66.71 – Sistemas Gráficos

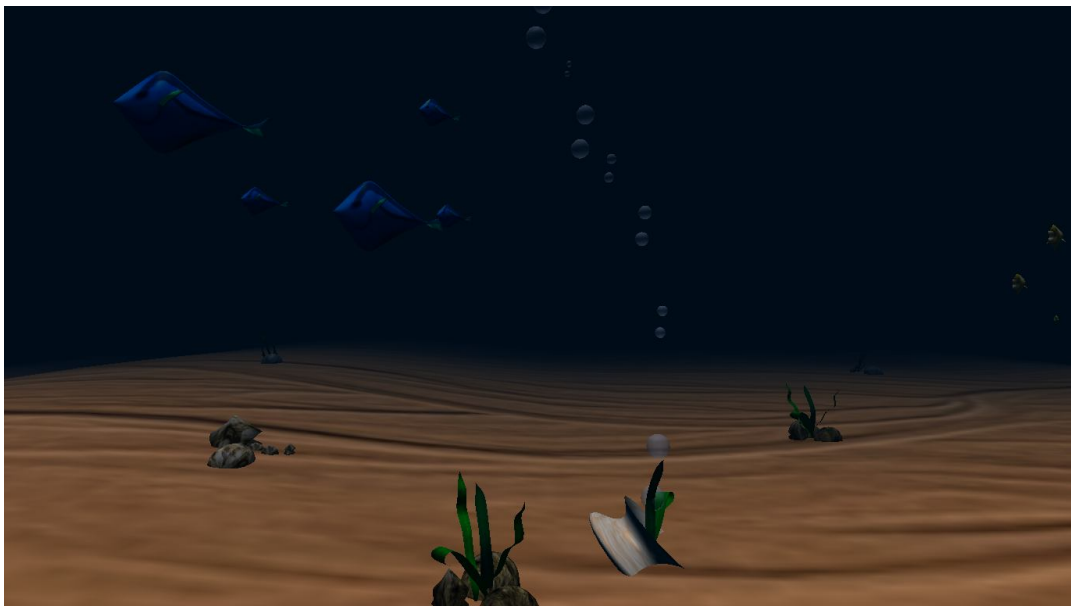
Trabajo Práctico Final – Fondo del Mar

Lucía Garbarini

2do Cuatrimestre 2009

- **Introducción**

Este trabajo fue realizado en OpenGL en el lenguaje C++ aplicando los conceptos vistos en la materia como son las superficies de barrido y revolución, iluminación, mapeo de texturas y utilizando las herramientas del pipeline gráfico de OpenGL.



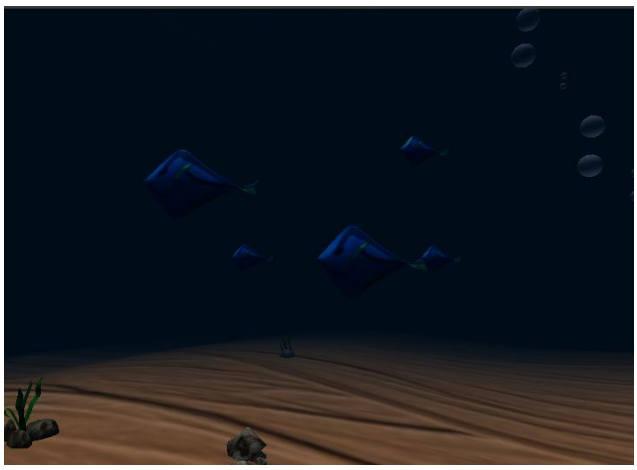
- **Peces**

Se modelaron 3 tipos de peces, cada uno instanciado varias veces para formar un cardumen. Los cardúmenes tienen peces en diferentes escalas y posiciones que siguen trayectorias cerradas. Las aletas y la cola de los peces fueron animadas mediante una "interpolación" de los puntos de control utilizando curvas bspline para que el movimiento sea suave y parezca detenerse y acelerar.

- Pez Koi

El cuerpo está hecho con una superficie de revolución y tiene movimiento de aletas, cola y bigotes.

- Pez Cirujano



El cuerpo esta modelado con una superficie de barrido cerrada y tiene movimiento de aletas y cola.

- Pez "Plateado"

El cuerpo es una superficie de barrido cerrada y tiene movimiento de aletas y cola.

- **Terreno**

El terreno está modelado con una superficie de barrido deformada, para dar la impresión de arena y con efecto niebla para simular el efecto del agua en la profundidad. Se agregaron grupos de algas y rocas y un jarrón con burbujas que utilizan transparencias. Las algas están animadas mediante una función senoidal en el eje z que deforma los puntos y desfasada en el tiempo para que la onda se propague entre cada cuadro de la animación.

- **Cámara**

La cámara tiene un modo "free look" y un modo "pez" en el que la cámara viaja con el cardumen.

