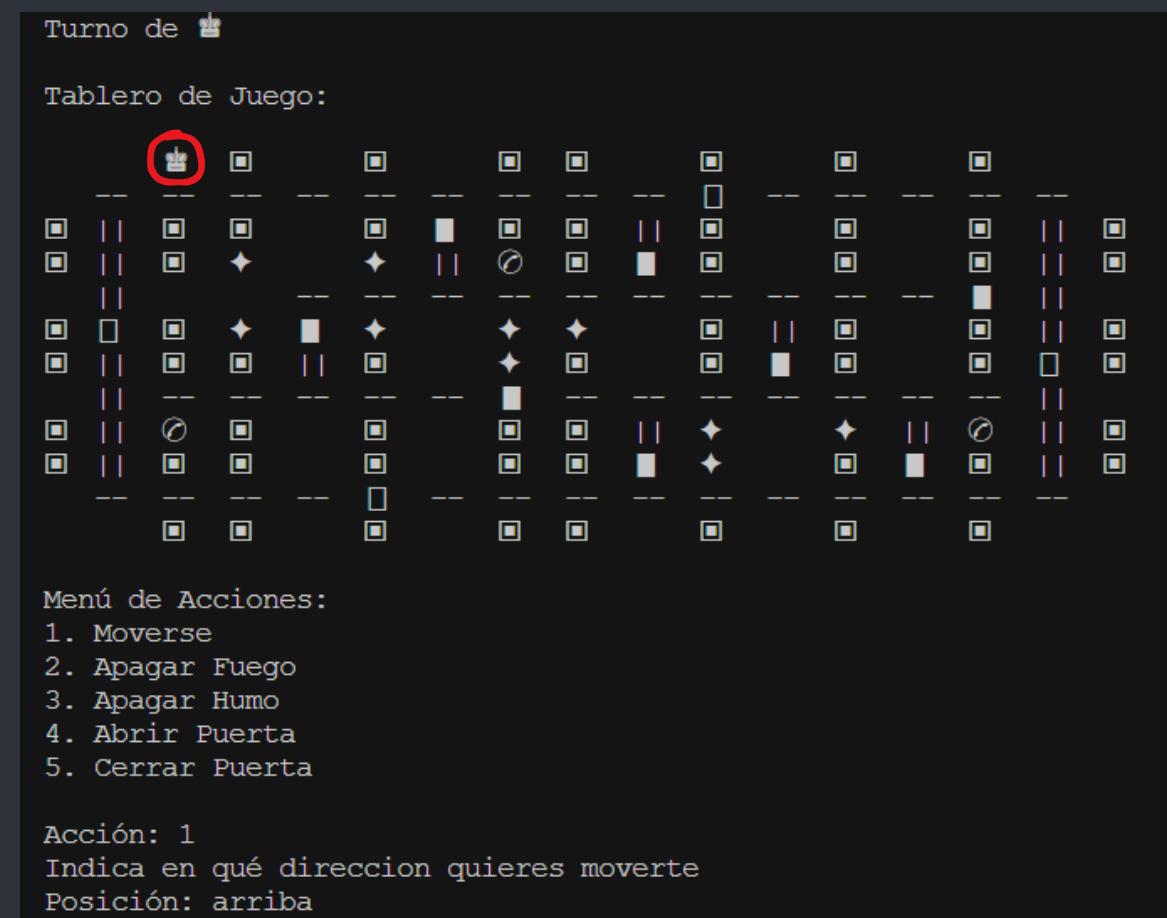
Robustez del código puede dificultar su comprensión o encontrar los fallos.

{

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

# Glitches {

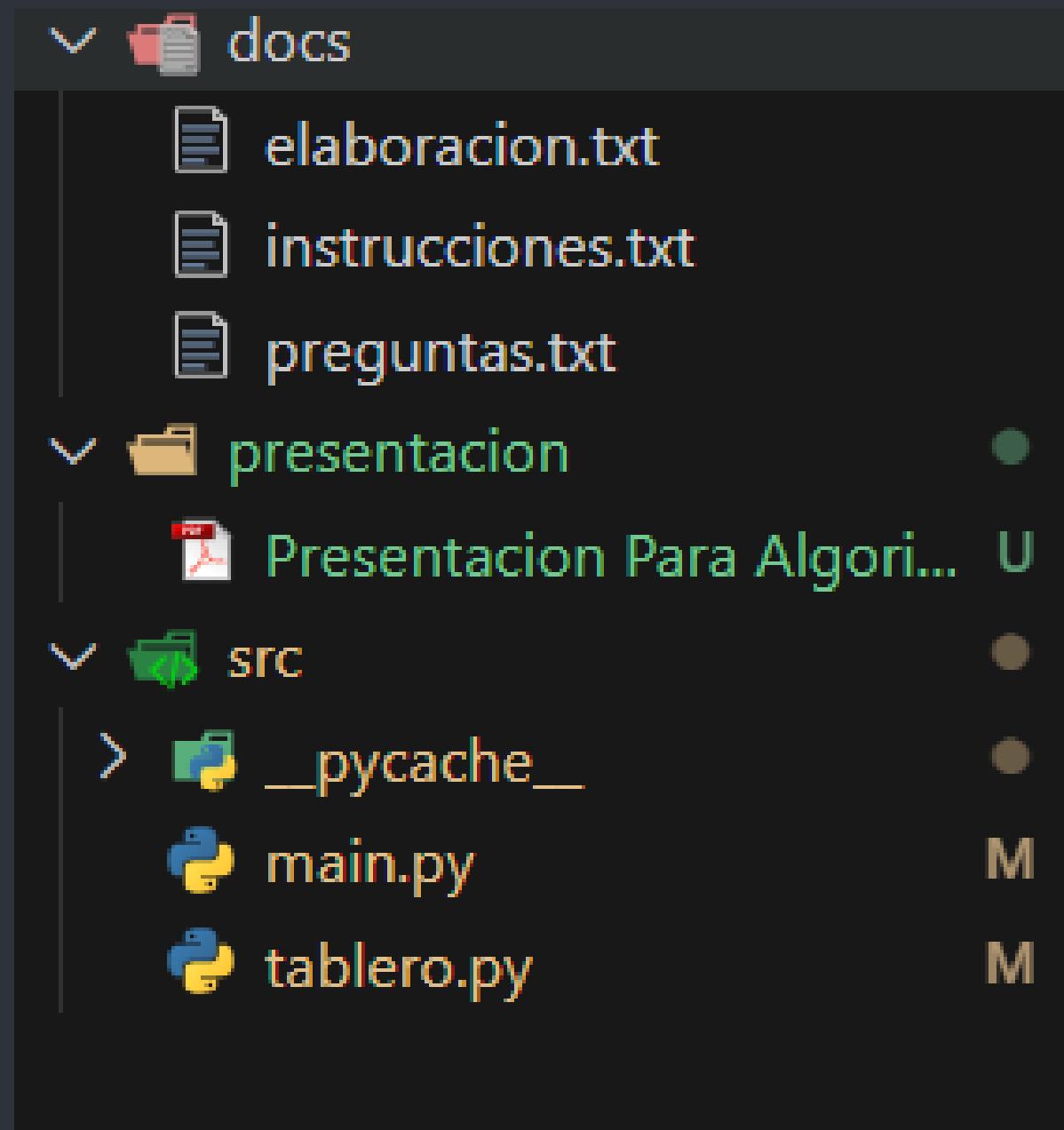
1. El jugador puede moverse fuera de los límites del tablero



