Tarea 2

Instrucciones Generales

- La tarea es individual.
- Este enunciado presenta varias opciones, escoja e implemente solo una de ellas.
- El plazo de entrega se indica en tareas/u-cursos.
- Esta tarea debe ser trabajada y entregada en un repositorio privado git bitbucket, proporcionando acceso al equipo docente. Instrucciones detalladas en guía-git-bitbucket.pdf disponible en material docente.
- Guarde todo su trabajo para esta tarea en una carpeta de nombre tarea2x, con x indicando la opción que haya escogido.
- DEBE utilizar: Python 3.5 (o superior), Numpy, OpenGL core profile, GLFW.
- *Las imágenes presentadas son solo referenciales, utilice un estilo propio para su trabajo.*

Entregables

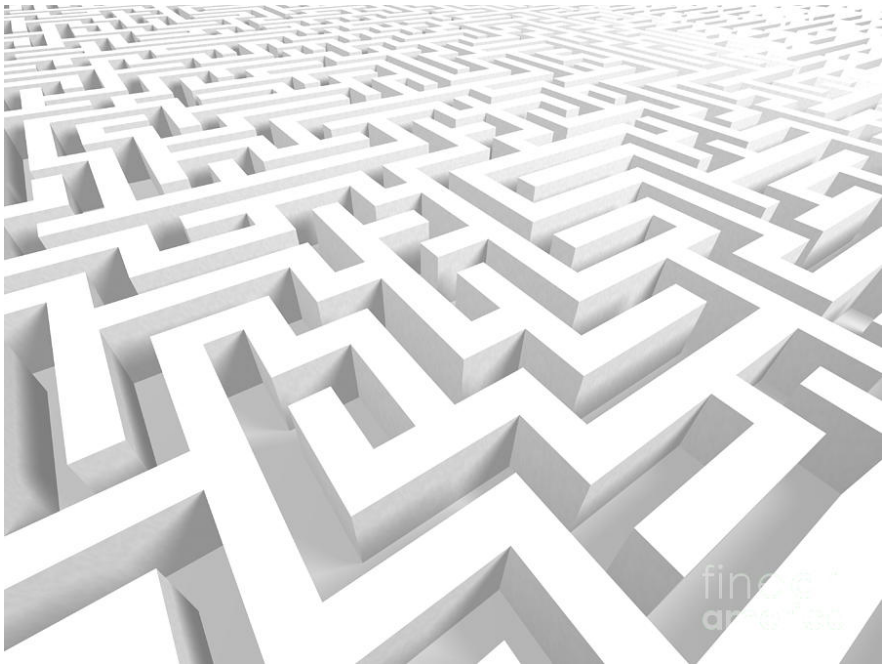
- Código que implemente su solución (4.5 puntos)
  - Su versión final DEBE utilizar archivos *.py, NO jupyter notebooks.
  - Incluya TODO lo que su código necesita, incluyendo archivos proporcionados en cátedras o auxiliares.
  - Al menos 5 commits en su repositorio git remoto ilustrando progreso en su trabajo. Nota 1 si no cumple con este item.
- 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)
- Todo lo anterior debe estar disponible en su repositorio git bitbucket remoto.

Objetivos

- Ejercitar el uso de OpenGL core profile en una aplicación en 3 dimensiones simple.
- Ejercitar curvas, texturas, control de cámara, iluminación e interacciones con el usuario.
- Consolidar uso de transformaciones, modelación jerárquica y el uso del patrón de diseño Modelo-Vista-Controlador.

Opción A: Laberintos en 3D

Don Pedro, aburrido de la *planitud* del espacio 2D de los laberintos, ha decidido explorarlos ahora en 3D. Como no se verá todo el escenario, el desafío es aún mas intenso.



Observación: No se requiere que haya realizado la tarea 1a. Puede construir los archivos *.npy requeridos con un pequeño script Python.

