IE–TEC

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
MP-6123 Procesamiento Digital de Imágenes
Profesor: MSc. Carl Michael Gruner Monzon
II Cuatrimestre, 2016

Proyecto 1

Pegasus
Módulo PTZR Digital

1. Problema

Nuestra empresa *Pegasus* se dedica a la instalación y mantenimiento de tuberías. Dada la naturaleza del trabajo, requerimos de equipo especial para inspeccionar secciones de difícil acceso. Uno de los equipos de uso más frecuente es una cámara delgada y flexible capaz de proveer retroalimentación visual de la integridad de estos lugares. La Figura 1 muestra una de estas cámaras.

Figura 1: Camara para inspección de tuberías

Las imágenes son transmitidas al módulo receptor mediante un protocolo propietario de baja latencia con limitado ancho de banda. Lo anterior nos confina a un tamaño de imágen de máximo 300×300 pixeles para tener un video fluído. Sin embargo, este tamaño de imagen no permite una exploración ágil de la tubería y los detalles de los sectores laterales pueden quedar fuera de la escena capturada.

La solución ideal sería dotar al sistema de un módulo que permita el movimiento mecánico del sensor. Sin embargo, por diferentes motivos, en este momento esa no es una solución viable. Se quiere, entonces, tener un mecanismo digital que emule el comportamiento de una cámara PTZ (*Pan, Tilt y Zoom* por sus siglas en inglés). La Figura 2 muestra un ejemplo de este tipo de cámara.

El sistema deberá capturar una imágen de *Full-HD* (1920×1080 pixeles) o *HD* (1280×720 pixeles), tomar un segmento de la misma con un tamaño y posición particular y escalarlo al tamaño final de

Figura 2: Cámara PTZ

300 × 300 pixeles. La posición del segmento simulará el *Pan* y el *Tilt*, mientras que el tamaño del mismo simulará el *Zoom*. Además, se requiere la capacidad de rotar la imágen final para facilitar su inspección, en caso de ser necesario. La Figura 3 ejemplifica el concepto anterior.

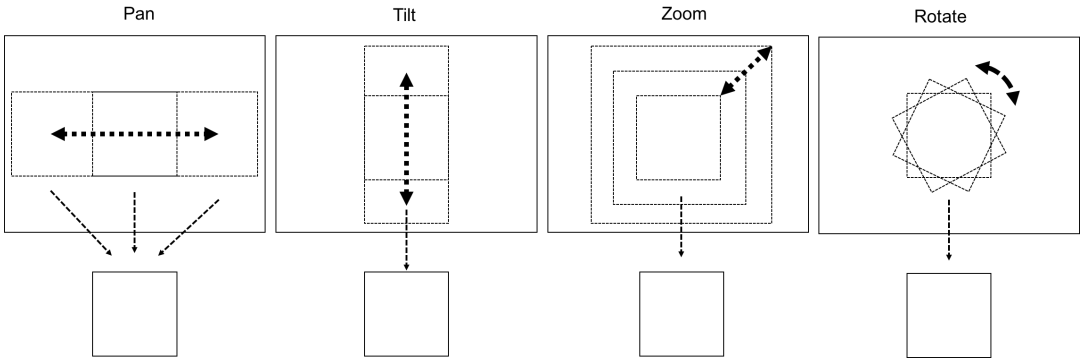

Figura 3: PTZR digital

2. Especificación del Sistema

El sistema deberá capturar una imagen de entrada de alta definición de 1920 × 1080 pixeles (o de 1280×720 pixeles si su computadora o cámara digital no lo permiten). El usuario especificará el nivel de *Pan*, *Tilt*, *Zoom* y *Rotate* mediante parámetros configurables en tiempo real. Como resultado, el sistema desplegará en una pantalla una imagen de 300 × 300 pixeles según los parátros elegidos.

