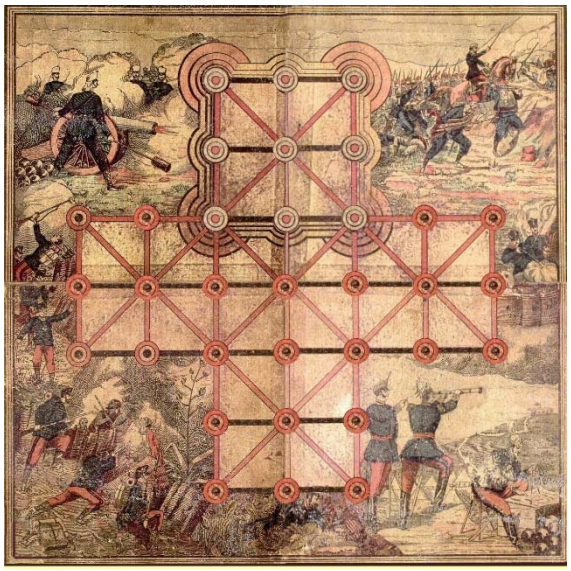
# Paradigmas de Programación

# Práctica I - Curso 2018/19

# Asalto (I)

## 1. Objetivo de la Práctica



El objetivo de la práctica es el desarrollo de una aplicación en lenguaje Python que implemente el famoso juego **Asalto**. Este es un juego para 2 jugadores que hizo furor a mediados del siglo XIX, y en su versión clásica consiste en una rejilla de 33 puntos en forma de cruz (5 cuadrados de 3x3 puntos, estando el cuadrado central solapado en los bordes con los otros 4). El cuadrado superior representa a un **castillo**, que debe ser defendido por 2 **oficiales** del asalto de 24 **rebeldes**.

El objetivo de los rebeldes es el de ocupar los 9 puntos del castillo o bien rodear a los 2 oficiales de forma que no puedan moverse. El objetivo de los oficiales es el de capturar rebeldes hasta que queden menos de 9 (de forma que no puedan ocupar totalmente el castillo) o bien aguantar hasta que ningún rebelde pueda moverse.

Las reglas del juego son[1]:

- Las posiciones iniciales de las fichas en el tablero están prefijadas.
- No se pueden *apilar* fichas (en todo momento cada punto contiene una o ninguna ficha)
- Se juega por turnos (comienzan los oficiales), moviendo una única ficha cada vez. Las reglas de movimiento son distintas para oficiales y rebeldes. En todo caso el movimiento siempre debe realizarse por las líneas que unen los puntos (denominamos **puntos contiguos** a los puntos vecinos que estén unidos por una única línea)
- Un rebelde sólo puede moverse a un punto contiguo que esté vacío y sólo si con ello **disminuye**[2] su distancia al castillo. Esta última restricción de distancia no se aplica si ambos puntos pertenecen al castillo. La distancia al castillo de un punto se mide por el número mínimo de líneas que le separa de algún punto que pertenezca al castillo.
- Un oficial puede moverse a cualquier punto contiguo que esté vacío (sin la restricción de distancia anterior) salvo que exista **captura obligada**.
- Un oficial puede **capturar** (eliminándolo del tablero) a un rebelde que esté en un punto contiguo saltando por encima de él, pero solo si existe una posición vacía justo después (no se puede cambiar de dirección durante el salto).

[1] En algunos textos o páginas web podéis encontrar la descripción de este juego con reglas ligeramente distintas. Las únicas reglas válidas para la práctica son la que se indican en este enunciado.

[2] Si un punto contiguo está a la misma distancia **no se puede mover a el** (salvo que ambos estén en el castillo)

