

# Juana de Arco

en el Castillo de las Montañas de Reims



Juana de Arco (también conocida como la Doncella de Orléans, o en francés, *la Pucelle*), fue una heroína militar nacida en la región de la Lorena, Francia. Desde muy jovencita se vio metida en los tumultuosos hechos guerreros de la Francia de principios del siglo XV luchando a favor del reino de Carlos VII. Entre numerosas hazañas, Juana de Arco supo dar ánimos a sus compatriotas y batalló contra los ingleses, que ocupaban en aquel momento la ciudad de Reims y alrededores. ([Wikipedia](#)).

Deberás guiar a nuestra guerrera Juana de Arco por el castillo situado de las Montañas de Reims y descifrar el enigma que según cuenta la leyenda destruirá el maleficio que ha permitido a los ingleses apoderarse de él. El castillo sirve de morada para el general Brayan (súbdito de Enrique VI, rey de Inglaterra) y sus soldados ingleses, los cuales han preparado innumerables peligros y trampas para evitar ser invadidos. ¡Cuidado! durante el juego Juana de Arco deberá comer regularmente para no morir de inanición.

En tus manos te dejamos el inicio del juego que te permitirá controlar los movimientos de esta la guerrera protagonista y guiarla por los rincones del castillo mediante las flechas de dirección del cursor, 'm' para pausar el juego para pensar por donde seguir y 'ESC' para salir.

Después de probarlo deberás acabarlo. Aquí podrás aplicar lo que has aprendido durante estos meses, modificando el código fuente del juego que te proporcionamos para cumplir con tus objetivos.

## El código base

El código base (ProjecteJuanaArco.tgz en el campus virtual) es el conjunto de clases que te proporcionamos para poder tener el juego funcionando con un mínimo de utilidades, incluida la que contiene el main. Fíjate en el estilo y documentación del código proporcionado y documenta apropiadamente tus propias clases.

## Estructuras de datos

El castillo está formado por varias plantas, cada una de ellas con un número cualquiera de salas (o habitaciones).

Cada una de las salas sobre las que se desarrolla el juego se divide en **celdas de un tamaño fijo** formando una cuadrícula de 25 x 17 celdas, sobre una celda podemos situar un muro, una puerta o nada (espacio libre de obstáculos).

La definición de cada sala se realiza mediante un fichero de texto plano que contiene determinados caracteres: 'P' puerta, 'X' muro, '0' (cero) suelo. Cada carácter corresponde a una de las celdas

de la cuadrícula. Los ficheros deben situarse en el directorio "**recursos**".

Aquí podemos ver el contenido del fichero "**h0\_0.txt**" que corresponde con una de las habitaciones del programa demostración que se incluye:

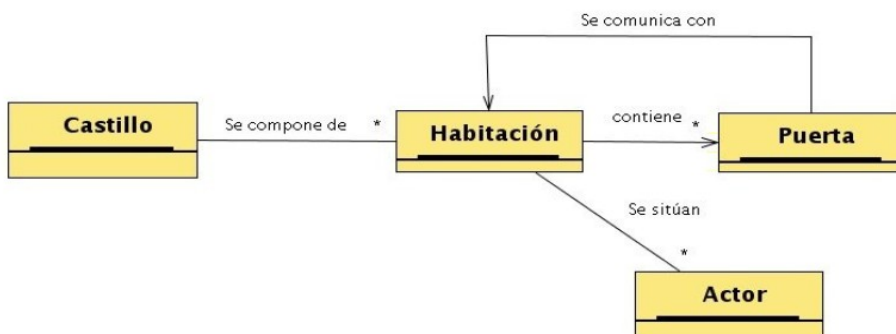
```

X,X,X,X,X,X,X,X,X,X,P,X,X,X,X,X,X,X,X,X,X,X,X,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,X,X,X,X,X,X,X,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,P
X,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,X,0,0,X
X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X

```

Podemos ver que contiene un muro periférico y 2 puertas, una situada al norte y otra al sureste. Así como un pasillo en forma de L invertida.

