Proyecto 2: 2D World

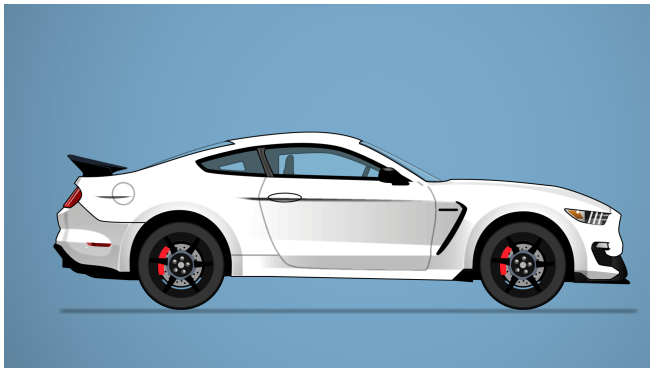
Profesor: Ernesto Rivera Alvarado

I. Introducción

En este proyecto, el estudiante se expondrá a la aplicación de los algoritmos vistos en clase para crear una imagen sintetizada por computadora a través de polígonos y curvas de Bezier. Se aplicarán texturas y coloreo de polígonos. La programación se hará en C sobre Linux.

II. La imagen

Los estudiantes recrearán la imagen que se muestra a continuación:



Cada uno de los elementos mostrados en la imagen debe representarse por un conjunto de polígonos. Cada polígono se representará como una secuencia de vértices en coordenadas universales (para una calidad aceptable se estima que se debe contar con al menos 1000 vértices en total). A la hora de mostrar el proyecto, se inicializará una ventana en coordenadas universales en donde se podrá apreciar la imagen completa. Para evitar distorsiones, la proporción de la ventana y el *framebuffer* serán iguales (vertical horizontal).

III. Despliegue

La imagen se podrá desplegar en cuatro modos diferentes que se describen a continuación:

1) Sin color: Únicamente se despliegan los bordes que componen cada polígono. Estos serán líneas rectas con el algoritmo de Bresenham.
2) Con color: Cada polígono será pintado con un color que se asemeje de la manera más cercana a la imagen original.
3) Con texturas: Todos los polígonos se pintarán texeles que provendrán de imágenes. Los estudiantes seleccionarán un repertorio de imágenes para utilizar como texturas.
4) Con una misma textura: Se utilizará para colorear todos los polígonos una sola textura, la cual será una versión recortada de la imagen provista en este documento con una resolución horizontal de 128 pixeles que muestra el carro completo.

IV. Modos de despliegue

Con una interfaz sencilla y a través de OpenGL/Vulkan o SDL (investigar cómo gestionar mouse y teclado), se podrá interactuar con el programa de la siguiente manera.

- Modo: Permite seleccionar entre el modo sin color, con color, con texturas y con una misma textura.
- Zoom-In/Zoom-Out: Podrá ser tanto de acercamiento como de alejamiento. Toda la imagen será redibujada.
- Pan Left/Right/Up/Down: La ventana podrá moverse hacia la izquierda, derecha, arriba y abajo.
- Rotación: Se podrán rotar los contenidos de la ventana en ambos sentidos de las manecillas del reloj. Los ángulos de rotación serán de 15 grados.
- Reiniciar: Vuelve a la vista inicial del mapa.
- Terminar: Se termina el programa.
- Firmar: A través de curvas de Bezier, se firmará la imagen mostrada en alguna ubicación con las firmas de ambos integrantes. La firma se verificará contra cédula.

El sistema debe tener la capacidad de combinar las operaciones de la manera en que se desee, por ejemplo, hacer zoom, pan left, rotar y rellenar con texturas.

Notar que la implementación de este proyecto requiere *clipping* de líneas y polígonos con los algoritmos vistos en clase y también relleno de polígonos (con o sin textura).

V. Puntaje opcional extra

Hacer que el Zoom in/ Zoom out tenga poco incremento o decremento de la ventana cada vez que se pida la operación (modo lento) o mediante alguna interfaz apropiada, por ejemplo presionando *shift* al mismo tiempo, hacer que este incremento o decremento sea más marcado (modo rápido).

Hacer que el Pan en cualquier sentido tenga poco incremento o decremento de la posición de la ventana cada vez que se pida la operación (modo lento) o mediante alguna interfaz apropiada, por ejemplo presionando *shift* al mismo tiempo, hacer que este incremento o decremento sea más marcado (modo rápido).

Hacer que la rotación del mapa en cualquier sentido sea por pocos radianes cada vez que se pida la operación (modo lento) o mediante alguna interfaz apropiada, por ejemplo presionando *shift* al mismo tiempo, hacer que la rotación sea por una mayor cantidad de radianes (modo rápido).

VI. Sobre la revisión y completitud del proyecto

Los puntos 2 y 3 se evaluarán únicamente si la fidelidad de la imagen es buena. La evaluación del proyecto se dividirá de la siguiente manera:

1) Fidelidad de la imagen pintada con líneas 35%.
2) Firma. 5%.

3) Características mencionadas en la descripción del proyecto.
   a) Pan, Zoom, Rotación. 15%.
   b) Relleno con color. 20%.
   c) Relleno con texturas. 15%.
   d) Combinación de operaciones: Pan, Zoom, Rotación. 10%.
4) Puntaje extra opcional 10%.

