Tarea 2 A

HexaPentaTetra

Pablo Pizarro R.
ppizarror1@gmail.com

Pablo Polanco
paropoga@gmail.com

Introducción

Esta tarea consiste en diseñar y programar un juego similar al popular *Super Hexagon*, en donde el usuario debe controlar un pequeño cursor en medio de la pantalla para evitar diversos bloques que se aproximan a la figura geométrica del centro [1]. Esta figura central delimitará las regiones por las cuales se aproximan los bloques. Las figuras que se piden en la tarea son: Un hexágono, un pentágono y un cuadrado [2].

Figura 1: Imagen del juego

[1] Si el cursor que maneja el usuario choca con un bloque entonces pierde el juego

[2] Se recomienda ver el funcionamiento del juego en YouTube. Este juego además se encuentra disponible en Steam, Android y iOs. `https://www.youtube.com/watch?v=ceKMbVfex1Q`

Descripción del juego

Una vez que el juego empieza se comenzará con una figura central (cuadrado, pentágono o hexágono) el cual debe ir cambiando de forma de manera aleatoria a medida que pasa el tiempo. En cada región que genera la figura (si es un cuadrado se generan 4 regiones, si es pentágono 5 y si es un hexágono son 6 regiones) aparecerán unos bloques que se irán aproximando a la figura central (hasta desaparecer cuando estos bloques atraviesen la arista de la figura central).

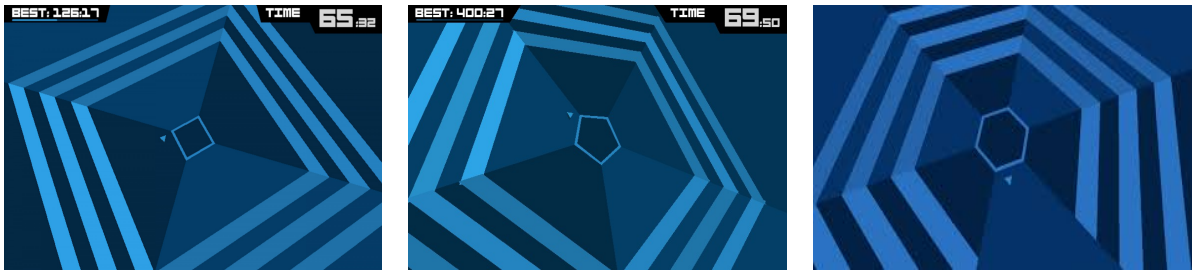
El usuario debe moverse con las flechas *izquierda* y *derecha* para moverse en rededor de la figura central (en una trayectoria circular, de radio fijo) con el fin de evitar estos bloques. A medida que el tiempo pasa la velocidad con la que estos bloques se aproximan es incrementada. Además de este movimiento se tiene que la cámara rota al rededor de la figura central, con lo cual se tiene un juego de gran dificultad a pesar de la sencillez de los controles.

El usuario ganará cuando llegue a una cantidad determinada de segundos sin chocar con un bloque. Pueden ser 60, 120 o 240 segundos, por ejemplo. Esta cantidad de segundo está directamente relacionada con la dificultad con la que se juege, si la dificultad es baja entonces se requiere de un menor número de segundos para ganar.

Dibujar la escena

Se pide implementar un programa escrito en Python, Pygame u PyOpenGL (implementación de OpenGL con Python e Pygame) el cual permita representar la escena 2D que define el juego. Esta escena debe considerar:

1. Una figura central, la cual puede ser un cuadrado, un pentágono o un hexágono (ver figura 2). **(0.5pt)**
2. El fondo de la aplicación debe tener a lo más 3 colores [3], e irán pintando cada región de manera que no hayan 2 regiones con el mismo color (ver figura 2 [4]). Estos colores además de ser estéticos permiten delimitar las regiones por las que los bloques se aproximarán. **(1pt)**



(a) (b) (c)

Figura 2: Ejemplo con un cuadrado (a), un pentágono (b) e un hexágono (c)

[3] Dese cuenta que con 3 colores se pueden pintar todas las aristas de cualquier polígono regular sin que hayan dos aristas con un mismo color.

[4] En esta figura se puede observar que al haber un cuadrado como figura principal sólo se tienen 2 colores en el fondo, en un pentágono se tienen 3 colores y en un hexágono sólo se necesitan 2 para pintar todo el fondo

3. Un cursor que girará en torno a la figura central (en un radio fijo que permita girar en torno a la figura central sin chocar con ella). Puede ser un triángulo, un cuadrado, o alguna figura geométrica que usted estime conveniente. **(0.5pt)**

4. La escena también debe contemplar los bloques aleatorios [5] que se acercan al usuario, estos deben tener un color acorde con la escena, pero que sean distintos a los colores del fondo (para que el usuario no se confunda). **(1.0pt)**

5. Por último se debe dibujar un temporizador en alguna esquina de la ventana, la cual indicará cuanto tiempo ha jugado el usuario en la partida actual. **(0.5pt)**

NOTA: Todas las figuras geométricas deben ser generadas utilizando Pygame o PyOpenGL. No se permite el uso de texturas externas (png, gif, etc).