En cada sala (o habitación) se situarán diversos objetos como alimentos, flechas,gemas, monedas, etc. a los que llamaremos **Actores**. El siguiente diagrama muestra como se relacionan todos los conceptos de los que hemos tratado hasta ahora:



Aquí tenemos la pantalla de bienvenida al juego:







Aquí tenemos una captura en un momento del juego, correspondiente al mapa que hemos dibujado

anteriormente (h0\_0.txt) :



## Personajes

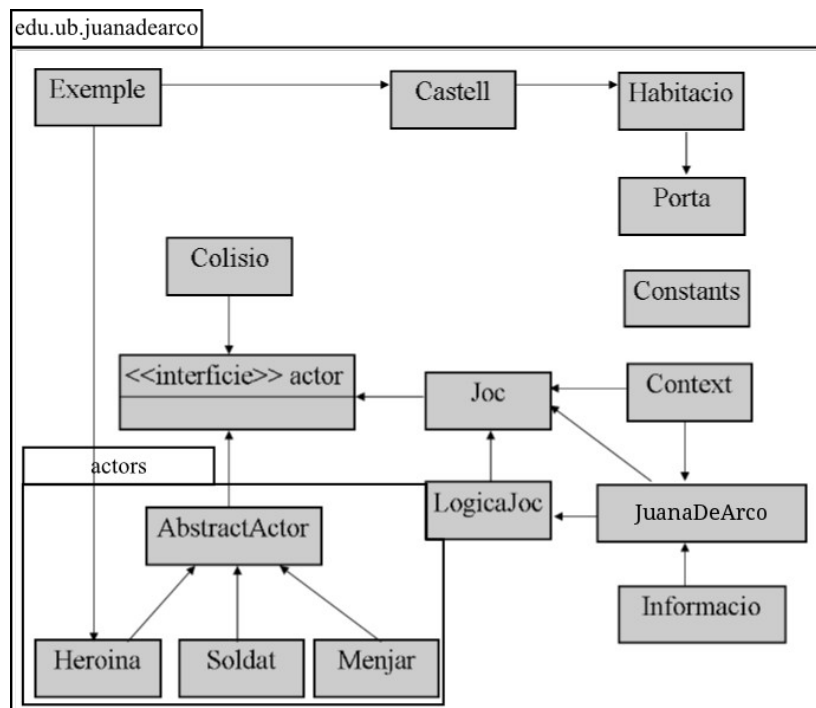
	<b>Juana de Arco:</b> Esta doncella es la guerrera protagonista de la historia. Está dispuesta a combatir con los soldados ingleses que se han apoderado del castillo de Reims, dónde teme que está secuestrado el hijo de Carlos VII.
	<b>Duque de Anjou :</b> Es el hijo predilecto del rey francés de la época. Ha sido secuestrado por guerrillas inglesas. El rey ha encargado a Juana de Arco rescatarlo sano y salvo cuanto antes.
	<b>General Brayan:</b> Temido general militar responsable del secuestro del duque. Ha raptado al hijo del rey con el fin de chantajearlo a cambio de más tierras y riquezas.
	<b>Soldados Ingleses:</b> Estos soldados son los más formados del ejército inglés y han sido contratados para vigilar y proteger el castillo de Reims. Están escondidos en cualquier parte del castillo ¡y no dudarán en atacarte si te descubren!

## Estructura del código:

El juego consta de distintas partes funcionales:

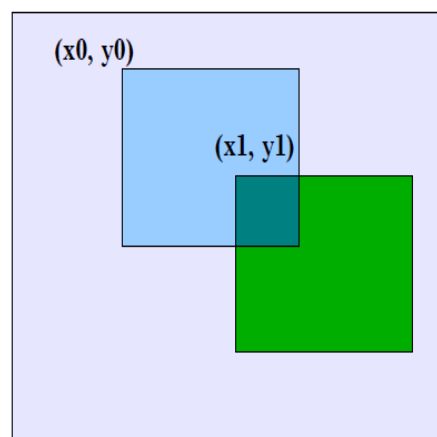
- **El motor del juego:** controla la animación y coordina las acciones que se han de realizar en cada momento,
- **La lógica del juego:** que establece unos estados: *Menú*, *Jugar*, *Fin de Juego (GameOver)* y *Salida (Exit)*,
- **Los tipos de datos:** son las clases que representan los datos sobre los que se desarrolla el juego: **Castell**, **Habitacio**, **Joc**, **Marcador**; y las clases que modelan cada uno de los objetos (**Actores**) que aparecen en el juego: **Heroína**, **Menjar**, **Fletxa**,.....

