Tarea 2

Instrucciones Generales

- La tarea es estrictamente individual.
- Este enunciado presenta varias opciones, escoja e implemente solo una de ellas.
- Guarde todo su trabajo para esta tarea en una carpeta de nombre tarea1x, con x indicando la opción que haya escogido y entregue eso en U-cursos.
- El uso de Git es altamente recomendado, aunque no es obligatorio.
- DEBE utilizar: Python 3.7 (o superior), Numpy, OpenGL core profile, GLFW.
- *Las imágenes presentadas son solo referenciales, utilice un estilo propio para su trabajo.*

Entregables

- Código que implemente su solución (4.5 puntos)
  - Su versión final DEBE utilizar archivos *.py, NO jupyter notebooks.
  - Incluya TODO lo que su código necesita, incluyendo archivos proporcionados en cátedras o auxiliares.
- Reporte de documentación de entre 1 y 2 planas. Formato pdf. Detalles disponibles en primera clase. (1.2 puntos)
- Video demostrativo de 20-30 segundos. (0.3 puntos)

Opción A: Flappy Bird 3D

Nuestro Flappy Bird 2D fue todo un éxito! Ahora usted decide regalarle el juego a su mejor amigo en una versión 3D para navidad. El nuevo Flappy Bird debe ser todo un éxito ¡Incluirá luz, cámaras y acción!

Su juego se debe ejecutar con la siguiente llamada:

```
python flappy_bird.py N L
```

Siendo N el puntaje necesario para que el jugador gane.
Siendo L el tiempo de un ciclo de un día y noche en segundos.

Considere lo siguiente:

- Se recomienda hacer un paisaje simple y agregar las tuberías por un camino central como se muestra en la imagen.
- De manera predeterminada debe existir una cámara por detrás de la nuca de nuestro personaje, que esté apuntando hacia delante. Este modo se vuelve a activar al presionar la tecla 1 (u otra que estime conveniente)
- Al presionar la tecla 2 (u otra que estime conveniente) se activa una cámara lateral, que ve al personaje de lado, que permite verlo en el mismo plano que las tuberías, como si fuera 2D.
- Al presionar la tecla 3 (u otra que estime conveniente) se activa una cámara en primera persona, es decir, se ve lo que ve el Flappy Bird.
- Existe una un sol que emite luz, que se va desactivando y activando según un parámetro L, simulando un sol y una noche. Se recomienda definidir dos luces, una de día pleno y otra de noche plena, y aplicar una interpolación entre ambos, tal que tome un tiempo L, pero la implementación es libre.
- El personaje va cayendo constantemente, también puede saltar como en el juego 2D anterior, subiendo hasta que la gravedad lo vuelve a hacer caer. Las velocidades se pueden suponer constantes o se puede usar aceleración.
- Puede elegir un modelo 3D descargado o hecho en OBJ/OFF para el Flappy Bird o la tubería, pero NO para ambos modelos.
- El modelo del Flappy Bird no puede ser una simple esfera, su amigo se hará bolita y llorará.
- El modelo del Flappy Bird debe mirar ligeramente hacia arriba cuando salta, y mirar ligeramente hacia abajo mientras cae. Este efecto se ve reflejado en la cámara en primera persona, la cual también mira más hacia abajo o arriba según sea el caso.
- Se considerará un bonus si permite mover hacia donde apunta la cámara con el mouse tanto en primera como en tercera persona (sin considerar la vista de lado).

Puntuación

- Iluminación: 1.0 puntos
- Cámaras: 1.5 puntos
- Lógica (Que se cuenten los puntos, que haya colisiones, que se gane y se pierda): 1.0 puntos

- Transformaciones Modelos (Rotación del Flappy Bird, traslaciones de las tuberías): 0.5 puntos
- Modelos: 0.5 puntos
- BONUS: 0.5 puntos [Nota máxima en 7.0, sirve también para errores en el informe o video]

Opción B: Lights Out

Usted quiere implementar una versión propia del nivel *Lights Out* del famoso juego de plataformas *Crash Bandicoot*. Para ello usted usará OpenGL y todo el conocimiento que ha adquirido en el curso de Computación Gráfica. Para recordar el nivel, puede ver el siguiente video

Su juego se debe ejecutar con la siguiente llamada:

```
python lights_out.py X Y Z
```

donde X corresponderá al largo que tendrán las zonas de saltos en el juego, Y corresponderá al largo del nivel en cantidad generaciones, y Z corresponderá al tiempo que la iluminación dura.

Considere lo siguiente:

- Debe implementar los modelos necesarios para el juego, es decir, al personaje principal, el suelo, paredes y objetos decorativos para el nivel, usando Opengl. Puede utilizar modelos 3D hechos en OBJ, pero solo para un objeto dentro del juego.
- El personaje principal debe saltar con la tecla *espacio*.
- Deberá utilizar texturas dentro del juego, al menos 1.
- Su juego deberá contar con dos cámaras, una en tercera persona tal como el juego original, y una en primera persona. Por defecto debe usar la cámara en tercera persona, y con la tecla L alternar entre las dos cámaras.

