Tarea 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 presentadas son solo referenciales, utilice un estilo propio para su trabajo.*

Entregables

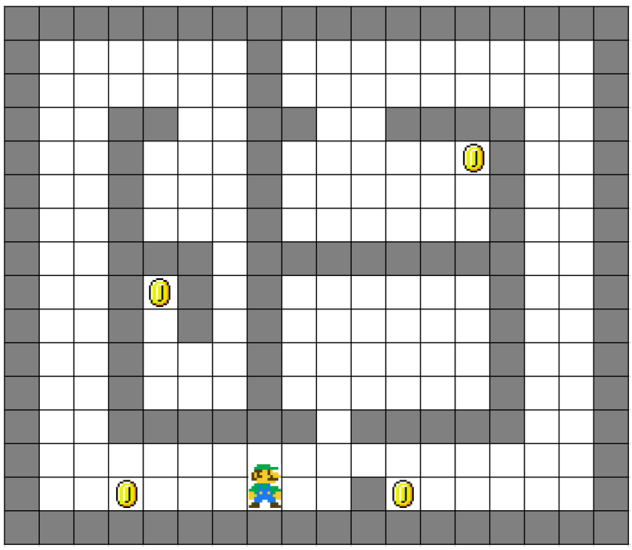
- Código que implemente su solución (4.5 puntos)
  - Su versión final DEBE utilizar archivos *.py, NO jupyter notebooks.
  - Incluya TODO lo que su código necesita, incluyendo archivos proporcionados en cátedras o auxiliares.
  - Al menos 5 commits en su repositorio git remoto ilustrando progreso en 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 2 dimensiones simple.
- Dominar el uso de matrices de transformación.
- Trabajar con interacciones de usuario vía GLFW.
- Implementar una animación simple.
- Hacer uso del patrón de diseño Modelo-Vista-Controlador y de la técnica de modelación jerárquica.

Opción A: Laberintos

Don Pedro, personaje del universo ficticio de CC3501, es un ávido jugador de laberintos. En esta tarea, implementaremos un editor de laberintos, y un juego que permita leerlos y encontrar tesoros ocultos en él.



Especificaciones

Su programa editor de laberintos se debe invocar como sigue:

```
python maze_maker.py maze.npy 10x20
```

Considere que:

- *maze.npy* es el nombre del archivo donde se guardará el laberinto.
- 10×20 es opcional e indica que se creará un laberinto de 10 celdas de alto por 20 celdas de largo. Si el archivo existe, el programa debe acusar error, no debe eliminar el archivo.
- Si no se indica el tamaño, se intentará abrir el archivo indicado para editarlo.
- Por supuesto, tanto el tamaño como el nombre del archivo pueden ser arbitrarios. Preocúpese solo de tamaños menores a 20×40.
- Al crear un nuevo laberinto, el programa debe presentar una grilla regular, donde todas las celdas son piso.

- Al hacer clic izquierdo en una celda, se debe alternar su valor entre muro y piso.
- Internamente, la grilla se debe almacenar en un arreglo numpy de enteros. Si la celda i,j es muro, se debe almacenar un 1 en la posición i,j del arreglo. Si no hay muro, se debe almacenar un 0 en dicha posición del arreglo.
- El laberinto se debe almacenar y leer utilizando la funcionalidad load/save de arreglos numpy.
- Al presionar *scroll* o 1, se define la posición de inicio para el jugador en la celda donde se encuentre el mouse. Internamente, se debe guardar un 2 la celda correspondiente del arreglo. Dibuje a Don Pedro en dicha celda.
- Al presionar clic derecho, se debe agregar un tesoro a la celda donde se encuentre el mouse. Esta moneda se representa almacenando un 3 en el arreglo numpy.
- Usted debe modelar los muros, el piso, a Don Pedro y a los tesoros. Debe utilizar todas las técnicas vistas en clases: modelos punto a punto, transformaciones y grafos de escenas.

Por otro lado, se debe implementar un lector de dichos laberintos que permita jugarlos. El programa para jugar, debe invocarse como sigue:

```
python maze_play.py maze.npy
```

Considere que:

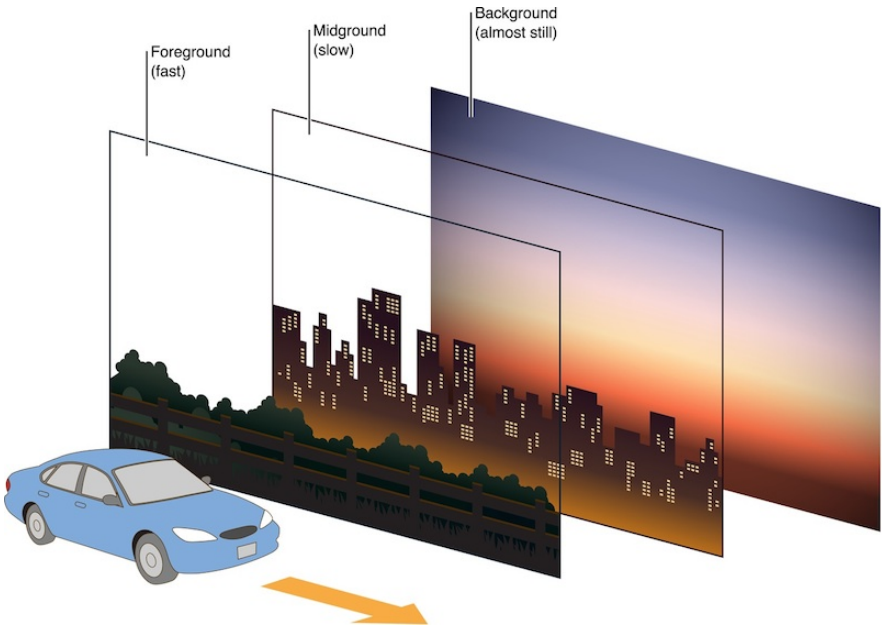
- El laberinto *maze.npy* debe ser cargado.
- Si no se ha definido la posición de inicio, o no hay tesoros, se debe acusar error.
- Al presionar las flechas del teclado, Don Pedro se debe mover una celda completa en la dirección indicada siempre y cuando no haya un muro en la celda destino.
- Si Don Pedro cae en una celda con tesoro, el tesoro debe desaparecer.
- Cuando se acaben todos los tesoros, el juego termina y se debe imprimir en consola el total de pasos que se han dado y el tiempo total transcurrido desde que se inició el juego.

Puntuación

- Almacenamiento y lectura de laberintos: 1 punto
- Edición del laberinto: 1 punto
- Modo Juego: 1 punto
- Modelos: 1.5 puntos

Opción B: Parallax

Una técnica muy utilizada en videojuegos para simular profundidad en 2D es el efecto Parallax. El truco es mover capas en el fondo lentamente, mientras las capas más cercanas a la cámara se mueven más rápido.



En esta tarea, modelaremos un paisaje con efecto Parallax que podrá ser recorrida horizontalmente con el nuevo y fantástico auto de Don Pedro (si, el mismo Don Pedro de la opción anterior).

Considere que:

- Su escena debe presentar un paisaje con sentido y estilo único: Bosques, Ciudad, Desierto, Granja, etc...
- La escena debe presentar un efecto parallax de 4 capas que se aprecie al deslizar el auto horizontalmente por la escena.
- Deben existir al menos 2 clases de objetos distintos que se muevan por razones no controladas. Ejemplos: nubes, aves, aviones, etc.
- El auto de Don Pedro destaca por su nivel de detalle y funcionalidades, esto es:
  - Las flechas del teclado le permiten avanzar hacia adelante y hacia atrás.
  - Sus ruedas rotan sin resbalar al deslizarse.
  - Sus luces se encienden o apagan con la tecla 1.

  - Presenta una funcionalidad extra espectacular que se activa con la tecla 2, pudiendo esta ser: animación descapotable, posibilidad de volar, de saltar, habilitar un propulsor gigante que le proporcionará mayor velocidad, etc... Al presionar 2 nuevamente, la funcionalidad extra debe ser desactivada.

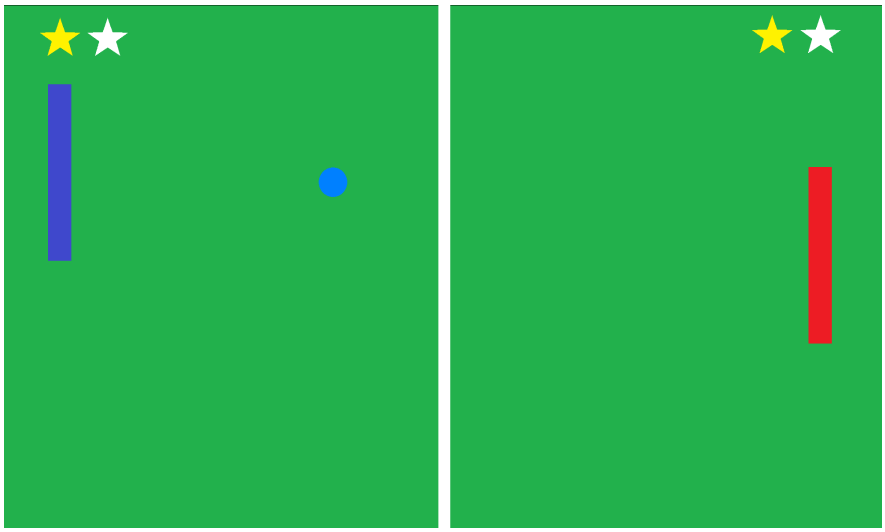
- Para el diseño de sus modelos debe utilizar cada una de las técnicas vistas: especificación punto a punto, transformaciones y grafos de escena.

Puntuación

- Efecto Parallax de 4 capas: 1 punto
- Animación no controlada de fondo: 0.5 puntos
- Control del auto de Don Pedro: 0.5 puntos
- Funcionalidad Extra: 1 punto
- Modelos: 1.5 puntos

Opción C: Pong

Usted quiere darle un lindo regalo de cumpleaños a su abuelito. Luego de pensar, se le ocurrió la fantástica idea de llevarlo de vuelta a su niñez con el famoso juego *Pong*.