- **Los recursos:** las imágenes que se usan para crear los escenarios (podéis encontrar más en <http://www.iconarchive.com> , <http://www.veryicon.com> , <http://www.iconseeker.com> , <http://images.google.com/images?q=sprite+sheet>, ...).



## Descripción del funcionamiento interno del juego:

La animación se ejecuta varias veces por segundo (la frecuencia es de unos 60 *frames* por segundo), para cada *frame* los actores actualizan su posición y estado, y pueden interactuar con otros actores, modificando alguno de sus atributos. Por ejemplo, el objeto **Menjar** interacciona con nuestra heroína aumentando su nivel de vida cada vez que "*colisionan*". Para detectar las colisiones, cada actor definirá su dimensión (ancho y alto en píxels) como un rectángulo situado en una posición X, Y. Dos actores en las posiciones (x0, y0) y (x1, y1) colisionarán cuando sus rectángulos delimitadores interseccionen en algún punto:



Cuando se detecta alguna colisión de este tipo, el método **tractarColisio(Colisio colisio)** de cada actor implicado es ejecutado, de forma que el actor podrá modificar su estado, si es necesario. La definición completa de la interfaz **Actor** es la siguiente:

```

public void inicialitzar();
public void setPosicioInicial(int x, int y);
public int[] getPosicioInicial();
public void setPosicio(int x, int y);
public int[] getPosicio();
public void actualitzar(Context context);
public Rectangle getLimits();
public void tractarColisio(Colisio colisio);
public float getVida();
public void setVida(float nivell);
public int getEstat();
public void setEstat(int estat);

```

**(Consulta la documentación para conocer más sobre cada método)**

Cada escena del juego transcurre en una única planta y sala. La clase **Castell** proporciona todos los métodos necesarios para modificar la sala y planta actuales, así como para añadir y acceder a las habitaciones.

```

public void addHabitacio(int planta, int numHabitacio, Habitacio habitacio)
public void setPlanta(int planta)
public int getPlanta()
public void setNumHabitacio(int habitacio)
public int getNumHabitacio()
public Habitacio getHabitacio()

public Habitacio[] getHabitacions(int numPlanta)
public int getNumPlantes()
public int getNumHabitacions(int planta)

```

Como podéis imaginar la clase Castell contiene un atributo del tipo tabla que guarda la estructura de las habitaciones que hay en cada piso:

```

// habitacions es una matriz de pisos y las habitaciones que hay en cada
// piso: el número de habitaciones es el mismo en todos los pisos
private Habitacio[ ][ ] habitacions;
...
// en el constructor de Castell se realiza la siguiente instrucción:
habitacions = new Habitacio[numPlantes][numHabitacionsPerPlanta];

```

Para comunicar las habitaciones entre sí utilizaremos la clase **Porta**, que contiene la sala y puerta con la que se comunica.

**Ejemplo:**

A título de ejemplo, se proporciona una clase principal Main.java que crea un juego con 3 salas y algunos objetos distribuidos por ellas. Aquí se muestra uno de los métodos que construye la sala inicial:

```

private Habitacio crearHabitacio0Planta0() {
    Habitacio h = Util.carregarHabitacio("h0_0.txt");
    Porta porta = h.getPorta(14, 24);
    porta.setNumPlantaDesti(0);
    porta.setNumHabitacioDesti(1);
    porta.setPosicioHabitacioDesti(h.getPosicioCela(1, 1));

    porta = h.getPorta(0, 10);
    porta.setNumPlantaDesti(0);
    porta.setNumHabitacioDesti(2);
    porta.setPosicioHabitacioDesti(h.getPosicioCela(14, 10));
    Fletxa fletxa = new Fletxa();
    int[] p = h.getPosicioCela(2, 23);
}

```

```

    fletxa.setPosicioInicial(p[0], p[1] - 10);
    fletxa.setDireccio(Constants.DIRECCIO_OEST);
    h.addActor(fletxa);
    return h;
}