13      }  
14 }

# Explicación {

0. Creamos la estructura de nuestro proyecto



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14 }

# Explicación {

## 1. Creamos la estructura estructura de nuestro tablero

```
# Creamos una función que nos facilite mostrar al usuario el tablero de juego con la librería tabulate
def imprimir_tablero(tablero):
    print(tabulate(tablero, tablefmt='plain'))
```

# Explicación {

2. Importamos las librerias y funciones que usaremos

```
# Importamos la libreria tabulate que nos ayuda a darle un formato al momento de imprimir una matriz
from tabulate import tabulate

# Importamos las funciones de nuestro tablero
from tablero import coordenadas_tablero
from tablero import logica_tablero
from tablero import preparacion_tablero
from tablero import imprimir_tablero

# Importamos la libreria random que usaremos a lo largo del problema
import random
```

13  
14 }

# Explicación {

3. Creamos una función para darle la bienvenida al usuario

```
1 #·Creamos·una·función·para·dar·la·bienvenida·al·usuario
2 def·bienvenida_usuario():
3     print("|BIENVENIDO A...|".center(65, '-'))
4     print("""\n
5
6
7
8
9
10
11
12
13
14     print("|Pulsa enter para comenzar|".center(65, '-'))
15     comenzar = input("")
```

# Explicación {

## 4. Creamos el sistema de funcionamiento de lanzar dados

```
1 # Debido a la estructura de nuestro tablero, crear un dado de 6 y 8 caras no es una opción
2 # por lo que crearemos una función que simule el funcionamiento que tendría el lanzar los dados
3
4 def lanzar_dados():
5     resultados_posibles = ['2.2', '2.3', '2.4', '2.5', '2.7', '2.8', '2.10', '2.11', '2.12', '2.13', '2.14',
6                             '3.2', '3.3', '3.4', '3.5', '3.7', '3.8', '3.10', '3.11', '3.12', '3.14',
7                             '5.2', '5.3', '5.5', '5.6', '5.7', '5.8', '5.9', '5.10', '5.12', '5.13', '5.14',
8                             '6.2', '6.3', '6.5', '6.6', '6.7', '6.8', '6.9', '6.10', '6.12', '6.13', '6.14',
9                             '8.2', '8.3', '8.4', '8.5', '8.6', '8.7', '8.8', '8.10', '8.11', '8.12', '8.14',
10                            '9.2', '9.3', '9.4', '9.5', '9.6', '9.7', '9.8', '9.10', '9.11', '9.12', '9.14']
11
12     # Escoge un resultado aleatorio dentro de resultados_posibles
13     resultado = random.choice(resultados_posibles)
14
15     # Lo dividimos en dos partes para obtener las coordenadas [i,j] cada una por individual
16     partes = resultado.split('.')
17     i = partes[0]
18     i = int(i)
19     j = partes[1]
20     j = int(j)
21
22     return i, j
23 }
```

# Explicación {

5. Creamos una función para saber la posición actual del jugador

```
1 # Creamos una función para conocer la posición actual del jugador y utilizarla en el juego.
2 def posicion_actual(tablero, jugador):
3     # Recorremos las filas de la matriz
4     for i, fila in enumerate(tablero):
5         # Recorremos las columnas de la fila actual
6         for j, valor in enumerate(fila):
7             # Comparamos el valor actual con el elemento buscado
8             if valor == jugador:
9                 # Si se encuentra el elemento, retornamos su ubicación
10                return i, j
11
12    return None
```

}

# Explicación {

## 6. Comenzamos con la función principal y declaración de variables

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14 }

```
# !----- CODIGO PRINCIPAL -----
# Empezamos con la función que se encargará de ejecutar nuestro juego
def main():
    # Empezamos la ejecución de nuestro código dando la bienvenida
    bienvenida_usuario()

    # Creamos las variables que usaremos a lo largo del proyecto
    tablero_coordenadas = coordenadas_tablero()
    tablero_de_juego = preparacion_tablero()
    tablero_estado_base = logica_tablero()
    jugadores_visual = ['▲', '▼', '■', '▲', '▼']
    jugadores = 0
    turno = 0
    coordenada = ''
    casillas_iniciales = ['0.0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '0.10', '0.11', '0.12', '0.13', '0.14', '0.15', '0.16',
                          '1.0', '2.0', '3.0', '4.0', '5.0', '6.0', '7.0', '8.0', '9.0', '10.0',
                          '11.0', '11.1', '11.2', '11.3', '11.4', '11.5', '11.6', '11.7', '11.8', '11.9', '11.10', '11.11', '11.12', '11.13', '11.14', '11.15', '11.16',
                          '1.16', '2.16', '3.16', '4.16', '5.16', '6.16', '7.16', '8.16', '9.16', '10.16'],
    casillas_iniciales_superiores = ['0.0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9', '0.10', '0.11', '0.12', '0.13', '0.14', '0.15', '0.16'],
    casillas_iniciales_izquierda = ['1.0', '2.0', '3.0', '4.0', '5.0', '6.0', '7.0', '8.0', '9.0', '10.0', '11.0'],
    casillas_iniciales_inferiores = ['11.0', '11.1', '11.2', '11.3', '11.4', '11.5', '11.6', '11.7', '11.8', '11.9', '11.10', '11.11', '11.12', '11.13', '11.14', '11.15', '11.16'],
    casillas_iniciales_derecha = ['1.16', '2.16', '3.16', '4.16', '5.16', '6.16', '7.16', '8.16', '9.16', '10.16'],
    cubos_dmg = 0
    pdi_en_tablero = 3
    direccion = ''
    i = 0
    j = 0
    puntos_de_accion = [6, 6, 6, 6, 6]
    puntos_de_interes = [True, True, True, True, True, True, True, True, True, False, False, False, False]
    victimas_salvadas = 0
    victimas_perdidas = 0
    # Variables que usamos para ciclos while
    no_perder = True
    casilla_valida = True
    direccion_valida = True
    apagar_fuego = True
    lanzamiento_valido = True

    print("-".center(65, "-"))
    print("\nComencemos a jugar...\n\n")
```

