MANUAL DE PRÁCTICAS DE LABORATORIO

Técnicas de Graficación

Programa Académico	**Ingeniero en Software**
Plan de Estudios	**2021**
Fecha de elaboración	**30/06/2025**
Versión del Documento	**1**

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

Dra. Martha Patricia Patiño Fierro
Rectora

Mtra. Ana Lisette Valenzuela Molina
Encargada del Despacho de la Secretaría General Académica

Mtro. José Antonio Romero Montaño
Secretario General Administrativo

Lic. Jorge Omar Herrera Gutiérrez
Encargado de Despacho de Secretario General de Planeación

Tabla de contenido

INTRODUCCIÓN

Como parte de las herramientas esenciales para la formación académica de los estudiantes de la Universidad Estatal de Sonora, se definen manuales de práctica de laboratorio como elemento en el cual se define la estructura normativa de cada práctica y/o laboratorio, además de representar una guía para la aplicación práctica del conocimiento y el desarrollo de las competencias clave en su área de estudio. Su diseño se encuentra alineado con el modelo educativo institucional, el cual privilegia el aprendizaje basado en competencias, el aprendizaje activo y la conexión con escenarios reales.

Con el propósito de fortalecer la autonomía de los estudiantes, su pensamiento crítico y sus habilidades para la resolución de problemas, las prácticas de laboratorio integran estrategias didácticas como el aprendizaje basado en proyectos, el trabajo colaborativo, la experimentación guiada y el uso de tecnologías educativas. De esta manera, se promueve un proceso de enseñanza-aprendizaje dinámico, en el que los estudiantes no solo adquieren conocimientos teóricos, sino que también desarrollan habilidades prácticas y reflexivas para su desempeño profesional.

Señalar en este apartado brevemente los siguientes elementos según corresponda:

- Propósito del manual
- Justificación de su uso en el programa académico
- Competencias a desarrollar
    - **Competencias blandas:** Habilidades transversales que se refuerzan en las prácticas, como la comunicación, el trabajo en equipo, el uso de tecnologías, etc.
    - **Competencias disciplinares:** Conocimientos específicos del área del laboratorio, incluyendo fundamentos teóricos y habilidades técnicas.
    - **Competencias profesionales:** Aplicación de los conocimientos adquiridos en escenarios reales o simulados, en concordancia con el perfil de egreso del programa.

IDENTIFICACIÓN

Nombre de la Asignatura		Técnicas de graficación	
Clave	061CP055	Créditos	
Asignaturas Antecedentes		Plan de Estudios	2021, Ingeniero en Software

Área de Competencia	Competencia del curso
Desarrollar software y servicios de soporte técnico y redes, con la finalidad de solucionar problemas y agilizar procesos en la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional, a través del análisis de problemas, comunicación, liderazgo e innovación.	Crear soluciones innovadoras utilizando técnicas especializadas para el diseño, implementación y optimización de aplicaciones gráficas en 2D y 3D, generando interfaces hombre-máquina mediante técnicas de trazado, manipulación y visualización de elementos en pantalla, fomentando el desarrollo de habilidades analíticas, comunicativas, de liderazgo e innovación.

Carga Horaria de la asignatura

Horas Supervisadas			Horas Independientes	Total de Horas
Aula	Laboratorio	Plataforma		
3	1	1	2	5

Consignación del Documento

Unidad Académica	Unidad Académica Hermosillo
Fecha de elaboración	30/06/2025
Responsables del diseño	Martin Humberto Anduaga Pinto martin.anduaga@ues.mx
Validación	
Recepción	Coordinación de Procesos Educativos

MATRIZ DE CORRESPONDENCIA

Señalar la relación de cada práctica con las competencias del perfil de egreso del Ingeniero en Software.

PRÁCTICA	PERFIL DE EGRESO
Práctica 1: Exploración y selección de entornos de graficación	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo, atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.
Práctica 2: Configuración de un entorno de desarrollo con OpenGL	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo, atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.
Práctica 3: Transformaciones geométricas en OpenGL	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo, atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.
Práctica 4: Iluminación básica y materiales	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo,

	atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.
Práctica 5: Desarrollo de un mini juego interactivo 2D	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo, atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.
Práctica 6: Introducción práctica a Unity 6	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo, atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.
Práctica 7: Proyecto: Desarrollo de videojuego 2D en Unity	• Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo. • Desarrollar soporte y asistencia técnica para la prevención y corrección de problemas en los sistemas de cómputo, atendiendo los requerimientos y políticas de la organización, garantizando la optimización y el uso responsable de los recursos.

NORMAS DE SEGURIDAD Y BUENAS PRÁCTICAS

Reglamento general del laboratorio

Asistencia y puntualidad

- El alumno debe ingresar puntualmente de acuerdo con el horario establecido.
- La asistencia es registrada por el docente o encargado técnico del aula de cómputo.

Comportamiento responsable y respetuoso

- Mantener el orden, respeto y silencio dentro del laboratorio.
- No se permite introducir ni consumir alimentos ni bebidas en el aula de cómputo.
- Los teléfonos celulares, tables o dispositivos móviles deben estar en modo silencioso o vibrador y su uso solo se permitirá con fines académicos.

Cumplimiento académico

- Cada alumno debe seguirá las instrucciones del docente y cumplir con las actividades asignadas dentro del tiempo establecido.
- La práctica debe realizarse de manera individual o en equipo, según se indique en las indicaciones de la práctica.

Salud y ergonomía

- Usar el mobiliario de forma adecuada (sillas y mesas al trabajar frente al equipo).
- Evitar bloqueos de las salidas de emergencia, pasillos y escaleras.

Reglamento de uniforme

Ropa formal adecuada: sin gorras, sandalias, pantalones rotos, pantalones cortos.

Uso adecuado del equipo y materiales

Encendido y apagado correcto del equipo

- Encender y apagar los equipos únicamente bajo indicación del docente.
- Apagar los equipos antes de salir del laboratorio.

Cuidado del equipo de cómputo

- No modificar configuraciones predeterminadas del sistema operativo, software de red ni BIOS.
- No instalar programas sin autorización expresa del docente.
- Evitar conectar o desconectar cables de red o energía sin supervisión.

