Documentación Proyecto Arquitectura

Documentación del Módulo Caché (src/Cache):

El sistema de caché implementado en nuestro proyecto de simulación 3D sigue un diseño set-associative que busca optimizar el acceso a los datos de geometría y texturas durante el proceso de renderizado. La arquitectura está compuesta por tres elementos fundamentales que trabajan en conjunto para reducir los tiempos de acceso a memoria.

En el núcleo del sistema encontramos las líneas de caché, que representan la unidad básica de almacenamiento. Cada línea contiene varios campos críticos: una etiqueta que identifica el bloque de memoria que almacena, un indicador de validez que muestra si los datos actuales son utilizables, una marca de modificación para implementar políticas write-back en futuras versiones, y un contador de accesos que soporta el algoritmo de reemplazo LRU. Estas líneas se organizan en conjuntos, cuya cantidad depende directamente de los parámetros de configuración de la caché.

La inicialización del sistema requiere tres parámetros clave: el tamaño total de la caché, el tamaño de cada bloque y el nivel de asociatividad. Estos valores determinan la estructura interna, calculando automáticamente el número de conjuntos disponibles. Por ejemplo, una caché de 32KB con bloques de 64B y 4 vías de asociatividad generará 128 conjuntos distintos. Esta configuración permite un balance entre flexibilidad y eficiencia en el acceso a los datos.

Cuando el simulador necesita acceder a una dirección de memoria específica, el sistema de caché ejecuta una secuencia de operaciones bien definidas. Primero determina el conjunto correspondiente mediante operaciones aritméticas sobre la dirección solicitada. Luego busca en todas las vías del conjunto para encontrar una coincidencia con la etiqueta calculada. Si encuentra el dato, registra un acierto y actualiza los contadores LRU. En caso contrario, produce un fallo de caché y procede a cargar el bloque desde memoria principal, reemplazando la línea menos recientemente utilizada según el algoritmo implementado.

Para mejorar aún más el rendimiento, el sistema incluye un mecanismo de prefetching que anticipa la carga de bloques adyacentes al accedido recientemente. Esta técnica aprovecha el principio de localidad espacial, cargando datos que probablemente serán necesitados en operaciones futuras, siempre que pertenezcan al mismo conjunto de caché.

El subsistema ofrece herramientas de monitoreo y depuración que permiten verificar su funcionamiento. Es posible consultar estadísticas globales y por conjunto sobre tasas de acierto, así como obtener un volcado completo del estado actual de la caché,

incluyendo todas las etiquetas almacenadas y sus contadores de acceso. Esta información resulta valiosa para optimizar los parámetros del sistema según las cargas de trabajo específicas del simulador 3D.

Actualmente, el sistema presenta algunas limitaciones que marcan el camino para futuras mejoras. Entre ellas destacan la implementación de políticas de escritura más sofisticadas y la inclusión de algoritmos de reemplazo alternativos. También se planea optimizar el diseño para entornos multinúcleo, donde la gestión de cachés compartidas puede ofrecer beneficios adicionales de rendimiento.

La integración de este subsistema con el simulador 3D busca principalmente acelerar las operaciones de transformación de vértices y el acceso a texturas. Su configuración flexible permite adaptarlo a diferentes arquitecturas hardware, ajustando parámetros como el tamaño de bloque para que coincida con las características específicas del procesador donde se ejecute la aplicación.

Documentación del Módulo de Vértices (Common/Vertice.hpp):

Este archivo define la estructura básica para representar vértices en el espacio 3D dentro de nuestro motor de simulación. Aunque minimalista, cumple un rol fundamental en el pipeline de renderizado.

La estructura Vertice encapsula las tres coordenadas espaciales necesarias para definir un punto en el espacio tridimensional:

x: Coordenada en el eje horizontal (derecha/izquierda)

y: Coordenada en el eje vertical (arriba/abajo)

z: Coordenada de profundidad (adelante/atrás)

Cada componente se almacena como un valor de punto flotante de precisión simple (float), lo que proporciona un balance adecuado entre precisión y consumo de memoria para la mayoría de casos de uso en nuestro simulador.