# Explicación {

1      7. Le mostramos al usuario las opciones que puede realizar

```
2
3     # ! ----- Menú de Opciones que puede hacer el jugador al empezar cada turno -----
4     while(no_perder):
5         opcion = int(input("""\n\nMenú de Opciones:
6             1. Ver Coordenadas del Tablero
7             2. Ver Estado Base del Tablero
8             3. Simbología del Tablero
9             4. Agregar jugadores
10            5. Jugar a Rescate!
11            6. Ver Estado Actual del Tablero
12            7. Rendirse
13            8. Ver estadísticas
14            Elección: """))
```

13      }

# Explicación {

## 8. Primeras tres opciones

```
1 # ! ----- Mostramos las coordenadas del tablero -----
2 if(opcion == 1):
3     print("\nCoordenadas del Tablero: \n")
4     imprimir_tablero(tablero_coordenadas)
5 # ! ----- Mostramos el estado base del tablero sin preparaciones -----
6 elif(opcion == 2):
7     print("\nEstado Base del Tablero: \n")
8     imprimir_tablero(tablero_estado_base)
9 # ! ----- Mostramos la simbología de los elementos de nuestro tablero -----
10 elif(opcion == 3):
11     print("""\nSimbología:
12         # Pared Vertical ..... ||| | |
13         # Pared Vertical Medio Rota ..... | |
14         # Puerta Abierta ..... □
15         # Puerta Cerrada ..... ▣
16         # Jugador ..... ♔ ♕ ♖ ♗ ♚ ♜
17         # Casilla ..... □
18         # Pared Horizontal ..... -- -
19         # Pared Horizontal Medio Rota ..... - -
20         # Fuego ..... ♦
21         # Humo ..... ♦
22         # Punto de interés ..... Ⓜ\n""")
```

{}

# Explicación {

## 9. Agregar jugadores

```
1      # !----- Empezamos con la primera configuración del juego, agregar los jugadores -----
2      elif(opcion==4):
3          continuar=True
4          # Usamos un ciclo while que se repita hasta que los jugadores ingresados sean válidos
5          while(continuar):
6              # Validamos que sea la primera vez que ingresa jugadores, si no ha ingresado jugadores entonces el valor de jugadores será de 0
7              if(jugadores==0):
8                  jugadores=int(input("\n¿Cuántos jugadores van a jugar?\n"))
9                  # Validamos que los jugadores ingresados sean válidos
10                 if(jugadores<=1 or jugadores>6):
11                     print("\nNúmero de jugadores no soportado. Mínimo 2 - Máximo 6\n")
12                     continuar=True
13                     jugadores=0
14                     # En caso de ser válidos se mostrará un mensaje de jugadores registrados y el ciclo while se terminará
15                 else:
16                     print("Jugadores Registrados...")
17                     continuar=False
18                     # Usamos una segunda validación para que el jugador ya no pueda ingresar jugadores si ya lo ha hecho una vez
19                     elif(jugadores>=1 or jugadores<=6):
20                         print("\nYa no puede agregar más jugadores una vez la partida ha iniciado!!!\n")
21                         continuar=False
22
23
24 }
```

# Explicación {

## 10. Selección de casillas iniciales

```
1      # ! -----Comenzamos con la ejecución principal de nuestro juego-----
2      elif(opcion == 5):
3          if(jugadores == 0): # Usamos un if para validar que el usuario ha ingresado anteriormente la cantidad de jugadores
4              print("\n\nPara jugar debes primero indicar cuántos jugadores van a jugar!!!")
5          else:
6              # El turno 0 es el turno en el que los jugadores establecen en qué casilla desean empezar
7              if(turno == 0):
8                  # Usamos un ciclo for que se repita por cada jugador que vaya a jugar
9                  for jugador in range(jugadores):
10                     casilla_valida = True
11                     # Usamos un ciclo while para validar que la casilla en la que desea iniciar el jugador sea válida
12                     while(casilla_valida):
13                         print("\nIndica las coordenadas de la casilla en la que deseas iniciar\nCasillas Disponibles:")
14                         # Imprimimos las casillas donde el usuario puede iniciar
15                         print("Casillas Superiores: ")
16                         for casilla in casillas_iniciales_superiores:
17                             print(casilla, end = ' ')
18                         print("\nCasillas Inferiores: ")
19                         for casilla in casillas_iniciales_inferiores:
20                             print(casilla, end = ' ')
21                         print("\nCasillas Derecha: ")
22                         for casilla in casillas_iniciales_derecha:
23                             print(casilla, end = ' ')
24                         print("\nCasillas Izquierda: ")
25                         for casilla in casillas_iniciales_izquierda:
26                             print(casilla, end = ' ')
27                         # Le pedimos al usuario que ingrese la casilla
28                         coordenada = input("\nCoordenada: ")
29                         # Separamos la coordenada en partes
30                         partes_coordenada = coordenada.split('.')
31                         # Validamos que la casilla sea una en la que puede iniciar
32                         if coordenada in casillas_iniciales:
33                             i = int(partes_coordenada[0])
34                             j = int(partes_coordenada[1])
35                             tablero_de_juego[i][j] = jugadores_visual[jugador]
36                             print('\nTablero de Juego: ')
37                             imprimir_tablero(tablero_de_juego)
38                             casilla_valida = False
39                         # En caso de no ser así, deberá volver a ingresar una casilla
40                         elif tablero_de_juego[i][j] in jugadores_visual:
41                             print("\nCasilla no válida")
42                             casilla_valida = True
43                         # Si ingresa otra cosa diferente a una casilla o no usa la sintaxis correcta, tendrá que volver a ingresar la casilla
44                         else:
45                             print("\nCasilla no válida")
46                             casilla_valida = True
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14 }

# Explicación {

## 11. Menú de acciones y validación de Acción de Movimiento

