

Reporte Tarea 3a

Pablo Muñoz Soto

17/07/2020

CC3501

Solución Propuesta:

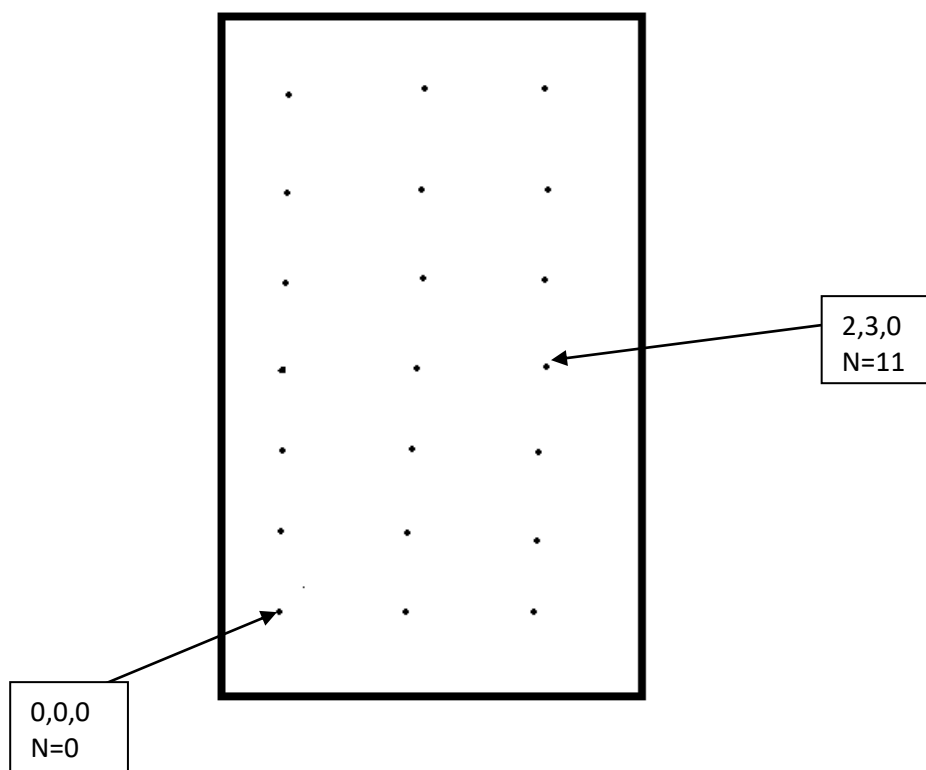
Aquarium-Solver

Se comienza por leer la información en el archivo problem-setup para asignar las variables del problema, para resolver la ecuación de calor del acuario usamos una discretización de un espacio tridimensional, con una distancia 0.25 entre puntos, y usando una aproximación de la derivada en cada punto con esto obtenemos una expresión del gradiente en función de los puntos vecinos, se clasifican los puntos por la cantidad de caras adyacentes, siendo 0 caras(interior del acuario), 1 cara, 2 caras y 3 caras(esquinas), dado que en cada caso varia la ecuación del punto, cada ecuación es almacenada en una matriz sparse que representa las variables y una matriz columna que representa el lado derecho de cada ecuación, para establecer la relación entre la matriz columna y el punto del cual es la ecuación se usan las funciones `getn()` y `getIJK()` para establecer una biyección entre ambas, luego se procede a resolver el sistema de ecuaciones y guardar la información de la matriz con los resultados con la función `np.save` de numpy.

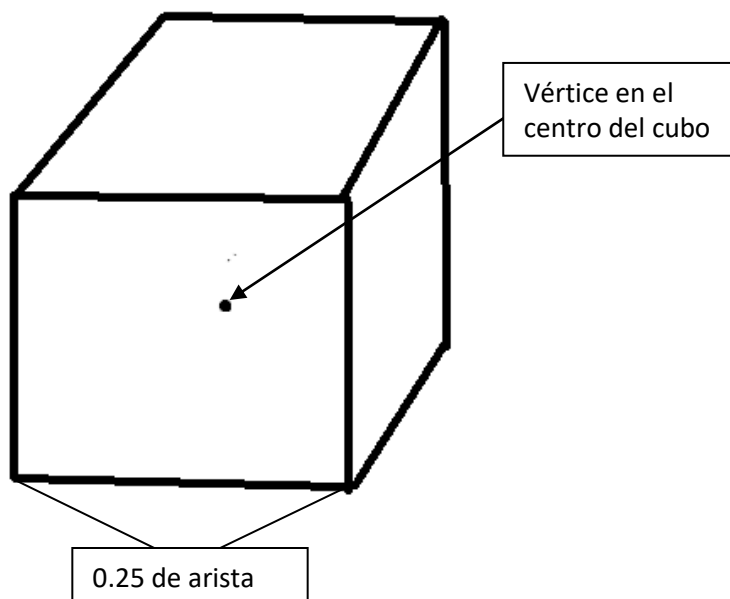
Aquarium-view

Se comienza leyendo la matriz con los resultados generada por Aquarium-solver y los parámetros entregados en view-setup, para luego revisar cada punto en la matriz solución para ver si esta dentro del rango de temperatura de los peces a, b y c, si están dentro del rango se almacenan en un arreglo, luego elegimos N_a vertices del arreglo a, N_b vértices del arreglo b y N_c vértices del arreglo con el módulo random, estos vértices corresponden a la posición de los peces en la visualización, luego se procede a generar el acuario, un cubo de arista 0.5 alrededor de cada vértice en el arreglo a, b y c y un pez en cada vértice seleccionado de manera aleatoria anteriormente, se uso el color rojo para el área a, verde para el área b y azul para el área c para poder distinguirlos fácilmente, para ahorrar poder de procesamiento se omite la creación de cubos en vértices ya rodeados por cubos, dado que de igual manera no serian visibles en la visualización.