El diseño deliberadamente simple permite:

Interoperabilidad fácil con APIs gráficas como OpenGL

Bajo overhead al procesar grandes mallas de polígonos

Flexibilidad para ser extendido cuando sea necesario (añadiendo normales, coordenadas UV, etc.)

La protección con #ifndef/#define garantiza que la definición no cause conflictos por inclusiones múltiples, una práctica esencial en proyectos con múltiples dependencias.

Este componente es utilizado principalmente por:

El cargador de modelos .obj

Los sistemas de transformación geométrica

El renderizador para pasar datos a la GPU

Documentación del Módulo Generador de Datos 3D (src/DataGenerators):

El subsistema Generador de Datos 3D proporciona herramientas fundamentales para la creación y análisis de estructuras geométricas básicas dentro del simulador. Ubicado en el directorio src/DataGenerators, este módulo cumple dos funciones principales: la generación de modelos tridimensionales primitivos y la creación de patrones de acceso a memoria optimizados.

En la parte de generación de modelos, el sistema ofrece métodos especializados para crear formas geométricas elementales. La función para generar cubos produce los ocho vértices necesarios para formar esta figura, posicionados simétricamente alrededor del origen del espacio tridimensional. Cada vértice se define con coordenadas precisas que garantizan las proporciones correctas según el tamaño especificado. De manera similar, la generación de pirámides crea una base cuadrada complementada con un vértice superior, permitiendo variar tanto la extensión de la base como la altura del ápice.

Para facilitar el trabajo de desarrollo y depuración, el módulo incluye capacidades de visualización básica. Estas permiten imprimir en consola tanto las coordenadas exactas de cada vértice como representaciones esquemáticas en formato ASCII de las figuras generadas. Esta funcionalidad resulta particularmente valiosa para verificar rápidamente la correcta formación de las estructuras sin necesidad de emplear visualizadores gráficos externos.

La segunda faceta importante del módulo se centra en la generación de patrones de acceso a memoria. Estos patrones están específicamente diseñados para evaluar el comportamiento del subsistema de caché, combinando accesos secuenciales que aprovechan la localidad espacial con un componente reducido de aleatoriedad. Esta combinación permite simular condiciones de uso más realistas que las obtenidas con patrones puramente secuenciales o completamente aleatorios.

La importancia de este generador radica en su capacidad para crear condiciones controladas que permiten analizar el rendimiento del sistema de memoria bajo diferentes configuraciones. Los patrones producidos ayudan a identificar cómo variables como el tamaño de bloque o el grado de asociatividad afectan la tasa de aciertos, proporcionando datos valiosos para la optimización del sistema completo.

Cada componente del módulo ha sido diseñado pensando en la claridad y la facilidad de uso. Los nombres descriptivos de los vértices, las representaciones visuales sencillas y los parámetros configurables hacen que esta herramienta sea accesible tanto para pruebas rápidas como para análisis más profundos del comportamiento del sistema. La arquitectura modular permite además extender las capacidades actuales mediante la incorporación de nuevas formas geométricas o patrones de acceso más sofisticados según las necesidades del proyecto.

Documentación del Módulo Gráfico (src/Graficos):

El directorio Graficos contiene los componentes centrales para la visualización 3D y la interacción de usuario en nuestro simulador. Este módulo se encarga de gestionar todo lo relacionado con la representación gráfica, el manejo de la cámara virtual y la interfaz de usuario.

El sistema se compone de varios elementos clave. El CameraController maneja el movimiento y rotación de la cámara virtual, implementando controles tipo FPS con movimiento suavizado y límites de rotación. Trabaja en conjunto con el InputHandler que procesa las entradas de teclado y mouse, convirtiéndolas en acciones para la cámara y la aplicación.

El Renderer es el núcleo del subsistema gráfico, encargado de transformar los modelos 3D en imágenes 2D mediante proyección perspectiva. Implementa técnicas como el recorte de geometría contra el plano cercano, ordenamiento por profundidad y diferentes modos de renderizado (sólido, alámbrico o mixto). Utiliza optimizaciones como el pre-cálculo de normales y la reutilización de buffers.