```
1
2
3 # ! ----- COMIENZO OFICIAL DEL JUEGO -----
4 # En caso de ya no ser el turno 0, eso quiere decir que puede comenzar oficialmente el juego
5 else:
6     print(f"Turno {turno}")
7     # Usamos un ciclo for que se repita para cada jugador
8     for jugador in range(jugadores):
9         print("-".center(100, "-"))
10        print(f"Turno de {jugadores_visual[jugador]}") # Mostramos de quién es el turno
11        puntos_de_accion = [6, 6, 6, 6, 6, 6] # Declaramos los puntos de acción y se reinician siempre al comenzar un turno
12        # Usamos un ciclo while que se repita hasta que el jugador se quede sin puntos de acción
13        while(puntos_de_accion[jugador] >= 1):
14            # Empezamos el juego oficialmente en el que el jugador deberá elegir que acción hacer
15            print("\nTablero de Juego:\n")
16            imprimir_tablero(tablero_de_juego)
17            # Le mostramos al usuario las acciones que puede hacer
18            print("\nMenú de Acciones:")
19            accion = int(input("1. Moverse\n2. Apagar Fuego\n3. Apagar Humo\n4. Abrir Puerta\n5. Cerrar Puerta\n\nAcción: "))
20            i, j = posicion_actual(tablero_de_juego, jugadores_visual[jugador]) # Obtenemos la posición actual del jugador para poder realizar las acciones
21            # Comenzamos con la ejecución para que el jugador pueda moverse
22            # ! -- MOVIMIENTO DEL JUGADOR
23            if(accion == 1):
24                tablero_de_juego[i][j] = ' ' # La posición actual del jugador se sustituye con una casilla vacía para que se visualice correctamente en el tablero
25                direccion_valida = True
26                # Usamos un ciclo while que valide que la dirección sea válida
27                while(direccion_valida):
28                    direccion_valida = True
29                    print("Indica en qué dirección quierés moverte (arriba - abajo - derecha - izquierda)")
30                    direccion = input("Posición: ")
31                    ...
32
33    }
```

# Explicación {

## 12. Explicación del movimiento

```
matriz = [[0.0, 0.1, 0.2],  
          [1.0, 1.1, 1.2],  
          [2.0, 2.1, 2.2]]
```

Supongamos que estamos en la coordenada 1.1, y que cada número le corresponde a i y j respectivamente

### Izquierda:

Si nos quisiéramos mover en una casilla hacia izquierda, el valor de j tendría que reducir. Resultado 1.0

### Derecha:

Si nos quisiéramos mover en una casilla hacia derecha, el valor de j tendría que aumentar. Resultado 1.2

### Arriba:

Si nos quisiéramos mover en una casilla hacia arriba, el valor de i tendría que reducir. Resultado 0.1

### Abajo:

Si nos quisiéramos mover en una casilla hacia abajo, el valor de i tendría que aumentar. Resultado 2.1

```
}
```

# Explicación {

## 13. Movimiento del jugador en código

```

1      # !-----ARRIBA-----
2      if(direccion == "arriba"):
3          # Hacemos validaciones para que la dirección en que se quiere mover el jugador es valida
4          # Si el jugador se mueve en una casilla en la que no se puede
5          if tablero_de_juego[i-1][j] in ['--', '--', '♦', '♦', '■'] + jugadores_visual:
6              print(f"No puedes moverte hacia {direccion}")
7              direccion_valida = True
8          # Si el jugador se mueve a una casilla en blanco o a una puerta se moverá automaticamente una casilla más o no
9          elif tablero_de_juego[i-1][j] == ' ' or tablero_de_juego[i-1][j] == '□':
10             if tablero_de_juego[i-2][j] in ['--', '--', '♦', '♦', '■'] + jugadores_visual:
11                 print(f"No puedes moverte hacia {direccion}")
12                 direccion_valida = True
13             # En caso de que la anterior opción sea falsa, significa que el jugador se puede mover sin problemas
14             else:
15                 tablero_de_juego[i-2][j] = jugadores_visual[jugador] # Se mueve el jugador dos casillas
16                 puntos_de_accion[jugador] -= 1 # Se consume un PA
17                 direccion_valida = False # Se termina el ciclo while que dictamina si la dirección es válida
18             # Si al momento de moverse hay una víctima
19             elif tablero_de_juego[i-1][j] == 'Ø':
20                 # De la matriz puntos_de_interes seleccionamos un valor. Si el valor es TRUE significa que hay una persona, si el valor es FALSE significa que era una falsa alarma
21                 pdi = random.choice(puntos_de_interes)
22                 puntos_de_interes.pop(pdi) # El valor obtenido de la selección se elimina del arreglo puntos_de_interes
23                 tablero_de_juego[i-1][j] = jugadores_visual[jugador] # Se mueve el jugador una casilla
24
25             # Si pdi es True entonces...
26             if(pdi):
27                 print("\nEs una persona!!.. El equipo de rescate ha llegado y la ha salvado.")
28                 victimas_salvadas += 1 # Se agrega la víctima a la variable victimas_salvadas
29                 puntos_de_accion[jugador] = 0 # MODIFICACION: Como en esta versión del juego no es necesario llevar a la persona afuera de la casa, para balancear, si es una persona gastará todos sus PA
30             else:
31                 print("\nEra una falsa alarma!!\n")
32                 puntos_de_accion[jugador] -= 1
33                 direccion_valida = False
34             # Si todas las anteriores validaciones son falsas el jugador se puede mover sin problemas y sin eventos especiales
35             else:
36                 tablero_de_juego[i-1][j] = jugadores_visual[jugador] # Se mueve el jugador una casilla
37                 puntos_de_accion[jugador] -= 1 # Se gasta PA
38                 direccion_valida = False # La dirección fue válida
39
40     }

```

Se repite el mismo proceso para cada dirección

# Explicación {

## 14. Apagar Fuego

