



Universidad Tecnológica Nacional  
Facultad Regional Buenos Aires

# Técnicas de Gráficos por Computadora



Trabajo Práctico – Estrategia

**Render Group**

1er Cuatrimestre 2014

María Guadalupe Uviña

Julián Selser

Jonathan P. Buzzetti

## Contenido

Ejemplo Creativo: Pirate Ship .....	3
Funcionalidades obligatorias para 1ra entrega: .....	3
Funcionalidades obligatorias para la 2da entrega: .....	4
Funcionalidades opcionales: .....	4
Introducción.....	5
Jugabilidad.....	5
Controles.....	5
Objetivo .....	<b>Error! Bookmark not defined.</b>
Poderes .....	<b>Error! Bookmark not defined.</b>
Construcción del Ejemplo .....	6
Modelos 3D .....	<b>Error! Bookmark not defined.</b>
Escenario .....	7
Océano.....	<b>Error! Bookmark not defined.</b>
Barcos.....	8
Shaders .....	9
Aplicados a los modelos .....	13
Aplicados a la imagen final .....	14
Colisiones .....	14
Sprites .....	14

## Ejemplo Creativo: Pirate Ship



### Funcionalidades obligatorias para 1ra entrega:

1. Construir un escenario en donde hay dos barcos en el océano que se disparan cañones.
2. El barco tiene que ser manejado por el usuario en tercera persona.
3. El otro barco tiene que ser manejado por Inteligencia Artificial.
4. El barco manejado por el usuario se tiene que poder desplazar por el agua, con los siguientes movimientos:
  - a. Aceleración
  - b. Desaceleración
  - c. Virar
5. El agua debe tener marea, con olas grandes que suben y bajas en tiempo real (sin llegar a romper). El agua del océano debe contar con los siguientes efectos gráficos:
  - a. El agua debe tener efectos de oleaje utilizando un vertex shader que altera la posición de los vértices y las normales en tiempo real.
  - b. Utilizar iluminación dinámica para el sol (ambient + diffuse + specular) en pixel shader.
  - c. Aplicar Environment Map utilizando un CubeMap.
  - d. El bote debe adaptarse en tiempo real a la marea. Deberá inclinarse correctamente para adaptarse a la superficie del agua en donde se encuentra.

6. El usuario tiene que poder disparar cañones. Los cañones son esferas 3D que deben ser disparadas con tiro parabólico. El barco enemigo tiene que hundirse luego de recibir una cierta cantidad de disparos.

#### **Funcionalidades obligatorias para la 2da entrega:**

1. Debe haber un efecto de lluvia simulando una tormenta fuerte.
2. La inteligencia artificial del barco enemigo debe seguir al barco manejado por el usuario siempre a una distancia X y disparar cañones donde solo un porcentaje Y aciertan contra el barco.
3. La velocidad de desplazamiento del barco deberá variar según qué tan inclinado se encuentre en el agua:
  - a. Cuesta arriba debe avanzar más lento.
  - b. Cuesta abajo debe avanzar más rápido.

#### **Funcionalidades opcionales:**

1. Agregar efecto de truenos que ponen en blanco la pantalla momentáneamente.
2. Agregar más de un barco enemigo.
3. Agregar efecto de explosión de partículas cuando un cañonazo golpea un barco.

## Introducción



## Jugabilidad

### Controles

Navegar con las teclas **W**, **A**, **S**, **D**.

Disparar con la tecla **P**.

Rotar la cámara con el botón izquierdo del mouse (solo cuando no está en tercera persona).

### Objetivo




Destruir la mayor cantidad de barcos de la armada real a cañonazos. A medida que se matan enemigos el mar se va tiñendo de rojo.

### Poderes

A través del panel de modifiers, se puede alterar el entorno para probar otras variantes del juego.

### Modifiers

**User control:** es un modifier personalizado formado por un UserControl que contiene 4 controles Button. Al hacer click con el mouse sobre cada button:

-  Reproduce una animación con una gaviota que vuela.
-  Permite seleccionar la tecnica de los distintos shaders para alternar entre el día y la noche.
-  Permite visualizar la animación de la lluvia y el efecto de postprocesado de los rayos.



