Universidad Simón Bolívar
Departamento de Computación y
Tecnología de la Información
CI-3815: Organización del Computador

Prof. Fernando Torre Mora

Proyecto 2 Septiembre – Diciembre 2022

Asteroids

1 Introducción

Asteroids es un juego clásico de arcada y uno de los primeros juegos de computadora comercialmente disponibles al público. Para recrear la Era de Oro de los Videojuegos, se le ha pedido que programe una versión simplificada del juego. Claro, para ser fieles al original, debe programarse en Assembly.

Asteroids se compone de un campo de juego en el cual el jugador controla una pequeña nave puntiaguda. Por todo el campo flotan asteroides, enemigos y vidas adicionales, los cuales el jugador debe esquivar y/o destruir.

2 Requerimientos del programa

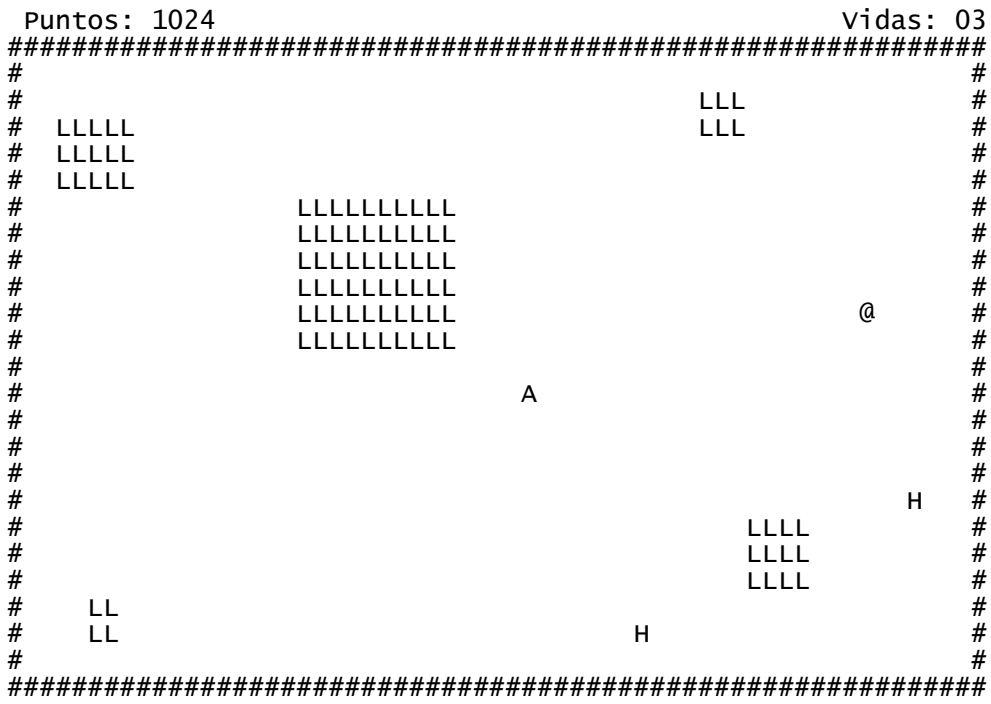
El programa se debe imprimir por cónsola, usando caracteres para indicar la posición de los diferentes elementos del juego. El campo de juego debe estar delimitado por un borde del caracter '#' para indicar donde termina la impresión de una pantalla y comienza la siguiente.

Cada impresión de una pantalla debe ocupar 60 caracteres de largo × 25 caracteres de alto

Se usan los siguientes símbolos para indicar los elementos del juego:

- A, <, >, V para representar la nave del jugador (dependiendo de en qué dirección esté apuntando)
- L para representar los asteroides
- H para representar los enemigos
- @ para representar las vidas adicionales

Los asteroides pueden tener tamaño variable, pero siempre serán cuadrados por simplicidad



Ejemplo del campo de juego con 5 asteroides, 2 enemigos y 1 vida adicional

La nave del jugador comienza siempre en el centro de un campo vacío y el jugador siempre comienza con 3 vidas.