Manejo responsable de software y simuladores

- Utilizar únicamente el software permitido por la institución y con la licencia de uso adecuadamente instalada.
- Guardar la información en medios externos personales.

Reportes y fallas

- Reportar inmediatamente cualquier falla de hardware o software al docente o responsable técnico.
- No intentar reparar por cuenta propia ningún dispositivo electrónico.

Cuidado del entorno físico

- Mantener el área de trabajo limpia y libre de objetos ajenos a la práctica.
- Al finalizar la sesión, dejar el equipo y mobiliario en el estado y posición original.

Procedimientos en caso de emergencia

Evacuación: Al activarse una alarma de evacuación, seguir las rutas de salida establecidas en orden y sin correr.

Atención a instrucciones: Atender exclusivamente las indicaciones del docente, brigadistas o personal de Protección Civil.

Concentración en puntos de reunión: Reunirse con el grupo en el punto designado y esperar la revisión de lista.

No regresar sin autorización: No volver al laboratorio hasta que la autoridad correspondiente lo autorice.

RELACIÓN DE PRÁCTICAS DE LABORATORIO POR ELEMENTO DE COMPETENCIA

Elemento de Competencia al que pertenece la práctica	**Elementos de competencia 1**
	Reconocer los entornos de graficación bajo condiciones específicas de diseño y desarrollo, con la finalidad de establecer las bases del desarrollo de software orientado a gráficos, en el contexto de aplicaciones interactivas, fomentando la capacidad analítica y la resolución de problemas.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 1	Exploración y selección de entornos de graficación.	Reconocer y comparar distintos entornos de graficación, identificando sus características técnicas, ventajas, desventajas y aplicaciones comunes, para seleccionar el más adecuado según el tipo de proyecto de software gráfico.

Elemento de Competencia al que pertenece la práctica	**Elementos de competencia 2**
	Aplicar los conceptos y técnicas fundamentales de graficación por computadora utilizando OpenGL bajo diferentes escenarios prácticos, con la finalidad de desarrollar aplicaciones gráficas eficientes y visualmente atractivas, en el contexto de proyectos de software interactivo, fomentando la creatividad y la capacidad de resolución de problemas.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 2	Práctica 2: Configuración de un entorno de desarrollo con OpenGL	Configurar un proyecto básico en OpenGL utilizando una librería auxiliar (como GLFW o GLUT) y un lenguaje de programación como C++ o C#.
Práctica No. 3	Práctica 3: Transformaciones geométricas en OpenGL	Aplicar transformaciones básicas como traslación, rotación y escalamiento a objetos 2D.
Práctica No. 4	Práctica 4: Iluminación básica y materiales	Simular efectos de iluminación ambiental, difusa y especular en escenas 2D simuladas en OpenGL.
Práctica No. 5	Práctica 5: Desarrollo de mini juego interactivo 2D	Desarrollar un juego interactivo simple (tipo Pong, Breakout o Snake) utilizando OpenGL.

Elemento de Competencia al que pertenece la práctica	**Elementos de competencia 3**
	Aplicar los fundamentos de los motores gráficos, centrándose en la generación de un videojuego en 2D empleando Unity, fomentando el trabajo en equipo y la creatividad en el proceso de desarrollo.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 6	Práctica 6: Introducción práctica a Unity 6	Familiarizarse con la interfaz de Unity 6, la creación de escenas y el manejo de objetos en un entorno 2D.
Práctica No. 7	Práctica 7: Proyecto colaborativo: Desarrollo de videojuego 2D en Unity	Aplicar los fundamentos del desarrollo de videojuegos en 2D en Unity, mediante el diseño, construcción y presentación de un prototipo funcional en equipo.

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

PRÁCTICAS

NOMBRE DE LA PRÁCTICA	Práctica 1: Exploración y selección de entornos de graficación
COMPETENCIA DE LA PRÁCTICA	Reconocer y comparar distintos entornos de graficación, identificando sus características técnicas, ventajas, desventajas y aplicaciones comunes, para seleccionar el más adecuado según el tipo de proyecto de software gráfico.

FUNDAMENTO TÉORICO

En el desarrollo de software gráfico e interactivo, las figuras primitivas representan los elementos más básicos y esenciales con los que se construyen escenas visuales. Estas primitivas incluyen puntos, líneas, triángulos, rectángulos y círculos. Además, son la base para representar objetos más complejos en aplicaciones, videojuegos y simulaciones gráficas.

Los entornos de graficación como OpenGL, DirectX, y bibliotecas como SDL o GLUT, utilizan estas primitivas como instrucciones que se transforman en visualización sobre la pantalla. Comprender cómo funcionan y cómo se programan estas figuras permite a los desarrolladores tener un mayor control sobre lo que se visualiza, cómo se optimiza el rendimiento, y cómo se interpreta la interacción del usuario.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Software de desarrollo (Visual Studio, Code::Blocks, NetBeans, Eclipse, o entorno equivalente).
- Librería gráfica o API (GLUT, GLFW o una configuración básica de OpenGL).
- Lenguaje de programación sugerido: Java, C, C++ o C#.
- Acceso a internet para consulta de documentación.
- Editor de texto como Word o equivalente para elaborar el reporte.
- Equipo de cómputo con las características recomendadas:
    - Procesador Intel Core i5 de 8ª generación o superior (o equivalente AMD Ryzen 5)
    - Memoria RAM: mínimo 8 GB (ideal: 16 GB)
    - Tarjeta gráfica: NVIDIA GTX 1050 o superior (o integrada con soporte DirectX 11/12)
    - Almacenamiento: SSD con al menos 20 GB disponibles
    - Sistema Operativo: Windows 10 64-bits (actualizado) o macOS 11+

PROCEDIMIENTO O METODOLOGÍA

1. Formar equipos de 2 a 3 personas para trabajar colaborativamente. El trabajo en equipo es clave en el desarrollo profesional del ingeniero en software.
2. Explorar y elegir una librería gráfica con apoyo del docente, seleccionen una librería gráfica con soporte para primitivas básicas (como GLUT o GLFW). Investigar su sintaxis y cómo se implementan las funciones para dibujar.
3. Desarrollar una serie de ejercicios prácticos para dibujar figuras primitivas:
    - Puntos individuales

- Líneas rectas (horizontal, vertical, diagonal)
- Triángulo equilátero y triángulo rectángulo
- Cuadrado y rectángulo
- Un círculo (simulado con polígonos)

4. Compilar y probar el código.
5. Documentar el proceso de implementación para generar el reporte de práctica respectivo.
6. Elaborar un reporte de práctica que incluya:
   - Introducción al tema de figuras primitivas
   - Descripción breve de la librería usada
   - Código comentado de cada figura
   - Capturas de pantalla de los resultados
   - Conclusiones individuales del aprendizaje adquirido
7. Entrar el archivo en formato PDF con el reporte completo a la plataforma institucional antes de la fecha límite indicada por el docente.