Implementación

Al programa implementado en el punto anterior, se deben agregar las siguientes funcionalidades:

- Un menú para elegir la dificultad del juego, las dificultades deben ser las siguientes: **Dificil**, **Muy dificil** e **Imposible**. A medida que la dificultad se incrementa aumenta la velocidad de giro del cursor (el que maneja el usuario), además aumenta la velocidad de rotación de la cámara y la velocidad con la cual los bloques se aproximan al usuario. **(0.5pt)**
- Con los botones `FLECHA IZQUIERDA` y `FLECHA DERECHA` se controlará el movimiento del cursor con el fin de evitar los bloques aleatorios. **(0.5pt)**
- La cámara debe girar en torno a la figura central. **(1.0pt)**
- Al presionar `ESC` se cierra el juego y al presionar `R` se reinicia el juego (comenzando desde cero manteniendo la misma dificultad). **(0.5pt)**

Bonus

Además de los puntos anteriores usted puede hacer los siguientes bonus, **MAX 1pto**:

- Agregar música al juego. **(0.3pt)**
- Añadir un *scoreboard*. Cada vez que un usuario muera se le pedirá su nombre y este se añadirá a una lista de los 10 mejores puntajes (entendiendo que mientras más el usuario dure en tiempo mayor será su puntaje). Este scoreboard debe ser desplegado en pantalla utilizando Pygame. **(1.0pt)**
- Hacer que la paleta de colores cambie al cambiar la figura central, esto es que cada vez que se pase de un cuadrado a un pentágono, o de un hexágono a un pentágono (de forma aleatoria) todos los colores de los bloques, el cursor, el fondo o los bloques cambie. **(0.5pt)**
- Añadir 3 tipos de figuras distintas además del cuadrado, el pentágono y el hexágono. **(0.5pt)**

[5] No necesariamente todos deben ser aleatorios, también usted puede programar secuencias de bloques las cuales se ejecutarán de forma *aleatoria* en el transcurso del juego.

- Permitir el movimiento del cursor con el `MOUSE`, en este caso el cursor apuntará al mouse lo cual le permitirá esquivar los bloques de una manera más sencilla. **(0.6pt)**

Informe

El informe debe incluir:

- Breve descripción del problema.
- Descripción de la solución.
- Responda las siguientes preguntas:
  - Indique explícitamente el uso del MVC. ¿Qué funciones cumplen la Vista y el Controlador en su tarea?
  - Explique y comente los modelos que utilizó, ¿Por qué decidió hacerlo en dicho modo? Apoye su explicación con un esquema del modelo.
  - Describa como decidió implementar las colisiones. ¿Es eficiente? ¿Cómo implementaría un método más eficiente para detectar colisiones?
  - Describa cómo implementó la generación aleatoria de bloques. ¿Su solución permite añadir más figuras centrales con distinto número de aristas? [6].
- Conclusiones.

Formato PDF: Los informes entregados en otro formato serán evaluados con nota **1.0**

Fecha de entrega: Miercoles **3** de Mayo a las 23:59.

Condiciones de entrega

- La tarea es individual y las copias serán penalizadas.
- Entregas solo vía U-Cursos.
- Adjuntar archivo Readme.txt con instrucciones de ejecución.
- Recuerde adjuntar **TODOS** los archivos en cada entrega, ya que los auxiliares y ayudantes sólo tienen acceso a la última entrega de su tarea.
- **NO** se aceptarán archivos atrasados una vez el plazo haya finalizado, aunque esto implique que su tarea no ejecute.

[6] Es decir, que si se tiene un octaedro como figura central es necesario reescribir todo el código para tener los bloques aleatorios, o su solución permite la generación de regiones y bloques para cualquier número de aristas

Recomendaciones

- Revisar documentación y tutoriales de Python, Pygame y PyOpenGL para obtener en detalle las funciones disponibles y ejemplos de uso.
- Consultas a través del foro de U-Cursos o auxiliares.
- Sea ordenado con su código, agregue comentarios y utilice nombre útiles en sus variables y funciones.
- Planifique su tiempo y comience su tarea con anticipación. No comience a programar directamente. Comprenda el problema, realice esquemas y plantee un algoritmo de solución.
- Necesita hacer uso de la programación orientada a objetos para desarrollar esta tarea.
- Realice un esquema de cómo generar las figuras y cómo mover la cámara.