2.1 Funcionamiento del programa

El programa opera al ritmo de la velocidad de refrescamiento de la pantalla, la cual debe ser de 6 refrescamientos por segundo. En ese momento, debe imprimirse por cónsola el estado del campo de juego, con todas las posiciones actualizadas.

2.2 Controles

El juego tiene los siguientes controles:

- La tecla W, la cual aumenta la velocidad de la nave del jugador en la dirección en la que está apuntando
- La tecla S, la cual aumenta la velocidad en la dirección contraria a la que el jugador está apuntando (permitiendo, así, frenar y retroceder)
- La tecla A, la cual gira la nave del jugador a la izquierda 90°
- La tecla D, la cual gira la nave del jugador a la derecha 90°
- La tecla P, la cual pausa el juego

La velocidad de la nave se mantiene constante a menos que el jugador la modifique. La velocidad máxima de la nave es de 5 caracteres por refrescamiento en X ó en Y.

Si el jugador llega a la orilla del campo de juego, debe hacer “warp” (teletransportarse) a la orilla opuesta y continuar en la dirección que llevaba.

2.3 Elementos del juego

2.3.1 Generación de asteroides

Cada asteroide tiene un tamaño entre 2 y 10 caracteres, los cuales dan el ancho del asteroide. Debido a que los caracteres de la cónsola son más altos que anchos, para que el asteroide se vea cuadrado, se debe multiplicar el ancho por $^{3}/_{5}$ y tomar el techo del resultado (sugerencia: revise si el residuo es cero para saber si debe aplicar la función techo o no).

Cada asteroide tiene una dirección designada dada por su velocidad en Y y su velocidad en X, no pudiendo exceder ninguna de estas a 3 caracteres por refrescamiento. El asteroide debe moverse a velocidad constante en esta dirección.

Los asteroides, al ser generados, comienzan a entrar lentamente cruzando la orilla del campo de juego, flotan hasta la orilla opuesta, cuando deben cruzarla lentamente hasta desaparecer. (Sugerencia: puede representar un campo de juego en memoria más grande del que imprime para poder llevar cuenta de cuánto del asteroide ha entrado en / salido del campo de juego en cada refrescamiento.)

Nótese que, si el asteroide entra por la orilla superior, su velocidad en Y debe ser hacia abajo, mientras que su velocidad en X sí puede ser negativa o positiva.

El juego debe generar un asteroide nuevo cada 5 segundos, pero nunca deben haber más de 8 asteroides en pantalla.

2.3.2 Generación de enemigos

Los enemigos se generan en una de las cuatro orillas, y siempre comparten la posición en X (en caso de generarse en las orillas horizontales) ó la posición en Y (en caso de generarse en las orillas verticales) con el jugador. Los enemigos vuelan hacia la orilla opuesta a la velocidad de 5 caracteres por refrescamiento.

El juego debe generar un enemigo cada 30 segundos, pero nunca deben haber más de 4 enemigos en pantalla

2.3.3 Generación de vidas adicionales

Las vidas adicionales se generan al igual que. Y se comportan de la misma manera que, los aterides, excepto que estas siempre ocupan un solo carácter.

El juego debe generar una vida adicional cada minuto, pero nunca debe haber más de 1 vida adicional en pantalla.

2.3.4 Números pseudo-aleatorios

Los tamaños y posiciones de los asteroides, así como la orilla de la cual sale un enemigo, se debe escoger usando un generador de números aleatorios

2.4 Detección de colisiones

Si el jugador choca contra un enemigo o un asteroide, ambos son eliminados. La colisión con los enemigos ocurre si ambos ocupan la misma posición (X,Y).

La colisión con los asteroides ocurre si el jugador se encuentra dentro de el *bounding box* del asteroide (es decir, si el asteroide es de 10×6 y ocupa las posiciones 30 – 40 en X y 42 – 48 en Y, si la nave del jugador se encuentra en cualquier posición de X entre 30 y 40 *y además* en cualquier posición de Y entre 42 y 48, el jugador ha chocado con el asteroide).