```

1 # !-----APAGAR_FUEGO-----
2 # Comenzamos con la ejecución para apagar fuego
3 elif(accion == 2):
4     apagar_fuego = True
5     # Comenzamos un ciclo while que se ejecutará hasta que se apague el fuego
6     while(apagar_fuego):
7         # Comprobamos si hay un fuego que apagar alrededor del jugador
8         if(tablero_de_juego[i-1][j] == '+' or tablero_de_juego[i+1][j] == '+' or tablero_de_juego[i][j-1] == '+' or tablero_de_juego[i][j+1] == '+' or ((tablero_de_juego[i-1][j] == '-' or tablero_de_juego[i-1][j] == '□') and tablero_de_juego[i-2][j] == '+') or ((tablero_de_juego[i+1][j] == '-' or tablero_de_juego[i+1][j] == '□') and tablero_de_juego[i+2][j] == '+') or ((tablero_de_juego[i][j-1] == '-' or tablero_de_juego[i][j-1] == '□') and tablero_de_juego[i][j-2] == '+') or ((tablero_de_juego[i][j+1] == '-' or tablero_de_juego[i][j+1] == '□') and tablero_de_juego[i][j+2] == '+')):
9             # Le preguntamos al usuario en qué dirección quiere apagar el fuego
10            direccion_fuego = input("Indica en qué dirección quieres apagar el fuego (arriba - abajo - derecha - izquierda)\nDirección: ")
11
12            # En la dirección en la que ingreso el usuario apagamos el fuego
13            if(direccion_fuego == "arriba"):
14                if(tablero_de_juego[i-1][j] == '+'):
15                    tablero_de_juego[i-1][j] = '-'.# Se sustituye el fuego por una casilla de humo
16                # Por la estructura de nuestro tablero, tenemos que hacer comprobaciones de más
17                elif((tablero_de_juego[i-1][j] == '-' or tablero_de_juego[i-1][j] == '□') and tablero_de_juego[i-2][j] == '+'):
18                    tablero_de_juego[i-2][j] = '-'.# Se sustituye el fuego por una casilla de humo
19                    puntos_de_accion[jugador] -= 1.# Se gasta el PA por apagar fuego
20                    apagar_fuego = False.# Si se ha apagado un fuego correctamente, se deja de repetir el ciclo while
21                # SE REPITE EL MISMO PROCESO CON CADA DIRECCIÓN
22                elif(direccion_fuego == "abajo"):
23                    if(tablero_de_juego[i+1][j] == '+'):
24                        tablero_de_juego[i+1][j] = '-'.
25                    elif((tablero_de_juego[i+1][j] == '-' or tablero_de_juego[i+1][j] == '□') and tablero_de_juego[i+2][j] == '+'):
26                        tablero_de_juego[i+2][j] = '-'.
27                        puntos_de_accion[jugador] -= 1
28                        apagar_fuego = False
29                elif(direccion_fuego == "izquierda"):
30                    if(tablero_de_juego[i][j-1] == '+'):
31                        tablero_de_juego[i][j-1] = '-'.
32                    elif((tablero_de_juego[i][j-1] == '-' or tablero_de_juego[i][j-1] == '□') and tablero_de_juego[i][j-2] == '+'):
33                        tablero_de_juego[i][j-2] = '-'.
34                        puntos_de_accion[jugador] -= 1
35                        apagar_fuego = False
36                elif(direccion_fuego == "derecha"):
37                    if(tablero_de_juego[i][j+1] == '+'):
38                        tablero_de_juego[i][j+1] = '-'.
39                    elif((tablero_de_juego[i][j+1] == '-' or tablero_de_juego[i][j+1] == '□') and tablero_de_juego[i][j+2] == '+'):
40                        tablero_de_juego[i][j+2] = '-'.
41                        puntos_de_accion[jugador] -= 1
42                        apagar_fuego = False
43                # En caso de que el jugador no ingrese una dirección válida
44                else:
45                    print("\nDirección Inválida")
46                    apagar_fuego = True
47                # En caso de que no se encuentre ningún fuego alrededor vuelve al menú de acciones
48                else:
49                    print("\nNo hay ningún fuego alrededor!!!")
50                    apagar_fuego = False

```

# Explicación {

## 15. Apagar Humo

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 } }

# ! ----- APAGAR HUMO -----
# Comenzamos con la lógica para apagar el humo
elif(accion == 3):
    apagar_humo = True
    # Comenzamos un ciclo while que se ejecutará hasta que se apague el humo
    while(apagar_humo):
        # Comprobamos si hay un humo que apagar alrededor del jugador
        if(tablero_de_juego[i-1][j] == '❖' or tablero_de_juego[i+1][j] == '❖' or tablero_de_juego[i][j-1] == '❖' or tablero_de_juego[i][j+1] == '❖' or ((tablero_de_juego[i-1][j] == '□' or tablero_de_juego[i-1][j] == '□') and tablero_de_juego[i-2][j] == '❖') or ((tablero_de_juego[i+1][j] == '□' or tablero_de_juego[i+2][j] == '❖') or ((tablero_de_juego[i][j+1] == '□' or tablero_de_juego[i][j+1] == '□') and tablero_de_juego[i][j+2] == '❖') or ((tablero_de_juego[i][j-1] == '□' or tablero_de_juego[i][j-1] == '□') and tablero_de_juego[i][j-2] == '❖')):
            # Le preguntamos al usuario en qué dirección quiere apagar el fuego
            direccion_humo = input("Indica en qué dirección quieres apagar el humo (arriba - abajo - derecha - izquierda)\nDirección: ")

            # En la dirección en la que ingresó el usuario apagamos el humo
            if(direccion_humo == "arriba"):
                if(tablero_de_juego[i-1][j] == '❖'):
                    tablero_de_juego[i-1][j] = '□' # Se sustituye el fuego por una casilla vacía
                    # Por la estructura de nuestro tablero, tenemos que hacer comprobaciones de más
                    elif((tablero_de_juego[i-1][j] == '□' or tablero_de_juego[i-1][j] == '□') and tablero_de_juego[i-2][j] == '❖'):
                        tablero_de_juego[i-2][j] = '□' # Se sustituye el humo por una casilla vacía
                    puntos_de_accion[jugador] -= 1 # Se consume los PA por apagar humo
                    apagar_humo = False # si se ha apagado un humo correctamente, se deja de repetir el ciclo while
            # SE REPITE EL MISMO PROCESO CON CADA DIRECCIÓN
            elif(direccion_humo == "abajo"):
                if(tablero_de_juego[i+1][j] == '❖'):
                    tablero_de_juego[i+1][j] = '□'
                    elif((tablero_de_juego[i+1][j] == '□' or tablero_de_juego[i+1][j] == '□') and tablero_de_juego[i+2][j] == '❖'):
                        tablero_de_juego[i+2][j] = '□'
                    puntos_de_accion[jugador] -= 1
                    apagar_humo = False
            elif(direccion_humo == "derecha"):
                if(tablero_de_juego[i][j+1] == '❖'):
                    tablero_de_juego[i][j+1] = '□'
                    elif((tablero_de_juego[i][j+1] == '□' or tablero_de_juego[i][j+1] == '□') and tablero_de_juego[i][j+2] == '❖'):
                        tablero_de_juego[i][j+2] = '□'
                    puntos_de_accion[jugador] -= 1
                    apagar_humo = False
            elif(direccion_humo == "izquierda"):
                if(tablero_de_juego[i][j-1] == '❖'):
                    tablero_de_juego[i][j-1] = '□'
                    elif((tablero_de_juego[i][j-1] == '□' or tablero_de_juego[i][j-1] == '□') and tablero_de_juego[i][j-2] == '❖'):
                        tablero_de_juego[i][j-2] = '□'
                    puntos_de_accion[jugador] -= 1
                    apagar_humo = False
            # En caso de que no se encuentre ningún humo alrededor vuelve al menú de acciones
            else:
                print("\nDirección Inválida")
                apagar_humo = True
            # En caso de que no se encuentre ningún humo alrededor vuelve al menú de acciones
            else:
                print("\nNo hay ningún humo alrededor!!!")
                apagar_humo = False

```

# Explicación {

## 16. Abrir Puerta

```
1
2
3     # ! ----- ABRIR PUERTA -----
4     # Comenzamos con la ejecución para Abrir Puerta
5     elif(accion==4):
6         # Comprobamos si hay una puerta alrededor de nosotros
7         if(tablero_de_juego[i-1][j] == '█' or tablero_de_juego[i+1][j] == '█' or tablero_de_juego[i][j-1] == '█' or tablero_de_juego[i][j+1] == '█'):
8             # En este caso no hay necesidad de preguntarle al usuario en que dirección quiere abrir la puerta
9
9
10        # Puerta en la dirección de arriba
11        if(tablero_de_juego[i-1][j] == '█'):
12            tablero_de_juego[i-1][j] = '□' # Se modifica la casilla, haciendo que la puerta ahora esté abierta
13            puntos_de_accion[jugador] -= 1 # Se consume el PA por abrir la puerta
14        # Puerta en la dirección de abajo
15        elif(tablero_de_juego[i+1][j] == '█'):
16            tablero_de_juego[i+1][j] = '□'
17            puntos_de_accion[jugador] -= 1
18        # Puerta en la dirección de izquierda
19        elif(tablero_de_juego[i][j-1] == '█'):
20            tablero_de_juego[i][j-1] = '□'
21            puntos_de_accion[jugador] -= 1
22        # Puerta en la dirección de derecha
23        elif(tablero_de_juego[i][j+1] == '█'):
24            tablero_de_juego[i][j+1] = '□'
25            puntos_de_accion[jugador] -= 1
26        # Se muestra que no hay una puerta alrededor y vuelve al menú de acciones
27    else:
28        print("\nNo hay ninguna puerta cerrada alrededor!!")
```

# Explicación {

## 17. Cerrar Puerta

```
1 # Comenzamos con la ejecución para Cerrar Puerta
2 elif(accion==5):
3     # Comprobamos si hay una puerta alrededor de nosotros
4     if(tablero_de_juego[i-1][j]== '□' or tablero_de_juego[i+1][j]== '□' or tablero_de_juego[i][j-1]== '□' or tablero_de_juego[i][j+1]== '□'):
5         # En este caso no hay necesidad de preguntarle al usuario en que dirección quiere cerrar la puerta
6
6     # Puerta en la dirección de arriba
7     if(tablero_de_juego[i-1][j]== '□'):
8         tablero_de_juego[i-1][j]= '■' # se modifica la casilla y se cierra la puerta
9         puntos_de_accion[jugador]-= 1 # se consume un PA por cerrar la puerta
10    # Puerta en la dirección de abajo
11    elif(tablero_de_juego[i+1][j]== '□'):
12        tablero_de_juego[i+1][j]= '■'
13        puntos_de_accion[jugador]-= 1
14    # Puerta en la dirección de izquierda
15    elif(tablero_de_juego[i][j-1]== '□'):
16        tablero_de_juego[i][j-1]= '■'
17        puntos_de_accion[jugador]-= 1
18    # Puerta en la dirección de derecha
19    elif(tablero_de_juego[i][j+1]== '□'):
20        tablero_de_juego[i][j+1]= '■'
21        puntos_de_accion[jugador]-= 1
22    # Se comprueba que no hay ninguna puerta cerrada alrededor
23    else:
24        print("\nNo hay ninguna puerta abierta alrededor!!")
25    # Si el jugador ingresa una acción invalida se le indicará
26    else:
27        print("\nAcción Inválida")
```

# Explicación {

## 18. Preparar el tablero - Colocar Humo (Explosiones)

```
1 # ! ----- PREPARAR EL TABLERO PARA EL SIGUIENTE TURNO -----
2 if(turno > 0): # Las explosiones tienen que ocurrir despues del turno 1
3     # Una vez se ha terminado el turno de todos los jugadores, tenemos que preparar el tablero para el siguiente turno
4     # 1. Comenzamos con las explosiones
5     colocar_humo = 0
6     # Usamos un ciclo while para que se coloquen los tres humos
7     while(colocar_humo != 3):
8         i, j = lanzar_dados() # Obtenemos una casilla donde caera el humo
9         print(f"Humo en las casillas: {i, j}") # Mostramos en que casilla cayó
10        print(tablero_de_juego[i][j])
11        # ! ----- EXPLOSION -----
12        if(tablero_de_juego[i][j] == '+'): # El humo ha caido en una casilla donde hay fuego
13            print("Explosion!!!")
14            colocar_humo += 1 # Se coloca un humo
15            # Comenzamos con la explosión, para cada lado
16            for direccion in ["arriba", "derecha", "abajo", "izquierda"]:
17                n = 0 # Usamos una variable global que recorrerá cada dirección hasta que pare la explosión
18                explosion_fin = True
19                while(explosion_fin):
20                    # ! ----- EXPLOSION -----
21
```

```
22 }
```

# Explicación {

## 19. Preparar el tablero - Colocar Humo (Explosiones)

```

#----- EXPLOSION ARRIBA -----
if(direccion == "arriba"):
    # Variable que nos ayudará a recorrer hacia arriba
    n += -1
    if i + n < 0: # Verificar que la explosión no salga de la matriz
        break
    # Comienzan los cambios por las explosiones
    if(tablero_de_juego[i + n][j] == '◻'): # Si hay una casilla vacía...
        tablero_de_juego[i + n][j] = '◆' # Se convierte en fuego
        explosion_fin = True # La explosión continua
    elif(tablero_de_juego[i + n][j] == '---'): # Si hay una pared completa...
        tablero_de_juego[i + n][j] = '-' # Se romperá a la mitad
        cubos_dmg += 1 # Se añade un cubo de daño
        explosion_fin = False # La explosión termina
    elif(tablero_de_juego[i + n][j] == '--'): # Si hay una pared media rota...
        tablero_de_juego[i + n][j] = '-' # Se romperá completamente
        cubos_dmg += 1 # Se añade un cubo de daño
        explosion_fin = False # La explosión termina
    elif(tablero_de_juego[i + n][j] == 'Ø'): # Si hay un PDI este perderá
        tablero_de_juego[i + n][j] = '◆' # Se convierte en fuego MURIÓ :(
        # Seleccionamos un valor de puntos_de_interes para eliminarlo
        pdi = random.choice(puntos_de_interes)
        puntos_de_interes.pop(pdi)
        # Si pdi es True entonces...
        if(pdi): victimas_perdidas += 1 # Perdimos a la víctima :(
        else: pass # No pasa nada, era una falsa alarma :)
        explosion_fin = True # La explosión continua
    elif(tablero_de_juego[i + n][j] == '◆'): # Si hay una casilla con humo
        tablero_de_juego[i + n][j] = '◆' # Esta se convierte en fuego
        explosion_fin = True # La explosión continua
    elif(tablero_de_juego[i + n][j] in jugadores_visual): # Si la explosión se expande hasta que le llega a un jugador
        # Identificamos que jugador es afectado por la explosión
        jugador_encontrado = jugadores_visual.index(tablero_de_juego[i + n][j])
        tablero_de_juego[i + n][j] = '◆' # La casilla se convierte en fuego
        # Tomamos una posición inicial del tablero y mandamos al jugador a ella
        posicion = random.choice(casillas_iniciales)
        partes = posicion.split('.')
        k = int(partes[0])
        l = int(partes[1])
        tablero_de_juego[k][l] = jugadores_visual[jugador_encontrado]
        explosion_fin = True # La explosión continua
    elif(tablero_de_juego[i + n][j] == '■'): # Si hay una puerta cerrada...
        tablero_de_juego[i + n][j] = '◻' # Esta se abre
        explosion_fin = False # La explosión termina
    elif(tablero_de_juego[i + n][j] == ' '): # Si hay un espacio vacío...
        tablero_de_juego[i + n][j] = '◆' # Se queda así
        explosion_fin = True # La explosión continua

```

Se repite el mismo proceso para cada dirección

# Explicación {

## 20. Preparar el tablero - Colocar Humo

```
1 # Verificamos las otras posibilidades
2 elif(tablero_de_juego[i-1][j] == '♦' or tablero_de_juego[i+1][j] == '♦' or tablero_de_juego[i][j-1] == '♦' or tablero_de_juego[i][j+1] == '♦'):
3     tablero_de_juego[i][j] = '♦' # Si en alguna casilla adyacente hay fuego, entonces se convierte automáticamente en fuego
4     colocar_humo += 1 # Se añade un humo
5 elif(tablero_de_juego[i][j] == '□'): # Si cae en una casilla vacía....
6     tablero_de_juego[i][j] = '♦' # Esta tendrá humo
7     colocar_humo += 1 # Se añade un humo
8 elif(tablero_de_juego[i][j] in jugadores_visual): # Si en la casilla hay algún jugador, este regresará al inicio
9     # Obtener el índice del jugador en la lista
10    jugador_encontrado = jugadores_visual.index(tablero_de_juego[i][j])
11    # Seleccionar una casilla inicial y el jugador lo mandaremos ahí
12    posicion = random.choice(casillas_iniciales)
13    partes = posicion.split('.')
14    k = int(partes[0])
15    l = int(partes[1])
16    tablero_de_juego[k][l] = jugadores_visual[jugador_encontrado]
17    tablero_de_juego[i][j] = '♦' # Se coloca un humo en la casilla del jugador
18    colocar_humo += 1 # Se añade un humo
```

12 }

13 }

14 }

# Explicación {

## 21. Preparar el tablero - Colocar PDI

```
1 # 2. Segundo verificamos que en el tablero haya tres puntos de interés
2 pdi_en_tablero = 0 # Inicializamos en cero para saber cuantos pdi hay.
3 for fila in tablero_de_juego: # Usamos un ciclo for para recorrer toda la matriz y buscar los pdi
4     for valor in fila:
5         if valor == 'Ø':
6             pdi_en_tablero += 1
7 # En caso de que los pdi no hayan sido tres, usamos un ciclo while para colocar en nuestro tablero los pdi que faltan
8 while(pdi_en_tablero != 3):
9     # Validamos que la posición en donde se colocará el pdi sea válida
10    lanzamiento_valido = True
11    while(lanzamiento_valido):
12        i, j = lanzar_dados() # Se lanzan los dados para saber en qué casilla caerá el PDI
13        print(f"Punto de Interés en la casilla: {i}, {j}")
14        # En caso de que la posición esté ocupada por un valor en el que no puede estar
15        if(tablero_de_juego[i][j] in ['-', '|', '||', '| ', '|Ø', '|Ø ']):
16            lanzamiento_valido = True # Se vuelve a lanzar
17        # En caso de que la posición esté ocupada por un jugador se desvelará automáticamente
18        elif(tablero_de_juego[i][j] in jugadores_visual):
19            # Seleccionaremos un valor de punto_de_interes
20            pdi = random.choice(puntos_de_interes)
21            # Lo eliminamos de la matriz
22            puntos_de_interes.pop(pdi)
23
24            # Si pdi es True entonces...
25            if(pdi):
26                print("\nHaz encontrado una víctima escondida. El equipo de rescate la ha salvado!!!")
27                victimas_salvadas += 1 # Se añade la víctima a victimas salvadas
28                lanzamiento_valido = True # Se vuelve a tirar el dado porque se descubrió a la víctima
29            else:
30                print("Era una falsa alarma")
31                lanzamiento_valido = True # Se vuelve a tirar el dado porque se descubrió a la víctima
32            # Si todo lo anterior es falso, se coloca el PDI
33        else:
34            tablero_de_juego[i][j] = 'Ø' # Se coloca en la casilla
35            lanzamiento_valido = False # El lanzamiento fue válido
36            pdi_en_tablero += 1 # Se coloca un PDI en el tablero
37
38 }
```

# Explicación {

## 22. Comprobación del juego

```
1 # ! ----- COMPROBACION DEL JUEGO -----
2 # Si hay 24 cubos de daño perdemos
3 if(cubos_dmg >= 24):
4     print("Oh no, el edificio se ha derrumbado!!!")
5     no_perder = False
6 # Si las victimas perdidas es mayor o igual a 6 perdemos
7 elif(victimas_perdidas >= 6):
8     print("Oh no, hemos perdido demasiadas victimas!!! No cumplimos con nuestro deber...")
9     no_perder = False
10 # Si salvamos a 7 victimas ganamos!!
11 elif(victimas_salvadas >= 7):
12     print("VICTORIA!!!! Hemos salvado a la mayoría de victimas.")
13     no_perder = False
14 print("-".center(75, "-"))
15 imprimir_tablero(tablero_de_juego)
16 turno += 1
```

# }

# Explicación {

## 22. Comprobación del juego

```
1 # ! ----- COMPROBACION DEL JUEGO -----
2 # Si hay 24 cubos de daño perdemos
3 if(cubos_dmg >= 24):
4     print("Oh no, el edificio se ha derrumbado!!!")
5     no_perder = False
6 # Si las victimas perdidas es mayor o igual a 6 perdemos
7 elif(victimas_perdidas >= 6):
8     print("Oh no, hemos perdido demasiadas victimas!!! No cumplimos con nuestro deber...")
9     no_perder = False
10 # Si salvamos a 7 victimas ganamos!!
11 elif(victimas_salvadas >= 7):
12     print("VICTORIA!!!! Hemos salvado a la mayoría de victimas.")
13     no_perder = False
14 print("-".center(75, "-"))
15 imprimir_tablero(tablero_de_juego)
16 turno += 1
```

# }

# Explicación {

## 23. Ultimas tres opciones

```
1 # ! ----- MOSTRAMOS EL ESTADO ACTUAL DEL TABLERO -----
2 elif(opcion == 6):
3     print('\nTablero de Juego: ')
4     imprimir_tablero(tablero_de_juego)
5 # ! ----- EL JUGADOR SE HA RENDIDO -----
6 elif(opcion == 7):
7     print("\nSer bombero no es trabajo para todos. Hasta la próxima!!! ")
8     no_perder = False
9 # ! ----- MOSTRAMOS LAS ESTADISTICAS DE LA PARTIDA -----
10 elif(opcion == 8):
11     estadisticas = [["Víctimas Perdida", victimas_perdidas], ["Víctima Salvadas", victimas_salvadas], ["Cubos de Daño", cubos_dmg]]
12     print("\n")
13     print(tabulate(estadisticas, tablefmt="rounded_grid", headers=["ESTADISTICA", "CANTIDAD"]))
```

}

# Explicación {

24. Mandamos a llamar nuestro código principal

```
# !-----SE EJECUTA NUESTRO CODIGO-----  
main()
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13 }  
14 }

Porteo hacia otros lenguajes {

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14 }



```
1 Nivel de Complejidad {  
2 |  
3 | Durante el desarrollo del algoritmo  
4 | hubo varios desafíos a los cuales nos  
5 | enfrentamos.  
6 |  
7 | El primero fue cómo crear el tablero de  
8 | juego y las paredes y que este funcione  
9 | correctamente.  
10 |  
11 | El segundo fue crear las explosiones y  
12 | que estas surgieran efecto.  
13 |  
14 }
```



1  
2  
3 Gracias por su  
4 Atención {  
5  
6  
7  
8  
9  
10  
11  
12  
13 }  
14

```
portada.html                                         contenido.html

1   PROYECTO FINAL ‘Algoritmos’ {  
2  
3       [Rescate! Juego de Mesa]  
4  
5           <li>  
6               Integrantes:  
7                   • Del Carmen Arteaga Jorge Eduardo  
8                   • Herrera Correa Luis Axel  
9                   • López Ordoñes David Esaú  
10                  • Luna Gómez Guadalupe de Jesús  
11                  • Rodríguez Medina Dennis Rogelio  
12           </li>  
13     }  
14
```

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO || FACULTAD DE INFORMÁTICA