RESULTADOS ESPERADOS

Al finalizar esta práctica, se espera que los estudiantes sean capaces de:

- Comprender el concepto de figuras primitivas y su papel dentro de la representación gráfica por computadora.
- Implementar con éxito el dibujo de formas geométricas básicas mediante primitivas gráficas utilizando una API o librería adecuada.
- Interpretar cómo la configuración de coordenadas, color y tipo de primitiva influye en la visualización de objetos.
- Fortalecer su lógica de programación al vincular estructuras geométricas con instrucciones gráficas en un entorno de desarrollo real.
- Comunicar de manera escrita y técnica el proceso de creación, solución de errores y presentación de resultados.

ANÁLISIS DE RESULTADOS

Durante el desarrollo de esta práctica, el estudiante deberá evaluar:

- La correcta visualización de cada figura dibujada en pantalla (forma, tamaño, ubicación y color).

- La relación entre el código fuente y el resultado visual (cómo las instrucciones gráficas se traducen en imagen).
- El manejo adecuado del sistema de coordenadas del entorno gráfico.
- Las dificultades técnicas encontradas al trabajar con la librería (instalación, configuración o errores de sintaxis) y cómo se solucionaron.

CONCLUSIONES Y REFLEXIONES

Al finalizar esta práctica, el alumno deberá reflexionar sobre:

- La importancia de dominar las figuras primitivas como punto de partida para desarrollar gráficos más complejos.
- Cómo el conocimiento adquirido en esta práctica se relaciona con el desarrollo de interfaces gráficas, videojuegos o simulaciones técnicas.
- El impacto que puede tener el uso adecuado de librerías gráficas en la optimización de recursos computacionales y la experiencia del usuario.
- El trabajo en equipo como una herramienta fundamental en el desarrollo de software gráfico, desde la distribución de tareas hasta la integración de código y pruebas.

ACTIVIDADES COMPLEMENTARIAS

Para reforzar los aprendizajes obtenidos en esta práctica, se sugiere realizar las siguientes actividades:

1. Lectura guiada:

   Revisar el capítulo introductorio del libro sobre gráficos por computadora, enfocado en primitivas gráficas y sistemas de coordenadas (recomendación: Fundamentals of Computer Graphics 3rd ed. - P. Shirley, S. Marschner. 2009).

2. Ejercicio individual opcional:

   Crear una escena compuesta que combine al menos 4 primitivas diferentes para simular un objeto o paisaje (por ejemplo, una casa, un árbol o una figura abstracta).

3. Foro de discusión en línea:

   Publicar en la plataforma institucional la experiencia personal sobre las dificultades encontradas al trabajar con la librería y cómo se solucionaron, promoviendo el intercambio de conocimientos entre compañeros.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	• Entrega de los algoritmos para generación de gráficas primitivas y su implementación. .
Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%).
Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: o Portada, o Introducción o Fundamentos teóricos o Objetivo de la práctica, hipótesis o Expectativa o planteamiento experimental o Materiales, equipamiento y/o reactivos o Procedimiento o metodología o Procesamiento de datos o Resultados o Análisis y discusión. o Conclusiones o Bibliografía o Anexos

NOMBRE DE LA PRÁCTICA	Práctica 2: Configuración de un entorno de desarrollo con OpenGL
COMPETENCIA DE LA PRÁCTICA	Configurar un proyecto básico en OpenGL utilizando una librería auxiliar (como GLFW o GLUT) y un lenguaje de programación como C++ o C#.

FUNDAMENTO TÉORICO

OpenGL (Open Graphics Library) es una API de propósito general diseñada para representar gráficos 2D y 3D, ampliamente utilizada en aplicaciones que requieren renderizado en tiempo real, como videojuegos, simulaciones, sistemas CAD y software interactivo. Su diseño multiplataforma y flexible la convierte en una herramienta estándar en la industria del desarrollo gráfico.

Para comenzar a trabajar con OpenGL, es necesario configurar un entorno de desarrollo que permita inicializar su contexto gráfico, crear una ventana de visualización y ejecutar el ciclo básico de renderizado. Esta configuración puede realizarse con diversos lenguajes de programación, siendo C y C# los más comunes en entornos educativos e industriales.

En el caso de C, se suele trabajar con bibliotecas como GLUT, GLEW o GLFW, mientras que en C# se pueden usar entornos como OpenTK. Para el desarrollo práctico, Visual Studio es una de las IDEs más completas para Windows, mientras que NetBeans se puede adaptar con plugins y librerías si se trabaja con C/C++ o bindings en C#.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Computadora con sistema operativo Windows 10 o superior.
- IDE instalado: Visual Studio 2022 (preferente) o NetBeans 12+
- Librerías OpenGL: GLUT, GLFW o GLEW para C; OpenTK para C#
- Lenguaje de programación: C o C#
- Acceso a internet para descargas y consulta de documentación
- Manual técnico de referencia (opcional)

PROCEDIMIENTO O METODOLOGÍA

1. Descargar e instalar Visual Studio 2022 Community con el paquete de desarrollo en C++ o C#.
2. Instalar la librería de OpenTK para C#
3. Crear un nuevo proyecto de consola (C o C#).
4. Integra las librerías de OpenGL según el lenguaje elegido.

5. Escribir un programa que genere una ventana vacía y muestre un color de fondo (por ejemplo, azul o gris oscuro).
    1. El programa debe incluir el ciclo de renderizado básico (main loop) y cerrar correctamente con un evento de tecla.
6. Compila y ejecuta el programa.
    1. Si no hay errores y se visualiza la ventana, la configuración ha sido exitosa.
7. Documentación del proceso
8. Incluir capturas de pantalla, código comentado y consideraciones técnicas.
9. Subir a la plataforma institucional el reporte en PDF con los apartados indicados, junto con el código fuente comprimido.