- Permite seleccionar la técnica de los distintos shaders para alternar entre mar congelado o no.

**Altura Marea:** es un control TrackBar con valores float entre 0.1 y 6.0 que modifica la altura de las olas cuando el océano no está congelado. El valor seleccionado es usado por la aplicación para modificar la altura máxima del heightmap y calcular la altura del barco en un punto.

**Fog Color:** es un control ColorDialog que permite al usuario seleccionar el color que se va a enviar como parámetro al shader aplicado en el océano para modificar el color de los píxeles. Es importante destacar que solo funciona cuando el escenario está en modo de día y sin hielo.

**Fog Start:** es un control TrackBar con valores float entre 50.0 y 7000.0. El valor seleccionado se envía al shader del océano como parámetro para calcular desde que valor de z empieza a calcularse la influencia del fog en el color final. Este modificador actúa solo cuando el agua no está congelada.

**Blend Start:** es un control como el de fog start, solo que envía al shader un valor que le indica desde que posición de z empezar a calcular el factor de alpha blending. Este valor actúa siempre y para todas las técnicas.

**Reflection:** es un control TrackBar con valores float entre 0.0 y 1.0. Su valor es enviado al shader del océano y representa el porcentaje de reflejo o influencia que tendrá la textura del cubemap del cielo sobre la textura del diffuse map. Actúa solamente cuando el océano no está congelado.

**CámaraEnBarco:** es un control CheckBox que permite al usuario seleccionar el tipo de cámara. Si tiene un valor true se usa la cámara en tercera persona que sigue al barco desde atrás. Si tiene un valor false se utiliza la cámara rotacional desde el centro del escenario y permite rotar la visión en todas las direcciones utilizando el mouse.

**Música:** es un control CheckBox que permite reproducir o detener la música.

## Construcción del Ejemplo

### Modelos 3D

Los meshes de los barcos fueron construidos completamente en el MeshCreator de la cátedra a partir de elementos disponibles como ejemplo. Dichos modelos son: el barco protagonista y las instancias de barco enemigo.

Además se agregó un muelle estático modelado en 3ds Max y exportado con el plugin de la cátedra “TgcSceneExporter.ms”.

## Escenario

El escenario se compone de 3 elementos principales: el skybox, el terreno y el agua.

El skybox se construyó con la clase provista por la cátedra y se le dio una textura que puede ser cambiada para simular el día o la noche.

El resto del escenario fue construido a partir de tres terrains, cada uno con un relieve dado por sus respectivos heightmaps: uno para el océano, uno para la cascada y otro para las montañas. Cada uno cuenta con un shader específico y una textura creada especialmente para el ejemplo a partir de la edición de imágenes fotográficas.

## Océano

El océano es una instancia de la clase SmartTerrain, por lo tanto requiere de una textura, que en nuestro caso es un heightmap negro de 64 x 64 px. Sólo es útil para darle el tamaño, es decir, la cantidad de vértices a la malla resultante, ya que el movimiento de los mismos en el eje Y se calcula en el GPU. Los cálculos de la posición se encuentran en el vertex shader aplicado:

```
float y = amplitud* sin(x/5 - time) * amplitud* cos(z/2 - time);
```

Generando un campo continuo que simula el movimiento ondulatorio de la marea. Para agregarle un poco de ruido se le suma a la altura otra función sinoidal pero esta vez con una amplitud más pequeña.

El trabajo requería conocer la normal del océano en todos los puntos, para adaptar la inclinación del barco y reflejar correctamente la iluminación. Para esto, se necesitó realizar el cálculo en 2 momentos diferentes. El barco necesitaba la información antes de entrar en el pipeline, por eso se replicó la fórmula en el código de C#. En cambio la normal en cada punto para la reflexión podía calcularse dentro del mismo shader.

La normal en un punto P se obtuvo en ambos casos tomando la altura del terreno en 4 puntos a una distancia  $\square < (\text{amplitud}/4)$  de P, formando una cruz. El producto cruz de estos 2 vectores co-planares da como resultado la normal. También se pueden reducir los cálculos tomando solo 2 puntos, pero la precisión es menor.