- Para el movimiento del personaje, considere que la velocidad con que el personaje se mueve en el eje vertical (eje Y) es constante. Esto le ayudará a simplificar el movimiento de los saltos.

- Debe implementar los casos de colisión en el juego para que el jugador no pueda *salirse* del nivel (por ejemplo en paredes, suelo, etc).

- Para los obstáculos de su juego, en un principio solo habrán caídas (en donde si el personaje cae, perderá). Si usted desea agregar otros obstáculos, puede hacerlo. Para esto, deberá generar aleatoriamente algunas características del mapa. Su mapa deberá ser una combinación intercalada de suelos y saltos, es decir, después de una parte de salto siempre vendrá un suelo, y viceversa (Esto para evitar que se generen dos saltos seguido y que el personaje no pueda saltar por la distancia). Con respecto a los largos de estos, X corresponde al largo que tendrán las zonas de salto, y con respecto al suelo, este se deberá generar con un largo aleatorio entre X y 2X.

- El suelo posee 4 tipos: suelo completo, suelo a la izquierda, suelo a la derecha, y suelo centrado. La diferencia de estos suelos es la manera en que cubren el espacio de izquierda a derecha. El suelo completo cubre desde la pared izquierda a la pared derecha. El suelo a la izquierda cubre 1/3 del espacio, pero apegado a la pared izquierda. El suelo a la derecha similar al anterior pero apegado a la derecha, y el suelo centrado cubre igualmente 1/3 del espacio pero centrado (no adyacente a ninguna pared). Cada vez que usted genere un suelo, deberá escoger aleatoriamente uno de estos tipos.

- El nivel deberá estar completamente a oscuras. Cuando el jugador presione la tecla *E*, deberá iluminarse en un radio limitado alrededor del personaje, para que así solo pueda ver los obstáculos cercanos a él, y no los que están lejos (al igual que el juego original). Usted debe definir como realizar este comportamiento, pero debe cumplir lo anterior.

- Para que el jugador no siempre esté iluminado, esta luz durará Z segundos. Cuando se acabe, la luz dejará de iluminar, apagándose instantáneamente, y el jugador no podrá ser capaz de ver nada en el nivel.

- El jugador podrá volver a activar la luz, luego de 5 segundos de que la luz se haya desactivado.

- Debe definir un final de nivel, para que cuando el jugador logre ganar, muestre una pantalla que indique que el jugador ganó. De la misma forma, cuando el jugador pierde, debe existir una pantalla de game over.

- BONUS (Este bonus reemplaza al punto donde se habla de la reactivación de la luz) [0.5 de bonificación]: El jugador podrá volver a activar la luz siempre y cuando haya avanzado más allá de donde activó la luz por última vez. Para evitar que el jugador avance, y luego quede esperando por la luz de nuevo, usted deberá asegurarse que el

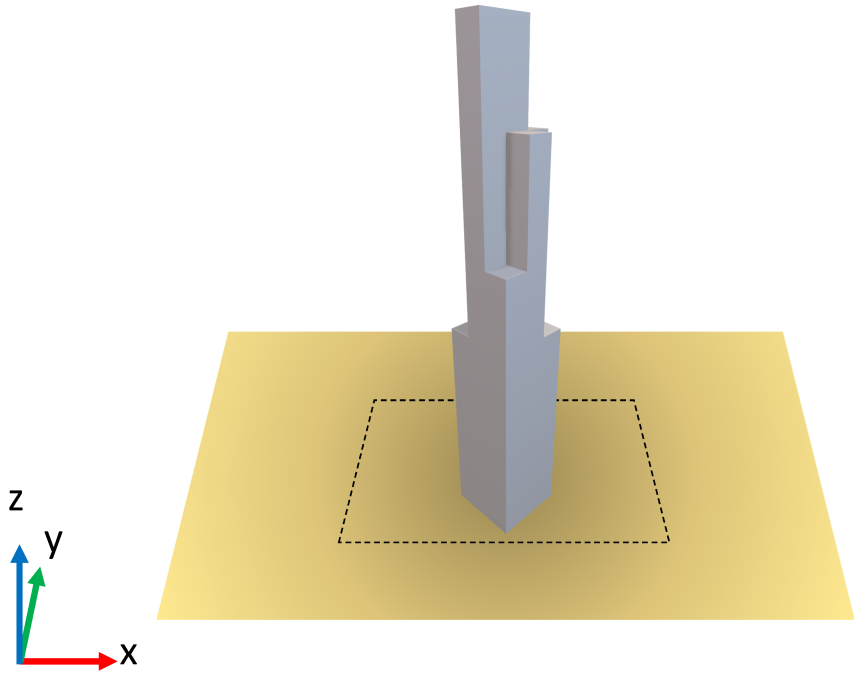
personaje no se quede más de 3 segundos dentro del mismo lugar (o generación en este caso) (Ver imagen adjunta).