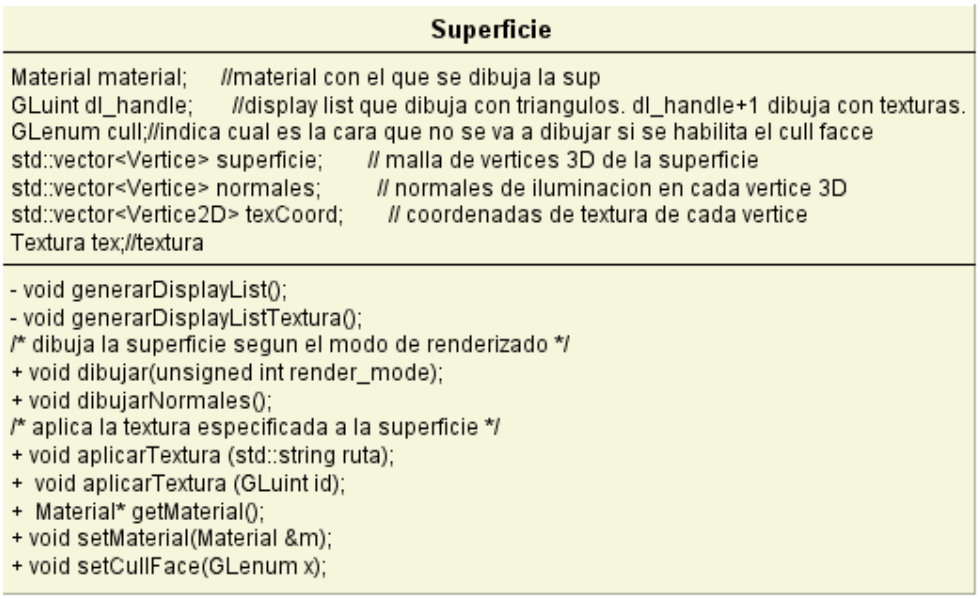
- **Estructuras de Datos**

El trabajo está organizado por las primitivas gráficas, que encapsulan la lógica para generar curvas de bezier y bspline, superficies de barrido y revolución, trayectorias, mapeo de textura, carga de imágenes y animación.

El Contenedor de Texturas se encarga de cargar y administrar las imágenes, para que una misma imagen no se cargue dos veces en memoria.

El Contenedor de Objetos es el que crea todas las superficies y animaciones. Es también quien sabe dibujar todos estos objetos. Esto permite tener una sola instancia de un objeto pero dibujarlo varias veces. Cada objeto está identificado con un ID para poder dibujar cada una de sus partes independientes.

El Controlador de Escena arma la escena, utilizando los objetos del contenedor y el stack de matrices de OpenGL. También se encarga de activar o desactivar las animaciones y mover la cámara en el modo "cámara pez"



ContenedorObjetos

{crea y dibuja los objetos}

```
Superficie* superficies[MAX_DIBUJOS];        //rocas y otros
Animacion* animaciones[MAX_ANIMACIONES];     //algas
Cardumen* cardumen[MAX_CARDUMEN];            //conj de peces
Superficie* cuerpos[MAX_TIPO_PECES];  //cuerpo de los peces
Animacion* colas[MAX_TIPO_PECES];            //cola de los peces
Animacion* aletas[MAX_TIPO_PECES];           //aleta de los peces
Vertice2D longitud[MAX_TIPO_PECES];          //y =longitud del centro a la cola del pez; x = ancho del pez en el 000;
Trayectoria *tray_burbujas;   //burbujas
GLuint handle_burbuja;   //handle para la display list de las burbujas
GLuint handle_agua;    //handle para la superficie del agua
GLuint idTexAgua;
```

```
// dibuja el objeto indicado por id
+ void dibujarObjeto(unsigned int id, unsigned int render_mode);
+ void dibujarAgua(unsigned int render_mode);
+ void dibujarCardumen(Cardumen* car, unsigned int render_mode);
+ void dibujarPez( uint id, uint render_mode, float escala);
+ void dibujarFlorero(uint render_mode);
+ Animacion* getAnimacion(unsigned int id);
+ Cardumen* getCardumen(unsigned int id);
+ void cambiarTexturaRoca(GLuint id);
+ void animarAlgas();
+ void animarPeces();
- void crearSuperficies();
- void crearAnimaciones();
- void crearCardumenes();
```