## Colisionante

Los elementos que interactúan entre sí lo hacen mediante un boundingSphere; Dichos elementos están generalizados por una clase “Colisionante” que se encarga del movimiento y detección de colisiones. Adicionalmente esta clase tiene la responsabilidad de mantener la dirección del mesh (un colisionante es un TgcMesh). Las bolas de cañón disparadas por los barcos también serán colisionantes que podrán chocar contra los barcos.

## Barcos

Existen dos tipos de barcos:

- El protagonista controlado por el jugador que es responsable de manejar la cámara para ser acompañado navegando el escenario recibiendo el input correspondiente.
- El enemigo está constantemente observando y siguiendo al jugador hasta que lo alcanza, deteniéndose a una distancia X para dispararle. Cuando un enemigo cae en combate se encargará de colocar en el escenario de 1 a 2 enemigos más de manera aleatoria.

Ambos tipos de barcos extienden de una clase más general que generaliza las características de todos los barcos, esto es: flotar, disparar, moverse, acelerar.

Es importante destacar la flotación ya que ah sido un punto central en el desarrollo del T.P. y se explica a continuación: Siendo que el Océano puede decirnos en cada punto su normal, esta misma será la normal del barco y mediante proyecciones y operaciones trigonométricas pueden conseguirse las rotaciones en X, Y, Z usando las siguientes fórmulas:

$$\text{Rotacion} = \arctg(\text{CatetoOpuestoProyectado} / \text{CatetoAdyacenteProyectado})$$

Sin embargo esto no contempla las rotaciones, para contemplarlas hay considerar en el numerador del cociente el ángulo de rotación perpendicular que llamaremos Y con alguna de las dos onda senoidales(Sen o Cos):

$$\text{Rotacion} = \arctg(\text{CatetoOpuestoProyectado} * \text{Cos}(\text{rotacionY}) / \text{CatetoAdyacenteProyectado})$$

Aunque para ambas se anularan en sus ceros, por lo que hay que sumar ambas para mantener la intensidad, luego:

$$\begin{aligned} \text{Rotacion} = & \arctg(\text{CatetoOpuestoProyectado} * \text{Cos}(\text{rotacionY}) / \text{CatetoAdyacenteProyectado}) \\ & + \\ & \arctg(\text{CatetoOpuestoProyectado} * \text{Sin}(\text{rotacionY}) / \text{CatetoAdyacenteProyectado}) \end{aligned}$$



## Shaders

Los shaders en el ejemplo fueron aplicados principalmente para lograr movimiento (en el agua), para la mejorar la representación del espacio (en todo el escenario) y la calidad visual en general.

La representación del espacio tridimensional en la bidimension fue uno de los principales focos de atención al idear el escenario. Los shaders fueron pensados para complementar los aspectos que parecían imposibles de lograr con el modelado del escenario estático.

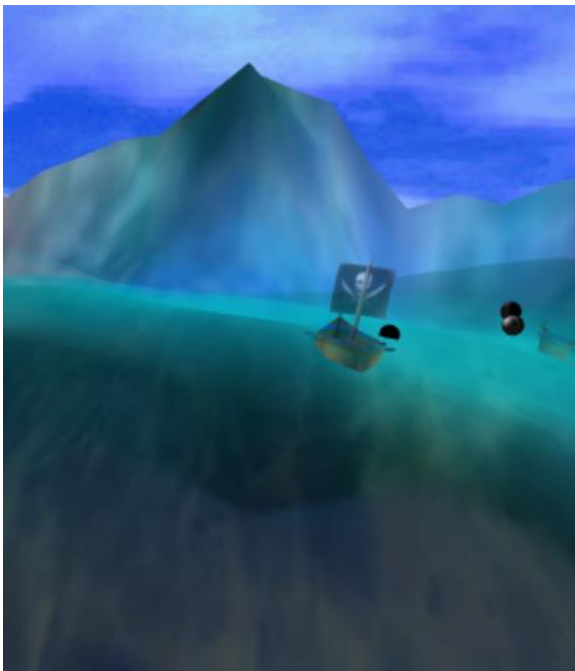
La representación en perspectiva apareció de forma bastante natural por las características de la cámara que ya viene implementada, el tamaño y la posición de los objetos fueron determinados al crear el escenario, pero la iluminación, la transparencia y la perspectiva atmosférica eran los indicadores de espacio que quedaban por modificar para lograr un espacio más realista, estos fueron los más afectados por los shaders para mejorar la calidad gráfica.

