**OBSERVACIONES DEL TALLER: VISUALIZADOR OBJ CON TRACKBALL**

**Curso**: Computación Gráfica  
**Fecha**: 3 de Septiembre 2025  
**Estudiantes**: Abel Albuez Sanchez, Ricardo Crus  
**Profesor**: Leonardo Florez-Valencia

**1. PLANTEAMIENTO DEL PROBLEMA**

**1.1 Ajustes para correrlo en mac**

* Se hicieron portables los includes de OpenGL/GLUT en BaseApp y Mesh para reconocer las rutas específicas de Apple.
* Se añadió la cabecera estándar necesaria para habilitar el uso de std::string en el header de BaseApp.
* Se incorporó Eigen como dependencia interna (header-only), descargando y colocando la carpeta Eigen dentro de lib/eigen3/Eigen.
* Se normalizaron las rutas de inclusión del compilador para apuntar a lib y lib/eigen3.
* Se adaptó el Makefile a macOS, enlazando con los frameworks nativos OpenGL y GLUT y manteniendo clang++ y las banderas de compilación.
* Se agregó un script run.sh y se configuraron sus permisos para facilitar la ejecución del binario con un archivo .obj.

**1.2 Objetivo**

Implementar un programa para inspeccionar modelos 3D en formato Wavefront OBJ utilizando el ratón con lógica de trackball.

**1.2 Requisitos Específicos**

* Lectura de archivos OBJ
* Control de rotación mediante trackball virtual
* Interacción intuitiva con el mouse
* Visualización en tiempo real

**1.3 Justificación**

La interacción tipo trackball es fundamental en aplicaciones de gráficos 3D porque permite rotación natural e intuitiva de objetos sin las limitaciones del control por ejes separados.

**2. MARCO TEÓRICO**

**2.1 Trackball Virtual**

El trackball virtual es una metáfora de interacción que simula una esfera imaginaria alrededor del objeto 3D. El usuario puede "rotar" esta esfera arrastrando el mouse, como si estuviera manipulando una pelota física.

**2.2 Fundamentos Matemáticos**

**2.2.1 Proyección 2D a 3D**

La posición del mouse (x,y) se proyecta sobre una esfera unitaria:

x' = (2x - ancho) / ancho

y' = (alto - 2y) / alto

z' = √(1 - x'² - y'²) si x'² + y'² ≤ 1

z' = 0 si x'² + y'² > 1

**2.2.2 Cálculo de Rotación**

* **Eje de rotación**: n = v₁ × v₂ (producto cruz)
* **Ángulo**: θ = arccos(v₁ · v₂) (producto punto)
* **Matriz de rotación**: Fórmula de Rodrigues

**3. METODOLOGÍA DE DESARROLLO**

**3.1 Análisis de Requerimientos**

Se identificaron las siguientes necesidades:

1. Carga eficiente de modelos OBJ
2. Rotación fluida en cualquier dirección
3. Control de zoom para inspección detallada
4. Desplazamiento (pan) para centrar áreas de interés
5. Normalización automática para diferentes escalas

**3.2 Diseño de la Solución**

Se optó por una arquitectura modular con separación de responsabilidades:

* **Trackball.h/cxx**: Encapsula la lógica de interacción
* **ShowOBJ.cxx**: Integra el trackball con la visualización
* **BaseApp/Mesh**: Clases base proporcionadas

**3.3 Implementación**

La implementación se realizó en dos fases:

1. **Fase 1**: Desarrollo de la clase Trackball con rotación básica
2. **Fase 2**: Integración de características adicionales (pan, zoom, normalización)

**4. RESULTADOS OBTENIDOS**

**4.1 Funcionalidades Implementadas**

| **Funcionalidad** | **Descripción** | **Control** |
| --- | --- | --- |
| Rotación 3D | Trackball virtual completo | Click izquierdo + arrastrar |
| Zoom | Acercamiento/alejamiento | Rueda del mouse, teclas +/- |
| Pan | Desplazamiento en plano XY | Click medio + arrastrar |
| Reset | Restaurar vista inicial | Tecla R |
| Modo wireframe | Alternar sólido/líneas | Tecla W |
| Normalización | Ajuste automático de escala | Automático al cargar |

**4.2 Características Técnicas**

* **Lenguaje**: C++ con OpenGL/GLUT
* **Librerías**: Eigen3 para álgebra lineal
* **Arquitectura**: Orientada a objetos, modular
* **Compatibilidad**: Linux, macOS, Windows (con ajustes)

**4.3 Ventajas de la Solución**

1. **Modularidad**: Clase Trackball reutilizable
2. **Intuitividad**: Control natural tipo trackball
3. **Robustez**: Manejo correcto de casos límite
4. **Eficiencia**: Renderizado fluido en tiempo real

**5. ANÁLISIS Y DISCUSIÓN**

**5.1 Comparación de Enfoques**

Durante el desarrollo se evaluaron dos enfoques:

**Enfoque A (Modular)**:

* Clase Trackball separada
* Mayor cantidad de código
* Mejor mantenibilidad

**Enfoque B (Integrado)**:

* Todo en un archivo
* Código más compacto
* Menor modularidad

Se eligió el Enfoque A por su mejor arquitectura y reusabilidad.

**5.2 Dificultades Encontradas**

1. **Gestión de memoria**: Patrón singleton en BaseApp
2. **Compatibilidad**: Diferencias entre plataformas
3. **Proyección matemática**: Casos límite en la esfera

**5.3 Soluciones Aplicadas**

* Manejo cuidadoso del ciclo de vida de objetos
* Makefile con detección automática de sistema
* Normalización de vectores y validación de rangos

**6. CONCLUSIONES**

**6.1 Logros**

1. Se implementó exitosamente un visualizador OBJ con trackball
2. Se superaron los requisitos mínimos añadiendo pan y normalización
3. Se logró una arquitectura modular y extensible
4. La interacción es intuitiva y fluida

**6.2 Aprendizajes**

* Comprensión profunda de transformaciones 3D
* Importancia de la modularidad en proyectos gráficos
* Técnicas de interacción en aplicaciones 3D
* Manejo de matrices y cuaterniones implícitos

**6.3 Trabajo Futuro**

* Implementar iluminación y shading
* Soporte para texturas y materiales
* Optimización con VBOs
* Interfaz gráfica adicional

**7. REFERENCIAS**

1. Shoemake, K. (1992). "ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse"
2. OpenGL Programming Guide (Red Book)
3. Documentación de Eigen3
4. Notas del curso de Computación Gráfica

**ANEXO: INSTRUCCIONES DE USO**

**Compilación**

make clean

make

**Ejecución**

./ShowOBJ archivo.obj

**Controles**

* **Rotar**: Click izquierdo + arrastrar
* **Pan**: Click medio + arrastrar
* **Zoom**: Rueda del mouse
* **Reset**: R
* **Wireframe**: W
* **Salir**: ESC/Q