Cardumen

{Agrupa objetos con mismo id que siguen una trayectoria

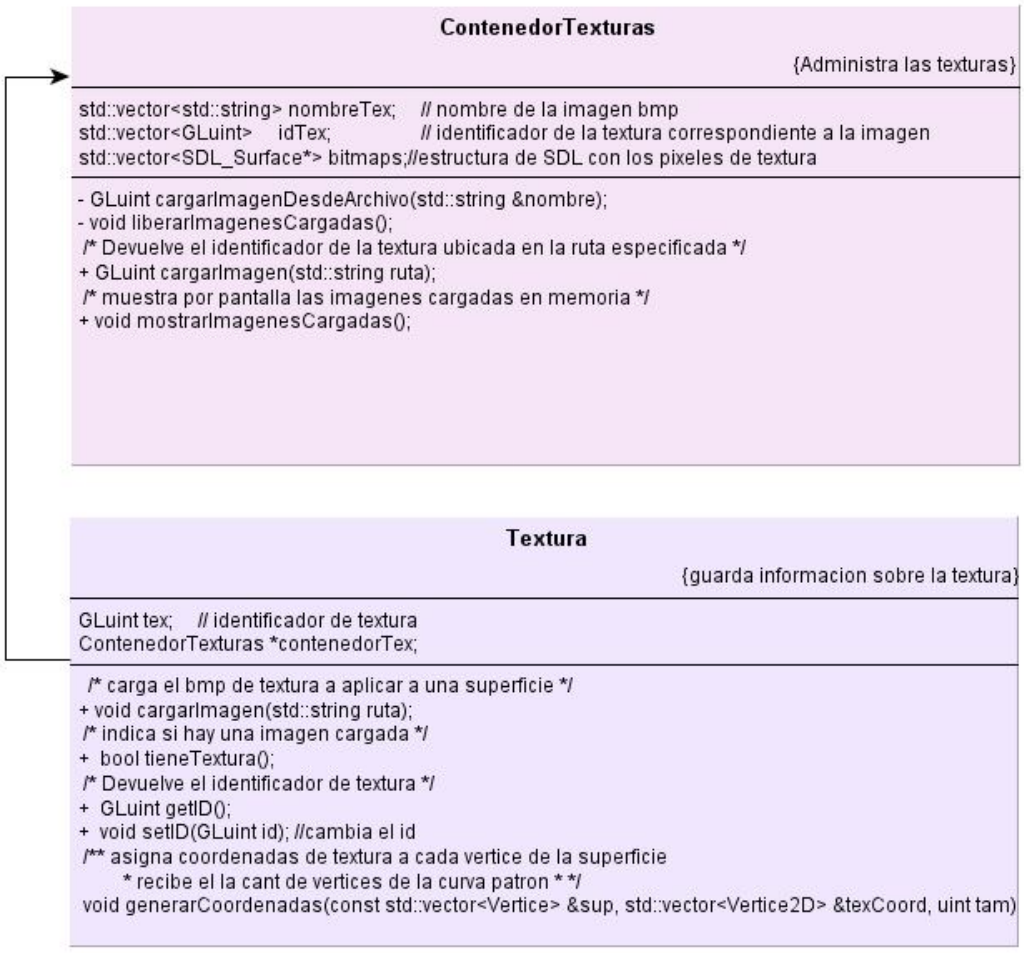
```
uint IDobjeto;
uint cantidad;        //cantidad de objetos a dibujar
float* volumen;       //escala de cada objeto (de 1 a 4)
Vertice* ubicacion;   //ubicacion de cada objeto en el grupo
Trayectoria* recorrido; //puntos que recorre el objeto
```

```
+ void viajar(); //avanza en su trayectoria
 //genera una ubicacion nueva para acomodar los peces en el cardumen
- void nuevaUbicacion(Vertice &v);
void init(); //genera escalas aleatorias para cada elemento.
```

ControladorEscena

{ubica los objetos en la escena y activa la animacion

```
+ bool estaAnimando();
+ void generarEscena();
+ void nextRenderMode();
+  unsigned int getRenderMode();
+ Camara* getCamara();
+ void nextAnimationMode();
+ void nextFrame();
+ int camaraCardumen(int nro);
+ void nextTrackDisplayMode();
```


ContenedorTexturas
{Administra las texturas}
std::vector<std::string> nombreTex; // nombre de la imagen bmp
std::vector<GLuint> idTex; // identificador de la textura correspondiente a la imagen
std::vector<SDL_Surface*> bitmaps;//estructura de SDL con los pixeles de textura
- GLuint cargarImagenDesdeArchivo(std::string &nombre);
- void liberarImagenesCargadas();
/* Devuelve el identificador de la textura ubicada en la ruta especificada */
+ GLuint cargarImagen(std::string ruta);
/* muestra por pantalla las imagenes cargadas en memoria */
+ void mostrarImagenesCargadas();
Textura
{guarda informacion sobre la textura}
GLuint tex; // identificador de textura
ContenedorTexturas *contenedorTex;
/* carga el bmp de textura a aplicar a una superficie */
+ void cargarImagen(std::string ruta);
/* indica si hay una imagen cargada */
+ bool tieneTextura();
/* Devuelve el identificador de textura */
+ GLuint getID();
+ void setID(GLuint id); //cambia el id
/** asigna coordenadas de textura a cada vertice de la superficie
* recibe el la cant de vertices de la curva patron * */
void generarCoordenadas(const std::vector<Vertice> &sup, std::vector<Vertice2D> &texCoord, uint tam)