El sistema deberá operar a 30 cuadros por segundo. El usuario podrá ajustar los parámetros en tiempo real y los cambios deberán verse reflejados en la imagen de salida inmediatamente.

- **Entrada**: Imagen en alta definición 1920 × 1080 pixeles (o 1280 × 720 pixeles).
- **Entrada**: Parámetro de nivel de *Pan*.
- **Entrada**: Parámetro de nivel de *Tilt*.
- **Entrada**: Parámetro de nivel de *Zoom*.

- **Entrada**: Parámetro de nivel de *Rotate*.
- **Frecuencia de operación**: 30 fps.
- **Salida**: Imagen resultante de 300×300 pixeles

3. Entorno de Desarrollo

3.1. OpenGL

El prototipo deberá ser implementado utilizando las facilidades del GPU en su sistema, mediante OpenGL. OpenGL es un estándar para la manipulación y procesamiento de gráficos digitales. Su arquitectura interna está diseñada para operaciones típicas de procesamiento de imágenes, como traslaciones, rotaciones, escalamientos, transformaciones geométricas, manejo de texturas, iluminación, sombras, entre otros. El problema propuesto puede ser resuelto de manera natural en este pipeline gráfico.

El pipeline (tubería) de OpenGL consta de múltiples etapas, tal y como se muestra en la Figura 4.

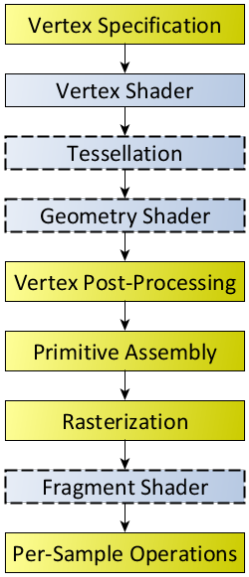

Figura 4: Pipeline de renderización de OpenGL

El proceso inicia mediante una especificación de vértices por parte del usuario. Seguidamente, estos vértices pasan por diferentes etapas hasta que finalmente se rasterizan en la ventana para su despliegue. Muchas de estas etapas son programables por el usuario, para lograr las manipulaciones personalizadas sobre los vertices. Dichos módulos programables se llaman *Shaders* y se programan mediante un lenguage denominado *GLSL* (*OpenGL Shading Language*).

Para este proyecto se recomienda el uso de dos *Shaders*:

- Vertex Shader

- Fragment Shader

3.1.1. Vertex Shader

Este shader permite al usuario realizar transformaciones geométricas a los vértices especificados. Para ello utiliza tres matrices de transformación configurables por el usuario [1]:

1. **Matriz de modelo**: Transforma de las coordenadas del modelo a coordenadas del mundo.
2. **Matriz de vista**: Transforma de las coordenadas del mundo a coordenadas de la cámara.
3. **Matriz de proyección**: Permite realizar transformaciones proyectivas sobre los vértices en coordenadas de la cámara.

La Figura 5 muestra estas transformaciones.

