

TAREA 2: OPENGL3D

UNIVERSIDAD DE CHILE
FCFM - DCC

CC3501
5 DE MAYO DE 2020

INSTRUCCIONES GENERALES

- La tarea es estrictamente 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 Git remoto privado, proporcionando acceso al equipo docente. Instrucciones detalladas disponibles 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.*
- Cada opción de tarea tiene una distinta lectura asignada.

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 ítem
 - *Importante: el mínimo de 5 commits deben corresponder a progresos en el código. Agregar reportes o videos al repositorio no contabiliza.*
- 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 remoto.

OBJETIVOS

- Ejercitarse el uso de OpenGL core profile en una aplicación en 3 dimensiones simple.
- Consolidar conocimientos adquiridos en el curso
- Estudiar una lectura relacionada y aplicar dichos conocimientos en el desarrollo de la tarea.
- Trabajar con interacciones de usuario vía GLFW

OPCIÓN A: GENERADOR DE BOSQUES

Las técnicas fractales son ampliamente utilizadas para generar contenido de manera automatizada (procedural generation). Y de esta forma, se disminuyen los tiempos para generar abundantes modelos similares, pero distintos. En este tarea, generaremos árboles utilizando una estrategia fractal.



Lo primero, es estudiar la lectura asociada *fractals.pdf* disponible en material docente. Con estos conocimientos, debe formular una estrategia para generar un árbol en 3D en base a ciertos parámetros. Usted debe definir parámetros que le permitan la generación unívoca de un árbol en específico.

```
python tree.py model.obj [params]
```

Su programa deberá: (1) generar una visualización del modelo 3D generado y, (2) almacenar dicho modelo como un archivo OBJ con el nombre indicado. Su visualización debe considerar efectos de iluminación.

En una segunda etapa, queremos utilizar la funcionalidad anterior para generar un bosque completo. Para evitar un piso completamente plano, utilizaremos una superficie compuesta por la suma de varias gaussianas, o funciones similares lo suficientemente grandes.

```
python forest.py model.obj [5 params]
```

- Su segundo programa deberá: (1) generar una visualización del bosque completo generado y, (2) almacenar el modelo del bosque completo como un archivo OBJ con el nombre indicado.
- En este caso, usted debe definir 5 parámetros para controlar la generación del terreno. Buenos ejemplos pueden ser: cantidad de gaussianas, número semilla para funciones aleatorias, densidad de árboles por unidad de área, etc. La decisión sobre qué parámetros utilizar es suya.
- No pueden ser más de 5 para facilitar la usabilidad del programa.
- En este caso, no importa si la generación del bosque no es unívoca.
- Su visualización debe considerar efectos de iluminación y permitir un movimiento de cámara tal que permita observar la escena desde distintos ángulos.

PUNTUACIÓN

- Generación de árbol con estrategia fractal: 1.5 puntos
- Generación de bosque: 1 punto
- Visualización 3D (ambas aplicaciones): 1 punto
- Almacenar modelos en formato OBJ (ambas aplicaciones): 1 punto

OPCIÓN B: CRAZY RACER

Con el propósito de sorprender a los jugadores de hoy, ya no basta con monótonas pistas planas a nivel del piso... En este proyecto, implementaremos una pista considerablemente más sofisticada, y le permitiremos al usuario moverse a través de ellas en un simple pero potente vehículo de carreras.



```
python crazy-racer.py
```

Utilizando una *Rounded Nonuniform Spline* de la lectura *nonuniform_splines.pdf* (disponible en material docente) usted aprenderá a generar una curva apropiada para generar un circuito de carreras. La tarea faltante, es asignarle una sección transversal rectangular y con ella generar una malla 3D que permita modelar la pista completamente.

- Modele un único vehículo en 3D utilizando una mezcla de las siguientes tres técnicas: grafos de escena, texturas y efectos de iluminación.
- Modele una única pista con la estrategia descrita. La pista debe poseer al menos 5 puntos de inflexión, y presentar distintas elevaciones. Utilice un color plano o una textura repetitiva para colorearla. Debe considerar efectos de iluminación.
- Implemente el control del vehículo de manera que le permita moverse sobre la pista 3D. Se debe permitir el desplazamiento hacia adelante

o hacia atrás utilizando las flechas arriba y abajo; y rotar, tanto la cámara como el vehículo, según la dirección de las flechas hacia los lados. Note que la rotación de la cámara debe ocurrir en torno al vehículo, de manera que este no tenga traslaciones de drifting.