Especificaciones

En esta tarea debe implementar un programa que permita jugar los mismos laberintos creados con el editor de laberintos de la tarea 1a y almacenados en archivos *.npy. La llamada a su programa debe ser:

```
python maze3d_play.py maze.npy
```

Considere que:

- El laberinto *maze.npy* debe ser cargado. Cada celda debe ser modelada como un cubo representando los muros que impiden el paso. Debe añadir al menos un plano para el suelo.
- Debe considerar modelos tridimensionales para los tesoros. Además, decore la escena utilizando distintas texturas.

- Si no se ha definido la posición de inicio, o no hay tesoros, o cualquier otro tipo de inconsistencia, se debe acusar error.
- La vista del juego es en primera persona, es decir, la cámara son los ojos de Don Pedro. En este caso no se requiere un modelo para Don Pedro.
- Al presionar las flechas del teclado, Don Pedro (i.e. la cámara) se debe mover suave y continuamente hacia los lados o hacia adelante y hacia atrás. Al mover horizontalmente el mouse, Don Pedro debe rotar la vista hacia los lados.
- Don Pedro recoge un tesoro al acercarse lo suficiente. El tesoro debe simplemente desaparecer.
- Cuando se acaben todos los tesoros, el juego termina y se debe imprimir en consola el tiempo total transcurrido desde que se inició el juego.
- Por supuesto, Don Pedro no posee la habilidad de atravesar muros, por lo que el juego debe preocuparse de esta restricción.

Modo Nocturno:

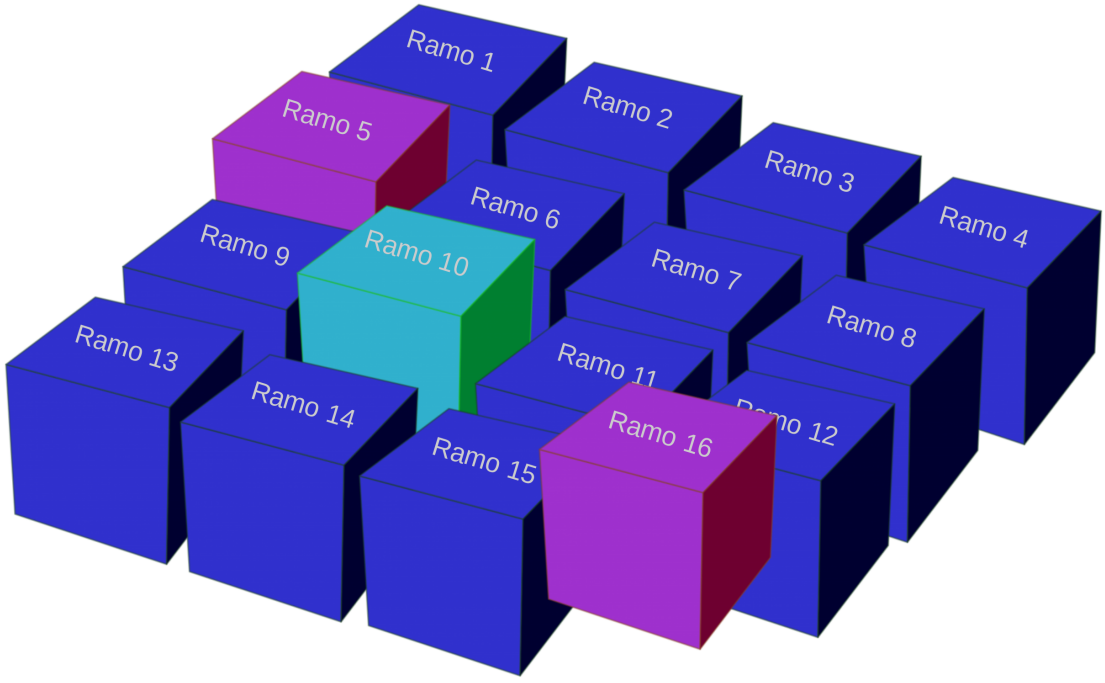
- Como dificultad adicional, estos laberintos cuentan con el modo nocturno. En este modo, Don Pedro posee una vela (o fuente de luz puntual omnidireccional) que permite iluminar su entorno cercano, mientras que el laberinto no se verá en la distancia.
- Para este modo debe usar la técnica de sombreado de Phong (o Gouraud, si su computador no es compatible). Recuerde que en ambos casos se necesitan las normales a las superfícies.
- Al presionar *espacio* se alterna entre los modos normal y nocturno. Es decir, con *espacio* debe alternar el programa de shaders en uso.

