

## 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.
- 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)
- Vídeo demostrativo de 20-30 segundos. (0.3 puntos)

## Objetivos

- Ejercitarse el uso de *OpenGL core profile* en una aplicación en 3 dimensiones simple.
- Dominar el uso de la cámara, modos de proyección, iluminación y texturas.
- Hacer uso del patrón de diseño Modelo-Vista-Controlador.
- Trabajar con interacciones de usuario vía GLFW.
- *Nota: No todas las opciones de tareas ejercitan TODOS los objetivos.*

## Opción A: Proyecto personal

La opción A comprende un proyecto personal. Para ello deben realizar, a más tardar el próximo miércoles **4 de Noviembre a las 23:59**, un documento de una a tres páginas en las que deben explicar su propuesta de tarea, el que debe ser subido a u-cursos en la tarea correspondiente (*Tarea 2A: Propuesta proyectos personales*). Guíese por los enunciados de las demás tareas (B, C, D, E) para:

1. Definir una introducción de su proyecto, adjuntar esquemas
2. Definir la escena 3D que harán, principales componentes, las luces, cámara, entre otros
3. Definir las mecánicas de la aplicación (input/output), la lógica que deben tener sus actores/modelos
4. Una carta Gantt con su trabajo (o bien una tabla) detallando la hoja de ruta de su proyecto, los principales hitos a realizar y en qué plazos ud. pretende llevar a cabo su desarrollo
5. Proponer un esquema de puntuación razonable

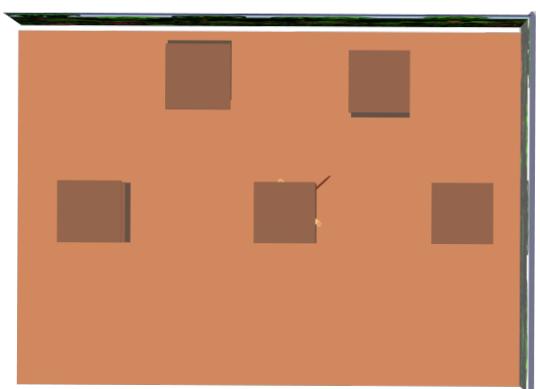
Una vez se hayan recibido las propuestas éstas se evaluarán y se dirá si corresponde o bien hay que modificar ciertas cosas (por ejemplo los componentes, las mecánicas o su esquema de puntuación). Se espera que el orden de dificultad de su proyecto sea comparable (o incluso superior) y que sea realizable dentro del plazo de la tarea; los proyectos personales tendrán el mismo tiempo de desarrollo que el resto de opciones.

## Opción B: Monkey Jump 3D

Esta vez el juego se ha actualizado, el monito deberá moverse en un ambiente 3D. Esta actualización traerá mayor dificultad, ya que nos encontramos en la selva, lugar donde hay enemigos que nos lanzan objetos desde los costados. Ahora hay plataformas falsas y se aumentó el número de columnas que contienen plataformas.



(a) Cámara en el frente



(b) Cámara de arriba

El juego se debe ejecutar con la siguiente llamada:

```
python monkey_jump.py structure.csv
```

Esta vez el archivo structure.csv estará compuesto por 5 columnas de datos, lo cual indica que habrá 5 columnas de plataformas. Si es un 0 indica que no existe plataforma. Si es un 1 indica que sí existe plataforma. Si hay una x indica que es una plataforma falsa, la cual desaparecerá si el personaje salta sobre ella.

El siguiente ejemplo de archivo csv se visualiza en la imagen anterior:

```
1,1,0,0,1  
0,1,x,0,1  
1,0,0,1,0  
1,0,1,1,x  
0,1,0,0,0
```

Considere lo siguiente:

- Su programa debe leer y cargar el archivo structure.csv con la descripción de la estructura del juego. La estructura visualizada debe seguir los mismos patrones mostrados en el archivo.
- El archivo csv no debe tener filas con sólo 1's, ya que eso no permite el avance del personaje. Ni tan poco filas con sólo 0's, ya que de esta forma no podría subir por las plataformas.
- La estructura tiene 3 columnas de plataformas en un plano mas adelante que las otras 2. También debe considerar las plataformas falsas dentro de la escena.
- Debe implementar un diseño para: el personaje, el premio, las plataformas y el fondo. El diseño de la escena no tiene restricciones (Se puede utilizar texturas y exportaciones de objetos, utilice su creatividad).
- Debe implementar sombreado e iluminación en la escena.
- Debe existir un movimiento de cámara para que se pueda visualizar el avance del personaje por la estructura.
- Con las teclas B, N y M deberá mostrar diferentes posicionamientos de la cámara. (Como por ejemplo los posicionamientos que se muestran en las imágenes)
- El personaje debe realizar movimientos horizontales y saltos verticales, utilizando las teclas A ,W , D. Pero en este caso el personaje debe hacer traslaciones con respecto a la profundidad de la escena, ya que no todas las plataformas están en el mismo plano (2 columnas mas atrás que las otras 3).

- Debe generar ataques desde los costados que impidan que el personaje suba. En esta actualización el personaje puede recibir hasta 3 disparos (el diseño y la implementación de este punto no tiene restricciones, utilice su creatividad).
- El juego termina cuando el personaje llega al final de la estructura y alcanza el premio. Si el personaje cae a la base de la estructura se termina el juego. En ambos casos debe haber una pequeña animación en 3D que indique el término del juego.

## Puntuación

- Lectura y visualización del archivo csv: 0.5 puntos.
- Diseños dentro del juego (personaje, premio, objetos lanzados, estructura y fondo): 1.0 punto
- Movimiento de cámara, lanzamiento de objetos y movimiento del personaje: 1.0 punto
- Iluminación y sombreado: 0.2 puntos
- Distintas posiciones de la cámara: 0.3 puntos
- Control del personaje: 1.0 punto.
- Animación final 3D: 0.5 puntos

## Opción C: Snake 3D

Su videojuego Snake tuvo mucho éxito en la industria. Ahora se le pide hacer un snake volumétrico considerando los últimos avances de la tecnología.



Su tarea debe cumplir los siguientes requisitos:

- El borde del mapa debe ser explícito y claramente distingible del centro, por donde se desplaza la serpiente.
- La serpiente crece por cada recompensa que toca. Al consumir el trofeo, aparece otro en algún lugar del mapa donde no esté la serpiente (Se eliminó inmediatamente como requisito, puede aparecer un rato después si así lo desea).
- La snake se debe mover a una velocidad razonable. No menos de 1 movimiento por segundo y no más de 30 por segundo.
- Al morir el personaje, se despliega una escena/textura que diga Game Over. Esta escena debe incluir un tipo de animación que estime conveniente.
- Puede utilizar las dimensiones que estime convenientes.

- La serpiente debe ser controlable con las teclas WASD o las flechas del teclado. Esta está siempre en movimiento, donde al presionar teclas se cambia la dirección hacia donde se dirige.
- El movimiento con las teclas debe ser relativo a la dirección donde mira la snake, como en los juegos FPS o de aventura tipo Resident Evil, Assassin Creed, GTA, Super Mario 64, etc. Este movimiento debe ser suave, no un giro de 90° instantáneo.
- La serpiente se mueve en un espacio tridimensional y tiene altura, así como los bordes y la manzana.
- El juego se inicia con una cámara que mira desde la nuca de la snake hacia donde ella mire (Como en juegos como GTA o Assassin Creed). Este modo se vuelve a activar con la tecla R si es que se modifica durante la ejecución del programa.
- Implemente otra cámara estática que se activa al apretar la tecla E. Esta cámara apunta desde arriba hacia abajo, tal que permita ver el mapa como si fuera una aplicación 2D, como ocurre en juegos como Brawl Stars o LoL, que están en 3D pero se ven desde arriba.
- Al presionar la tecla T, se activa otra cámara estática que mira desde arriba hacia abajo de forma diagonal, mirando con vista en perspectiva hacia el centro de la cámara.
- Invente una mecánica para agregarle iluminación al juego. El único requisito es que la iluminación experimente un cambio al ocurrir ciertos eventos o bien presente un movimiento continuo. Por ejemplo: 1) Cada cierta cantidad de premios surge una manzana(u otro objeto) dorada(o) muy brillante. 2) cada manzana que coma la serpiente cambia el color de la escena a través de la luz (por ejemplo, todo parte oscuro y se adquiere color a medida que la serpiente crece), 3) Hay una luz en las paredes de la escena que va girando, apuntando a la cabeza de la serpiente.
- Las nuevas generaciones se aburrieron del modelo de manzana como recompensa. Prefieren que sea un OBJ a elección de quien diseña el programa.
- (OPCIONAL) Hay un obstáculo que se crea al iniciar el programa. Su posición es aleatoria. Puede implementar un efecto especial cuando la serpiente lo toca, como cambiar la iluminación de la escena, cambiar su color por un rato, aumentar su velocidad, invertir las teclas WASD, impedir su paso o bien hacerla perder instantáneamente. Este obstáculo no puede ser una figura básica de las otorgadas en material docente. Puede ser un OBJ importado.

