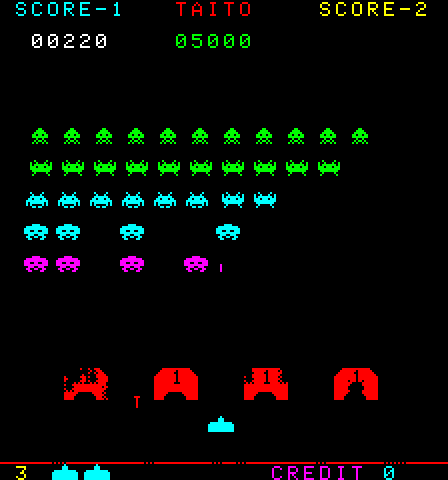
¿Por qué descomponer en clases?

Cuando tenemos que realizar un proyecto grande, será necesario descomponerlo en varios subprogramas, de forma que podamos repartir el trabajo entre varias personas.

Esta descomposición no debe ser arbitaria. Por ejemplo, será deseable que cada bloque tenga unas responsabilidades claras, y que cada bloque no dependa de los detalles internos de otros bloques.

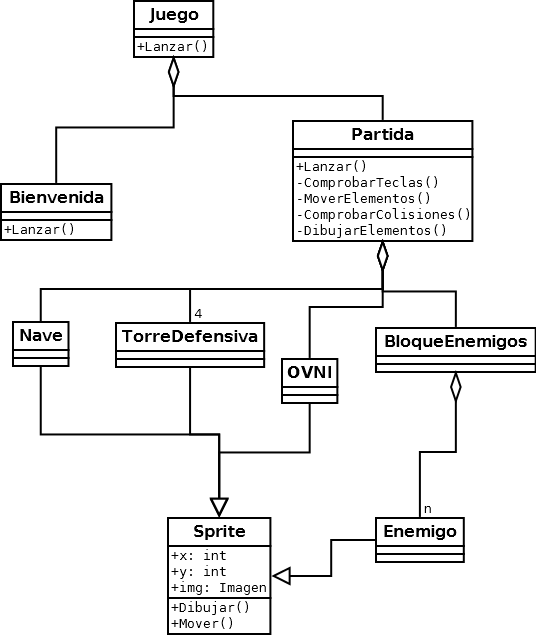
Existen varias formas de descomponer un proyecto, pero posiblemente la más recomendable consiste en tratar de verlo como una serie de "objetos" que colaboran entre ellos, cada uno de los cuales tiene unas ciertas responsabilidades.

Como ejemplo, vamos a dedicar un momento a pensar qué elementos ("objetos") hay en un juego como el clásico Space Invaders:



De la pantalla anterior, se puede observar que nosotros manejamos una "nave", que se esconde detrás de "torres defensivas", y que nos atacan (nos disparan) "enemigos". Además, estos enemigos no se mueven de forma independiente, sino como un "bloque". En concreto, hay cuatro "tipos" de enemigos, que no se diferencian en su comportamiento, pero sí en su imagen. También, aunque no se ve en la pantalla anterior, en ocasiones aparece un "OVNI" en la parte superior de la pantalla, que nos permite obtener puntuación extra. También hay un "marcador", que muestra la puntuación y el record. Y antes y después de cada "partida", regresamos a una pantalla de "bienvenida", que muestra una animación que nos informa de cuántos puntos obtenemos al destruir cada tipo de enemigo.

Para diseñar cómo descomponer el programa, se suele usar la ayuda de "diagramas de clases", que muestran de una manera visual qué objetos son los que interaccionan para, entre todos ellos, formar nuestro proyecto. En el caso de nuestro "Space Invaders", un diagrama de clases simplificado podría ser algo como:



Algunos de los detalles que se pueden leer de ese diagrama son:

* La clase principal de nuestro proyecto se llama "Juego" (el diagrama típicamente se leerá de arriba a abajo).
* El juego contiene una "Bienvenida" y una "Partida" (ese relación de que un objeto "contiene" a otros se indica mediante un rombo en el extremo de la línea que une ambas clases, junto a la clase "contenedora").
* En una partida participan una "Nave", cuatro "Torres" defensivas, un "BloqueDeEnemigos" formado por varios "Enemigos" (que, a su vez, podrían ser de tres tipos distintos, pero no afinaremos tanto por ahora) y un "Ovni".
* Tanto la "Nave" como las "Torres", los "Enemigos" y el "Ovni" son tipos concretos de "Sprite" (esa relación entre un objeto más genérico y uno más específico se indica con las puntas de flecha, que señalan al objeto más genérico).
* Un "Sprite" es una figura gráfica de las que aparecen en el juego. Cada sprite tendrá detalles (atributos) como una "imagen" y una posición, dada por sus coordenadas "x" e "y". Será capaz de hacer operaciones (**métodos**) como "dibujarse" o "moverse" a una nueva posición. Cuando se programa toda esta estructura de clases, los atributos serán variables, mientras que los "métodos" serán funciones. Los subtipos de sprite "**heredarán**" las características de esta clase. Por ejemplo, como un Sprite tiene una coordenada X y una Y, también lo tendrá el OVNI, que es una subclase de Sprite.
* El propio juego también tendrá métodos como "comprobarTeclas" (para ver qué teclas ha pulsado el usuario), "moverElementos" (para actualizar el movimiento de los elementos que deban moverse por ellos mismos), "comprobarColisiones" (para ver si dos elementos chocan, como un disparo y un enemigo, y actualizar el estado del juego según corresponda), o "dibujarElementos" (para mostrar en pantalla todos los elementos actualizados).

En este punto, podríamos empezar a repartir trabajo: una persona se podría encargar de crear la pantalla de bienvenida, otra de la lógica del juego, otra del movimiento de los enemigos, otra de las peculiaridades de cada tipo de enemigo, otra del OVNI...

Nosotros no vamos a hacer proyectos tan grandes (al menos, no todavía), pero sí empezaremos a crear proyectos sencillos en los que colaboren varias clases, que permitan sentar las bases para proyectos más complejos.