El uso del color fue otro de los aspectos que se tuvo en cuenta para mejorar la calidad visual del ejemplo. El uso de texturas y su modificación en tiempo real fue posible gracias al uso de shaders. Utilizando las posibilidades ofrecidas por los pixel shaders se logró el efecto que alterna entre el día y la noche, y entre el escenario congelado y sin congelar, y el efecto del mar que se tiñe de rojo al matar enemigos, entre otros.

## Algunos efectos logrados con shaders



1-cambios en la iluminación y los detalles de textura para simular la noche.



2-perspectiva atmosférica con efecto fog y alpha blending.



3-efecto de barco roto con alpha blending



4-efecto mar y barco rojos después de matar varios barcos enemigos.

### **Uso de texturas:**

**Para el ejemplo se crearon distintas texturas con diferentes usos que se envían al shader como parámetros para lograr varios efectos:**

- Texturas para diffuse map: se diseñaron para simular la luz natural del día y se aplicaron los pixel shader para adecuarlos a los cambios en tiempo de ejecución.
- Cube maps: se diseñaron 2 cubemap distintos utilizando GIMP2 para exportarlos en formato .dds, uno para el día con nubes y cielo celeste y otro para la noche con una luna y estrellas. Los dos se utilizaron para mejorar el realismo del agua al simular la reflexión, como textura auxiliar al diffuse map. Los 2 cubemap contienen además de la textura del cielo la información de las montañas y la cascada.
- Mapa de recorte: textura de tipo mascara auxiliar que se utilizó para lograr calar los meshes de los barcos.

### **Indicadores de espacio:**

**Fueron muy considerados diferentes indicadores a la hora de investigar, desarrollar e implementar los shaders en los modelos:**

- Color: el modelado del color es el efecto principal para lograr la ilusión de espacio tridimensional. Para lograr un buen manejo de este recurso es que se confeccionaron a medida todos los diffuse map aplicados al ejemplo. Se trató de aplicar texturas lo más optimizadas posibles y en su mayoría son imágenes de color indexado en formato .png.
- Transparencia con alpha blending: la transparencia se usó principalmente en el agua y como complemento de otros efectos.
- Perspectiva atmosférica con fog: la perspectiva atmosférica es uno de los recursos más utilizados por el lenguaje visual tradicional para simular la distancia en los espacios al aire libre. Se usa en general en combinación con otros indicadores como las relaciones de tamaño y el nivel de detalle y simula la distancia modificando las propiedades de saturación del color. A medida que los objetos se acercan al horizonte los colores pierden saturación y se hacen más grises o celestes para simular las capas de aire de la atmosfera, este efecto es el que se resolvió utilizando el fog, y algo de alpha blending, aplicado en profundidad.
- Superposición: la superposición es uno de los recursos más básicos para indicar relaciones espaciales. Al aplicar el alpha blending al agua se puede ver a través para indicar que el espacio continúa por debajo del plano del mar o de la cascada.

La reflexión del cielo y las montañas sobre el agua también fue de ayuda con la representación de la ubicación espacial entre los objetos en el escenario.

- Nivel de detalle con alpha blending y fog: para cambiar la iluminación del escenario y que la escena resulte creíble fue necesario modificar las texturas en tiempo de ejecución. Cuando se hace de noche, por ejemplo, la intensidad de la luz disminuye, y por lo tanto los objetos deberían ser menos visibles, para esto se aplicó un fog de color negro que actúa no solamente restando luz a las texturas sino también restándole detalles a las cosas que están más alejadas de la cámara. En el caso de la nieve es el ejemplo inverso, la nieve real refleja más luz blanca en todas direcciones y por lo tanto se pierden los detalles por la cantidad de reflejos en los objetos. La propiedad de alpha blending también se usó para mejorar estos efectos.