## Puntuación

- Diseños dentro del juego (serpiente, bordes y fondo): 1.0 puntos

- Implementaciones de la cámara: 1.0 puntos
- Mecánicas del juego: que la serpiente se agrande, aparezcan las recompensas, morir al tocar un borde. 0.8 puntos
- Movimientos y control de la serpiente: 0.7 puntos
- Implementaciones de iluminación: 0.7 puntos
- Animación final: 0.3 puntos

## Opción D: Sistema Planetario 3D

¡Su aplicación anterior ha sido aprobada! Ahora se le ha pedido que la visualización sea en tres dimensiones, donde los cuerpos celeste sean modelos 3D, agregando iluminación generada por el astro central. Además de implementar una funcionalidad extra.

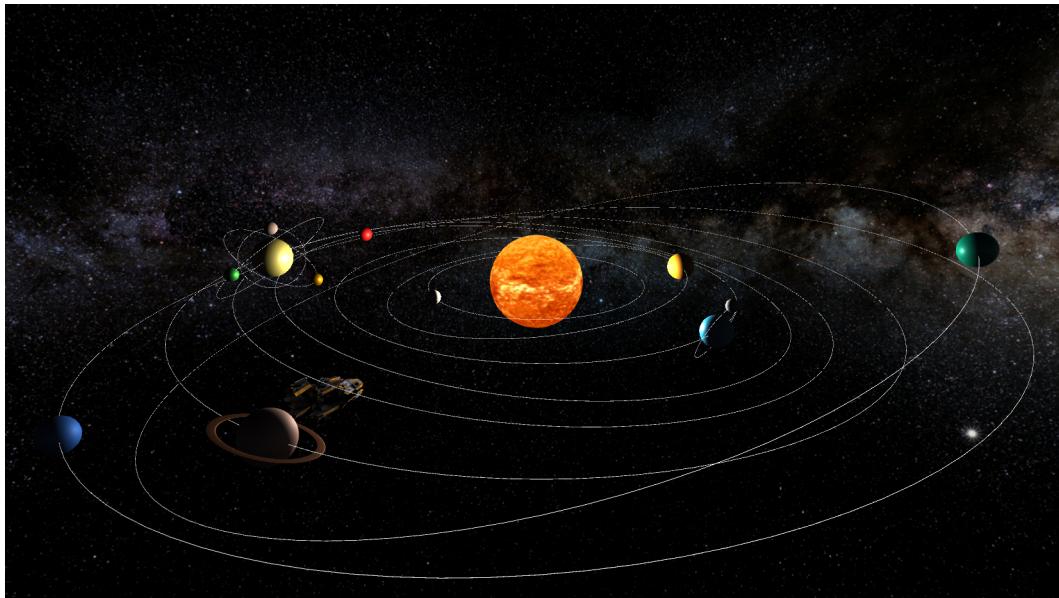


Figura 1: Primer modo de visualización

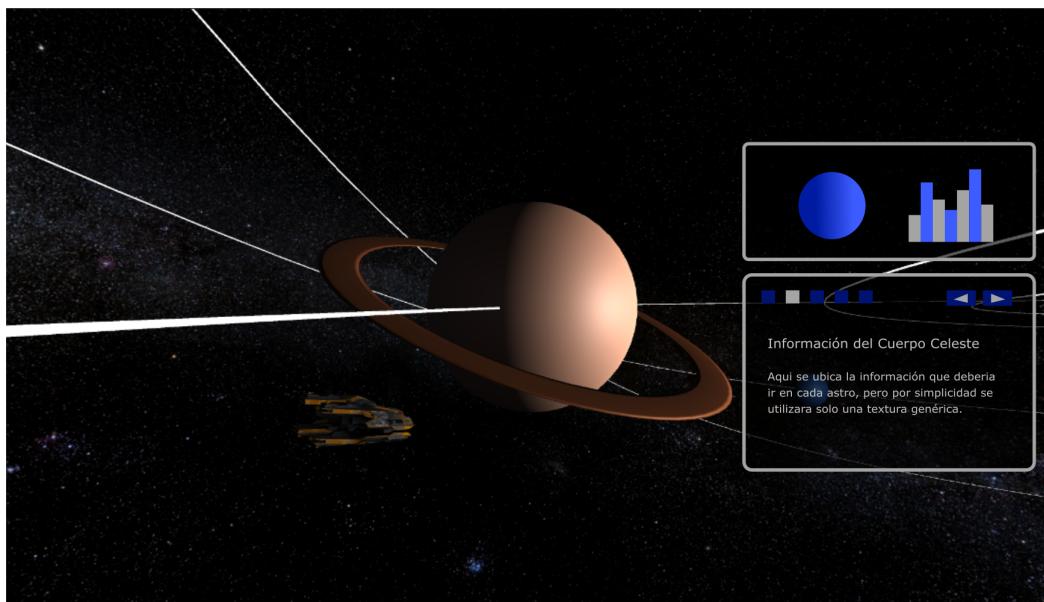


Figura 2: Segundo modo de visualización

La nueva aplicación debe añadir los siguientes requisitos:

- Los cuerpos celestes deben ser representados por un OBJ, indicando el directorio en el archivo json, si no se indica, debe generar un esfera unitaria con OpenGL. Donde los campos “Color” y “Radius” se refieren al color asignado al modelo y a la escala aplicada respectivamente. (Debe entregar al menos un OBJ)
- Dado la incorporación de una nueva dimensión, las orbitas deben tener una inclinación señalada como un ángulo con respecto a algún eje a su criterio, especificado en el archivo json. Se mantiene la linea que representa la orbita.
- Tiene que implementar iluminación de tal manera que parezca que el astro central (primer cuerpo del json) sea la fuente de luz, así como este cuerpo no presente sombreado.