VII. Requisitos indispensables

La ausencia de uno solo de los siguientes requisitos vuelve al proyecto "no revisable" y recibe un 0 de calificación inmediata:

- Todo el código debe de estar escrito en C sobre Linux.
- Las funciones de OpenGL/Vulkan o SDL se limitan a crear una ventana, gestionar la interacción con el usuario (mouse y teclado) y apagar y encender pixeles en colores definidos.
- No debe presentarse "segmentation fault" por ninguna razón.
- La colaboración entre grupos se considera fraude académico.
- Todo el código debe estar escrito en C (no C++).
- El proyecto debe compilar y ejecutar en Linux. Todo debe estar **integrado**, explicaciones del tipo "*todo está bien pero no pudimos pegarlo*"[1] provocan la cancelación automática de la revisión.
- Todos los algoritmos gráficos deben haber sido desarrollados por los estudiantes. El uso de alguna biblioteca (MESA / GLUT / OpenGL / Vulkan / SDL o equivalente) se limita a crear una ventana, alterar el color de los píxeles, encender el píxel y manejar la interacción con el usuario.
- La presentación debe ser de mucha calidad.
- No debe dar "Segmentation Fault" bajo ninguna circunstancia.
- Hacer la demostración en una máquina que levante Linux de manera real (puede ser dual), es decir no usar máquinas virtuales.
- La presentación debe hacerse en clase en la computadora del estudiante.
- Debe seguir a cabalidad lo descrito en la **Sección II, III y IV** del presente documento.
- Los estudiantes deben demostrar dominio completo de su proyecto en la presentación oral.
- Los estudiantes deben utilizar el sistema de control de versiones Git tal como se describe en este documento.

La evaluación de este proyecto está sujeta a la presentación oral del mismo, en la que el estudiante debe mostrar un dominio completo del trabajo realizado a nivel de código. En caso de que el estudiante no muestre un dominio completo del trabajo, se considerará el trabajo como "no revisable" y se asignará la nota de cero. La defensa del proyecto será grabada.

[1]esto incluye los supuestos casos cuando alguien del grupo de trabajo no hizo su parte – el profesor no está interesado en sus problemas de organización.

En caso de trabajar en pareja, ambos integrantes deberán tener un dominio completo del trabajo que están presentando. El profesor podrá hacerle preguntas a cualquiera de los integrantes y este deberá contestar acordemente. En caso de que se evidencia que uno de los integrantes desconoce el funcionamiento de alguna parte del trabajo presentado, se le asignará la nota de cero a ambos.

VIII. Desarrollo del proyecto

El proyecto está pensado para desarrollarse individualmente, sin embargo, los estudiantes que deseen reforzar la habilidad de trabajo en equipo pueden entregar el proyecto en parejas. El profesor les hace la aclaración de que el trabajo con un compañero conlleva dificultades de coordinación, división de trabajo y sobre todo de "pegar o juntar ambas partes". En experiencias propias del profesor, se les comenta que en ocasiones el trabajo de juntar, acoplar y corregir partes desarrolladas por diferentes personas conlleva más tiempo y trabajo que la realización individual.

IX. Git

Los estudiantes deberán utilizar el sistema de control de versiones Git, específicamente a través de GitLab. Para ello configurarán desde sus máquinas Gitlab, para poder hacer cualquier acción (ejemplo *push*, *pull*, *commit*) desde **consola de texto**, sin la necesidad de ingresar la contraseña constantemente. Además agregarán al repositorio del proyecto al usuario Ernesto.cursos, donde el profesor podrá observar los avances y desarrollos del proyecto por semana, día, hora, minuto y segundo.

X. Consideraciones sobre plagio

El trabajo presentado por los estudiantes (ya sea a nivel individual y de parejas) debe ser de su propia autoría. No se permite utilizar código realizado por un tercero (sin importar la licencia de dicho código) o entre compañeros del mismo curso, y en caso de que esto ocurra, será considerado plagio por lo que se seguirá el proceso correspondiente. Códigos encontrados en "github" de los cuales solo se tomaron "partes" será considerado plagio, el estudiante es el responsable de desarrollar absolutamente todo su código, a excepción del código que se menciona como base en el presente documento. De igual manera, cualquier código que el estudiante no tenga la capacidad de explicar, será considerado como plagio.

XI. Fecha de entrega

Las demostraciones se harán en la **semana 8** asignadas por citas distribuidas aleatoriamente en las dos lecciones de la semana.

La carpeta comprimida `.zip` de su proyecto debe contener los archivos fuentes `.c` y complementarios, además del archivo `Makefile` que genera el ejecutable del código fuente. El nombre de la carpeta comprimida debe de ser los apellidos de integrantes del grupo de trabajo (por ejemplo `rivera-alvarado.zip`) y este será subido al correo del profesor y al Tec Digital el día de la revisión a antes de las

6 am. Los estudiantes deberán defender el trabajo realizado y mostrar un dominio completo del trabajo desarrollado. El no tener conocimiento de cómo funcionan partes del proyecto invalida la entrega y se asignará la nota de cero. El título del correo será [Proyecto 2 - Gráficos por Computadora].