Fundamentos de la programación

Práctica 1. Moon lander

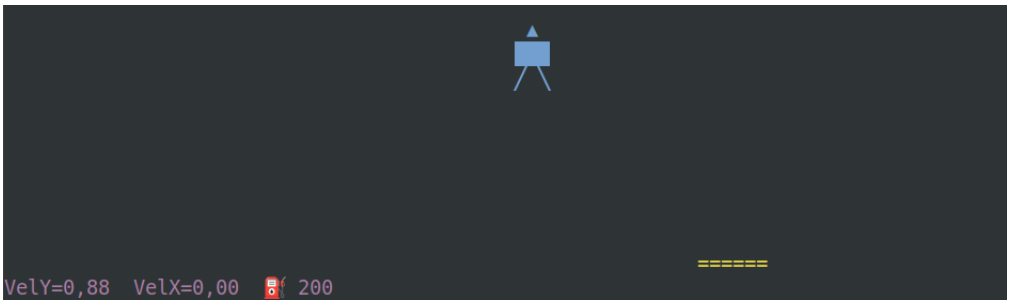
Indicaciones generales:

- La línea 1 del programa y siguientes deben contener los nombres de los alumnos de la forma (una línea por alumno):

  `// Nombre Apellido1 Apellido2`

- **Lee atentamente** el enunciado y desarrolla el programa tal como se pide, con la representación y esquema propuestos.
- El programa, además de correcto, debe estar bien estructurado y comentado. Se valorarán la claridad, la concisión y la eficiencia.
- La entrega se realizará a través del campus virtual, subiendo únicamente el archivo `Program.cs`, con el programa completo.
- El **plazo de entrega** finaliza el 3 de noviembre.

Vamos a implementar un juego de alunizaje, que tendrá el siguiente aspecto en la consola:



En la parte superior se muestra el módulo lunar, en azul, que está en caída libre, sujeto a la aceleración de la gravedad lunar. La franja horizontal amarilla representa la plataforma de aterrizaje. La línea inferior de texto (HUD) indica la velocidad vertical y horizontal del módulo, y el combustible restante.

El objetivo es posar suavemente el módulo sobre la plataforma. Si cae fuera de ella o sobre ella, pero con demasiada velocidad, el módulo se destruye y pierde el jugador. El módulo posee tres propulsores: uno de impulsión hacia arriba para contrarrestar la gravedad y dos hacia los lados para moverlo

Figura 1: Plantilla del programa (archivo `Program.cs`)

```
static Random rnd = new Random(); // generador de aleatorios para colocar la plataform
const int ANCHO = 100, ALTO = 30, // ancho y alto del área de juego
          ANCHO_PLAT = 6;         // ancho de la plataforma
const double G_LUNAR = 0.05;      // gravedad lunar (aceleración hacia abajo)

const bool DEBUG = false;         // datos para depuración

static void Main() {
    // Estado del módulo de aterrizaje
    double posX = ANCHO / 2, posY = 2, // posición inicial
           velX = 0, velY = 0,       // velocidad inicial
           gravedad = 0.04,    // gravedad lunar
           empuje = -0.3,      // aceleración del propulsor vertical
           lateral = 0.05;     // aceleración de los propulsores horizontales
    int combustible = 200,     // unidades de combustible iniciales
        plataformaX,
        plataformaAncho = 6;        // ancho de la plataforma
    bool landed = false,    // true si ha aterrizado correctamente
         destroyed = false; // true si se ha destruido
```

lateralmente. Se activan durante un breve periodo de tiempo con las teclas ''`awd`'' (izquierda, arriba, derecha). Cada vez que se activa un propulsor se consume una unidad decombustible y una vez agotado el combustible, los propulsores dejan de funcionar.

Para implementar el juego, se proporciona una plantilla (archivo `Program.cs`), que contiene ya parte del código necesario, tal como muestra la Figura 1. El generador de números aleatorios `rnd`, funciona con la llamada `rnd.Next(a,b)` que devuelve un entero en el rango `[a,b)` (se utilizará para colocar la plataforma en una posición horizontal aleatoria). Las constantes `ANCHO` y `ALTO` determinan el tamaño del área de juego y `ANCHO_PLAT` el ancho de la plataforma. La constante `G_LUNAR` es la aceleración de la gravedad lunar. La

constante `DEBUG` puede ponerse a `true` para mostrar datos adicionales en pantalla (los que se consideren oportunos) para ayudar a depurar el programa.

El resto de variables determinan el estado del módulo lunar: posición, velocidad, aceleraciones, combustible y posición de la plataforma. Como se aprecia en el dibujo, el módulo lunar ocupa 3×3 posiciones en pantalla. Las coordenadas (`posX`,`posY`) representan la posición central del módulo (que se utilizará para mostrar en pantalla, detectar colisiones, etc.). La plataforma se situará en la parte inferior del área de juego (la coordenada vertical es fija), y su posición horizontal se almacenará en `plataformaX`, que determina la posición izquierda de la plataforma. El ancho de la plataforma viene dado por `plataformaAncho`.

