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 que el jugador debe esquivar y/o destruir. Debido a que no existe forma de simular la inserción de monedas adicionales a la máquina de arcada, el juego ofrecerá vidas adicionales periódicamente, las cuales deben flotar al igual que los asteroides, como ocurre en las versiones para cónsolas que se conectan a televisores.

2 Requerimientos del programa

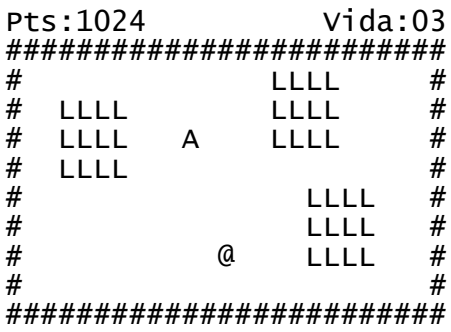
El programa debe funcionar en la cónsola MMIO simulada de MARS. Debe imprimir, usando caracteres para indicar la posición de los diferentes elementos del juego, el estado del juego cada 0,2 segundos. El campo de juego debe estar delimitado por un borde del caracter '#' para indicar donde termina la impresión de una pantalla y comienza la siguiente.

Cada impresión de una pantalla debe ocupar 25 caracteres de largo × 10 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
- @ para representar las vidas adicionales

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



Ejemplo del campo de juego con 3 asteroides 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. Los puntos son la cantidad de tiempo que el jugador ha logrado sobrevivir sin agotar sus vidas, medido en refrescamientos de la pantalla.

2.1 Funcionamiento del programa

El programa opera al ritmo de la velocidad de refrescamiento de la pantalla, la cual debe ser de 5 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 velocidad de la nave se mantiene constante a menos que el jugador la modifique. La velocidad máxima de la nave es de 3 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 es un rectángulo de 4×3 caracteres. Cada asteroide tiene una dirección designada dada por su velocidad en Y y su velocidad en X, no pudiendo exceder ninguna de estas a 2 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 4 asteroides en pantalla.

2.3.2 Generación de vidas adicionales

Las vidas adicionales se generan al igual que, Y se comportan de la misma manera que, los ateroides, 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.

2.4 0,2 segundos

La espera de 0,2 segundos se puede realizar con la llamada a sistema 32, la cual detiene la ejecución del programa por la cantidad de milisegundos en $a0. Se puede invocar con:

```
li $v0,32
li $a0,200
syscall #sleep
```

Para asegurar que los refrescamientos, en efecto, ocurran cada 0,2 segundos, debe contemplarse el tiempo de efectuar el refrescamiento como tal. Para ello, se almacena la hora en milisegundos al inicio del refrescamiento, y nuevamente al final. La hora se puede obtener con la llamada a sistema 30, la cual coloca esta información en $a1:

```
li $v0,30
syscall #time
```

Se resta la hora obtenida al inicio de la obtenida al final para obtener el tiempo que tomó el refrescamiento, y se resta este número del resultado, así:

```
li $v0,30
syscall #time
move $s6,$a1

########################################
##Espacio para Código de refrescamiento##
########################################

li $v0,30
syscall #time
sub $s6,$s6,$a1 #tiempo transcurrido

li $t7,200
sub $a0,$t7,$s6 #tiempo restante
li $v0,32
syscall #sleep
```

Nótese, sin embargo, que el tiempo de reanudar el programa luego de un *sleep* es no-deterministico, por lo que puede ser útil medir el tiempo del *sleep* y sumar el exceso a lo que se le restará al siguiente *sleep* (sin embargo, esto no es obligatorio).

2.5 Números pseudoaleatorios

Las velocidades y posiciones iniciales de los asteroides y vidas adicionales se deben escoger usando un generador de números pseudoaleatorios. Se pueden generar números pseudoaleatorios con la llamada a sistema 42, la cual se usa de la siguiente manera:

Antes de la primera llamada, debe fijarse el valor de la semilla con la llamada a sistema 40, para asegurarse de que siempre sea diferente. La práctica común es usar los milisegundos de la hora actual. Ya que la llamada al sistema 30 almacena este valor en $a0, y la llamada a sistema 40 requiere este valor en $a1, podemos fijar la semilla con:

```
li $v0,30
syscall #time
move $a1,$a0
li $a0,1
li $v0,40
syscall #set seed
```

El sistema puede tener tantos generadores de números aleatorios como hay enteros; el valor en $a0 se utiliza para distinguirlos. En el código anterior, se inicializa el primer generador. Este valor debe usarse cada vez que se genere un número aleatorio. Así, para generar un número en el rango [0; 25), podemos usar:

```
li $v0,42
li $a0,1
li $a1,25
syscall #random int range
```

El número generado se fija en $a0.