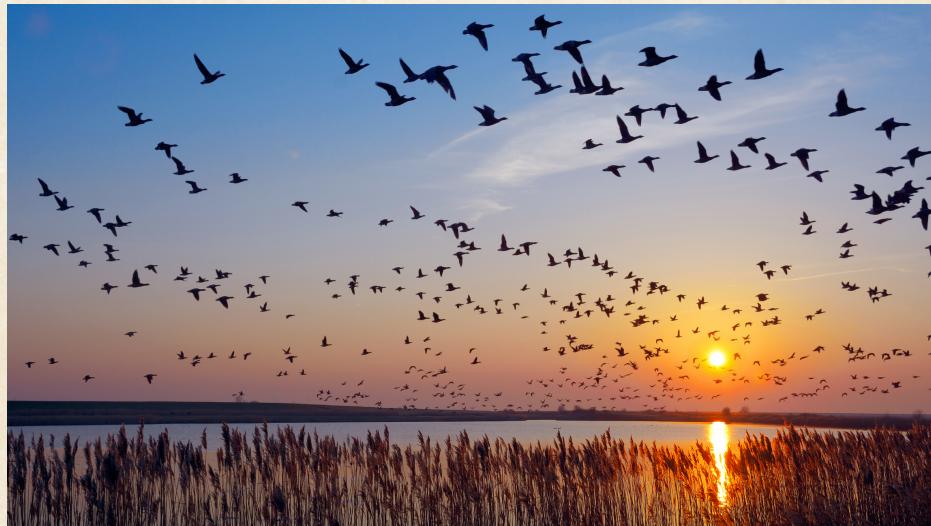
- Decore la escena con texturas y al menos 2 animaciones simples. Ejemplos: nubes, aves, humo proveniente del motor, contador de vueltas, etc...
- Si lo desea, puede considerar supuestos que simplifiquen el problema como: (1) el auto siempre estará dentro de la pista; (2) las secciones transversales de la pista son siempre paralelas al piso; etc.

PUNTUACIÓN

- Pista: 2 punto
- Modelo del vehículo: 1 punto
- Control del vehículo sobre la pista: 1 punto
- Decoración de la escena: 0.5 puntos

OPCIÓN C: MIGRACIÓN DE LAS AVES

En esta tarea modelaremos la migración de las aves utilizando un modelo articulado para animar el vuelo de cada una de ellas.



El primer paso, es estudiar la lectura asociada *skeletal-animations.pdf* disponible en material docente. Con esto, usted debe formular un modelo apropiado para modelar un ave. Debe considerar un mínimo de 3 articulaciones. Con este primer paso, generaremos un primer programa.

```
python bird.py
```

- Al mover el mouse hacia arriba o abajo, sobre la ventana de la aplicación, el ave debe mover sus alas.
- La visualización debe considerar efectos de iluminación

Luego, queremos animar 5 aves volando a lo largo de una trayectoria curvilínea especificada por el usuario.

```
python bird-herd.py path.csv
```

Considere lo siguiente:

- Debe configurar una única cámara que visualice la escena y que se pueda rotar hacia los lados y hacia arriba y abajo según el movimiento del mouse.
- Decore un escenario de paisaje con al menos 3 texturas distintas.
- El archivo *path.csv* incluye una secuencia arbitraria de puntos 3D. Estos puntos deben

ser interpolados por una spline de Catmull-Rom para formar la trayectoria de las aves.

- Utilice la trayectoria anterior y su modelo de ave, para instanciar al menos 5 aves que se muevan a través de ella. Por supuesto, las aves deben utilizar su animación interna de aleteo.
- Por supuesto, las aves deben estar separadas por alguna distancia razonable y su escena debe considerar efectos de iluminación.
- Usted debe proporcionar 3 archivos csv con distintas trayectorias que puedan ser observadas desde la ubicación de cámara que usted ha definido.

Consideré el siguiente archivo csv de ejemplo:

```
0,1,0  
0,1,1  
0,3,3  
0,2,1  
5,5,0
```

PUNTUACIÓN

- Modelo de ave y aleteo controlado: 1.5 puntos
- Animación de aves siguiendo la trayectoria: 1.5 puntos
- Trayectorias en archivos csv: 0.5 puntos
- Control de la cámara y decoración de la escena: 1 punto