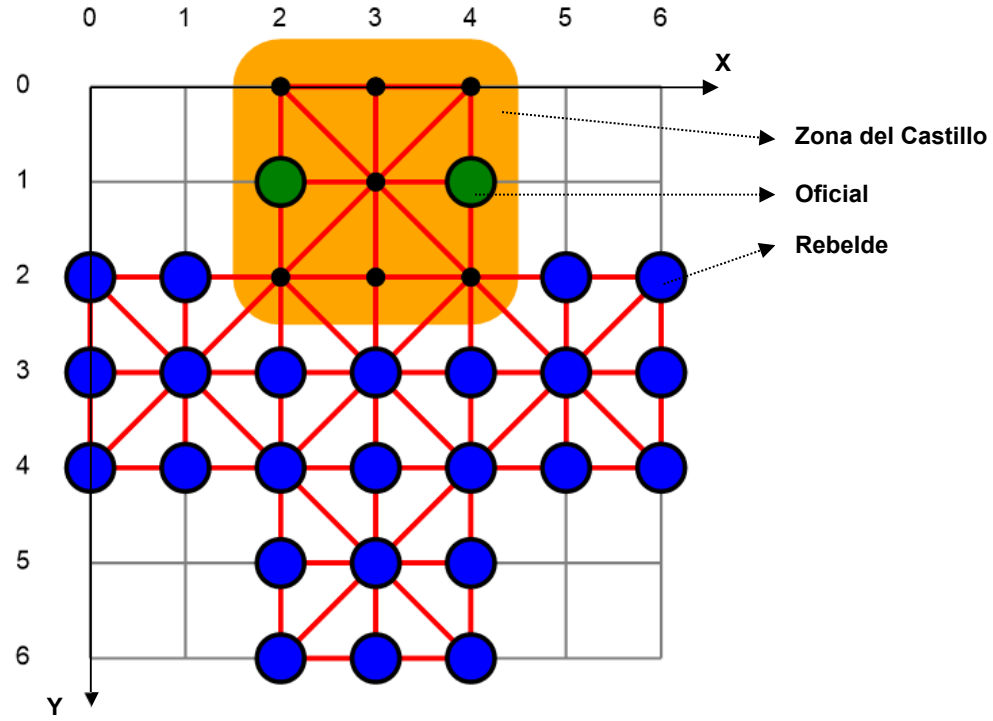
- **Captura múltiple**: Si al capturar a un rebelde el oficial acaba en un punto desde el que puede capturar a otro rebelde, puede (y debe) hacerlo en el mismo turno, volviéndose a aplicar esta norma tantas veces como existan capturas en las posiciones por donde pase el oficial.
- **Las capturas son obligatorias**: Si en el turno de los oficiales uno (o ambos) tienen posibilidad de captura, no pueden omitirla haciendo un movimiento sin captura. Si hay varias posibilidades de captura se puede elegir cualquiera de ellas.
- Si en algún punto (posterior al inicial) de una captura múltiple el oficial está en una posición donde tiene varias alternativas de captura, se escogerá una de ellas de acuerdo al procedimiento que se detallará en el apartado siguiente[3].
- Los oficiales no pueden ser capturados por los rebeldes.
- Si es el turno de los oficiales y no hay ningún movimiento válido, ganan los rebeldes.
- Si es el turno de los rebeldes y no hay ningún movimiento válido, ganan los oficiales.
- Si los rebeldes ocupan los 9 puntos del castillo, ganan los rebeldes.
- Si quedan menos de 9 rebeldes en el tablero, ganan los oficiales.

La aplicación incluirá las acciones de iniciar un nuevo juego, continuar uno guardado, salvar un juego sin terminar y modificar los ajustes de cómo se controla cada jugador (por una persona o por el ordenador).

En esta primera práctica el juego tendrá una entrada/salida **por consola**. En la segunda práctica se le dotará de una interfaz gráfica.

## 2. Descripción técnica

En el siguiente diagrama se muestra la **disposición inicial** del tablero y el sistema de coordenadas que se va a usar para representar los puntos del tablero:




[3] Esta norma no existe en el juego original, se ha incluido para simplificar el desarrollo de la práctica, ya que con ella la secuencia de capturas está completamente determinada sin ambiguedad con solo indicar el oficial y el primer rebelde que va a capturar.

En lo que sigue vamos a suponer que los **puntos** se representan como **tuplas** $(x, y)$ de Python.

Para representar el **estado del tablero** necesitamos conocer que tipo de ficha se encuentra en cada punto (hay 3 posibilidades: ninguna, rebelde u oficial, lo lógico es usar un entero), por lo que las opciones obvias son usar una lista de listas de enteros (usando las coordenadas del punto como índices y almacenando el entero que indica el tipo) o un diccionario donde las claves sean las coordenadas (tuplas) y los valores el tipo de ficha (entero). Se recomienda analizar cuidadosamente cual de estas posibilidades es mejor antes de empezar a programar.