Lo primero que debe hacerse es inicializar la posición de la plataforma (`plataformaX`) en una posición horizontal aleatoria, dejando al menos 5 posiciones libres a cada lado, para que no quede demasiado ajustada a los lados. A continuación arranca el bucle principal del juego, que se ejecuta mientras el módulo no llegue al suelo (a la plataforma o fuera de ella). En cada vuelta del bucle se harán las acciones que se especifican a continuación **en el orden indicado**. No obstante, para facilitar el desarrollo es muy recomendable implementar en primer lugar el renderizado, ya que nos permitirá ver en pantalla el estado del juego en cada momento y depurar el resto de acciones.

Las acciones que debe realizar el bucle principal son (**en este orden**):

- Recogida del input del jugador de teclado (ya implementado). Si se hiciese la lectura con el habitual `Console.ReadLine()`, la entrada sería *bloqueante*: el programa queda a la espera del input de usuario y la ejecución se bloquea hasta que se pulse *intro*. Para evitar esta parada y que el juego fluya utilizaremos una *lectura no bloqueante*: si hay pulsación de teclado se recoge el carácter correspondiente (sin esperar intro); si no, continúa la ejecución.

- Procesamiento del input de usuario: si se ha pulsado alguna de las teclas

''awd'' y queda combustible, se consume una unidad de combustible y se aplica la aceleración, en función de la tecla pulsada: se suma la aceleración a la velocidad (lateral o vertical) correspondiente.

- Motor de física: se actualiza la velocidad vertical sumando la aceleración de la gravedad. Luego se aplican las velocidades vertical y horizontal a la posición del módulo. Si el módulo se sale del área de juego por los lados se ajusta la posición al tope lateral y se pone la velocidad horizontal a cero. Si se sale por arriba se ajusta la posición al tope vertical (pero no se modifica la velocidad vertical).

- Renderizado en consola: se limpia la pantalla y se dibuja la plataforma, el módulo lunar y el HUD. Es importante observar que las coordenadas del módulo están expresadas en coma flotante (double) para permitir una mayor precisión en los cálculos. Pero para dibujarlo en pantalla hay que redondear esos valores a entero (int).

  El método `Console.SetCursorPosition(left,top)` sitúa el cursor en la posición `(left,top)` de la pantalla. La esquina superior izquierda de la pantalla es la posición `(0,0)`. Para cambiar el color del texto se utiliza `Console.ForegroundColor = ConsoleColor.Blue` (cambia el color a azul, en este caso). Para restaurar el color se utiliza `Console.ResetColor()`. Para limpiar la pantalla se utiliza `Console.Clear()`.

  Si `DEBUG=true` se muestran por debajo los datos adicionales con posición y aceleraciones y posición de la plataforma para depuración. El módulo puede dibujarse escribiendo tres caracteres en cada una de las tres filas que ocupa.

- Colisiones: si el módulo ha llegado al suelo se comprueba si ha aterrizado sobre la plataforma y si la velocidad vertical es menor o igual a 0.5. Si es así, el aterrizaje es correcto y gana el jugador (`landed=true`); en otro caso, el módulo se destruye (`destroyed=true`).

- Retardo: para que el usuario tenga tiempo de reacción y el juego sea *jugable*, al final de cada vuelta del bucle se incluye la instrucción `System.Threading.Thread.Sleep(DELTA)`, que para la ejecución durante `DELTA` milisegundos.

El bucle principal termina cuando se detecta alguna de las colisiones (`landed` o `destroyed`). Fuera del bucle se muestra un mensaje indicando si el jugador ha ganado o perdido, y se despide.

1. Posibles mejoras

- Cuando el módulo se sale por los lados, en lugar de ajustar la posición al tope lateral y poner la velocidad horizontal a cero, se puede hacer que aparezca por el lado contrario (efecto pantalla toroidal). Por ejemplo, si se sale por la izquierda, aparece por la derecha.
- Cuando se utiliza el propulsor inferior se puede renderizar un cono de fuego bajo el módulo de la forma  . Se puede hacer algo parecido con los propulsores laterales.
- Para evitar el parpadeo de la pantalla, en lugar de limpiar la pantalla y volver a dibujar todo, se puede dibujar directamente sobre la pantalla sin limpiarla. Para ello, antes de dibujar cada elemento cambiante (módulo, HUD) hay que dibujar espacios en blanco en las posiciones que ocupaba el elemento en el estado anterior.
- (Idea de Ismael) Implementar el juego para dos jugadores, con dos módulos lunares que compiten por aterrizar en la misma plataforma. Un jugador utiliza las teclas ''`wad`'' y el otro las teclas ''`jil`''. Gana el primero que aterrice correctamente. Además los módulos pueden destruirse entre sí si colisionan. En ese caso pierden los dos jugadores.