Puntuación

- Modelos: 0.5 puntos
- Generación aleatoria del nivel: 1 punto
- Movimiento del personaje: 0.5 puntos
- Colisión del personaje: 0.5 puntos
- Iluminación y su lógica: 1 punto
- Cámaras: 0.5 puntos
- Pantalla de ganador y game over: 0.5 puntos
- BONUS: 0.5 puntos

Opción C: Visualizador de edificios

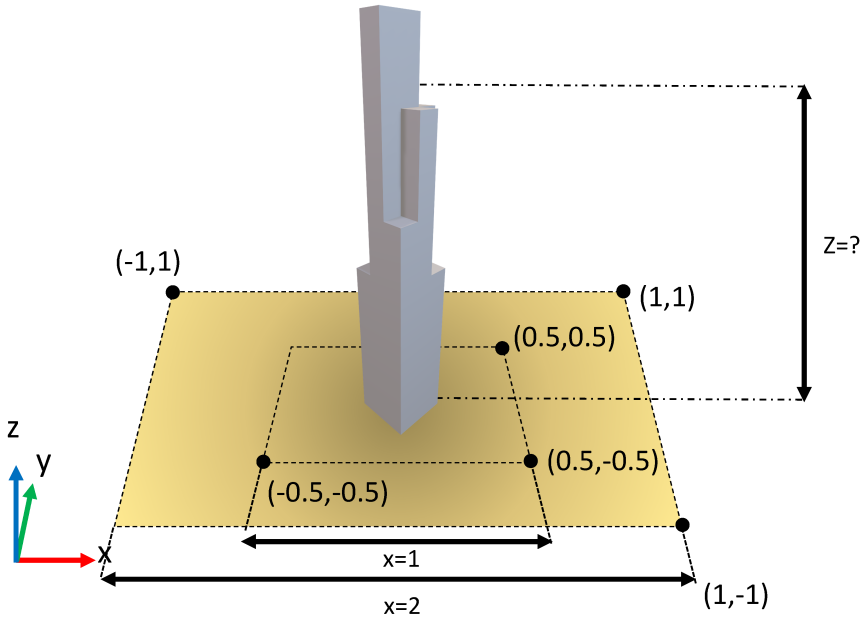
El objetivo de esta tarea es implementar un visualizador de distintos edificios icónicos del mundo, con el objetivo de servir como un programa interactivo para la importante conferencia a desarrollar en alguna parte del mundo. Este visualizador posee tres estructuras distintas las cuales se pueden intercambiar (Usando distintas teclas), cada edificio es dibujado en un plano, tal como se ilustra en la Figura.



Su programa se debe ejecutar con la siguiente llamada:

```
python building_viewer.py
```

- El modelo se debe representar en un mundo de tamaño $2x2$, en donde el edificio está centrado y posee una dimensión en planta (Plano xy) máximo de $1x1$, la altura de la estructura puede ser cualquiera, la que estime conveniente. La siguiente figura ilustra una parametrización del modelo:



- Los edificios deben ser modelados utilizando las metodologías vistas de OpenGL 3D, el color queda a su criterio.
- Se deben modelar los siguientes edificios:

  1. Empire State Building, ubicado en la icónica ciudad de New York, Estados Unidos.
  2. Willis Tower, Chicago, Estados Unidos.
  3. Burj Al Arab, Dubai, Emiratos Árabes Unidos.

- El plano del edificio debe considerar la textura en donde éste está ubicado (recortarla de google maps).
- La iluminación (color y posición) debe cambiar segun dónde esté ubicado el edificio. Por ejemplo, en Dubai debe ser más amarillo.

Considere que:

- El visualizador permite ver solo un edificio a la vez.
- Configure estratégicamente 4 cámaras para visualizar la totalidad de cada sector modelado. Las teclas `1`, `2`, `3` y `4` deben acceder a dichas cámaras. Debe haber cámaras con distintos tipo de vista (perspectiva y ortográfica)

- Además, configure una cámara móvil, que se mueva en un radio alrededor del edificio. Esta cámara, también debe poder desplazarse hacia arriba y hacia abajo, trasladándose en forma de cilindro. Para utilizar esta cámara deberá pulsar la tecla número 5. Esta camara siempre debe apuntar hacia el edificio.
- La tecla `E` permite visualizar el edificio Empire State, la tecla `W` permite cargar el edificio Willis Tower, por último con `B` se carga el edificio Burj Al Arab. Al cargar un edificio también debe cambiar la textura del plano.
- NO puede usar OBJ para simular un edificio totalmente, como mucho podrá usarlo para una parte de estos.
- Configure una luz intensa para modelar el sol, y que al pulsar la tecla `L`, esta luz pase a simular la luz de la luna. Al pulsar L, la luz debe comenzar a interpolarse, hasta llegar a la otra luz. Es decir, su escena estará de día, y al pulsar la tecla, su escena de a poco irá cambiando la iluminación, hasta llegar a simular la noche.

Puntuación

- Iluminación: 1.0 puntos
- Cámaras: 1.5 puntos
- Texturas: 0.5 puntos
- Modelos: 1.0 puntos