CC3501 Modelación y Computación Gráfica
Profesor: Daniel Calderón
Aulmno: Nicolás García Ríos

Reporte de Documentación - Tarea 3

25 de julio de 2020

Identificación

El presente documento describe la estructura del problema y solución de la Tarea 3 del curso **CC3501 - Modelación y Computación Gráfica para Ingenieros**. La opción elegida para resolver fué la Tarea 3-a, visualización científica para el problema del Acuario, el cuál fué resuelto con funciones de archivos .py de Auxiliares y cátedras del curso, y codificado en Jupyter Notebook.

Solución Propuesta

Para la resolución del problema de EDP se eligió una discretización (steps) de 0.2 para poder generar una gran cantidad de cuadros para la matriz. Además, hay que recordar que para los calentadores y la parte destapada de la pecera tendrán condiciones de Newman (que son conocidas y dadas por el usuario), mientras que en las esquinas se tendrán condiciones de Dirchlet que se deben calcular por cada caso (dependiendo de la pérdida de temperatura).

Ahora, con lo anterior, hay que iterar en cada cuadro de la matriz para calcular su temperatura dependiendo de todo lo anterior, por lo que se caerán en una gran cantidad de casos dependiendo por cada cara, arista y condición de Newman o Dirchlet. Cada caso por cara, arista, esquina y cada ecuación está especificada para finalmente guardar la matriz de temperaturas generada en un archivo .npy con el nombre especificado por el usuario y necesaria para poder visualizar los voxeles (utilizando la función createcolorcube de auxiliares anteriores) en la escena.

Para la visualización de los peces y acuario se crearon 4 archivos hechos con grafos de escena. 3 de ellos para cada tipo de pez, los cuales fueron formados solo con cubos transformados y con distintos colores de forma que sean totalmente distinguibles entre sí (Peces cril verdes y pequeños, peces globo medianos y peces payaso grandes). Además, la pecera se realizó también con cubos transformados y una base simple para distinguirlo como el fondo de la pecera.

Para una visualización más cómoda no se agregó ningún tipo de polígono para representar el agua, si no que simplemente se dibujó toda la escena sobre un fondo color celeste agua para crear un efecto de agua de pecera.

Para poder generar cada tipo de pez en su área correspondiente se decidió crear una lista de trios de vértices que cumplen las temperaturas por cada tipo de pez, esto con el fin de elegir de forma completamente random las coordenadas donde se dibujará cada pez. Con esto se guardan los peces a dibujar en una lista con el fin de ir uno por uno dibujando cada pez en una posición aleatoria (también se genera un ángulo aleatorio para que cada pez mire a un lado distinto). Para complementar lo anterior y evitar que los peces sobresalgan de la pecera, se escala tanto los voxeles de la matriz recibida como la pecera por un factor de 1.1, un número suficientemente bajo para que los peces no se alejen de sus áreas de hábitat y lo suficientemente alto para disminuir notablemente una vista sobresalida de la pecera.

Para el caso de los voxeles, se escala la matriz por cada coordenada dependiendo de W, L, H manteniendo siempre un constante de visualización de (3,6,4). Esto debido a que el tamaño de la pecera, áreas, cámara y otras componentes dependen del tamaño de la matriz, llevando a que

se generen muchas dependencias por cada archivo. Con esto, sin importar el tamaño de la matriz, siempre se verá con el mismo escalamiento (visualización).

Implementación

Para implementar la tarea en la consola es necesario crear primero la matriz dependiente de los datos entregados por un archivo .json. Así, se llama al archivo *aquarium-solver.py* junto con el archivo .json designado por el usuario que generará automáticamente un archivo .npy para poder implementar la visualización del acuario. Ya generado el archivo .npy basta llamar a continuación al archivo *aquarium-view.py* junto a otro archivo .json de condiciones iniciales designado por el usuario. Esta última llamada abrirá una ventana con la visualización de toda la pecera:

```
python aquarium-solver.py problem-setup.json   # Generará archivo .npy
python aquarium-view.py view-setup.json        # Abre ventana de visualización
```

Ya en la ventana será posible usar las flechas del teclado será posible moverse en la escena (izquierda y derecha modifican el ángulo mientras que arriba y abajo aumentan o disminuyen el zoom). Además de esto, con las teclas a, b y c se dibujarán o se desactivará la vista de los sectores de los peces tipos a, b y c respectivamente.

Resultados

La visualización logra cumplir los objetivos más importantes requeridos por la tarea. Se pueden observar los sectores por cada tipo de pez dependientes de los casos iniciales puestos en la resolución de la EDP, y además se pueden distinguir cada tipo de pez. Sin embargo, los peces no tienen una animación de cola y aletas (debido al tiempo se ha decidido enviar de esta forma).

