Paulo Sandoval Suazo

Nancy Hitschfeld K.

CC3501 - Computación Gráfica

16 Mayo 2017

Tarea 2 : HexaPentaTetra

Breve descripción del problema

El problema consiste en simular un videojuego llamado “Super Hexagon”, el cual está disponible en varias plataformas incluyendo Windows y Android.

Este juego consiste en una figura central (hexágono, pentágono, cuadrado) que va rotando. Mientras esta figura rota, aparecen bloques que se dirigen hacia las aristas de la figura central y deben ser esquivados por un “cursor” el cual es controlado por el jugador.

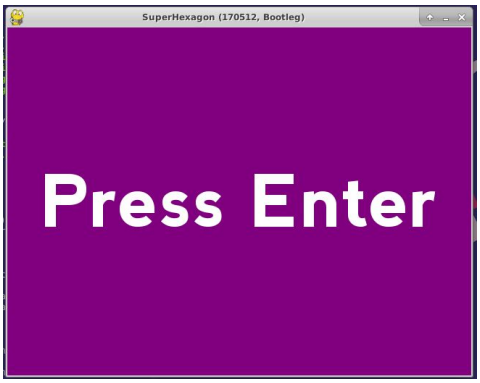
Apenas un bloque colisione con el cursor del jugador, el juego se termina.

Descripción de la solución.

La mayoría de la solución está basada en la clase “CenteredFigure” provista por el auxiliar Pablo Pizarro.

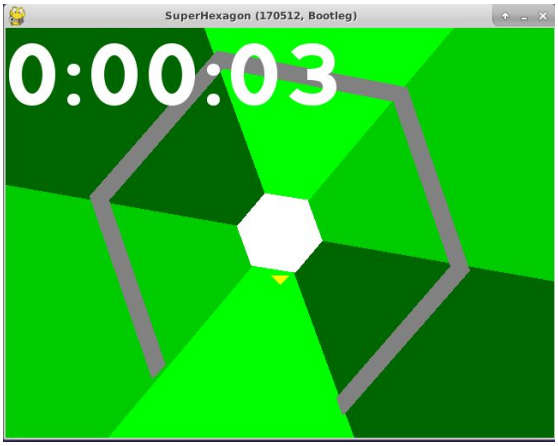
Esta clase contiene un par de métodos que son bastante útiles para comenzar a modelar el videojuego, en particular la función de escala, rotación e colisión.

Pygame se utilizó para crear la ventana y los comandos del juego (mover el cursor, reiniciar el juego, salir, etc)



Otra clase provista por el auxiliar fue pygame-menu, la cual se encarga de tener una abstracción simple para la creación de un menú para aplicaciones.

Mi solución en particular se enfocó en el core del videojuego más que en detalles.



Las features implementadas son:

- Timer en el juego
- Si el jugador sobrevive 60 segundos, gana el juego
- El sentido de rotación cambia despues de cierto tiempo, dependiendo de la dificultad
- Las distintas dificultades modifican el movimiento del cursor y de background
- Uso de MVC
- Tecla ESC cierra el videojuego (vuelve al menú) y R reinicia el juego (reiniciando el timer manteniendo la dificultad)

1.- Indique explícitamente el uso del MVC. ¿Qué funciones cumplen la Vista y el Controlador en su tarea?

El uso de Modelo Vista Controlador está dado por las siguientes componentes:

Modelos:

- CentralFigure : Objeto Central
- Background: Objeto con fondo de colores.
- Attack: Objeto que contiene bloques de ataque generados

Vista y Controlador:

La mayoría de las vistas están en parte de los modelos (funciones draw() y otras).

En el archivo tarea2.py, se encuentra todo el controlador del videojuego. Este controlador tiene dependencias en los modelos CentralFigure, Background, Ataque, pygame-menu y CenteredFigure.

El controlador contiene las acciones del usuario, el menú, la ventana contenedora y la inteligencia detrás del videojuego.

2.- Explique y comente los modelos que utilizó, ¿Por qué decidió hacerlo en dicho modo?

Apoye su explicación con un esquema del modelo.

Los 3 modelos que utilicé heredan de CenteredFigure, esto es más que nada para mantener un estándar de las funciones de rotación y escala.

- CentralFigure: Figura Central del videojuego. Como debe estar permanentemente rotando, necesita tener su propio método de rotación. Este objeto se instancia con el número de aristas (o vértices) que requiera. De 3 a 6 está probado que funciona bien. Se sitúa en la mitad de la ventana (de 640x480, se centra en 320,240). No tiene función de cambio de figura, por lo que por cada juego nuevo se genera al azar una figura nueva con un nuevo número de aristas.

- Background: El fondo del videojuego, como CentralFigure, también se instancia respecto al número de aristas. Si el número de aristas excede 4, se utilizan 3 colores para pintar el fondo. El background tal como CentralFigure también se debe estar moviendo constantemente, por lo cual tiene métodos para rotación.

- Attack: Los bloques que atacan estan en este sector. En particular hice que los ataques fuesen en todos las aristas menos 1 elegida al azar, por lo que los caminos de escape son más difíciles que lo que deberían. La generación de estos bloques no está totalmente correcta puesto que no siempre están alineados con las aristas de la figura central. El movimiento de los bloques se hace através del método de escala de CenteredFigure utilizando un factor menor a 1.

3.- Describa cómo decidió implementar las colisiones. ¿Es eficiente? ¿Cómo implementaría un método más eficiente para detectar colisiones?

Las colisiones están implementadas utilizando la función “collides” o intersects de CenteredFigure, en cada iteración del while principal del juego pregunto si el cursor (triángulo) intersecta con todos los ataques que están declarados en el programa en ese minuto.

No es eficiente puesto que en ninguna parte de mi código mato los ataques no colisionados, mi código pregunta por choques durante todo el trayecto de los ataques. Para implementar algo más eficiente, debería preguntar por ataques durante ciertos rangos de pixeles (relacionados al movimiento del triángulo) y eliminar los ataques que no colisionaron con el cursor para gastar menos ciclos en las iteraciones de preguntas de colisión.

4.- Describa cómo implementó la generación aleatoria de bloques. ¿Su solución permite añadir más figuras centrales con distinto número de aristas?

La generación aleatoria de bloques consiste en instanciar objetos de clase Ataque con cierto número de aristas. Cada n segundos (puede ser 1 o 2 o más) mi código agrega un nuevo ataque al arreglo de ataques del programa principal.

El objeto de ataque se genera utilizando 1 bloque (de una región definida por las aristas de la figura principal) y luego copiandola hacia las demás aristas. Un número al azar define qué arista no tendrá bloque de ataque (por ende un camino para que el jugador pueda sobrevivir).

Estos bloques de ataque como están definidos por el número de aristas, pueden soportar triángulos, cuadrados, pentágonos, hexágonos, heptágonos etc. siempre y cuando sean polígonos regulares.

Conclusiones.

Si bien no completé al 100% la tarea (No se cambia la figura central una vez ejecutado el videojuego), noté lo estresante que es hacer calzar los objetos en un videojuego.

El modelamiento tampoco me quedó 100% exacto, los ataques no calzan bien con las aristas de la figura central ni el background en todas las instancias, lo cual me hace pensar que quizás los ataques deberían estar controlados por el modelo de figura central o background, pero no alcancé a implementar esto.

La clase CenteredFigure me ayudó bastante para la realización de esta tarea, sin ella tendría que haber consultado mucha documentación de pygame respecto a las operaciones de escalamiento y rotación. Muchas gracias a Pizarro por su aporte al curso.