RESULTADOS ESPERADOS

- Instalación y configuración completa de un entorno de desarrollo con OpenGL funcional.
- Integración correcta de librerías gráficas en Visual Studio.
- Creación y ejecución de un programa básico que abra una ventana con un fondo renderizado.
- Código fuente funcional, claro y comentado.
- Reporte de práctica que explique el proceso de instalación, configuración y validación del entorno.

ANÁLISIS DE RESULTADOS

Durante el análisis se deberán considerar los siguientes aspectos:

- ¿El programa compila y corre sin errores? ¿Por qué?
- ¿Fue fácil o complicado vincular las librerías en el entorno elegido? ¿Qué dificultades surgieron?
- ¿Qué diferencias notaste entre trabajar en C y C#, si hiciste pruebas en ambos?
- ¿Qué herramientas o documentación te ayudaron más a entender la configuración?
- ¿Qué elementos del entorno son críticos para que funcione la ventana de renderizado?

CONCLUSIONES Y REFLEXIONES

Al finalizar esta práctica, el estudiante podrá reflexionar sobre:

- La importancia de contar con un entorno de desarrollo bien configurado para trabajar en proyectos de graficación.
- Las habilidades técnicas adquiridas al manipular el sistema de construcción del proyecto (enlaces, referencias, dependencias).
- Cómo este primer paso abre la puerta al desarrollo de interfaces visuales, simulaciones y videojuegos.
- La capacidad de enfrentarse a errores técnicos reales (compilación, incompatibilidades, rutas de librerías) como parte del proceso formativo.

ACTIVIDADES COMPLEMENTARIAS

1. Grabar un video corto (2-3 minutos) donde se explique el proceso de configuración.
2. Participar en un quiz en línea donde se evalúen conocimientos sobre librerías OpenGL, diferencias entre C y C#, estructura del programa y pasos de configuración.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	• Codificación y ejecución del programa en el entorno indicado.
Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%).

Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: ○ Portada, ○ Introducción ○ Fundamentos teóricos ○ Objetivo de la práctica, hipótesis ○ Expectativa o planteamiento experimental ○ Materiales, equipamiento y/o reactivos ○ Procedimiento o metodología ○ Procesamiento de datos ○ Resultados ○ Análisis y discusión. ○ Conclusiones ○ Bibliografía ○ Anexos

NOMBRE DE LA PRÁCTICA	Práctica 3: Transformaciones geométricas en OpenGL
COMPETENCIA DE LA PRÁCTICA	Aplicar transformaciones básicas como traslación, rotación y escalamiento a objetos 2D.

FUNDAMENTO TÉORICO

Las transformaciones geométricas son uno de los pilares fundamentales en el desarrollo de gráficos por computadora. Permiten modificar la posición, el tamaño y la orientación de los objetos dentro de una escena gráfica, y son ampliamente utilizadas en animaciones, interfaces gráficas, videojuegos y simulaciones interactivas.

En OpenGL, estas transformaciones se implementan manipulando matrices, ya sea directamente (en versiones modernas) o a través de funciones específicas (en versiones clásicas). Las tres transformaciones básicas son:

- Traslación: Mueve un objeto de un punto a otro en el espacio.
- Rotación: Gira un objeto respecto a un eje.
- Escalamiento: Aumenta o reduce el tamaño del objeto en uno o más ejes.

Estas transformaciones pueden aplicarse en combinación, y el orden en el que se ejecutan es clave para obtener el resultado esperado. Comprender cómo funcionan no solo permite controlar los objetos en la pantalla, sino que también entrena al estudiante en razonamiento espacial, organización de estructuras gráficas y solución de problemas visuales.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Entorno de desarrollo: Visual Studio (C/C# con OpenGL)
- Librerías gráficas: OpenTK (para C#)
- Computadora con tarjeta gráfica integrada o dedicada
- Acceso a internet y documentación técnica

PROCEDIMIENTO O METODOLOGÍA

1. Participar activamente en la explicación del docente sobre el uso de matrices de transformación y la implementación de traslación, rotación y escalamiento en OpenGL.
2. Desarrollar un programa base que dibuje un objeto simple (por ejemplo, un triángulo o un cuadrado) en la pantalla usando primitivas gráficas.
3. Implementar funciones para:
    - Trasladar el objeto a una nueva posición

- Rotarlo en sentido horario y antihorario
- Escalarlo en proporciones distintas

Aplicar las transformaciones de forma individual y combinada.

4. Ejecutar el programa y verificar visualmente los efectos de cada transformación. Probar con diferentes parámetros (ángulos, vectores, factores de escala).
5. Elaborar un reporte que incluya:
    - Breve explicación teórica de cada transformación
    - Código fuente con comentarios claros
    - Capturas de pantalla del objeto original y transformado
    - Reflexión personal del aprendizaje
6. Subir a la plataforma institucional el PDF del reporte y el código comprimido antes de la fecha límite establecida.

RESULTADOS ESPERADOS

- Comprensión teórica y práctica de las transformaciones geométricas básicas.
- Implementación funcional de traslación, rotación y escalamiento en objetos 2D mediante OpenGL.
- Visualización clara y correcta del objeto transformado en diferentes escenarios.
- Desarrollo de un código estructurado y documentado.
- Análisis crítico del orden de aplicación de las transformaciones y su impacto visual.

ANÁLISIS DE RESULTADOS

- ¿Qué efecto tiene cada transformación por separado?
- ¿Cómo cambia el resultado si se alteran el orden o los valores de las transformaciones?
- ¿Qué función cumple el sistema de coordenadas en los desplazamientos?

- ¿Qué dificultades se encontraron al implementar las funciones de transformación?
- ¿Cómo pueden aplicarse estas transformaciones en interfaces gráficas, juegos o simuladores?

CONCLUSIONES Y REFLEXIONES

Esta práctica permite al estudiante:

- Confirmar la importancia de las transformaciones geométricas en el desarrollo gráfico por computadora.
- Valorar el papel de la programación gráfica como un proceso lógico y visual al mismo tiempo.
- Identificar que las herramientas básicas son clave para cualquier proyecto que implique movimiento, interacción y manipulación de objetos visuales.
- Reconocer que la lógica matemática son habilidades esenciales en el perfil del ingeniero en software.