#### Aplicados a los modelos

##### **Shader de las montañas:**

Para las montañas se diseñó una textura con mezcla de piedras y arrecife de coral en la parte que cubre el mar. Para que esta textura se adapte a los cambios de luminosidad se aplicó un pixel shader que básicamente oscurece o aclara el diffuse map modificando los valores RGB de la textura y agregando más color blanco para simular la nieve y más negro para la noche.

En algunas de las técnicas se aplicó un factor de fog lineal que depende de la distancia de cada vértice con respecto de la cámara para lograr efecto de perspectiva atmosférica, este efecto es el que permitió que los tonos y la cantidad de luz sobre una región cambie a medida que la cámara se mueve.

##### **Shader de los barcos:**

Los barcos cuentan con un pixel shader que mediante el uso del alpha blending logra dos efectos muy importantes: uno es el que les permite desaparecer a la distancia y el otro es el uso de un mapa de recorte o máscara para simular que se van rompiendo a medida que reciben disparos.

La desaparición de los barcos a la distancia fue una decisión de diseño para adecuarse a las características del océano. Para lograr profundidad en un escenario chico se aplicó un factor de alpha blending que actúa usando la posición de z del vértice transformado.

Desde la aplicación se envía un mapa de recorte al shader y un factor de calado que depende de cuantos disparos recibió el barco. Este mapa es una textura en negro y distintos tonos de rojo. El pixel shader calcula que tan calado está un barco dependiendo de la composición RGB del pixel de la máscara, si el factor de calado enviado desde la aplicación supera un cierto nivel el pixel es descartado y no se ve por pantalla modificando

el canal alpha del pixel. El color negro indica que esa parte nunca se cala con los disparos, y el rojo que sí.

Del mismo modo que en las montañas también se aplicaron los cambios en la luminosidad para adaptarse al día, la noche y la nieve modificando los valores RGB del diffuse map.

### **Shader de la cascada:**

La cascada cuenta un vertex shader muy simple que cambia las coordenadas de textura usando una variable tiempo que es enviada desde la aplicación para simular la caída del agua cuando no hay hielo y un pixel shader que agrega los efectos de luz dinámica, la reflexión del cubemap del cielo y los cambios de luminosidad para el día y la noche como el del shader de las montañas.

### **Shader del océano:**

El shader aplicado al océano es el más complejo del ejemplo porque hace casi todo lo que hacen los otros shaders: el pixel shader cuenta con una combinación de iluminación dinámica, alpha blending, fog, cambios de luminosidad y el vertex shader mueve los vértices, las normales y las coordenadas de textura en tiempo real para simular el movimiento de las olas.

Este shader cuenta también con un efecto que altera el color del mar a medida que mueren los barcos del juego. Por cada barco hundido la aplicación aumenta el valor de un contador que se envía al shader como parámetro para calcular que tan roja se ve el agua. Es un efecto muy simple pero, de la misma forma que el efecto de calado de los barcos, se agregó para brindar información visual al usuario sobre el estado de las variables del ejemplo sobre las que tiene control.

[Aplicados a la imagen final](#)

Para lograr el efecto de rayos cuando esta activada la lluvia se usó un shader de postprocesado que altera los valores RGB de la textura auxiliar del render target que se va a renderizar para que la pantalla se ponga blanca por unos segundos.

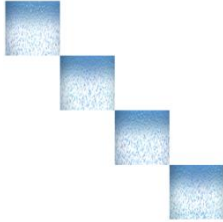
## **Colisiones**

Descripción

## **Sprites**

Los sprites utilizados en el ejemplo sirven más que nada como complemento visual y para resolver algunas cuestiones de requerimientos de manera simple. Fueron realizados utilizando Adobe Photoshop como imágenes en formato .png para permitir transparencias y se usaron para agregar las animaciones:

- Un sprite sheet animado para simular la lluvia que cuenta con cuatro fotogramas.



- Otro sprite sheet animado con una gaviota con 4 fotogramas que además implementa animación por transformaciones lineales de traslación y escalado.



- Un sprite 2d de timón de barco que rota siguiendo la rotación absoluta de la cámara en tercera persona



- Un sprite 2d fijo con textura de madera.

