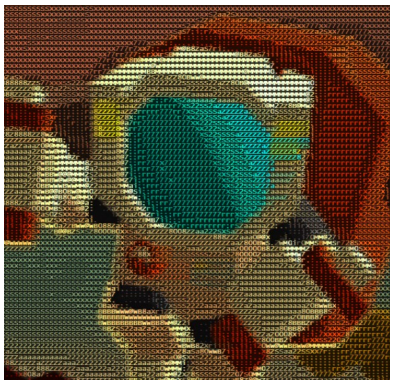
Proyecto FPII de Enrique Juan Gamboa

MARS: MARS - CMD Ed.



¿Qué es Mars:Mars?

Mars: Mars es un pequeño juego creado por Pomelo Games para Android e iOS en el que controlas un pequeño astronauta cuyo objetivo es moverse de una plataforma a otra. Para ello se vale de una pequeña mochila propulsora con la cual maniobrará por el aire, teniendo cuidado con aterrizar en una plataforma sin ir demasiado rápido contra el suelo. Eso sí, la mochila tiene una cantidad de combustible limitada, con que tendrá que tener en cuenta su consumo.

Gameplay trailer

Objetivos de esta versión

1. El nivel: la idea es utilizar la generación procedural que utilizamos en la Práctica 1 y editarla de manera que se pueda generar progresivamente el nivel, e implementar los métodos necesarios para que se pueda generar zonas llanas donde aterrizar.

2. Las físicas: se harán de trasfondo, cada frame, y se buscará una jugabilidad divertida antes de que sean realistas.

3. El renderizado: será muy simple, mostrará con qué cohete se está propulsando, el terreno y las plataformas. Debajo del renderizado del juego

se mostrará el número de la última plataforma a la que se llegó y el número máximo al que se ha llegado.

4. Puntuación y guardado: el juego guardará las puntuaciones junto al nombre de usuario que se use, y leerá todos los usuarios para averiguar quién tiene la puntuación más alta y enseñarla. Si un usuario que ya existe juega, comenzará donde se quedó la anterior partida.

Partida habitual

El juego empezará en el menú principal, donde se permitirá introducir un nombre con el que se cargará la partida (si existe) o creará una nueva.

Ya en el gameplay el jugador pulsará, desde el suelo, A o D para propulsarse a la izquierda o derecha. También podrá pulsar ambos botones para propulsarse rápidamente hacia arriba.
Tras el despegue, cada vez que el jugador pulse una de las teclas, su nivel de gasolina, representada debajo de la pantalla de juego, disminuirá.
Si el jugador aterriza con un descenso limpio y sin chocarse con nada, el mapa se desplazará hacia la izquierda para que el jugador pueda ver más mapa. En cambio, si el jugador aterrizara demasiado rápido o se chocara por el lateral, morirá y reaparecerá en su punto de partida anterior. Se sabrá si la velocidad de descenso es mortal si el color del astronauta cambia.

Una vez que el jugador quiera salir del juego, podrá entrar en el menú de pausa (pulsando P) para salir del juego, donde se guardará su progreso hasta la última llanura en la que aterrizó.

Representación interna

El juego a realizar contará, en principio, con 4 clases:

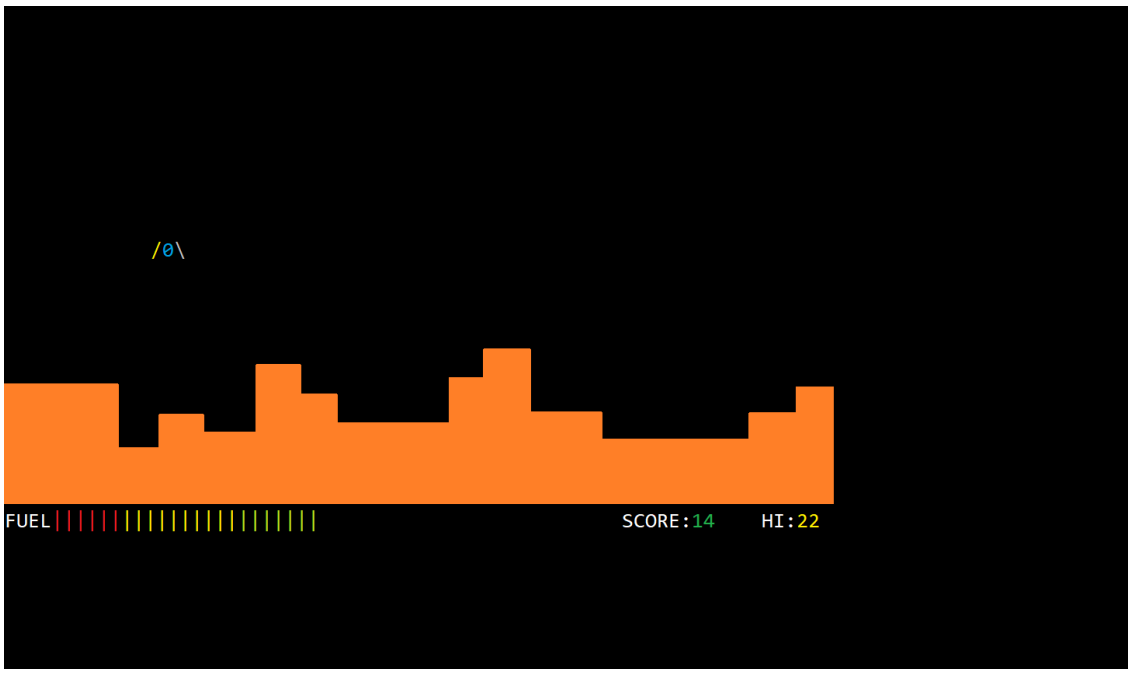
1. **Clase principal** (Program.cs): aquí se realizará todo el bucle de juego, la llamada a diferentes clases, el renderizado, etc. También se encargará del nivel de gasolina y de leer y guardar la puntuación.
   Esta clase utilizará variables simples como int y hará uso de arrays y structs para ordenar información y que quede un código más limpio.

2. **Listas:** a pesar de que se podría realizar el juego de manera sencilla sin usar listas, al requerirse en la práctica las utilizaremos para guardar la última plataforma a la que ha llegado el jugador y permitir retroceder.

3. **PhySick:** esta clase será el respaldo de todo el juego: va a ser el motor físico. Este motor contará con funciones para simular un campo gravitatorio dependiendo de la masa, aceleración, gravedad e incluso resistencia al aire de un objeto, una vez inicializado con los parámetros deseados.
   Va a contar con una amplia gama de funciones para simular movimiento, variables get/set de elementos como la masa y posición de objetos y funciones de propulsión física. Para todo ello hará un gran uso de structs y arrays.

4. **LvlGen:** esta clase podría introducirse en la clase principal, pero por orden y POO he optado por mantenerlo en una clase aparte.
   Esta clase está basada en una función que nos fue aportada para la práctica 1 de esta asignatura, con la que se generaba un nivel proceduralmente.
   En esta iteración será modificado para que sólo genere el suelo (ya que está programada para que genere una cueva) con parámetros como altura máxima e irregularidad de terreno y que sea capaz de generar terreno llano en intervalos para permitir que el astronauta aterrice.

Concepto de renderizado



Concepto de juego realizado en PhotoShop