ACTIVIDADES COMPLEMENTARIAS

1. Crear una pequeña animación donde un objeto rote y se desplace simultáneamente, utilizando funciones de temporizador o entrada por teclado.
2. Organizar un debate sobre qué transformación es más crítica en el diseño de un videojuego y por qué. Argumentar desde el punto de vista técnico y visual.
3. Diseñar una figura más compleja (por ejemplo, una letra o símbolo) y aplicar transformaciones combinadas para generar efectos visuales dinámicos.
4. Investigar cómo se utilizan las transformaciones geométricas en motores como Unity o Unreal Engine.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	• Correcta implementación de las transformaciones

Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%)
Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: ○ Portada, ○ Introducción ○ Fundamentos teóricos ○ Objetivo de la práctica, hipótesis ○ Expectativa o planteamiento experimental ○ Materiales, equipamiento y/o reactivos ○ Procedimiento o metodología ○ Procesamiento de datos ○ Resultados ○ Análisis y discusión. ○ Conclusiones ○ Bibliografía ○ Anexos

NOMBRE DE LA PRÁCTICA	Práctica 4: Iluminación básica y materiales
COMPETENCIA DE LA PRÁCTICA	Simular efectos de iluminación ambiental, difusa y especular en escenas 2D simuladas en OpenGL.

FUNDAMENTO TÉORICO

La iluminación es uno de los componentes más importantes para lograr realismo y profundidad en una escena gráfica. A través de la simulación de cómo la luz interactúa con los objetos, es posible representar materiales con texturas, sombras y brillos que aportan mayor credibilidad visual a las aplicaciones gráficas o videojuegos.

En gráficos por computadora, los modelos de iluminación más comunes son:

1. Iluminación ambiental (ambient light):

   Representa la luz general del entorno, que se refleja de manera uniforme en todos los objetos.
2. Iluminación difusa (diffuse light):

   Simula la luz que incide directamente en una superficie y se dispersa. Depende del ángulo entre la luz y la normal de la superficie.
3. Iluminación especular (specular light):

   Representa los reflejos brillantes en materiales como metal o vidrio. Depende de la dirección del observador.

En OpenGL, estos modelos se configuran mediante propiedades de los materiales y fuentes de luz, usando funciones específicas o shaders (dependiendo de la versión). Esta práctica busca que los estudiantes experimenten estos conceptos desde la programación, desarrollando una escena básica con efectos de iluminación.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Visual Studio con C/C# y librerías OpenGL (OpenTK)
- Computadora con soporte para aceleración gráfica
- Documentación técnica de OpenGL
- Plantilla base de escena simple en OpenGL (provista por el docente)

PROCEDIMIENTO O METODOLOGÍA

1. Participar en la exposición del docente sobre los tipos de iluminación y su representación matemática.
2. Crear un programa básico en OpenGL que muestre un objeto tridimensional (como un cubo o esfera) en el centro de la escena.

3. Agregar iluminación ambiental básica a la escena y visualizar su efecto general sobre el objeto.
4. Configurar una fuente de luz direccional y aplicar iluminación difusa para resaltar el contorno y volumen del objeto.
5. Simular reflejos mediante iluminación especular. Ajustar los parámetros para observar cambios en el brillo.
6. Definir las propiedades del material del objeto (colores, coeficientes, shininess) para simular diferentes superficies (mate, brillante, metálico).
7. Elaborar un reporte donde se explique la teoría, la implementación del código, capturas de pantalla del resultado y reflexiones finales.
8. Subir a plataforma el reporte en PDF con el código fuente comprimido.

RESULTADOS ESPERADOS

- Comprensión clara de los tipos de iluminación y su efecto visual.
- Implementación funcional de iluminación ambiental, difusa y especular en una escena con OpenGL.
- Generación de código estructurado, comentado y funcional.
- Elaboración de un reporte técnico que relacione teoría con práctica.

ANÁLISIS DE RESULTADOS

Durante el análisis, se deberá reflexionar sobre:

- ¿Qué tipo de iluminación produjo un mayor efecto visual?
- ¿Cómo se comporta la luz cuando se cambian los coeficientes de los materiales?
- ¿Cuál fue la transformación visual más notoria al ajustar los parámetros especulares?
- ¿Qué desafíos técnicos se encontraron al trabajar con múltiples fuentes de luz?
- ¿Cómo influye la posición de la cámara en la percepción de los reflejos?

CONCLUSIONES Y REFLEXIONES

Esta práctica permitirá al estudiante:

- Comprender cómo la teoría matemática de la iluminación se traduce en efectos visuales reales.
- Valorar la importancia de los materiales y la luz en el realismo gráfico.
- Desarrollar habilidades de configuración de escenas, ajustando no solo el código sino también la percepción visual.

ACTIVIDADES COMPLEMENTARIAS
1. Crear dos escenas similares, una con iluminación activa y otra sin iluminación, para comparar sus efectos. Discutir en clase los resultados. 2. Resolver un breve cuestionario sobre las fórmulas de iluminación y su relación con los parámetros usados en OpenGL. 3. Publicar una reflexión en la plataforma institucional sobre cuál tipo de iluminación consideran más útil en el desarrollo de videojuegos y por qué. 4. Modificar la escena para simular diferentes materiales (plástico, metal, cerámica) ajustando los coeficientes de los materiales.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	• Documentación e implementación clara de los conceptos de iluminación.
Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%).

Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: ○ Portada, ○ Introducción ○ Fundamentos teóricos ○ Objetivo de la práctica, hipótesis ○ Expectativa o planteamiento experimental ○ Materiales, equipamiento y/o reactivos ○ Procedimiento o metodología ○ Procesamiento de datos ○ Resultados ○ Análisis y discusión. ○ Conclusiones ○ Bibliografía ○ Anexos

NOMBRE DE LA PRÁCTICA	Práctica 5: Desarrollo de un mini juego interactivo 2D
COMPETENCIA DE LA PRÁCTICA	Desarrollar un juego interactivo simple (tipo Pong, Breakout o Snake) utilizando OpenGL.