Puntuación

- Visualización de laberintos: 1 punto
- Movimiento continuo: 0.5 puntos
- Modo Nocturno (Iluminación): 1 punto
- Modelos: 0.5 puntos
- Recolección de tesoros: 0.5 puntos
- Colisión con muros: 1 punto

Opción B: Explorador de malla curricular

El objetivo de esta tarea es implementar un visualizador de la malla curricular del DCC (o carrera de preferencia), con el objetivo de servir como un programa interactivo para que los alumnos puedan ver los requisitos y otra información de los ramos.



Considere lo siguiente:

- Se selecciona el Ramo 10, el Ramo 5 es requisito del Ramo 10 y a su vez el Ramo 10 es requisito del Ramo 16.
- Inicialmente se encuentra seleccionado un curso por default (da lo mismo cual, puede ser fijo o seleccionarse de manera aleatoria). Para cambiar el curso seleccionado, el usuario utilizará las teclas A (izquierda), S (abajo), D (derecha) y W (arriba).
- Se deben incluir bloques 3D que representan a cada ramo.
- Se debe agregar el nombre y código de cada curso como textura utilizando una imagen con esa información y un color de fondo (mismo color para todos los cursos).
- Al seleccionar un curso, éste debe sobresalir de la malla y cambiar de color. Al mismo tiempo, todos los cursos que son requisito del curso seleccionado o que tienen como requisito al curso selecionado deben sobresalir también y cambiar a un tercer color (diferente del color del curso seleccionado y de los cursos no relacionados).

- Se debe incluir un fondo con una textura de su preferencia.
- Se deben agregar los siguientes controles de cámara:
    1. Una cámara que pueda moverse continua y suavemente en un plano superior al de la malla mirando siempre hacia abajo (i.e. hacia la malla). Este movimiento es en 2 ejes de coordenadas cartesianas, y ocurre utilizando las flechas del teclado.
    2. Una cámara que persiga el curso activo. Esto es, al presionar 1, la cámara debe moverse desde su posición actual hasta ubicarse frente al curso actual seleccionado. Este movimiento debe seguir una trayectoria dada por una curva de las vistas en clases. Usted debe generar parámetros convenientes que permitan dicha animación.
    3. Al presionar 2, se activa o desactiva el modo de *seguimiento continuo de la cámara*. En este modo, la cámara siempre seguirá al curso activo a través de una trayectoria curva (de la misma forma que se especifica en el punto anterior). Luego, las flechas del teclado no tienen efecto en este modo.

Puntuación

- Modelos 3D con texturas: 1.5 puntos
- Configuración de cámaras: 1.5 puntos
- Animación de requisitos: 1.5 puntos

Opción C: Roller Coaster

En esta tarea usted debe modelar un Roller Coaster utilizando curvas de Catmull-Rom.

Su programa de debe ejecutar como:

```
python roller-coaster.py track.csv
```

Donde el archivo track.csv posee 3 números separados por comas en cada línea

```
1,0,0
2,0,0
6,0,0
7,1,0
6,2,0
0,0,4
0,0,1
```

Considere lo siguiente:

- Su programa debe leer y cargar el archivo track.csv con la descripción de la pista del roller-coaster.
- Para su conveniencia, la pista es un simple mono-riel, por lo que puede ser modelado por pequeños trozos cilíndricos convenientemente transformados para ilustrar el circuito.
- Debe configurar 2 cámaras estáticas que permitan ver todo el roller-coaster desde distintos ángulos. Presionando 1 y 2 se accede a dichas cámaras.