```

## Compilación y Ejecución

Desde el directorio "**src**":

– Para compilar:

```
> javac edu/ub/juanadearco/Main.java
```

– Para ejecutar el programa:

```
> java edu/ub/juanadearco/Main
```

– A veces es conveniente borrar todos los **.class** para volver a compilar todo de nuevo, para borrar todos los **.class** podéis utilizar esta instrucción:

```
> rm -r `find . -name *.class`
```

– para generar la documentación:

```
> javadoc -d ../doc/ -charset UTF-8 *.java
```

– después la puedes ver con tu navegador, por ejemplo desde la línea de comandos puedes ejecutar el comando:

```
> firefox ../doc/index.html
```

Objetivo	Grupo A	Grupo B	Grupo C	Grupo D	Grupo F
Crear un código base: 1,2,3,4	X	X	X	X	X
5 vida	X	X			
6 serpiente	X	X			
7 sorpresa			X		
8 soldado				X	
9 Brayan					X
10 Final por Escudo					X
11 Final por Hijo	X	X			
12 Final por Códigos			X		
13 Final por Llave				X	

1 - Crea un paquete con las iniciales de tu nombre dentro del paquete **edu.ub.juanadearco**. Por ejemplo: "edu.ub.juanadearco.abc"

2 - **Clase inicial:** Crea un fichero con el nombre **Practica.java** que contenga un método **main**. Esta clase será al estilo de la clase de ejemplo Main.java que se proporciona. Ten en cuenta que entonces deberás compilar el fichero Practica.java y no Main.java, como hasta ahora.

3 - **Configuración del territorio:** Crea un castillo con al menos 3 plantas y 3 habitaciones por planta. Para cada habitación deberás crear un fichero **.txt** con la definición de la habitación. Si lo deseas, puedes copiar el

fichero "h0\_2.txt" que define una sala vacía y modificarlo según desees.

4 - Crea las puertas que comunican las salas entre sí según desees.

5- Crea un nuevo actor con el nombre **Vida**. Si la heroína colisiona establece su vida a 100. Distribuye 2 vidas por el castillo. Puedes inspirarte en la clase Menjar.java, pero ten en cuenta que en este caso cada vida de una habitación se crea y se posiciona en una celda concreta definida al crearse la habitación misma.

6 - Crea un nuevo actor con el nombre de clase **Serp**. Si la heroína colisiona con una serpiente establece su vida a 0. Distribuye el número de serpientes que quieras por el castillo (mínimo 3) acorde con la distribución de los muros y puertas, ten cuidado en no situar una serpiente cerca de una puerta ya que el jugador nunca podría acceder. Puedes inspirarte en la clase Menjar.java, pero ten en cuenta que en este caso cada serpiente de una habitación se crea y se sitúa en su sitio al crearse la habitación misma.

7 - Crea un nuevo actor con el nombre **Sorpresa**. Al colisionar con nuestra heroína producirá uno de estos efectos aleatoriamente:

- Aumentar su nivel de vida en 50 unidades
- Disminuir su nivel de vida en 50 unidades
- Trasladar a la heroína de forma aleatoria a otra habitación de la misma planta

Crea de 3 a 5 sorpresas y distribúyelas aleatoriamente por las habitaciones del castillo. Puedes inspirarte en la clase Menjar.java, y en el método **distribuirMenjar()** de la clase Main.java

8 - Modifica el método **actualitzar()** de la clase **Soldat** de forma que se mueva por la habitación cambiando de dirección cada vez que choca con un muro. Crea más soldados, 3 por planta y colócalos en las habitaciones que creas.

9 – Crea un nuevo actor con el nombre de Brayan (Puedes inspirarte en el actor Menjar.java). Esta clase representa el general Brayan y nuestra heroína perderá el juego si colisiona con él. En la clase Practica crea un objeto de la clase Brayan y sitúalo aleatoriamente en una de las habitaciones del castillo. Añade el método **haTrobatBrayan()** de la clase Heroína de manera que retorne *true* si se ha encontrado el general Brayan. Añade también en esta clase el método **setHaTrobatBrayan()** que permita establecer que se ha encontrado. El método **setHaTrobatBrayan()** debería llamarse cuando la heroína colisiona con el general Brayan. En la clase JuanaDeArco.java, cuando se actualiza (método **actualitzarJoc()**) el juego, se debería tener en cuenta si la heroína ha encontrado o no al general. En esta misma clase modificar el método **mostrarGameOver** para que muestre también el mensaje de "GAME OVER" cuando el juego se acabe porque la heroína ha colisionado con el general. Inspírate en la clase Menjar.java, para posicionar el general Brayan en una habitación aleatoria del castillo.