FUNDAMENTO TÉORICO

El desarrollo de videojuegos involucra una combinación de disciplinas técnicas y creativas. A nivel de programación gráfica, uno de los primeros pasos para entender cómo se comportan los objetos, las interacciones y las escenas es la creación de juegos simples en 2D.

¿Qué es un juego?

Un juego es una estructura interactiva que posee un objetivo claro, reglas que lo rigen, y un conjunto de controles que permiten al jugador interactuar con el entorno. Para ser considerado funcional, un juego debe incluir al menos:

1. Objetivo: La meta que el jugador debe alcanzar (por ejemplo: recolectar monedas, evitar obstáculos, llegar al final de un mapa).
2. Controles: Los métodos mediante los cuales el jugador interactúa con el juego, como teclas de dirección, botones de acción o gestos en pantalla.
3. Reglas: Normas que definen lo que el jugador puede o no hacer, así como las condiciones para ganar, perder o avanzar.

En esta práctica, se busca aplicar todos los conocimientos aprendidos en las prácticas anteriores (primitivas, transformaciones, iluminación) para crear una experiencia interactiva básica, pero funcional, que sirva como base para proyectos más complejos.

El motor gráfico utilizado es OpenGL, el cual permite controlar directamente el renderizado gráfico y la lógica del juego mediante programación estructurada, generalmente en C o C#, lo que favorece una comprensión profunda del funcionamiento interno de los videojuegos.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Lenguaje: C o C#
- Librerías: OpenGL (GLUT o OpenTK)
- Entorno: Visual Studio o NetBeans con extensiones adecuadas
- Computadora con soporte gráfico básico
- Plantilla base provista por el docente (si aplica)

PROCEDIMIENTO O METODOLOGÍA

1. Diseñar los componentes del juego
    - Definir el objetivo del juego (ejemplo: “Evitar que el personaje toque obstáculos durante 30 segundos”).

- Establecer las reglas básicas: condiciones para ganar, perder, o reiniciar el juego.
- Determinar los controles del jugador: teclas de flechas, barra espaciadora, etc.

2. Diseñar el escenario gráfico básico
   - Crear los sprites de los elementos del juego (jugador, enemigos, fondo, etc.) usando primitivas básicas de OpenGL.
   - Aplicar transformaciones para mover, escalar o rotar los objetos si es necesario.
3. Definir la lógica de interacción
   - Programar el movimiento del personaje controlado por el jugador.
   - Incluir detección de colisiones simples.
   - Programar condiciones de victoria o derrota.
4. Realizar las pruebas de funcionamiento
   - Validar que las reglas y controles se comporten como se diseñaron.
5. Documentar el proceso de elaboración
   - Crear un reporte de práctica en PDF que contenga:
     - Descripción del juego
     - Código fuente comentado
     - Capturas de pantalla
6. Subir a plataforma el archivo del proyecto comprimido (.zip) y el reporte de práctica en PDF.

RESULTADOS ESPERADOS

- Diseño funcional de un juego en 2D con lógica, controles y objetivo definidos.
- Implementación de una escena interactiva utilizando primitivas gráficas.
- Aplicación de técnicas previas: dibujo, transformaciones, y manejo de eventos.
- Desarrollo de pensamiento lógico aplicado a la programación de interacciones.

ANÁLISIS DE RESULTADOS

Durante el análisis de los resultados, el estudiante deberá reflexionar sobre:

- ¿Qué tan claras y funcionales fueron las reglas del juego?
- ¿Funcionaron adecuadamente los controles definidos? ¿Hubo necesidad de modificarlos?
- ¿Qué técnicas gráficas resultaron más útiles para representar los elementos del juego?
- ¿Cómo fue la experiencia de integrar diferentes aspectos del curso en un solo proyecto?
- ¿Qué dificultades técnicas y creativas se presentaron durante el desarrollo?

CONCLUSIONES Y REFLEXIONES

Esta práctica permite al estudiante:

- Integrar habilidades de programación gráfica en un proyecto creativo y funcional.
- Comprender la importancia de los conceptos base de diseño de juegos: objetivo, controles y reglas.
- Valorar el proceso de planeación, codificación, prueba y mejora continua en el desarrollo de software interactivo.
- Experimentar una introducción práctica al diseño de videojuegos como posible línea de especialización profesional.

ACTIVIDADES COMPLEMENTARIAS

1. Agregar un sistema de puntuación o niveles de dificultad al juego, y documentar cómo se implementó.
2. Investigar cómo se implementan juegos similares en otros motores gráficos (como Unity) y redactar un resumen sobre las diferencias y similitudes.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	• Presentación de la codificación y ejecución del juego esperado.

Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%).
Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: ○ Portada, ○ Introducción ○ Fundamentos teóricos ○ Objetivo de la práctica, hipótesis ○ Expectativa o planteamiento experimental ○ Materiales, equipamiento y/o reactivos ○ Procedimiento o metodología ○ Procesamiento de datos ○ Resultados ○ Análisis y discusión. ○ Conclusiones ○ Bibliografía ○ Anexos

NOMBRE DE LA PRÁCTICA	Práctica 6: Introducción práctica a Unity 6
COMPETENCIA DE LA PRÁCTICA	Familiarizarse con la interfaz de Unity 6, la creación de escenas y el manejo de objetos en un entorno 2D.

FUNDAMENTO TÉORICO

¿Qué es Unity Hub y Unity 6 (LTS)?

Unity Hub es una herramienta de administración centralizada para trabajar con diferentes versiones del motor gráfico Unity. Permite crear, abrir y gestionar proyectos, así como instalar diferentes versiones de Unity de forma organizada.

Unity 6 (LTS), lanzado con soporte de Long-Term Support, es la versión más reciente y estable del motor Unity para proyectos profesionales. Esta versión está enfocada en la estabilidad, mejoras de rendimiento, herramientas gráficas avanzadas, y es ideal para proyectos educativos y comerciales que requieren soporte prolongado.

¿Qué son los Sprites?

Los sprites son imágenes 2D que representan personajes, objetos o elementos visuales dentro de un videojuego. En Unity, los sprites se importan como archivos de imagen (.PNG, .JPG) y se manipulan a través de componentes como el Sprite Renderer, que permite posicionarlos, escalarlos, rotarlos y animarlos.