- Debe tener dos modos de visualización, el primero es una vista global en la que se puede navegar, y la segunda equivale a una cámara fija con respecto al movimiento de un cuerpo seleccionado, fijando al astro en la pantalla, para que en un lado se vea la información/textura especificada en el primera aplicación. Se cambia de modo con la tecla **V**.
- Para el primer modo de visualización tiene que implementar una cámara que se mueva en coordenadas cilíndricas, donde en todo momento se apunta al astro central. Con las teclas **A** y **D** se maneja el ángulo  $\varphi$ , con **W** y **S** la altura  $z$  y con **Z** y **X** la distancia  $\rho$ .
- Para el segundo modo (reemplaza el despliegue de información de la primera aplicación) se tiene que cambiar de cuerpo seleccionado con las flechas izquierda y derecha, debe observarse siempre la totalidad del astro, así como debe poder apreciarse en el fondo, su desplazamiento por el sistema (debe seguir moviéndose).
- Debe implementar una skybox con texturas que envuelva el sistema tal que represente el fondo del espacio (estrellas, galaxias, etc). Su cámara debe alcanzar a dibujarlo.

El programa se correrá con la siguiente llamada:

```
python system_3dview.py bodies.json
```

Donde system\_3dview.py es el nombre de su script que ejecuta la aplicación y el archivo bodies.json contiene una lista jerarquizada de cuerpos celestes igual al de la primera aplicación, pero que agrega los siguientes campos:

- Model, directorio del modelo en OBJ.
- Inclination, ángulo en radianes que presenta la orbita del cuerpo con respecto a su parent.

Un ejemplo de los campos que tiene cada nodo de un archivo json:

```
[  
  {  
    "Color": [ 0.0, 0.2, 0.8 ],  
    "Radius": 1.5,  
    "Distance": 0.25,  
    "Velocity": 0.4,  
    "Model": "models/planet.obj",  
    "Inclination": 0.18,  
    "Satellites": "NULL"  
  }  
]
```

Por ultimo se le pide que implemente **solo una** de las siguientes funcionalidades:

- Un tercer modo de visualización que consiste en una cámara esférica apuntando al cuerpo seleccionado.
- Implementar al menos 2 modelos adicionales de astros que presenten más detalles y entregar un archivo json representando una escena visualmente bien construida (valores coherentes para los tamaños, distancias, velocidades, etc) incluyendo sus modelos. Puede hacer los modelos con algún programa de modelado como Blender.
- Agregar un modelo de nave espacial que se desplace por el sistema describiendo una curva especificada por usted, de tal manera que su recorrido sea suave.
- Implementar un modelo de asteroide, con el cual creara un cinturón de asteroides, marcando el límite del sistema planetario.
- Implementar materiales para cada cuerpo, esto es que cada astro se dibuje con sus constantes  $ka$ ,  $kd$  y  $ks$  especificados en el archivo json. Debe agregar estos campos a la estructura del json y modificar la función que dibuja el grafo de escena.

Puede tomar los supuesto que estime conveniente para su funcionalidad extra, pero debe ser correctamente informado en el reporte.

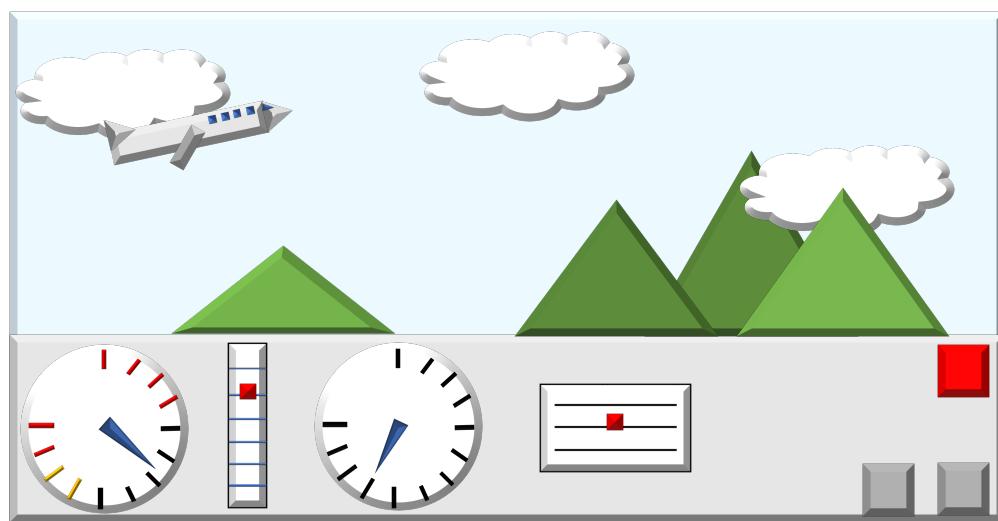
## Puntuación

- Lectura del archivo json y OBJ : 0.3 puntos
- Archivo OBJ entregado e implementación skybox : 0.3 puntos
- Jerarquía y movimiento de los cuerpos celestes : 1.0 puntos

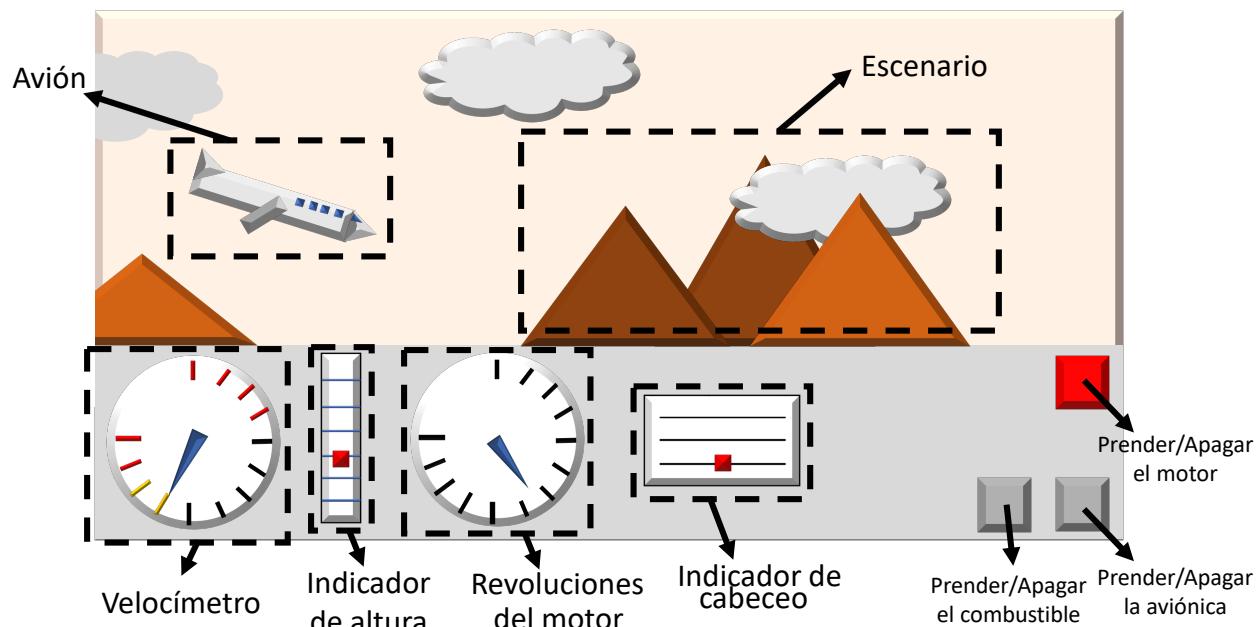
- Implementación de la iluminación : 0.5 puntos
- Implementación del primer modo de visualización : 0.8 puntos
- Implementación del segundo modo de visualización : 0.8 puntos
- Funcionalidad extra : 0.8 puntos

## Opción E: CC3501 Flight Simulator 3D

Dado el éxito de su anterior proyecto se le ha encomendado la tarea de añadir una **tercera dimensión** al simulador de vuelo. La aplicación y mecánicas son idénticas al trabajo anterior, con la excepción que ahora el escenario, el avión y el panel son en tres dimensiones. Las siguientes figuras ilustran el esquema de la aplicación (aplique imaginación para pensar que el bisel de las geometrías indica la dimensión extra de profundidad).



La siguiente figura describe los principales elementos de la escena:



La aplicación debe cumplir con los siguientes requisitos:

- A diferencia de la tarea anterior, ahora se debe crear toda la escena usando OpenGL 3D.
- (*idem*) La vista debe estar compuesta por una visualización de la escena y el panel inferior de la aviónica.
- La escena debe estar compuesta al menos de nubes y cerros creados con las primitivas de OpenGL en 3D. Estas deben moverse hacia la izquierda a medida que el jugador avanza para dar el efecto de movimiento. Note que puede proponer nuevos elementos en la escena como edificios, un sol, o estrellas. Los cerros los puede modelar como pirámides de base cuadrada o bien conos ubicados a diferentes "profundidades" para simular la cercanía o lejanía a la cámara. Las nubes las puede modelar con paralelepípedos y el terreno con un plano 2D de grandes dimensiones (que parezca infinito a simple vista). Ud. también puede proponer el terreno como una superficie triangulada de tal manera que parezca una cordillera (buscar por *terrain mesh triangulation*), de hacerlo de esta manera el color del terreno debe variar en función de la altura, para terrenos altos usar color blanco/gris (nieve), intermedios cafés, y bajos de color verde.
- La escena debe contener una luz (sol). Defina ud. los valores de la luz como atenuación o colores para encontrar los mejores resultados. Si lo desea puede incorporar más de una luz al simulador.
- El avión debe estar dibujado sólo con primitivas de OpenGL 3D, este debe ser capaz de rotar en su conjunto según la posición de cabecero del avión (giro en el plano para indicar que gana o pierde altitud) y debe poder moverse en el eje z para simular la altura. El modelo de la Figuras anteriores esta compuesto de 9 primitivas (pirámides y paralelepípedos). La forma del avión la puede proponer ud. **Usted también puede** importar un archivo .OBJ de una avión de su preferencia, las funciones de importación y el estilo de los archivos los debe proponer usted (puede basarse del internet).
- (*idem*) El panel inferior debe desplegar la aviónica. Esta, al menos, debe incorporar lo siguiente:
  - Velocímetro: Indica la velocidad del avión (*knots*).
  - Indicador de altura: Es una barra segmentada que contiene un punto rojo al medio que indica la altura del avión.
  - Revoluciones del motor: Velocidad en RPM del motor.
  - Indicador de cabeceo: Indica el ángulo del avión con respecto al plano principal de vuelo. Indica si se gana o pierde altura.

- Botones: Al menos el panel debe ofrecer tres botones, prender y apagar el motor, prender y apagar la aviónica (si está apagado ningún panel funciona), y prender y apagar el combustible.

Note que el tablero también debe modelarse con elementos 3D, y debe estar ubicado en frente de la cámara de tal manera que tanto la escena de fondo (avión + paisaje) como el tablero sean visibles. Piénselo como un cierto tipo de *virtual cockpit*.

■ (*idem*) Debe presentar unas físicas simples:

- El avión comienza en el suelo.
- Al prender el motor el usuario puede acelerar o desacelerar el motor con los botones **A** y **W**. El avión sólo puede despegar si la velocidad es mayor a 50, bajo esta velocidad el avión entra en caída libre (*stall*). Si el avión gana mucha velocidad puede resultar dañado y explotar. Note los límites superior e inferior del velocímetro. Si se choca con el suelo a una velocidad muy alta el avión puede explotar y el juego termina.
- La velocidad sólo controla la ganancia de altura, si se deja de acelerar el avión debe perder altura de manera gradual (dado el roce con el aire).
- El cabeceo del avión puede ser controlado con la teclas **FLECHA ARRIBA** y **FLECHA ABAJO**. El cabeceo permite controlar la tasa de ganancia de altura a velocidad constante. Ésta debe ser reflejada en la aviónica. De igual manera si el avión gana mucha altura (límite superior) el motor pierde empuje con el aire y la velocidad decae drásticamente perdiendo altura.
- El motor puede ser prendido o apagado con el botón del panel. Al estar prendido el motor el jugador puede interactuar con el motor. Además puede prender o apagar la alimentación de combustible (*mixture fuel*) y el panel (*avionics*).

Ud. sin embargo puede proponer sus propias físicas, aunque éstas deben ser consistentes. Puede basar su implementación en los simuladores reales para entender las principales mecánicas de la aviación. Además puede implementar otros botones como subir o bajar las ruedas (*landing gear*), prender o apagar baterías, etc.

■ (*idem*) Por último, el juego debe presentar al menos dos paletas de colores para el escenario. En las figuras anteriores se ilustran dos ejemplos (día) y (tarde). Puede proponer cualquier otra como por ejemplo: noche, nublado, etc.

El programa se correrá con la siguiente llamada:

```
python simulador3d.py
```

El simulador debe elegir la paleta de colores de manera aleatoria, o bien ud. puede proponer algún tipo de menú en consola para realizar todos los ajustes que quiera (elegir el avión, el escenario, etc...). Sea creativo en su solución.

## Puntuación

- Dibujo del escenario de manera estática en 3D (sin moverse en la medida que el jugador avanza): 1.0 puntos
- Implementación de las luces: 0.5 puntos
- Movimiento del escenario a medida que el jugador avanza: 0.2 puntos
- Dibujo y movimiento del avión (eje Z y rotación para el cabeceo) en 3D: 1.0 puntos
- Interacción con el usuario (teclas y clicks de los botones): 0.3 puntos
- Dibujo del panel de aviación en 3D: 1.0 puntos
- Física del avión, interacción con la aviación (ídem a trabajo anterior): 0.5 puntos