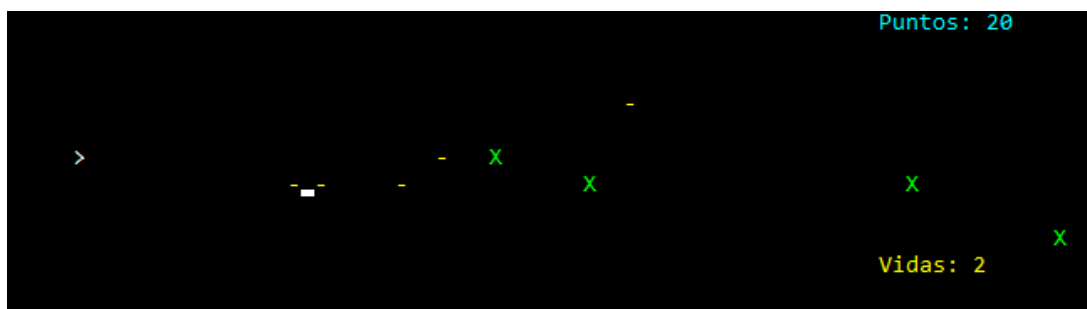


Programación

Práctica - Desarrollo de un videojuego



En esta práctica vamos a desarrollar un videojuego en modo consola en un proyecto llamado **Spaceship**. La estructura y mecánica del juego va a ser algo similar al proyecto *ConsoleInvaders* desarrollado en clase, pero con algunos cambios.

Esta es la apariencia general del juego:



1. Estructura de clases

El proyecto deberá contar con las siguientes clases y características (muchas de ellas modificadas de *ConsoleInvaders*):

- Clase **Configuracion** con algunos parámetros de configuración, como por ejemplo el ancho y alto de la pantalla (el alto esta vez estará limitado a unas 10 o 15 filas aproximadamente), el máximo de disparos permitidos simultáneamente, vidas iniciales, puntos por destruir un enemigo y otras cosas que puedas necesitar.
- Clase padre **Sprite** para los distintos tipos de sprite. Almacenará las coordenadas X e Y del *Sprite* junto con su imagen, y los constructores, *getters/setters* y otros métodos que puedan resultar útiles (dibujar, comprobar colisiones...). Tendrá como subtipos:
 - **Nave**: manejada por el usuario, podrá moverse verticalmente, y disparar hacia la derecha. Su imagen será el símbolo  del color que quieras
 - **Enemigo**: generados al azar desde el margen derecho, se moverán hacia la izquierda hasta desaparecer por el otro lado, chocar con la nave o ser destruidos por un disparo. Tendrá como imagen una  del color que quieras
 - **Disparo**: generados por la nave, hasta un tope
- Pantallas: igual que en *ConsoleInvaders*, dispondremos de las clases **Bienvenida** y **Partida**, gestionadas desde el programa principal **Spaceship**: se cargará la pantalla de bienvenida y, al pulsar la barra espaciadora, se pasará a la partida. Al pulsar escape desde esta pantalla volveremos a la de bienvenida, y al pulsar escape desde bienvenida saldremos del juego.

2. Mecánica del juego

El juego comenzará con la nave principal colocada a media altura de la pantalla, a unas 4 o 5 casillas del borde izquierdo. Podrá moverse sólo en vertical. La nave tendrá un máximo de D disparos simultáneos (definidos como constante en la clase *Configuración*. Por ejemplo, D puede ser 5 disparos). Estos disparos se iniciarán al pulsar espacio.

Se permitirá un máximo de E enemigos simultáneos en pantalla (por ejemplo, $E = 200$). Cada enemigo se generará con una probabilidad P entre 0 y 100. Para ello, generamos un número aleatorio de 0 a 100, y si es menor que P , generamos un enemigo en el borde derecho de la pantalla (coordenada Y aleatoria dentro de los límites permitidos), y hacemos que se mueva hacia la izquierda. Un valor adecuado de P para empezar puede ser 10.

Los enemigos se moverán hacia la izquierda siempre que permanezcan activos. El enemigo dejará de estar activo cuando:

- Llegue al borde izquierdo
- Choque con la nave (quitamos una vida a la nave)
- Choque con algún disparo de la nave (sumamos puntos)

En cualquiera de estos casos, el enemigo pasa a estar inactivo y puede reemplazarse por otro nuevo.

En la esquina superior derecha mostraremos el contador de puntos, y en la esquina inferior derecha las vidas. No puede haber naves ni disparos en ninguna de estas dos filas.

La partida terminará cuando nos quedemos sin vidas. Para ir incrementando poco a poco la dificultad, podemos hacer que cada X puntos se incremente la probabilidad de generar un enemigo (dato P), por ejemplo, de 10 en 10.

3. Criterios de calificación

- **(0,5 puntos)** Gestión de pantallas: paso adecuado entre las dos pantallas. Se valorará aquí la vistosidad de la pantalla de bienvenida. Por ejemplo, podéis dibujar el título del juego "SPACESHIP" con colores diferentes en cada letra.
- **(1 punto)** Clase *Sprite* y subclases, con información básica de cada una, constructores apropiados, getters/setters y métodos de dibujado
- **(0,5 puntos)** Clase *Configuración* con constantes y datos adecuados para poder ajustar los parámetros importantes del juego sin tocar el código de ninguna otra clase
- **(1 punto)** Movimiento adecuado de la nave arriba/abajo dentro de los límites permitidos (sin contar primera y última fila, donde se dibujará el HUD)
- **(2 puntos)** Generación aleatoria de enemigos en coordenadas Y aleatorias, y con una frecuencia adecuada. Se incluye aquí también el movimiento automático hacia la izquierda y el reemplazo por enemigos nuevos al quedar inactivos
- **(2 puntos)** Gestión de los disparos de la nave, hasta el máximo permitido. Se incluye aquí el movimiento del disparo a la derecha y su reemplazo por otro nuevo al quedar inactivo
- **(1 punto)** Incremento paulatino de la dificultad aumentando la probabilidad de generar enemigos

- **(1 punto)** Actualización del HUD con cada enemigo que se destruya con un disparo o cada vida que se pierda al chocar con un enemigo.
- **(1 punto)** Bucle del juego, incluyendo pausas, condición de terminación y secuenciación de pasos (dibujados, movimientos, colisiones, etc).