Ejemplo discretización con Width=1, Length=2, en $k=0$, los valores se asignan de izquierda a derecha, de atrás al frente, de abajo hacia arriba



Ejemplo voxel de un punto perteneciente a un área en la que puede estar un pez



Instrucciones de ejecución:

Aquarium-Solver:

Para utilizar Aquarium-solver son necesarias las librerías numpy, scipy, json y sys

Se ejecuta usando la caja de comandos con la llamada

```
python aquarium-solver.py problem-setup.json
```

En donde problem-setup.json contiene las variables del problema, el programa generara un archivo con la matriz solución con el nombre indicado en el archivo

Aquarium-view

Para utilizar Aquarium-view son necesarios los modulos transformations, basic_shapes, easy_shaders, scene_graph, ex_curves y las librerías json numpy, OpenGL, glfw, sys

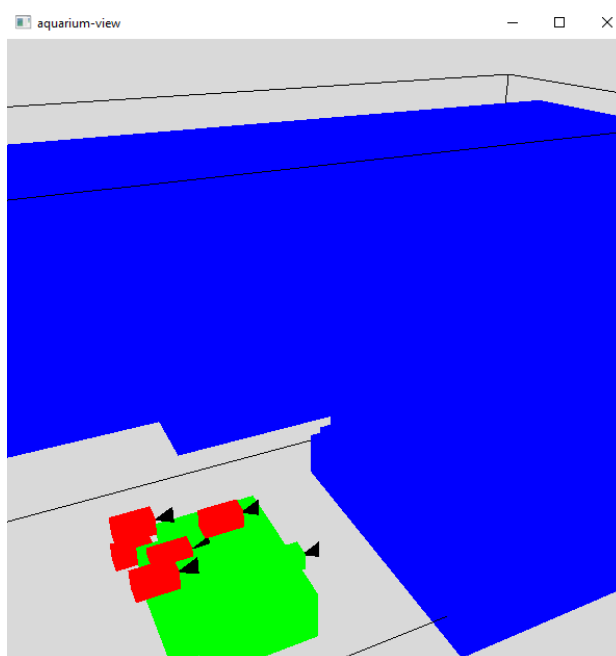
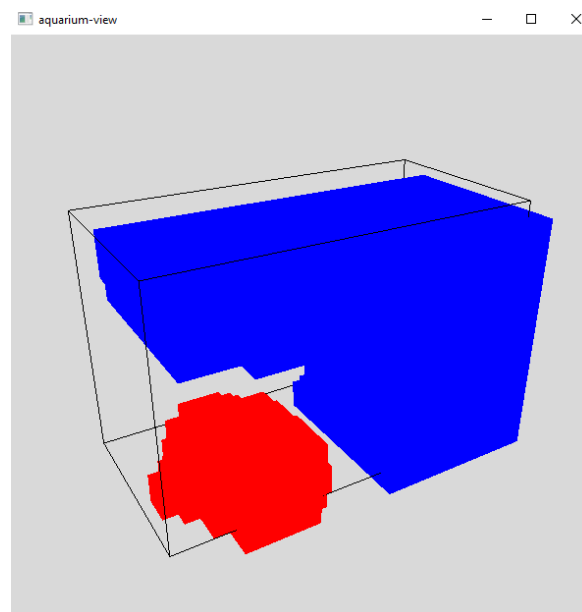
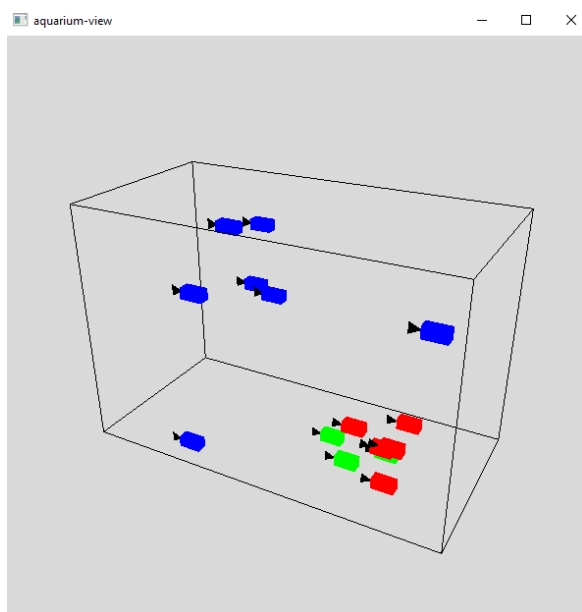
Se ejecuta usando la caja de comandos con la llamada




```
python aquarium-view.py view-setup.json
```

En donde view-setup contiene información sobre el rango de los peces y la cantidad de peces

A continuación se mostrara una ventana con la distribución de los peces, con las flechas izquierda y derecha es posible rotar la cámara alrededor del acuario, con las flechas arriba y abajo se puede acercar y alejar la cámara y con las teclas a, b, c se activan/desactivan la visualización de las regiones a,b,c respectivamente

Screenshots:



 Reporte Tarea 3a	12-07-2020 20:59	Documento de Mi...	44 KB
 solution.npy	12-07-2020 20:00	Archivo NPY	30 KB
 view-setup.json	06-07-2020 13:36	Archivo JSON	1 KB