Nota importante: solo debe fijarse la semilla una vez en toda la ejecución del programa.

2.6 Detección de colisiones

Si el jugador choca contra un asteroide, ambos son eliminados. La colisión ocurre si el jugador se encuentra dentro del *bounding box* del asteroide (por ejemplo, si el asteroide ocupa las posiciones 20 – 24 en X y 2 – 5 en Y, si la nave del jugador se encuentra en cualquier posición de X entre 20 y 24 *y además* en cualquier posición de Y entre 2 y 5, 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=19 y se mueve a 3 caracteres por refrescamiento, en el siguiente refrescamiento estará en X=22; no habrá interactuado con la frontera del asteroide en x=20, pero debe, aún así, detectarse la colisión.

Al detectarse una colisión del jugador, se debe ocultar la nave del jugador, eliminar el asteroide, 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. No es necesario tomar un rango para las vidas adicionales; requerirle al jugador frenar para ganar una vida adicional es una forma válida de jugar.

Si dos asteroides ocupan el mismo espacio, no se considera que colisionan, pasan uno delante del otro.

Las vidas adicionales también pasan por delante o por detrás de los asteroides.

3 Sugerencias Adicionales

- Comience con un juego basado en prompts usando la cónsola por defecto (pestaña Run I/O) y luego agregue el manejador de interrupciones para procesar la entrada por MMIO, y de tercero la rutina de impresión en la cónsola simulada de MMIO.
- Comience haciendo que llegar a la orilla detenga al jugador, asteroide, o vida adicional, y luego programe el “warping” del jugador y la desaparición progresiva del asteroide
- Comience con un campo de juego más pequeño (por ejemplo, 7×7 para que la velocidad máxima cause un “warp” luego de un solo refrescamiento) 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 rutina de impresión 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 vidas (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 (presionar la bala espaciadora genera una bala frente a la nave que viaja en la dirección en la que miraba el jugador. Las balas viajan a 5 caracteres por segundo y se denotan con el símbolo +)
- Enemigos (los enemigos aparecen de la misma forma que los asteroides y disparan una vez por segundo en la dirección en la que viajan; se denotan con el símbolo H y no debe haber más de 2 en pantalla)
    - En el *Asteroids* original existen dos tipos de enemigo: los que disparan en la dirección en la que viajan y los que disparan vagamente en dirección al jugador. Esta segunda clase de enemigo se contará por separado y se denota con el símbolo X.

- *Break apart* (en el *Asteroids* original, cualquier impacto que recibiera un asteroide – ya fuera del jugador, una bala, o un enemigo – hacía que se convirtiera en dos más pequeños. En este caso, serían dos asteroides de 2×2, y se ser golpeado nuevamente, de 1×1. Ambas "mitades" deben viajar en direcciones opuestas perpendiculares a la dirección del asteroide original. Un asteroide sólo es borrado del mapa en este esquema, si es de 1×1. Estos fragmentos no cuentan para el límite de 4 asteroides en pantalla)
    - Las colisiones entre enemigos y asteroides se contarán por separado para este fin, pero deben destruir al enemigo, ya que una estrategia válida en el juego original para librarse de un enemigo es tratar de que los enemigos choquen con un asteroide
- 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 (y de ser así, cuáles). 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 Evaluación

El proyecto tiene una ponderación de 10 puntos. Se asignarán

- 5 puntos por código
    - 1 punto por tener un manejador de la entrada usando el simulador MMIO
    - 1 punto por tener un manejador de la salida usando el simulador MMIO
    - 1 punto por la forma de almacenar el campo de juego
    - 1 punto por la forma de actualizar el campo de juego
    - 1 punto por sus rutinas para la generación de los elementos del juego
- 5 puntos por ejecución
    - 1 punto por el campo de juego
    - 1 punto el movimiento del jugador
    - 1 punto por el movimiento de los asteroides
    - 1 punto por el movimiento de las vidas adicionales
    - 1 punto por su detección de colisiones y actualización del número de vidas

El programa debe correr sin errores.