Esta opción consiste en implementar el juego *Pong* utilizando sus habilidades en Python y OpenGL. Para ello deberá modelar el fondo, las barritas, la pelotita e iconos que representen los puntos/vidas y los remaches disponibles de cada jugador. Además tendrá que implementar las funcionalidades del juego, como por ejemplo: que la pelotita rebote, si algún jugador marca un punto se debe iniciar una nueva ronda, etc.

Considere lo siguiente:

- Para la creación de los objetos deberá usar las 3 estrategias vistas en clases: punto a punto, a través de transformaciones, y grafo de escena. Distintos objetos pueden ser modelados utilizando distintas técnicas. Debe aprovechar la interpolación de colores automática que permite OpenGL.
- El movimiento de las barritas debe ser con las teclas W y S para el jugador 1, y, *flecha arriba* y *flecha abajo* para el jugador 2.
- Al golpear la pelota con la barrita, se debe modificar levemente la dirección de rebote dependiendo del sentido de movimiento de la barrita. De esta forma, es posible direccionar el tiro.
- Debe añadir al juego un sistema de vidas o puntaje, a través de un modelo gráfico. Por ejemplo, un sistema de vidas podría estar basado en corazones, donde al marcar un punto se le elimina un corazón al jugador contrario; un sistema de puntos añadiría estrellas por cada punto conseguido; etc

Puntuación

- Modelos: 1.5 puntos
- Movimiento y rebote de la pelota: 1.5 puntos
- Control de usuario, incluyendo direccionamiento del tiro: 1 punto
- Sistema de puntuación: 0.5 puntos

Opción D: Estados del Agua

En esta tarea estudiaremos visualmente el comportamiento de la molécula de agua en sus distintos estados: sólido, líquido y gaseoso. Para esto, implementaremos un visualizador que ilustre como varía la agitación térmica según temperatura y estado.

Su simulador debe invocarse como sigue:

```
python h2o.py N
```

Considere que:

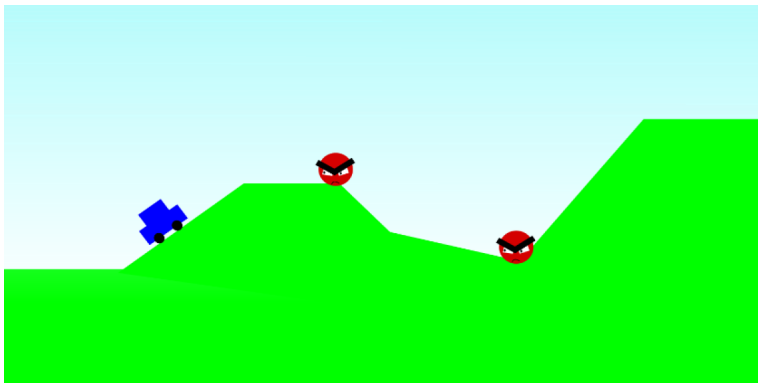
- La ventana presentará N moléculas de agua moviéndose arbitrariamente en ausencia de gravedad por toda la ventana. No es necesario que las moléculas reboten entre si.
- En la parte inferior de la ventana debe ubicar una barra slider utilizando solo OpenGL. Al mover dicha barra con el mouse, variaremos la temperatura entre -20^oC y 120^oC.
- En la medida que cambia la temperatura, las moléculas deben modificar su comportamiento. Ordenándose desde la estructura cristalina del hielo, un movimiento restringido en el caso del estado líquido y hasta un completo caos para el estado gaseoso. La agitación térmica debe ser siempre observable y el cambio de temperatura debe apreciarse en su movimiento.
- Para los estados de animación anterior, considere usted supuestos razonables que le permitan ilustrar el fenómeno de manera simple. No es necesaria una representación fidedigna de todos los procesos físico-químicos involucrados, pero sí debe apreciarse un cambio en la animación para los distintos estados.
- La tonalidad de la escena debe cambiar su color dependiendo de la temperatura. Esto es, el color debe cambiar desde blanco para las temperaturas más bajas, colores normales, anaranjado y finalmente rojizo.
- Debe lograr el efecto anterior añadiendo una variable uniform vec3 al vertex shader y que este module (multiplique) el color final que se enviará al fragment shader.

Puntuación

- Interacción con usuario: 1.5 puntos
- Tonalidad de la escena: 1 punto
- Animación a través de temperaturas y estados del agua: 1.5
- Modelos: 0.5 puntos

Opción E: 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.



Su juego debe invocarse como sigue:

```
python death_race.py scene.txt
```

Donde el archivo scene.txt posee 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.

Considere además que:

- Debe implementar un modelo de carrito y de un enemigo.
- Su programa debe el archivo especificado 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 invisible 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.

Puntuación

- Modelos: 1.5 puntos
- Generar escena: 1 punto
- Movimiento del carrito por la escena: 1.5 puntos
- Interacción con usuario: 0.5 punto