Las siguientes funciones pueden simplificar mucho la programación de la aplicación:

- **Comprobación de si un punto pertenece o no al tablero:**

```
def pto_valido((x,y)):
    return (0 <= x < 7 and
            0 <= y < 7 and
            max(min(x, 6-x), min(y, 6-y)) > 1)
```

- **Comprobación de si un punto pertenece o no al Castillo:**

```
def pto_castillo((x,y)):
    return 2 <= x < 5 and 0 <= y < 3
```

- **Distancia de un punto al castillo** (devuelve un valor entero):

```
def dist_castillo((x,y)):
    return max(0, max(abs(x-3), abs(y-1)) - 1)
```

- **Punto al que salta un oficial cuando captura a un rebelde:**

```
def pto_captura((x_ofi,y_ofi), (x_reb,y_reb)):
    return (2*x_reb - x_ofi, 2*y_reb - y_ofi)
```

- **Recorrido de las celdas válidas accesibles desde un punto:**

```
DVEC = [(0,-1),(1,-1),(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1)]

def vecinos((x,y)):
    for (dx,dy) in DVEC:
        if not pto_valido((x+dx, y+dy)):
            continue
        if dy == 0 or dx == 0:
            yield (x+dx, y+dy)
        elif (abs(x-y) % 2 == 0 and
              pto_valido((x+dx, y)) and pto_valido((x, y+dy))):
            yield (x+dx, y+dy)
```

La función anterior es un **generador**[4], y la forma de utilizarla es mediante un bucle for. Por ejemplo, si deseamos conocer los puntos accesibles desde el (4,1), que corresponde a la posición inicial del oficial derecho, escribiríamos:

```
for pto in vecinos((4,1)):
    print(pto)
```

→

```
(4,0)
(4,2)
(3,1)
```

Otra posibilidad es convertir el iterador en una lista usando **list**:

```
list(vecinos((4,1)))
```

→

```
[(4,0),(4,2),(3,1)]
```

Puede ser muy conveniente el modificar la función *vecinos* para que solo nos devuelva los puntos que almacenan un tipo de ficha concreto (indicado por parámetro), ya que en el código nos vamos a encontrar muchas veces con la necesidad de conocer las celdas vacías contiguas a una ficha, los rebeldes contiguos a un oficial, etc.

## 2.1 Estructura de la aplicación

La aplicación comienza mostrando un menú similar a éste:

```
------------------ ASALTO -----------------------
- Practica de Paradigmas de Programacion 2018-19 -
-------------------------------------------------

  1. NUEVA PARTIDA
  2. CARGAR PARTIDA
  3. SALVAR PARTIDA
  4. CONTROL OFICIALES: [HUMANO]  AZAR
  5. CONTROL REBELDES:  [HUMANO]  AZAR   BASICO
  6. SALIR

Indique opción: _
```

Se solicita al usuario que introduzca el número de la opción deseada. Salvo la primera opción, las opciones restantes tienen un procesado muy sencillo:

- **Cargar Partida:** Se pide al usuario que introduzca el nombre de un fichero de texto que contenga las jugadas de una partida sin finalizar, se lee el fichero, se calcula el tablero resultante de aplicar esas jugadas y se pasa a la pantalla de juego (igual que en opción 'Nueva Partida' sólo que en esa opción se comienza con el tablero inicial).

  Se puede suponer que el fichero existe y es correcto, no es necesario crear código que trate esos casos.

- **Salvar partida**: Se pide al usuario el nombre de un fichero de texto y se escriben en él las jugadas de la última partida jugada en esa sesión. Si no se ha jugado ninguna partida todavía se avisa al usuario (y no se realiza la acción).

