Informe Trabajo Práctico N°2
Asteroids

Malena Díaz Falvo y Agustín Ormazábal

30 de Junio de 2019

Introducción

El actual proyecto consta de la implementación de una reversión del juego Asteroids en lenguaje ISO-C99 utilizando la biblioteca gráfica SDL2. Para su recreación se implementó el concepto de TDA, entre otras funcionalidades, incorporados a lo largo del curso.

Desarrollo

Para abordar este proyecto, se utilizaron y modificaron algunos de los archivos brindados para el TP1 (reversión del Lunar Lander). Además, se diseñaron y utilizaron diferentes TDAs para los objetos del juego, obteniendo finalmente los siguientes `.h`:

- `caracteres.h`
- `config.h`
- `dibujar.h`
- `diccionario.h`
- `movimiento.h`
- `nave.h`
- `asteroides.h`
- `disparos.h`
- `graficador.h`
- `vector.h`
- `iterador.h`
- `listas.h`

Cada uno de estos archivos está asociado a su correspondiente `.c` (a excepción de `config.h`), tres de los cuales fueron suministrados por la cátedra, y cada uno posee una descripción detallada de las pre y postcondiciones de cada función junto con, valga la redundancia, la función de las mismas. Además, se diseñó un diccionario para traducir caracteres a los vectores que se encuentran en `caracteres.h` con el fin de poder utilizar la librería gráfica a nuestra voluntad.

Al inicio de cada partida, la cual se ve delimitada por el aterrizaje o destrucción de la nave, se genera un terreno aleatorio nuevo, para esto se utilizó la función `float **terreno_crear(size_t *n);` brindada en el enunciado del trabajo práctico. Además, se posicionarion los mensajes con sus respectivos valores, que se modificarán a lo largo del transcurso del juego, de forma tal que se asemeje al diseño original del mismo.

Dificultades

Personalmente, la mayor dificultad que se me presentó fue comenzar a utilizar la función `SDL_RenderDrawLine(renderer, x0, y0, x1, y1);` de la biblioteca SLD2 y llegar a la conclusión de que era conveniente utilizar vectores dinámicos, para representar a la nave y al chorro, que sean redefinidos en cada ciclo de `while`. Esto es necesario debido al

hecho de que la nave debe rotar sobre sí misma, dependiendo de la interacción asincrónica que se establece con el o la usuaria, mientras se traslada por el plano xy. Si, por el contrario, los vectores se definen por fuera del `while`, cada traslación los modificaría y se perdería el rastro del orígen de coordenadas para las figuras, por lo que terminarían describiendo una circunferencia a lo largo del plano en lugar de rotar sobre sí mismos.

Por otro lado, la modularización contribuyó bastante a tener un código ordenado y prolijo. Sin embargo, si no se lo actualiza con cuidado regularmente, se puede llegar a olvidar sacar algún prototipo de una función borrada de los módulos. Esto no implica un error de compilación pero podría generar cierta confusión para aquel o aquella que quiera utilizar nuestros `.h` como una librería. Además al ser un enunciado de gran extención, a pesar de la lectura y relectura del mismo, siempre parecía que algun dato pasaba desapercibido.

Adicional

En la versión original del Lunar Lander se hace un zoom al momento de aterrizar. Con esta base, y teniendo dos diseños para la figura (nave chica y nave grande), se definió una distancia mínima al terreno y un intervalo del ángulo de la nave, y dependiendo de si se cumplen o no dichas condiciones, se dibujarán en pantalla la **nave de aterrizaje** (nave grande) o **la nave de vuelo** (nave chica).

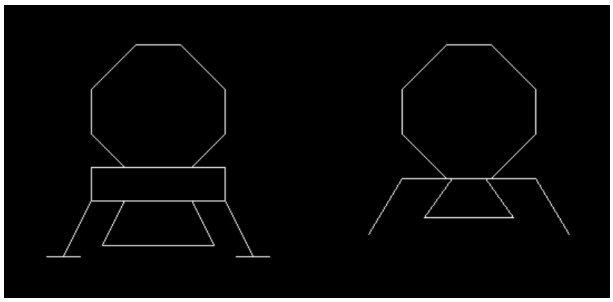

Figura 1: Naves suministradas.

Esta elección se debió a que, a pesar de que ambas naves poseen sus patas desplegadas para el aterrizaje, la primera posee un aspecto más sólido, con una mayor resistencia ante el impacto de la nave contra la superficie (en caso de que éste sea un aterrizaje violento), mientras que la segunda parece más frágil e inestable.

Además, se le agregaron estrellas al fondo para recrear el escenario de la superficie lunar con su paisaje de la forma más acertada posible. Para esto se definió una matriz de coordenadas para la representación de una estrella, junto con las funciones necesarias para crear una cantidad determinada de manera aleatoria e imprimirlas.

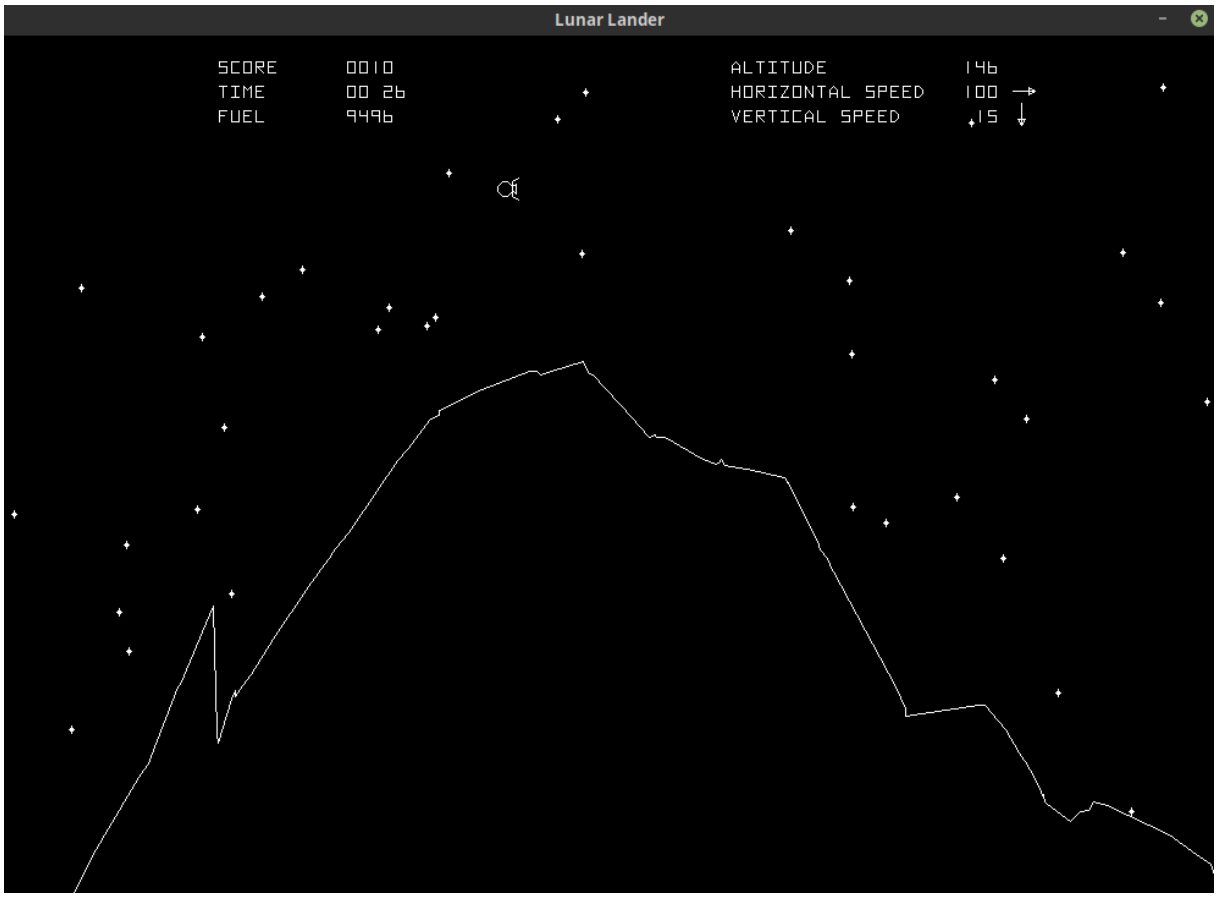

Figura 2: Superficie lunar en el juego.

Al no ser modificaciones complicadas de implementar, que aportan a la estética del juego, creí conveniente su aplicación ya que no interfiere con los campos principales a ser evaluados ni modifica los demás requisitos exigidos en el enunciado del trabajo práctico.