Departamento de Ciencias de la Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
CC3501-1 Modelación y Computación Gráfica

ACUARIO

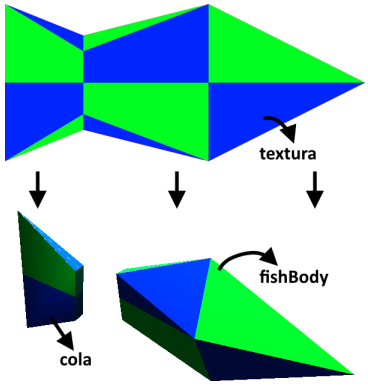
Tarea 3: EDPs y Visualización Científica

Alumno: Alfredo Escobar
Profesor: Daniel Calderón
Auxiliares: Nelson Marambio Q.
Alonso Utreras
Ayudantes: Tomás Calderón R.
Nadia Decar
Beatriz Graboloza M.
Heinich Porro Sufan
Fecha: 20 de julio de 2020

1. Solución Propuesta

1.1. local_shapes.py

Primero se han definido funciones para la creación de las GPUShapes correspondientes a las partes del pez (el cuerpo y la cola). En cada una de estas funciones, se comienza por definir la posición de los vértices, asociar cada uno a un punto de una imagen de textura, y luego son entregados en orden a las funciones "createTextureNormals—Indexation" para obtener nuevas listas de vértices, normales e índices. Luego, en la función "crearPez" se unen las partes del pez mediante nodos de grafo de escena, los que nos permitirán manipular la animación del pez aplicando rotaciones.



Se ha definido la función "createTextureVoxel", la cuál crea un cubo texturizado en una posición especificada.

También se definieron tres funciones para la creación de los elementos estéticos del acuario (el borde, el interior y los reguladores térmicos).

1.2. aquarium-solver.py

Comenzamos por obtener las variables especificadas en el archivo .json de configuración. Definimos una discretización y una parametrización que nos permitirá trabajar con el espacio tridimensional de manera más ordenada (en donde "N" será la cantidad de puntos cuya temperatura desconocemos).

Definiremos un sistema de ecuaciones diferenciales mediante matrices: $A \cdot x = b$. En la matriz $A_{(N \times N)}$ guardamos los coeficientes de las incógnitas, y en $b_{N \times 1}$ guardamos el lado derecho de cada ecuación. Estas matrices son de tipo dispersas list-on-list, lo que nos ahorrará uso de memoria y rendimiento al momento de modificar el grado de dispersión (al cambiar el valor de un 0).

Iteramos sobre los "N" puntos definiendo las EDPs mediante stencils de 7 puntos (los 6 puntos a su alrededor más el punto mismo). Dependiendo de la posición del punto se decidirá su ecuación. Se han definido un total de 29 tipos de ecuaciones:

- 1 para los puntos al interior del acuario.
- 2 para los puntos correspondientes a los reguladores térmicos.
- 6 para los puntos en las caras del acuario.

- 12 para los que están en las aristas.
- 8 para los que están en las esquinas.

Luego de definidas las matrices, se resuelve el sistema, se añaden los valores de temperatura de los casos conocidos (cara superior), y se guardan los resultados en un archivo .npy.

1.3. aquarium-view.py

Comenzamos por obtener las variables especificadas en el archivo .json de configuración. Luego, leemos el archivo .npy obtenido con el programa anterior, y guardamos sus valores en la matriz "u".

Se ha definido la función "zonasAptas", que retorna listas con posiciones cuya temperatura es apta para cada una de las especies de pez. Estas listas se utilizarán para posicionar a los peces y para generar a los voxeles.

Luego, la función "merge" une los valores de múltiples cuerpos geométricos (vértices, índices, referencias posiciones en imágenes o colores) para retornar un solo cuerpo con isosuperficie.

Se compilan distintos shaders: 3 para cuerpos con textura e iluminación, 1 para cuerpos con texturas sin iluminación, y 1 para cuerpos simples sin iluminación. Se llama a las funciones que generan los elementos visuales del acuario en sí (en local_shapes.py). Luego, llamamos 3 veces a la función de instancing de peces para generar a los 3 grupos (cada uno con una textura distinta).

Después, llamamos a la función "zonasAptas". Con estos datos, creamos voxeles posicionados (con la función "createTextureVoxel") y los unimos (con la función "merge") para generar 3 cuerpos de isosuperficie, uno para cada especie de pez. Después, posicionamos aleatoriamente a los peces dentro de su sector adecuado, asignándole también a cada uno un valor de rotación en el eje Z.