A continuación se dejarán imágenes de cada parte importante de la visualización:

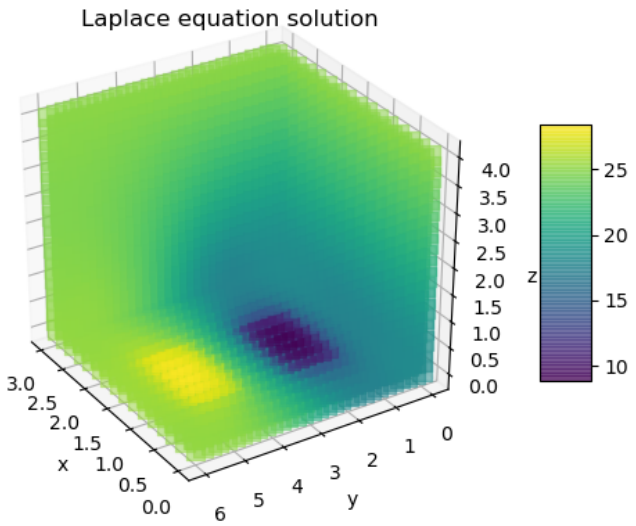

Figura 1: Visualización de la solución de la EDP usando las condiciones de ejemplo

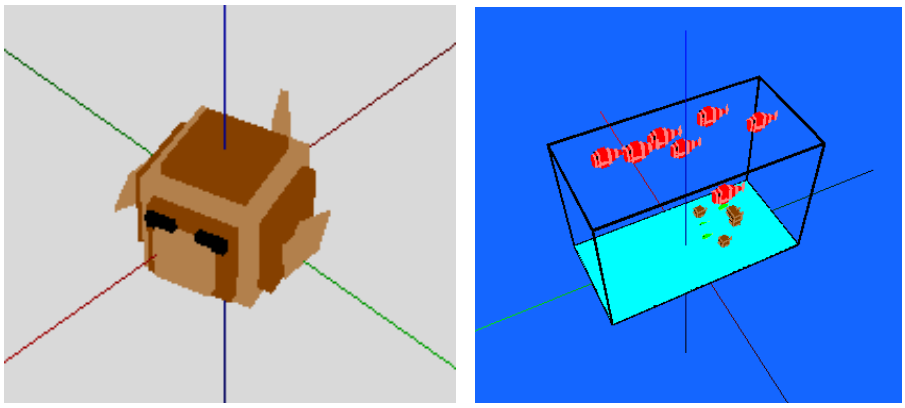

Figura 2: Izquierda: Ejemplo de pez, en este caso, del archivo pez2.py o pez globo. Derecha: Prototipo y primera versión de la pecera sin voxeles y con peces

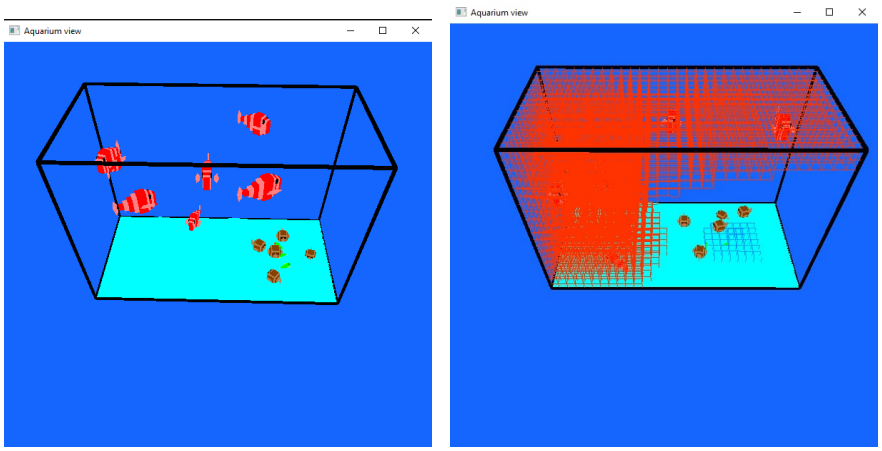

Figura 3: Izquierda: Pecera final sin sectores activados y angulos de peces aleatorios. Derecha: Pecera final con sectores c y b activados (a no activado para visualizar b)

Comentarios Adicionales

Por motivos que no puedo explicar, mi archivo de visualización dibujaba de forma correcta la escena y las áreas de los peces, pero al cambiar el archivo para poder recibir las condiciones de los archivos .json estos eran ilógicos y errados. Esto me llevó a solicitarle a dos compañeros que probaran mi código, y los resultados de estos se visualizaban de la forma esperada y tal como se observan en las imagenes anteriores. Dejo este comentario como constancia de que pese a intentar otros métodos de visualización o intentar arreglar esto, en mi computador se sigue viendo de forma completamente distinta a lo que mis compañeros visualizan pese a que se utiliza mi código y los mismos archivos de condiciones iniciales entregados por mí.