¿Qué es una Escena?

Una escena en Unity es el entorno o espacio virtual donde ocurren los eventos del juego. Es como un nivel o pantalla donde se colocan objetos, luces, cámaras y scripts. Cada escena puede representar un menú, un nivel del juego o una sección diferente de la aplicación.

Dominar estos elementos básicos es el primer paso para trabajar con Unity de forma efectiva. Esta práctica tiene como finalidad que el estudiante se familiarice con la interfaz, cree una escena básica 2D y comprenda cómo usar sprites para diseñar un prototipo simple e interactivo.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Computadora con al menos:
  - Procesador Intel i5 o AMD Ryzen equivalente
  - 8 GB de RAM (recomendado 16 GB)
  - 10 GB de espacio libre en disco
  - Tarjeta gráfica con soporte para DirectX 11 o superior
- Conexión a internet
- Unity Hub (última versión)

- Unity 6 LTS instalado desde Unity Hub
- Cuenta en Unity ID
- Editor Visual Studio Community (instalado con Unity)
- Recursos gráficos (sprites) proporcionados por el docente o descargados desde el Asset Store

PROCEDIMIENTO O METODOLOGÍA

1. Instalar las herramientas requeridas
    - Instalar Unity Hub desde el sitio oficial: https://unity.com
    - Crear una cuenta Unity ID (si no se tiene)
    - Desde Unity Hub, instalar Unity 6 (LTS) junto con el módulo de desarrollo 2D y Visual Studio
2. Crear el proyecto
    - Crear un nuevo proyecto en 2D con nombre y ubicación definidos
    - Identificar las diferentes herramientas de la interfaz de Unity: jerarquía (Hierarchy), escena (Scene), vista del juego (Game), inspector (Inspector) y proyecto (Project)
3. Importar los sprites necesarios
    - Importar al proyecto imágenes en formato PNG como sprites
    - Organizar los recursos en carpetas dentro del panel Project
    - Añadir los sprites a la escena arrastrándolos desde el panel al área de trabajo
4. Construir la escena
    - Posicionar los elementos gráficos: fondo, personajes, objetos interactivos
    - Aplicar componentes como Rigidbody2D y BoxCollider2D
    - Crear una animación sencilla (por ejemplo, cambio de posición al presionar una tecla)
5. Configurar los controles simples
    - Programar scripts sencillos en C# para mover un personaje usando el teclado
    - Probar el juego desde la vista Game
6. Subir el reporte en PDF a plataforma, que incluya:
        - Capturas de pantalla del proceso y de la escena final

- Código del proyecto
- Archivo comprimido del proyecto (.zip)

RESULTADOS ESPERADOS

- Instalación correcta y uso básico de Unity Hub y Unity 6 LTS
- Creación funcional de una escena en 2D con sprites y controles simples
- Comprensión y uso de la interfaz de Unity
- Uso de componentes como Rigidbody2D y Collider

ANÁLISIS DE RESULTADOS

En esta parte se deberá reflexionar sobre:

- ¿Qué tan intuitiva fue la interfaz de Unity para comenzar un proyecto?
- ¿Fue sencillo importar y manipular los sprites en la escena?
- ¿Qué dificultades se presentaron al programar el movimiento del personaje?
- ¿Se logró representar visualmente lo planeado desde el diseño inicial?
- ¿Qué se podría mejorar para que la escena parezca más profesional o interactiva?

CONCLUSIONES Y REFLEXIONES

Esta práctica permite al estudiante:

- Introducirse al desarrollo de videojuegos con un enfoque visual e interactivo
- Desarrollar confianza en el uso del entorno profesional de Unity
- Comprender los conceptos clave: sprites, escenas, jerarquía y componentes
- Abrir paso hacia proyectos más avanzados, como el desarrollo completo de un juego en Unity

ACTIVIDADES COMPLEMENTARIAS

1. Descargar y utilizar al menos un recurso gratuito desde el Unity Asset Store, integrarlo a tu escena y explicar su función.

2. Crear una segunda escena con un fondo diferente y enlázala desde un botón o tecla. Esto permitirá practicar la navegación entre escenas.
3. Grabar un video corto mostrando tu prototipo en ejecución y explícalo como si fueras a presentarlo a un cliente.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	• Revisión del código e implementación de las instrucciones en Unity
Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%).
Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: ○ Portada, ○ Introducción ○ Fundamentos teóricos ○ Objetivo de la práctica, hipótesis ○ Expectativa o planteamiento experimental ○ Materiales, equipamiento y/o reactivos ○ Procedimiento o metodología ○ Procesamiento de datos ○ Resultados ○ Análisis y discusión. ○ Conclusiones ○ Bibliografía ○ Anexos

NOMBRE DE LA PRÁCTICA	Práctica 7: Proyecto: Desarrollo de videojuego 2D en Unity
COMPETENCIA DE LA PRÁCTICA	Aplicar los fundamentos del desarrollo de videojuegos en 2D en Unity, mediante el diseño, construcción y presentación de un prototipo funcional en equipo.

FUNDAMENTO TÉORICO

Desarrollar un videojuego completo implica planificación estratégica, organización del equipo, y aplicación técnica. No basta con tener habilidades de programación, también se requiere visión creativa, diseño estructurado y comunicación efectiva.