Esto permite evitar errores porque el jugador esté yendo muy rápido para detectar la colisión con el borde externo (por ejemplo, si el jugador se encontraba en X=39 y se mueve a 5 caracteres por refrescamiento, en el siguiente refrescamiento estará en X=44; no habrá interactuado con la frontera del asteroide en x=42, pero debe, aún así, detectarse la colisión – Nótese, sin embargo, que si el asteroide es más pequeño que la distancia por refrescamiento que recorre el jugador, el jugador es capaz de evitarlo volando directamente hacia él; a este comportamiento lo llamaremos *phasing*)

Al detectarse una colisión del jugador, se debe descontar una vida, esperar 3 segundos, y luego volver a colocar la nave del jugador en el centro.

Si las vidas llegan a cero, se deben, en su lugar, imprimir "Game Over" y el tiempo que el jugador sobrevivió contando en refrescamientos.

Si el jugador "colisiona" con una vida adicional, se debe incrementar las vidas del jugador, y eliminar la vida adicional del campo de juego. El jugador sigue moviéndose en la dirección que llevaba. (Obsérvese que, en este caso, el phasing dificulta el juego)

3 Sugerencias Adicionales

- Comience con un juego basado en prompts, donde la pantalla sólo se actualiza si el jugador presiona una tecla, y luego agregue el manejador de interrupciones.
- Comience con un campo de juego más pequeño para poder detectar errores más fácilmente
- Comience con una forma predecible de generar los asteroides y enemigos para poder detectar errores más fácilmente (por ejemplo, que vayan saliendo de la posición 1,1; luego 1,2; luego 1,3; etc) y sólo incorpore el manejo de números aleatorios una vez verificado que esto funciona bien
- Utilice funciones para hacer con el mismo código todas las funcionalidades comunes (por ejemplo, todos los elementos se pueden desplazar usando la misma función – el elemento que se está calculando puede ser un parámetro)
- Designe un espacio en la memoria para armar el String de salida antes de enviarlo a la cónsola y utilícelo como mapa del estado del juego
- Almacene por separado la información de los elementos de la información de la pantalla – es decir, no intente derivar la velocidad de un elemento comparando la pantalla actual con los refrescamientos anteriores; es más sencillo simplemente tener la velocidad almacenada en memoria.

4 Puntos Extra

Se considerará para puntos extra agregar funcionalidades que hagan que su juego se asemeje más al *Asteroids* original, incluyendo:

- *Warping* de asteroides y enemigos (la especificación anterior sólo requiere que el jugador pueda teletransportarse al llegar a la orilla; en el juego original todos los elementos lo hacían)
- Cañon de la nave (las balas viajan a 10 caracteres por segundo y los enemigos disparan continuamente hacia delante a un ritmo de 1 por segundo)
- *Break apart* (en el *Asteroids* original, cualquier impacto que recibiera un asteroide, hacía que se convirtiera en dos – así, golpear un asteroide de 10×6 hace que sea reemplazado por dos asteroides de 5×3 en esa posición, viajando en direcciones opuestas perpendiculares a la dirección

del asteroide original; un asteroide sólo es eliminado si es de 1×1. Estos fragmentos no cuentan para el límite de 8 asteroides en pantalla)

- Colisiones entre enemigos y asteroides (una estrategia válida en el juego original es tratar de que los enemigos se generen en donde chocarán por cuenta propia con un asteroide, destruyendo así al enemigo)
- Eliminación de *phasing*
- Detectar si ganó (si el jugador logra limpiar el campo de juego de todos los asteroides y enemigos en un momento dado, se dice que ha ganado)
- *High Score* (un aspecto característico de los juegos de arcada es que almacenan las 10 puntuaciones más altas logradas)

5 Readme

No es necesario entregar informe. Sólo debe incluir un archivo de texto llamado “readme” (léame) que indique el estado del programa y si decidieron optar por los puntos extra. De resto, se espera que la documentación de su código sea lo suficientemente clara y concisa para especificar y justificar todas las estructuras usadas.

6 Requerimientos de la entrega

El proyecto debe ser subido al Moodle de la materia en la sección marcada como “ Proyecto #” hasta el [fecha]. Sólo deberá efectuar una entrega por grupo.

7 Evaluación

El proyecto tiene una ponderación de 10 puntos. Se asignarán

- 5 puntos por código
- 5 puntos por ejecución

El programa debe correr sin errores.