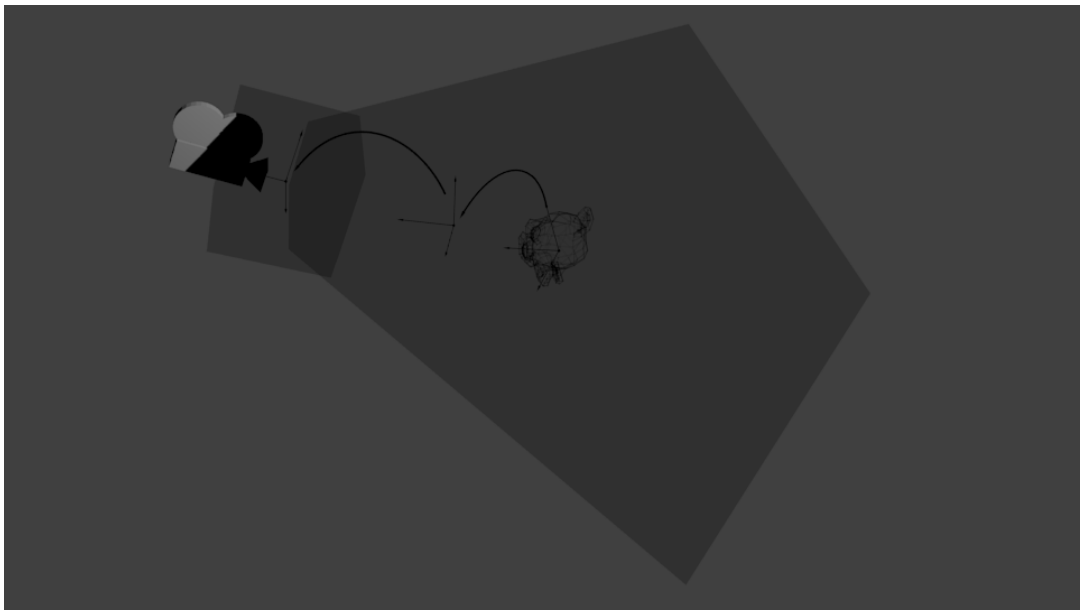

Figura 5: Matrices de transformación de OpenGL

Utilizando estas transformaciones, es posible lograr efecto de *Pan*, *Tilt*, *Zoom* y *Rotate* deseado. La Figura 6 muestra una posible solución.

3.1.2. Fragment Shader

Este shader permite al usuario realizar transformaciones sobre el valor de los pixeles una vez estos han sido transformados geométricamente.

Utilizando este shader, es posible renderizar la imagen capturada sobre el conjunto de vértices especificados al inicio del pipeline.

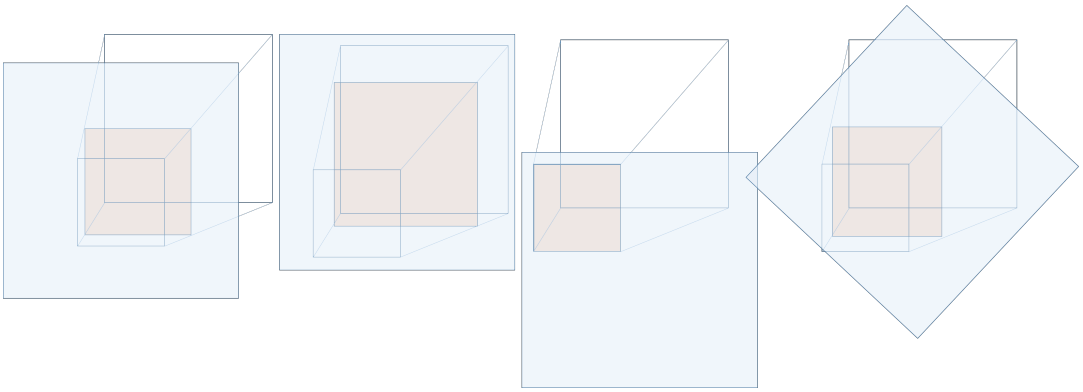

Figura 6: Propuesta de PTZR utilizando OpenGL

4. Entregables y Fecha Límite

- Prototipo en GNU/Octave o Matlab que valide las matrices de transformación.
- Código fuente de aplicación.
- Interfaz gráfica que incluya:
  - Captura de imagen en alta definición
  - Despliegue de la imagen resultante
  - Habilidad para especificar parámetros de configuración.
- Artículo científico en el formato estándar de la IEEE con extensión de no más de 5 páginas.

El proyecto tiene una extensión de dos semanas. Dadas las vacaciones del TEC (del 4 al 17 de julio), el proyecto deberá entregarse el lunes 25 de julio antes de las 23:59.

Referencias

[1] Anónimo. Tutorial 3 : Matrices, 2010. Available from: http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/.