[4] Los generadores (una clase restringida de **corutinas**) se estudian brevemente en el tema 2 (Paradigma Imperativo) pags. 35-38. Si tiene interés también puede examinar el tutorial oficial de Python secciones 9.8 a 9.10

- **Salir**: Se termina la aplicación.
- **Opciones de control de oficiales y rebeldes**: Estos aparatados permiten elegir la forma en que se van a obtener las jugadas en cada turno: La opción [Humano] indica que se pediran al usuario, la opción [Azar] que se escogerá una jugada al azar de entre las jugadas válidas y la opción [Basico], disponible sólo para los rebeldes, que se aplicará el algoritmo básico (explicado posteriormente) para su elección.

Las opciones de control sólo sirven para dar valor a los parámetros que usarán cuando se pase a la pantalla de juego. Cuando el usuario introduzca el número de esa opción, se asigna como valor de control el **siguiente** al seleccionado y se vuelve a mostrar el menú, es decir no se elige directamente el valor, sino que se van seleccionando cíclicamente todas las posibilidades.

Ejemplo:

```
------------------ ASALTO ------------------------
- Practica de Paradigmas de Programacion 2018-19 -
--------------------------------------------------

  1. NUEVA PARTIDA
  2. CARGAR PARTIDA
  3. SALVAR PARTIDA
  4. CONTROL OFICIALES: [HUMANO]  AZAR
  5. CONTROL REBELDES:   HUMANO   AZAR  [BASICO]
  6. SALIR

Indique opción: 5
```

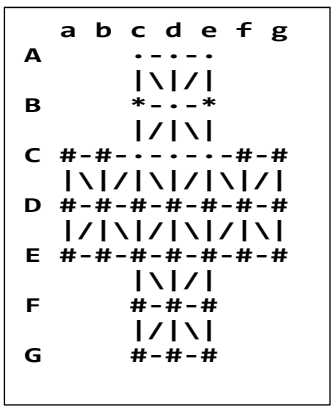
→

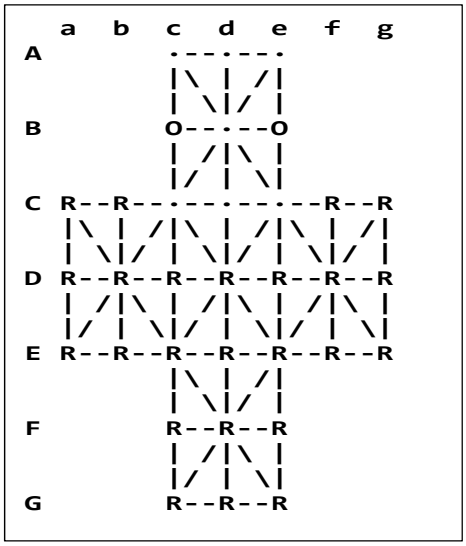
```
------------------ ASALTO ------------------------
- Practica de Paradigmas de Programacion 2018-19 -
--------------------------------------------------

  1. NUEVA PARTIDA
  2. CARGAR PARTIDA
  3. SALVAR PARTIDA
  4. CONTROL OFICIALES: [HUMANO]  AZAR
  5. CONTROL REBELDES:  [HUMANO]  AZAR  BASICO
  6. SALIR

Indique opción: _
```

## 2.2 Pantalla de Juego

Cuando el usuario selecciona “Nueva Partida” o “Cargar Partida” la aplicación pasa en la pantalla de juego, que consiste en repetir las siguiente secuencia de acciones:

1. Mostrar una representación del estado del tablero en pantalla. Las coordenadas de los puntos se muestran con letras (mayúsculas para las filas y minúsculas para las columnas). Es necesario mostrar o sugerir las líneas que conectan los puntos[5]. Ejemplos:




2. Calcular la **lista de todas las jugadas válidas disponibles** para las fichas de ese turno. Este es el apartado más importante y complejo de la aplicación, y se tratará en detalle en la sección 2.3

[5] En python el carácter ‘\’ tiene un significado especial, hay que especificarlo de esta forma: ‘\\’

3. Comprobar si se cumple alguna de las **condiciones de finalización** de partida:

    a) Bloqueo de las fichas del jugador de ese turno (lista de jugadas vacía)

    b) Todos los puntos del castillo están ocupados por rebeldes

    c) Quedan menos de 9 rebeldes

    Si se da alguna de las condiciones anteriores se muestra un mensaje informando al usuario y se termina la partida volviendo a mostrar el menú principal.