Planificación del juego

1. Toda experiencia de juego tiene un hilo conductor. Aunque sea un juego sencillo, tener una narrativa mínima ayuda a darle coherencia y motivación al jugador (por ejemplo: “Un robot debe escapar de una fábrica evitando enemigos para regresar a su planeta”).
2. Definir el look visual del juego: ¿será pixel art, caricaturesco, minimalista, realista? Esto influye en la elección de los recursos, los colores y la ambientación.
3. Un juego se compone de niveles o etapas que deben aumentar la dificultad progresivamente. Cada nivel puede tener un diseño distinto, enemigos nuevos o retos adicionales.
4. Elementos que desafían al jugador. Pueden tener comportamientos programados (seguir al jugador, disparar, bloquear el paso). Es fundamental definir cuántos tipos habrá y cómo interactúan.
5. Las reglas del juego: cómo se mueve el personaje, qué puede hacer (saltar, disparar, esquivar), cómo se pierde o se gana, y qué interacciones hay con el entorno.
6. El gameplay es la experiencia real del jugador: cómo se siente jugar, cuán fluido es, si es entretenido o frustrante. Cada parte del juego debe estar dividida en escenas en Unity: por ejemplo, una para el menú, otra para el nivel 1, otra para “Game Over”, etc.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Unity 6 LTS
- Visual Studio 2022 Community
- Recursos gráficos y de sonido (propios o libres de derechos)
- Software de edición de imagen (opcional): GIMP, Piskel, Aseprite)
- Computadora con mínimo 8 GB RAM, procesador i5 o superior

- Plataforma institucional para entrega del proyecto y reporte

PROCEDIMIENTO O METODOLOGÍA

1. Formar el equipo de trabajo
    - Máximo 4 integrantes
    - Asignar roles (programador, diseñador, líder de proyecto, etc.)
2. Definir la estructura de proyecto de juego a implementar
    - Documentar con:
        - Nombre del juego y sinopsis
        - Objetivo del juego
        - Estilo gráfico
        - Número y tipo de escenas
        - Descripción de enemigos, obstáculos y reglas
        - Diseño de niveles (mínimo 2 niveles funcionales)
3. Desarrollar del videojuego
    - Crear escenas en Unity (menú, niveles, final, Game Over)
    - Programar los movimientos del jugador, interacciones y colisiones
    - Aplicar animaciones, sonidos y HUD (marcador, vidas, tiempo, etc.)
    - Realizar las pruebas, ajustes y depuración necesarias.
4. Presentar
    - Realizar una demo del juego implementado
    - Entregar el proyecto Unity (.zip), reporte técnico en PDF y presentación (PowerPoint o PDF)

RESULTADOS ESPERADOS

- Videojuego funcional con al menos dos niveles, menú y final
- Aplicación de conceptos de sprites, escenas, animaciones y mecánicas

- Uso correcto del motor Unity 6 y buenas prácticas de desarrollo

ANÁLISIS DE RESULTADOS

- ¿Qué tan funcional y divertido es el juego desde el punto de vista del jugador?
- ¿Qué aspectos técnicos fueron más retadores?
- ¿Qué decisiones de diseño afectaron la jugabilidad positiva o negativamente?
- ¿Lograron cumplir con el diseño inicial o hubo ajustes significativos?

CONCLUSIONES Y REFLEXIONES

- Experimentar un proyecto completo desde su planeación hasta su ejecución
- Enfrentar retos reales como en un entorno profesional de desarrollo
- Entender la importancia de planificar antes de programar
- Valorar la experiencia del usuario como un factor clave en el desarrollo de videojuegos
- Reconocer el poder de Unity como motor gráfico profesional para videojuegos

ACTIVIDADES COMPLEMENTARIAS

1. Organizar una sesión de exposición donde todos los equipos jueguen y evalúen los proyectos de los demás. Se puede premiar categorías como: Mejor Gameplay, Mejor Diseño Visual, Juego más Innovador.
2. Aplicar una encuesta breve a quienes prueben el juego (usabilidad, diversión, dificultad), e incluir sus resultados en el reporte final.
3. Como actividad extendida, se puede subir el juego a una plataforma gratuita como itch.io y compartirlo con otros compañeros o familiares.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	• Presentación del Demo del videojuego implementado.
Rúbricas o listas de cotejo para valorar desempeño	• Rúbrica: Reporte de práctica de laboratorio (100%).
Formatos de reporte de prácticas	• Documento institucional editable de “reporte de prácticas” que contiene: ○ Portada, ○ Introducción ○ Fundamentos teóricos ○ Objetivo de la práctica, hipótesis ○ Expectativa o planteamiento experimental ○ Materiales, equipamiento y/o reactivos ○ Procedimiento o metodología ○ Procesamiento de datos ○ Resultados ○ Análisis y discusión. ○ Conclusiones ○ Bibliografía ○ Anexos

FUENTES DE INFORMACIÓN

Bosque Orero, J. L. (2007). Introducción a OpenGL: (ed.). Madrid, Spain: Dykinson. Recuperado de https://elibro.net/es/lc/ues/titulos/35680

McReynolds, T., &Blythe, D. (2005). Advanced graphics programming using OpenGL. Morgan Kaufmann

Publishers Inc. https://www.sciencedirect.com/book/9781558606593/advanced-graphics-programming-using OpenGL

Chen, J. X., &Chen, C. (2008). Foundations of 3D graphics programming: Using JOGL and Java3D (2nd ed.). Springer-Verlag London. https://doi.org/10.1007/978-1-84800-284-5

Iznaga Benítez, A. M. (2006). Fundamentos de la gráfica por computadora: (ed.). La Habana, Cuba: Editorial

HEARN, Donald &M. Pauline Baker, (1995). Gráficas por computadora 2ª edición, Ed. Prentice Hall Hispanoamericana. México.

FOLEY, James &Andries Van Dam, (1996). Introducción a la graficación por computador, Ed. Addison Wesley Iberoamericana.

DEMEL, John T. &Michael J. Miller, Gráficas por computadora., Ed. McGraw Hill

Unity Technologies. (2024). *Unity Manual: Unity 6 LTS Documentation*. Recuperado de https://docs.unity3d.com/Manual/index.html

Unity Technologies. (2024). *Unity Learn Platform*. Recuperado de https://learn.unity.com/ y https://unity.com/es

Universidad Estatal de Sonora. (2022). Secuencia didáctica de Técnicas de Graficación, disponible en https://www.ues.mx/archivos/secuencias/022_21_061CP055.pdf

NORMAS TÉCNICAS APLICABLES

Aunque no existe una norma ISO específica para Unity, se pueden considerar las normas y marcos de referencia internacionales aplicables al desarrollo de software:

Norma / Estándar	**Aplicación**
ISO/IEC 25010	Calidad del software (funcionalidad, mantenibilidad, usabilidad, etc.)
IEEE 830	Requisitos de software
SCRUM / Agile	Gestión de proyectos de desarrollo de videojuegos
ISO/IEC 9126 *(antecesor del 25010)*	Atributos de calidad en productos software
GDPR / Leyes de privacidad	Si el juego recopila datos del jugador


UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu