Tareas 1

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 tarea1x, con x indicando la opción que haya escogido.
- DEBE utilizar: Python 3.5 (o superior), Numpy, OpenGL core profile, GLFW.
- *Las imágenes son solo referenciales, utilice un estilo propio para su trabajo.

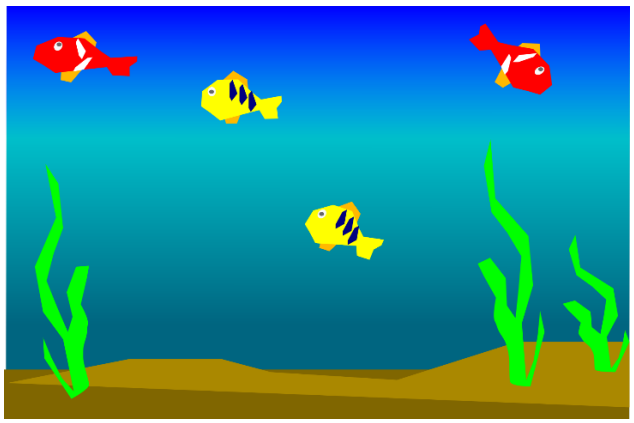
Entregables

- Reporte de documentación (1-2 planas). Formato pdf. Detalles disponibles en primera clase. (2%)
- Código que implemente su solución. Incluya TODO lo que su código necesita, incluyendo archivos proporcionados en clases. (9%)
- Al menos 4 commits en su repositorio git remoto ilustrando progreso en su trabajo. (1%)
- Video demostrativo de 20-30 segundos. (0.5%)
- Todo lo anterior debe estar disponible en su repositorio git bitbucket remoto.
- Puntaje total: 12.5 % del curso.

Objetivo

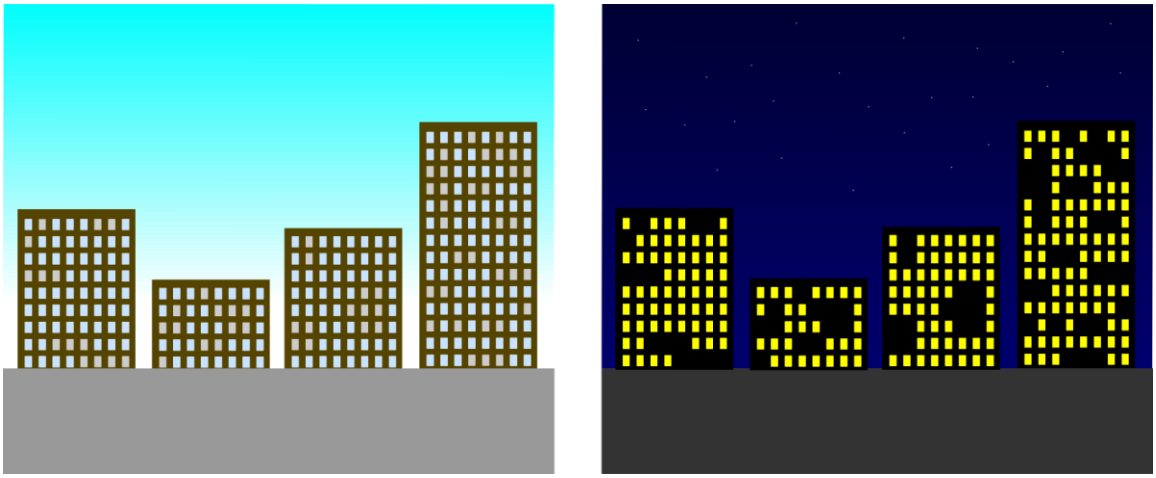
- Ejercitar el uso de OpenGL core profile en una aplicación en 2 dimensiones simple.
- Aplicar convenientemente matrices de transformación
- Trabajar con interacciones de usuario vía GLFW
- Implementar una animación simple

Opción A: Acuario



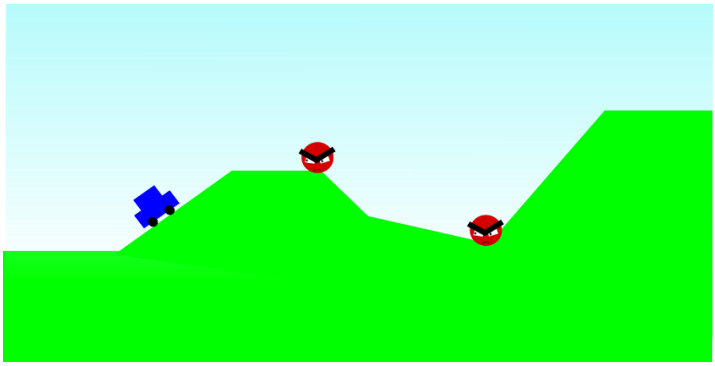
- Modele 3 peces diferentes, definiéndolos convenientemente vértice a vértice.
- Modele un fondo submarino utilizando al menos 3 figuras simples. Tome ventaja de la interpolación de colores de OpenGL. Ejemplos: rocas, algas, arena, burbujas, barco naufragado.
- Utilizando transformaciones, genere una escena que presente inicialmente 5 peces con posiciones arbitrarias dentro de la ventana.
- Genere una función de animación simple para mover los peces dentro del acuario. Utilice distintas funciones matemáticas: sinusoides, parábolas, movimiento en zig-zag, etc. Los peces no deben salir de la ventana de visualización. La salida de esta función debe ser una matriz de transformación. Entregando distintos parámetros a esta función usted puede generar distintas animaciones.
- Al presionar ‘enter’, se debe añadir un nuevo pez al acuario. Tanto el pez como su posición inicial deben ser aleatorios. Lo mismo los parámetros que definan su animación.
- Al hacer clic sobre un pez, este debe desaparecer :(.
- Utilice estratégicamente pequeñas transformaciones de shearing periódicas para simular un efecto de distorsión.
- Puntajes:
    - Modelos: 3 puntos
    - Configuración de la escena: 2 punto
    - Animaciones: 2 puntos
    - Interacción con el usuario: 2 puntos

Opción B: Ciudad



- Implemente una función que dado una altura h, genere un modelo de edificio de dicha altura. Debe agregar la cantidad de ventanas que sea requerida para ilustrar su tamaño.
- Implemente una función que dado una cantidad entera N, genere una cantidad N de edificios organizados horizontalmente modelando una vista de ciudad. Debe utilizar convenientemente transformaciones y la función anterior. La altura de cada edificio debe ser determinada arbitrariamente. Puede dejar un espacio entre cada edificio.
- Su programa debe recibir como argumento el número N de edificios. Es decir, su programa se debe ejecutar como: python tarea2b.py 15, generando 15 edificios.
- Decore la escena con al menos 3 elementos: cerros, nubes, sol. Tome ventaja de la interpolación de colores de OpenGL.
- Al presionar la tecla ‘espacio’, cambie el “shader program”, utilizando uno que presente la misma escena, pero de noche, solo cambiando los colores. En la escena vista de noche, algunas ventanas de los edificios deben mostrar sus luces encendidas.
- Al presionar Q y W debe acercar o alejar la escena suavemente en estilo zoom. Con las flechas del teclado, debe mover trasladar la escena en la dirección especificada. Esta funcionalidad debe ser implementada utilizando funciones de escalamiento y traslación convenientemente.
- Puntajes:
    - Edificio: 2 puntos
    - Escena decorada y con varios edificios: 4 punto
    - Interacción con el usuario: 3 punto

Opción C: Death Race



- Esta tarea consiste en un juego donde un carrito se mueve por una escena, pudiendo chocar con enemigos si es que no los elimina antes.
- Implemente un modelo de carrito y de un enemigo.
- La escena se describe en un archivo de texto con el siguiente formato:

```
0.0, 0.3
0.4, 0.3
0.6, 0.7, e
0.7, 0.5, e
1.0, 0.1
1.3, 0.3
```

  Cada línea específica un punto (X,Y) de la pista y la posibilidad de que exista un enemigo en dicha ubicación (cuando terminan en ",e"). Los puntos están estratégicamente organizados para siempre aumentar la coordenada X.
  En este ejemplo, primero se tiene un sector horizontal, luego sube, hay un enemigo, luego baja, hay otro enemigo, luego baja aún más, y finalmente sube. La imagen presenta otro ejemplo.
- Su programa debe recibir el nombre del archivo de escena como argumento del programa, i.e., se debe ejecutar como: `python tarea1c.py escena.txt`
- Su programa debe leer dicho archivo y generar una animación del carrito moviéndose sobre dicho camino. Para esto:
  - El carrito debe considerar una transformación de rotación y traslación vertical según la inclinación del tramo sobre el que se encuentre.
  - El escenario debe considerar una transformación de traslación horizontal de forma que se traslade con velocidad constante.
  - En otras palabras, el fondo se moverá, pero el carrito solo rotará y trasladará verticalmente, produciendo la ilusión de movimiento.
- Al presionar la tecla 'espacio', el carrito emite una onda expansiva que elimina a los enemigos dentro de un radio definido convenientemente.
- Al chocar con un enemigo, o al llegar al final de la pista, el juego se debe congelar indicando su término. Al presionar 'escape' se termina el programa.
- Puntajes:
  - Modelos: 2 puntos
  - Generar escena: 2 puntos
  - Movimiento del carrito por la escena: 2 puntos
  - Interacción con usuario: 3 puntos