10 – **Final por Escudo:** Haz que el juego se acabe cuando la protagonista encuentre al general Brayan pero haya conseguido un escudo previamente.

- Crea un nuevo actor en el paquete **edu.ub.juanadearco.actors** con el nombre de clase **Escut** (Puedes inspirarte en el actor Menjar.java)
- En la clase Practica, crea un objeto de la clase **Escut** y sitúalo aleatoriamente en una de las habitaciones del castillo.
- Modifica el método **haTrobatEscut()** (y añade los métodos que hagan falta) de la clase Heroína de manera que retorne *true* si se ha encontrado el escudo.
- Sitúa un objeto de la clase **Escut** en la última habitación de la última planta.
- Piensa en modificar de forma conveniente la colisión entre **Escut** y **Heroína** para indicar que se ha encontrado el escudo.
- En la clase **JuanaDeArco**, cuando se actualiza el juego, se debería tener en cuenta si la heroína ha encontrado o no al general y el escudo a la vez. Si ha encontrado al general pero no el escudo el

juego se acabará con el mensaje de "GAME OVER" i "Estàs mort! Has trobat el General Brayan. Un altre dia potser ..." pero si ha encontrado el escudo previamente, el juego se acabará con el mensaje "T'has pogut defensar del General Brayan!"

Ayuda: Podéis modificar la clase Brayan para que al colisionar ponga la vida de la heroína a 0 y modificar la clase adecuada para que aparezca el mensaje de "GAME OVER" únicamente cuando la vida de la heroína es 0 y el mensaje "Estàs mort! Has trobat el General Brayan" si además ha encontrado el General Brayan.

11 – **Final por Hijo:** Haz que el juego se acabe cuando la protagonista encuentre al hijo secuestrado de Carlos VII, el duque de Anjou.

- Crea un nuevo actor en el paquete **edu.ub.juanadearco.actors** con el nombre de clase **Fill**. (Puedes inspirarte en el actor Menjar.java)
- En la clase Practica, crea un objeto de la clase **Fill** y sitúalo en la última habitación de la última planta.
- Añade el método **haTrobatFill()** de la clase Heroína de manera que retorne *true* si se ha encontrado el hijo. Añade también en esta clase el método **setHaTrobatFill()** que permita establecer que se ha encontrado.
- El método **setHaTrobatFill()** debería llamarse cuando la heroína colisiona con el hijo.
- Añade un método en la clase Marcador que se llame **pintarMissatgeFill()** para que aparezca un mensaje de "Soc aquí!!!" cuando la heroína acceda a la misma sala donde está el hijo. Fíjate en como se hace para que aparezcan los indicadores de la sala y la planta en las que está la heroína.
- En la clase **JuanaDeArco**, cuando se actualiza el juego, se debería tener en cuenta si la heroína ha encontrado o no al hijo. Cuando acabe el juego aparecerá el mensaje ""HAS TROBAT EL FILL !!!", si la heroína lo ha encontrado.

12 - **Final por Códigos:**

Haz que el juego se acabe cuando la protagonista haya recogido un conjunto de 5 códigos como mínimo. Un código es un color. Estos son los códigos que permitirán descifrar el enigma que según cuenta la leyenda destruirá el maleficio que ha permitido a los ingleses apoderarse del castillo.