- Debe modelar un carrito que navegará por el roller-coaster. Este carrito debe tener un nivel razonable de detalle.
- Con la tecla *espacio*, el carrito comenzará a moverse a través del roller-coaster.
- Su escenario completo debe estar embebido al interior de una caja *sky-box* que modele un fondo razonable (Ej: bosque, ciudad, montañas y nubes).
- Al presionar el número 3, se activa una cámara móvil que persigue al carrito en tercera persona. Esto es, si el carrito se mueve, la cámara se traslada y rota con él según la pista. Así se observará la trayectoria completa, con toda la adrenalina que esto implica. *Hint:* El vector *forward* de la cámara tendría la misma dirección que la tangente a la curva.

Puntuación

- Trayectoria del Roller-Coaster y cámaras estáticas: 1.5 puntos
- Carrito: 0.5 puntos
- Movimiento continuo del carrito: 0.5 puntos
- Cámara en tercera persona: 1 punto
- Skybox: 1 punto

Opción D: Visita virtual al Partenón o Panteón

En esta tarea usted debe modelar el Partenón (Atenas) o Panteón (Roma), o en su defecto, cualquiera de las 12 casas de los caballeros dorados (Caballeros del Zodiaco). El objetivo es visitar virtualmente dichos lugares, para luego crear una animación de trayectoria virtual a través de ellos.

Partenón

Panteón

Considere lo siguiente:

- Debe implementar solo uno de los modelos propuestos. Para los modelos de fantasía, se espera una dificultad comparable a la del Partenón o Panteón. Cualquiera sea el modelo escogido, puede estilizarlo.
- Debe prestar especial atención a los pilares, estos deben modelarse utilizando curvas para generar volúmenes de revolución. Además, su Modelo debe estar correctamente texturizado.

- Como intentamos una visualización realista, la iluminación de la escena debe mostrar el avance de la luz del sol. Un ciclo de sol completo debe tardar 30 segundos. El movimiento del sol se debe poder pausar con la tecla *1*. El fondo debe ser de un simple color que varíe según el momento del día.
- Para lograr la iluminación de la escena, debe usar la técnica de sombreado de Phong (o Gouraud, si su computador no es compatible). Recuerde que se necesitan las normales a las superfícies.
- Implemente una cámara móvil que le permita navegar por todo el escenario utilizando las flechas del teclado (o las teclas W,A,S,D) y el mouse.
- La cámara se debe mover suave y lentamente permitiendo un preciso control de ella.
- La ruta recorrida por la cámara se debe poder almacenar en un archivo de texto csv (*comma separated values*). Para esto, su programa debe contar con un modo especial de grabación.
  - Con la tecla *enter* se activa y desactiva el modo de grabación.
  - Al activarse, se inicia el almacenamiento de la trayectoria de la cámara en un archivo de texto de nombre *camera0001.csv*. Los 4 dígitos finales irán en aumento si el archivo ya existe.
  - Cada 0.5 segundos, se almacenan en una línea 9 valores numéricos separados por comas, estos valores serían: 3 coordenadas para el vector *eye*, 3 para el vector *at*, y 3 coordenadas para el vector *up*. Al desactivar la grabación se termina un archivo.

Si el programa se ejecuta con un archivo de texto como argumento. Su aplicación debe iniciar una animación de la cámara siguiendo la ruta ahí definida. Ejemplo:

```
python panteon.py camera0002.csv
```

En este caso, se deben leer los valores para la trayectoria de la cámara: *eye*, *at* y *at*. El programa debe interpolar estos valores para producir una animación suave y continua a través de la trayectoria especificada. Esta interpolación debe implementarse utilizando splines de Catmull-Rom.

Puntuación

- Modelos: 1 punto
- Movimiento de cámara: 1 punto
- Ciclo del sol: 1 punto

- Modo grabación: 0.8 puntos
- Visualización de grabación: 0.7 puntos

Hints!

- Python posee librerías que le facilitan la lectura de archivos .npy, .csv, .json, etc...
- Utilice la tarea como campo de entrenamiento para consolidar sus conocimientos teóricos.
- Concéntrese primero en los requisitos mínimos, luego proceda con el refinamiento de su trabajo.