El ModelViewer actúa como coordinador principal, integrando todos los componentes para mostrar los modelos 3D. Gestiona el bucle de renderizado, sincroniza las actualizaciones y maneja los eventos del sistema. Implementa lógica para visualización tanto en modo gráfico como en consola.

La interfaz de usuario es manejada por UIHandler, que muestra información relevante como estadísticas de rendimiento, posición de la cámara y controles disponibles. Utiliza texto renderizado con fuentes TrueType y se integra con el sistema de eventos de SFML.

El sistema sigue principios de diseño orientado a objetos, con responsabilidades claramente separadas. Los componentes están diseñados para ser extensibles, permitiendo fácil incorporación de nuevos modos de visualización o técnicas de renderizado. El código hace uso intensivo de SFML para el renderizado 2D acelerado por hardware, mientras que las transformaciones 3D se implementan manualmente para mantener el control preciso sobre el pipeline gráfico.

La arquitectura permite visualizar diferentes modelos 3D (como cubos y pirámides) con iluminación básica, mostrando simultáneamente información técnica sobre el estado de la simulación. Todo el subsistema está diseñado para trabajar con tasas de cuadros estables, incluso en hardware modesto, gracias a técnicas de optimización como el cálculo diferido de transformaciones y el uso eficiente de los recursos gráficos.

Documentación del Módulo Principal (../principal.cpp):

El archivo principal.cpp constituye el núcleo central de la aplicación, funcionando como punto de entrada principal y coordinador general del sistema. Este módulo integra todos los componentes del programa gestionando el flujo de ejecución, la interacción con el usuario y la coordinación entre los distintos subsistemas.

La implementación comienza con la inclusión de diversas bibliotecas esenciales que proporcionan funcionalidades clave. Se incluyen componentes para manejo de tiempo, operaciones de entrada/salida, gestión de excepciones y verificación de archivos. Adicionalmente, se incorporan las bibliotecas gráficas SFML para el renderizado 2D y manejo de ventanas, junto con los módulos personalizados del proyecto para caché, generación de datos y visualización 3D.

El código establece varias funciones auxiliares que mejoran la experiencia de usuario y la organización del programa. La función limpiarTerminal proporciona una forma portable de borrar la pantalla de consola, implementando lógica específica para diferentes sistemas operativos. La función esperarEnter simplifica la creación de puntos de pausa en la interfaz de consola, mejorando la legibilidad de la salida.

Para la interacción con el usuario, se implementan dos menús principales. mostrarMenuPrincipal presenta las opciones globales del sistema, permitiendo acceder a las estadísticas de caché, al modelado 3D o salir de la aplicación. mostrarMenuModelado ofrece opciones específicas para visualización de figuras geométricas tridimensionales. Ambos menús incluyen validación robusta de entrada para garantizar una experiencia de usuario fluida.

La función mostrarEstadisticasCache se encarga de presentar información detallada sobre el rendimiento del sistema de caché, incluyendo tasas de acierto y tiempo de

simulación. cargarFuente proporciona un mecanismo para cargar tipografías desde archivo, con manejo adecuado de casos de error.

En la función main se coordina toda la ejecución del programa. Se inicializa el sistema de caché con parámetros predefinidos y se genera una secuencia optimizada de direcciones de memoria. El código mide con precisión el tiempo de ejecución de la simulación de caché utilizando la biblioteca chrono.

La aplicación detecta automáticamente la disponibilidad de soporte gráfico y adapta su comportamiento en consecuencia. El bucle principal gestiona la navegación entre menús y la ejecución de las diferentes funcionalidades. Para la visualización 3D, se crea una ventana SFML específica y se invoca al visualizador correspondiente según la figura seleccionada por el usuario (cubo o pirámide).

El sistema incluye un manejo robusto de errores mediante bloques try-catch que capturan y presentan adecuadamente cualquier excepción no controlada. La aplicación finaliza de manera controlada, devolviendo códigos de salida apropiados que indican éxito o fallo en la ejecución.

El diseño modular del código facilita el mantenimiento y la extensión futura, permitiendo la incorporación de nuevas funcionalidades con mínimas modificaciones a la estructura existente. La separación clara de responsabilidades entre las diferentes funciones y componentes asegura un alto nivel de cohesión y bajo acoplamiento.