- Crea un nuevo actor en el paquete **edu.ub.juanadearco.actors** con el nombre de clase **Codi**. (Puedes inspirarte en el actor Menjar.java)
- En la clase Practica, distribuye los códigos o letras por las habitaciones del castillo de forma aleatoria. El número de códigos total en el castillo es el valor de la constante **Constants.NUM\_CODIS** que puedes utilizar. (Puedes fijarte en el método **distribuirMenjar()** de la clase **Main**)
- Añade el método **haTrobatElsCodis()** de la clase Heroína de manera que retorne *true* si ha encontrado todos los códigos. Añade también en esta clase el método **addCodi()** que permita incrementar el número de códigos que va consiguiendo la heroína.
- El método **addCodi()** debería llamarse cuando la heroína colisiona con un código. En este método procura guardar la letra del código en una tabla además de incrementar el número de códigos.
- Añade un método en la clase Marcador que se llame **pintarCodi()** para que muestre también el número de códigos encontrados. Fíjate en como se hace para que aparezcan los indicadores de la sala y la planta en las que está la heroína.
- En la clase **JuanaDeArco**, cuando se actualiza el juego, se debería tener en cuenta si la heroína ha encontrado todos los códigos o no. Cuando acabe el juego aparecerá el mensaje "HAS TROBAT ELS X CODIS" (dónde X es el número de códigos).

13 – **Final por llave:**



Haz que el juego se acabe cuando la protagonista haya recogido la llave de la celda donde está el duque de Anjou y esté con un nivel de vida de más del 70 %, en caso de colisionar con la llave y el nivel de vida sea menor del 70% se acabará el juego y la protagonista perderá.

- Crea un nuevo actor en el paquete **edu.ub.juanadearco.actors** con el nombre de clase **Clau**. (Puedes inspirarte en el actor Menjar.java).
- En la clase Practica, posiciona la llave en una de las habitaciones del castillo de forma aleatoria. (Puedes fijarte en el método **distribuirMenjar** de la clase **Main**).
- Añade el método **haTrobatClau()** de la clase Heroína de manera que retorne *true* si se ha encontrado la llave. Añade también en esta clase el método **setHaTrobatClau()** que permita establecer que se ha encontrado.
- El método **setHaTrobatClau()** debería llamarse cuando la heroína colisiona con la llave.
- En la clase **JuanaDeArco**, cuando se actualiza el juego, se debería tener en cuenta si la heroína ha encontrado la llave. Si lo ha bebido con más de un 70% de vida el juego acabará con el mensaje "Has pogut trobar la clau a temps!!!". Si no encuentra la llave en estas condiciones aparecerá el mensaje de **"\* GAME OVER \*"**.

## Ayudas

### • Demostración

Junto con el código os proporcionamos el fichero "JuanaDeArco.jar" que contiene una demo ejecutable con algunos ejercicios solucionados, para ejecutarla:

```
> java -jar JuanaDeArco.jar
```

### • Tabla de posición de imágenes:

Nombre	Fichero	Posición X	Posición Y	Ancho	Alto
Brayan	brayan.png	0	0	25	50
Clau	objectes.png	675	50	20	30
Codi 1	objectes.png	595	95	30	30
Codi 2	objectes.png	760	6	30	30
Codi 3	objectes.png	760	45	30	30
Codi 4	objectes.png	760	85	30	30
Escut	objectes.png	4	85	32	36
Fill	fill.png	0	0	25	50
Fletxa 1	fletxa.png	0	0	27	15
Fletxa 2	fletxa.png	30	0	27	15
Foc	foc.png	0	0	28	30
Heroína 1	heroína.png	0	0	25	50
Heroína 2	heroína.png	25	0	25	50
Heroína 3	heroína.png	50	0	25	50
Heroína 4	heroína.png	75	0	25	50
Menjar 1	objectes.png	540	14	30	22
Menjar 2	objectes.png	439	14	27	23
Menjar 3	objectes.png	97	100	30	20
Serp 1	objectes.png	700	0	40	42
Serp 2	objectes.png	700	42	40	42
Serp 3	objectes.png	700	84	40	42
Soldat	objectes.png	675	0	20	50
Sorpresa	objectes.png	198	41	28	35
Vida	objectes.png	174	12	26	24

- **Obtención de números aleatorios:**

Para obtener un número aleatorio puedes utilizar la función **Math.random** de java, que retorna un número **mayor o igual a 0 y menor de 1**. Ejemplos:

```
// número aleatorio entre 0 y 10
```

```
int r = (int) (Math.random() * 11);
```

```
0
```

```
int r = (int) (Math.round(Math.random() * 10));
```