Ya en el loop principal, se verifica el estado de las teclas que controlan la cámara, y en base a esto se calculan los ángulos de barrido y el nivel de zoom para luego construir la matriz de vista.

Luego, se aumenta el valor de la variable "cola_theta" en función del tiempo. Esta se usa para determinar la transformada de rotación de "sgnCola". La rotación de "sgnPez" se calcula en base a "pez_theta", que a su vez corresponde al valor de "cola_theta" con un desfase.

Finalmente, dibujamos los objetos. Según "controller.voxels" decidiremos si dibujar o no uno de los conjuntos de voxeles. Si dibujamos alguno, será con el pipeline para cuerpos texturizados sin iluminaación. Este pipeline se usa también para dibujar el agua del acuario (aparecerá sólo cuando no estemos mostrando voxeles). Los peces son dibujados con uno de los 3 pipelines para cuerpos con textura e iluminación (aunque "controller.showFish" puede determinar si son dibujados o no). Dibujamos también los demás objetos estéticos del acuario.

2. Instrucciones de Ejecución

2.1. Argumentos

El primer programa se ejecuta con la siguiente llamada:

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

En donde problem-setup.json corresponde al archivo con los parámetros del acuario (dimensiones y temperaturas). Por ejemplo:

```
{
    "height" : 4,
    "width" : 3,
    "lenght" : 6,
    "window_loss" : 0.01,
    "heater_a" : 5,
    "heater_b" : 30,
    "ambient_temperature" : 25,
    "filename" : "solution.npy"
}
```

El segundo programa se ejecuta con la siguiente llamada:

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

En donde view-setup.json corresponde a al archivo con los parámetros correspondientes a los peces (cantidad y temperatura preferida). Por ejemplo:

```
{
    "filename" : "solution.npy",
    "t_a" : 15,
    "t_b" : 10,
    "t_c" : 25,
    "n_a" : 5,
    "n_b" : 3,
    "n_c" : 7
}
```

2.2. Control

aquarium-solver.py terminará de ejecutarse una vez haya generado un archivo .npy.

En bird-herd.py se puede controlar la cámara con las teclas de dirección. Además, con **Shift Izquierdo** se puede hacer Zoom In, y con **Control Izquierdo** se puede hacer Zoom Out. Con las teclas **A**, **B** y **C** se muestran los voxeles correspondientes a los sectores del acuario apropiados para cada pez. Con la tecla **1** se dejarán de mostrar los voxeles. Con la tecla **2** se puede alternar la visualización de los peces. Con la tecla **3** se puede alternar la visualización de los reguladores térmicos. Se puede cambiar el pipeline de sombreado con las teclas **Q** (Flat), **W** (Gouraud) y **E** (Phong). Finalmente, se puede cerrar el programa mediante la tecla **ESCAPE**.

3. Resultados

3.1. aquarium-solver.py

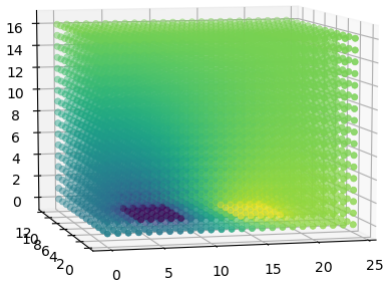
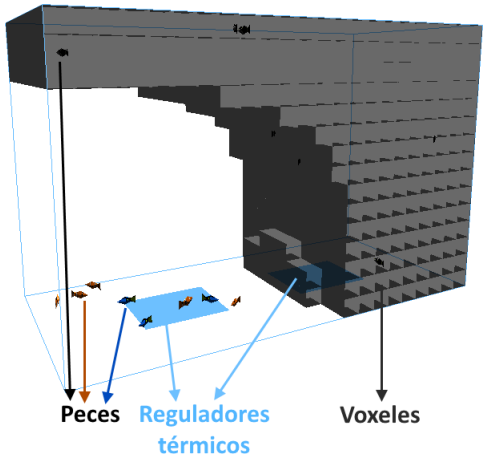


Figura 1: Matriz resultante representada mediante el programa view_graph.py adjunto

Tras ejecutar este programa, obtendremos un archivo .npy que contiene la matriz de temperatura en todos los puntos (discretizados) del acuario.

3.2. aquarium-view.py



En este programa se puede observar a tres grupos de peces dentro de un acuario, cuya temperatura está dada por el archivo .npy obtenido con el programa anterior. Podemos rotar la cámara, acercarnos y alejarnos con las teclas indicadas en la sección **2.2 Control**. Podemos también visualizar los sectores del acuario con temperaturas adecuadas para cada una de las especies de pez por medio de voxeles.