4. Se muestra por pantalla a quien corresponde el turno actual y el listado de jugadas posibles.

5. **Selección de la jugada a realizar**: Dependiendo del turno actual y de la opción de control elegida para ese tipo de fichas, se realiza lo siguiente:

    - Control **"Humano"**: Se pide al usuario que introduzca la jugada, en el formato "Punto de origen"-"Punto de destino", donde los puntos se indican mediante 2 carecteres, el primero la fila (letra mayúscula) y el segundo la columna (letra minúscula).

      Se busca en la lista de jugadas si existe alguna en la que coincidan sus dos primeros puntos[6], si es el caso se devuelve esa jugada y en caso contrario se informa al usuario de que la jugada es erronea y se vuelve a pedir que la introduzca.

      Si el usuario introduce la cadena "FIN" entonces no se realiza ninguna jugada y se sale al menú principal (se termina la partida, que puede ser salvada si en el menú principal el usuario elige "Salvar Partida").
    - Control "**Azar**": Se devuelve una jugada escogida al azar de la lista.
    - Control "**Básico**": Se utiliza el algoritmo descrito en la sección 2.4

6. Como último paso se actualiza el tablero con la jugada seleccionada, se cambia de turno y se añade la jugada al historial (la lista de jugadas realizadas, para que puedan salvarse en fichero si nos salimos de la partida y seleccionamos la opción "Salvar partida" en el menú principal).

## 2.3 Obtención de la lista de jugadas

Si es el turno de los rebeldes la lista se genera recorriendo las posiciones de todos los rebeldes del tablero y para cada uno de ellos se recorren las posiciones vecinas (se puede usar la función *vecinos* proporcionada al principio de éste apartado), cualquier posición vecina que esté vacía se comprueba si es un posible movimiento para ese rebelde (tiene que estar a una distancia menor del castillo o bien ambos puntos pertenecer al castillo) y si es el caso se incluye la jugada correspondiente.

Si es el turno de los oficiales la generación se divide en dos fases, en la primera se buscan e incluyen las jugadas que sean capturas y solo si no hay ninguna captura es cuando se inicia la segunda fase de incluir las jugadas que sean movimientos.

En la primera fase se recorren los oficiales, para cada uno se recorren sus posiciones vecinas que contengan rebeldes y se comprueba si se puede capturar (si el punto devuelto por *pto_captura* pertenece al tablero y contiene una posición vacía). Para cada captura es necesario **completar** la

[6] Una jugada **puede constar de más de dos puntos** si es el caso de una captura múltiple en que se capturan más de dos piezas. En esos casos está garantizado que los dos primeros puntos bastan para distinguir esa captura de cualquier otra que se produzca en esa misma jugada.

jugada, lo que supone **simular** la captura[7] (quitando al rebelde y moviendo al oficial) y comprobar si en la nueva posición el oficial tiene nuevas oportunidades de captura[8], y así sucesivamente. Por supuesto, al final de todo este proceso de completado de la jugada se debe **restaurar el tablero** al estado anterior.

La segunda fase (movimientos posibles de los oficiales si no existen capturas) es similar a la obtención de movimientos de los rebeldes descrita anteriormente, salvo que los oficiales no tienen la restricción de tener que acercarse al castillo.

## 2.4 Dotación de inteligencia a los rebeldes

El algoritmo que se propone para dotar de un poco de consistencia al movimiento de los rebeldes es el siguiente: Se recorre la lista de posibles jugadas de los rebeldes, se calcula una puntuación para cada jugada y se escoje la jugada **de mejor puntuación** o bien, si hay varias con la misma puntuación, se escoge al azar alguna de ellas.

La fórmula para calcular la puntuación es $\boxed{ncas - 2 \cdot ncap}$, donde *ncas* vale 1 si la jugada hace que el rebelde pase a ocupar una posición del castillo (no estando antes en él) y 0 en caso contrario, y *ncap* es el número máximo de capturas que pueden conseguir los oficiales en el siguiente turno.

Para calcular *ncap* se simula la jugada del rebelde, se obtiene la lista de jugadas de los oficiales, se recorre esa lista para detectar la jugada de los oficiales que consigue el máximo número de capturas, y se deshace el efecto de la jugada.

La justificación de este método es que los rebeldes deben realizar la jugada que menos capturas (idealmente 0) proporcione a los oficiales en el turno siguiente, y que a igualdad de capturas son mejores las jugadas que consiguen tener un rebelde más en el castillo.

## 2.5 Ejemplo de desarrollo de un juego

```
------------------ ASALTO ------------------------
- Practica de Paradigmas de Programacion 2018-19 -
--------------------------------------------------

  1. NUEVA PARTIDA
  2. CARGAR PARTIDA
  3. SALVAR PARTIDA
  4. CONTROL OFICIALES: [HUMANO]  AZAR
  5. CONTROL REBELDES:  [HUMANO]  AZAR  BASICO
  6. SALIR

Indique opción: 1
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     O-·-O
      |/|\|
C R-R-·-·-·-R-R
  |\|/|\|/|\|/|
D R-R-R-R-R-R-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: OFICIALES.
Jugadas posibles: Bc-Ac, Bc-Bd, Bc-Cc,
Be-Ae, Be-Ce, Be-Bd
Introduzca jugada o FIN: Bc-Cc
```

[7] Este paso (simular la captura) es necesario porque es posible, en una captura múltiple, que al desaparecer alguno de los rebeldes "capturados" esto genere un hueco que pueda posibilitar una captura posterior dentro de la misma jugada.
[8] Si existen varios rebeldes que pueden ser capturados, se elige capturar **al primero que sea devuelto por la función *vecinos*.**

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-·-O
      |/|\|
C R-R-O-·-·-R-R
  |\|/|\|/|\|/|
D R-R-R-R-R-R-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: REBELDES. Jugadas posibles:
Dd-Cd, Cf-Ce, Df-Ce, De-Ce, Dd-Ce
Introduzca jugada o FIN: Dd-Cd
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-·-O
      |/|\|
C R-R-O-R-·-R-R
  |\|/|\|/|\|/|
D R-R-R-·-R-R-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: OFICIALES. Jugadas posibles:
Cc-Ce
Introduzca jugada o FIN: Cc-Ce
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-·-O
      |/|\|
C R-R-·-·-O-R-R
  |\|/|\|/|\|/|
D R-R-R-·-R-R-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: REBELDES. Jugadas posibles:
Ee-Dd, Ed-Dd, Ec-Dd, Dc-Cc, Db-Cc, Cb-Cc
Introduzca jugada o FIN: Db-Cc
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-·-O
      |/|\|
C R-R-R-·-O-R-R
  |\|/|\|/|\|/|
D R-·-R-·-R-R-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: OFICIALES. Jugadas posibles:
Be-Ae, Be-Bd, Ce-Dd, Ce-Cd, Ce-Bd
Introduzca jugada o FIN: Ce-Bd
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-O-O
      |/|\|
C R-R-R-·-·-R-R
  |\|/|\|/|\|/|
D R-·-R-·-R-R-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: REBELDES. Jugadas posibles:
Ec-Db, Eb-Db, Ea-Db, Da-Db, Ca-Db, Cc-
Bc, Ee-Dd, Ed-Dd, Ec-Dd, Cc-Cd, Cf-Ce,
Df-Ce, De-Ce
Introduzca jugada o FIN: Df-Ce
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-O-O
      |/|\|
C R-R-R-·-R-R-R
  |\|/|\|/|\|/|
D R-·-R-·-R-·-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: OFICIALES. Jugadas posibles:
Bd-Df-Dd-Db-Bd, Bd-Db-Dd-Df-Bd
Introduzca jugada o FIN: Bd-Db
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-O-O
      |/|\|
C R-R-·-·-·-R-R
  |\|/|\|/|\|/|
D R-·-·-·-·-·-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: REBELDES. Jugadas posibles:
Ec-Db, Eb-Db, Ea-Db, Da-Db, Ca-Db, Ee-Dd,
Ed-Dd, Ec-Dd, Cb-Cc, Ec-Dc, Cf-Ce, Cg-Df,
Dg-Df, Eg-Df, Ef-Df, Ee-Df, Ee-De
Introduzca jugada o FIN: FIN
```

```
------------------ ASALTO ------------------------
- Practica de Paradigmas de Programacion 2018-19 -
--------------------------------------------------

  1. NUEVA PARTIDA
  2. CARGAR PARTIDA
  3. SALVAR PARTIDA
  4. CONTROL OFICIALES: [HUMANO]  AZAR
  5. CONTROL REBELDES:  [HUMANO]  AZAR   BASICO
  6. SALIR

Indique opción: 3

Nombre del fichero: ejemplo.txt
```

```
------------------ ASALTO ------------------------
- Practica de Paradigmas de Programacion 2018-19 -
--------------------------------------------------

  1. NUEVA PARTIDA
  2. CARGAR PARTIDA
  3. SALVAR PARTIDA
  4. CONTROL OFICIALES: [HUMANO]  AZAR
  5. CONTROL REBELDES:  [HUMANO]  AZAR   BASICO
  6. SALIR

Indique opción: 2

Nombre del fichero: ejemplo.txt
```

```
  a b c d e f g
A     ·-·-·
      |\|/|
B     ·-O-O
      |/|\|
C R-R-·-·-·-R-R
  |\|/|\|/|\|/|
D R-·-·-·-·-·-R
  |/|\|/|\|/|\|
E R-R-R-R-R-R-R
      |\|/|
F     R-R-R
      |/|\|
G     R-R-R

Turno: REBELDES. Jugadas posibles:
Ec-Db, Eb-Db, Ea-Db, Da-Db, Ca-Db, Ee-Dd,
Ed-Dd, Ec-Dd, Cb-Cc, Ec-Dc, Cf-Ce, Cg-Df,
Dg-Df, Eg-Df, Ef-Df, Ee-Df, Ee-De
Introduzca jugada o FIN: _
```

Contenido del fichero ejemplo.txt:

```
Bc-Cc
Dd-Cd
Cc-Ce
Db-Cc
Ce-Bd
Df-Ce
Bd-Db-Dd-Df-Bd
```

## 3. Presentación y Evaluación de la práctica

La práctica se realizará **por parejas** (para otras alternativas consulte con su profesor de prácticas) y su evaluación se divide en dos etapas:

1. Presentación electrónica del fichero(s) que componen la práctica (ficheros ***.py**). Para ello se habilitará en el Aula Virtual de la E.T.S. Informática (www.inf.uva.es -> menú Aula Virtual) una tarea de subida de ficheros cuya fecha límite será el **domingo 24 de marzo a las 23:59.** Al principio de todos los ficheros debe aparecer un comentario con el nombre de quienes la han desarrollado.
2. Evaluación **presencial**, en laboratorio, ante el profesor. Se realizará en el lugar, día y hora correspondiente al horario de prácticas del subgrupo al que pertenezca durante la semana del 25 al 29 de marzo.

En el caso de realización por parejas (la situación habitual), tan sólo es necesario que uno cualquiera de ellos realice la presentación electrónica. En la evaluación, sin embargo, si es necesaria la presencia de **ambos** y la evaluación puede ser distinta para cada uno de ellos.

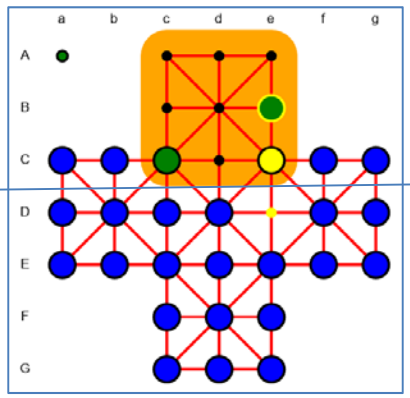
En la evaluación de la práctica se tendrá en cuenta, entre otros, los siguientes factores:

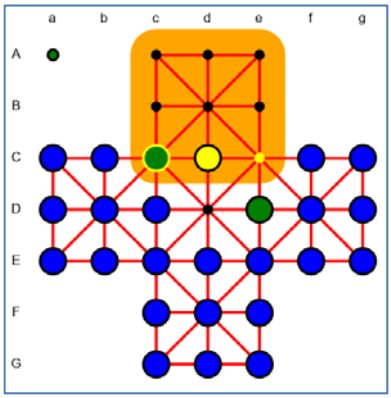
- Autoría y participación en la misma.
- La correcta resolución del problema, así como la modularidad, documentación y robustez de la solución presentada. El uso de las técnicas impartidas hasta ese momento en la asignatura tiene una influencia positiva en la evaluación.

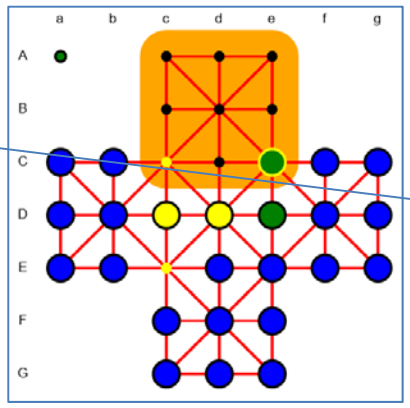
# Apéndice A: Ejemplo de partida completa

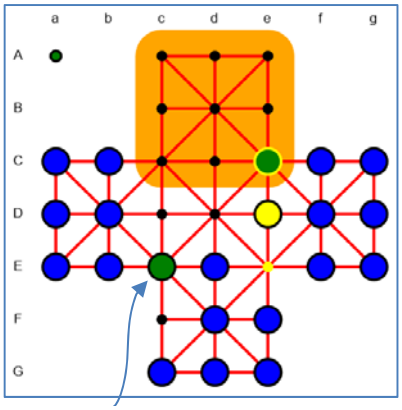
Se muestra el estado del tablero en algunas posiciones concretas:

```
Bc-Cc
De-Ce
Be-De
Dd-Cd
Cc-Ce
Ec-Dd
Ce-Ec-Cc
Fc-Ec
De-Ce
Ec-Dc
Cc-Ec
Ee-De
Ce-Ee
Df-Ce
Ec-Dd
Ce-Cd
Dd-Bd
Dg-Df
Ee-Ec
Fd-Ee
Bd-Cc
Df-Ce
Ec-Dd
Ce-Cd
Dd-Bd
Eg-Df
Bd-Be
Gc-Fc
Cc-Dc
Db-Cc
Dc-Bc
Ca-Db
Be-Ce
Fc-Ec
Ce-Eg
Cg-Df
Eg-Ce-Cg
Ec-Dc
Bc-Cc
Ee-Dd
Cc-Ca-Ec-Cc-Ee-Eg
```

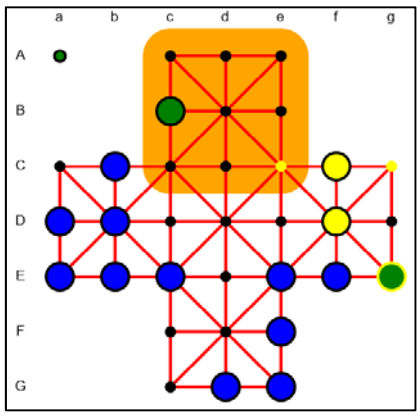


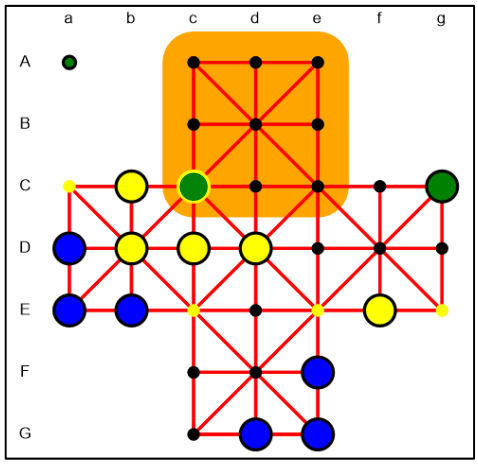


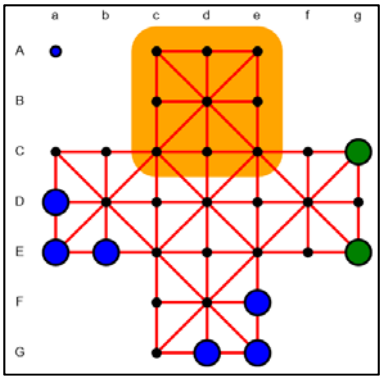




También podría capturar







La partida termina en este punto con victoria de los oficiales.