Tema 2. Arquitecturas orientadas a las RV

TÉCNICAS Y DISPOSITIVOS DE REALIDAD VIRTUAL

MASTER EN INFORMÁTICA GRÁFICA,
JUEGOS Y REALIDAD VIRTUAL

Pablo Toharia
pablo.toharia@urjc.es

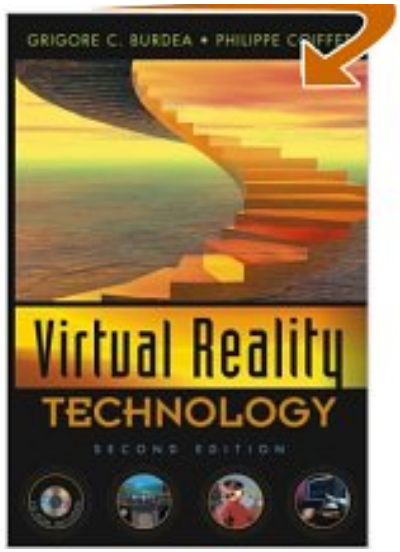
Marcos García
Miguel Ángel A. Otaduy

Universidad
Rey Juan Carlos

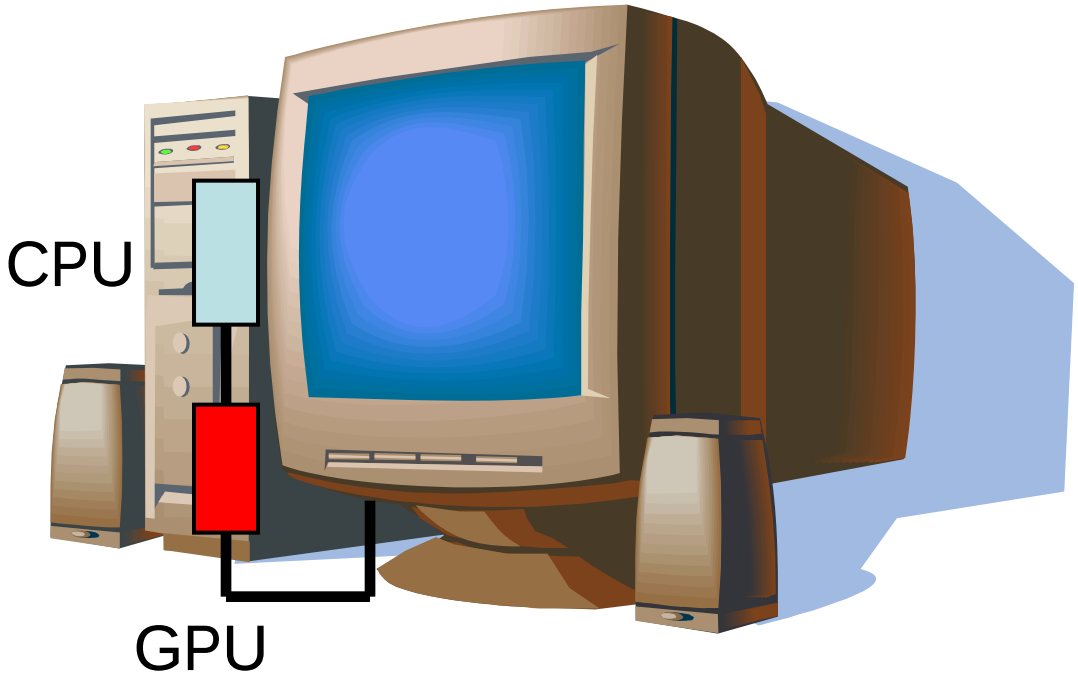
Bibliografía

“Virtual Reality Technology” Ed. Wiley-Interscience (Second Edition). Grigore C. Burdea & Philippe Coiffet.

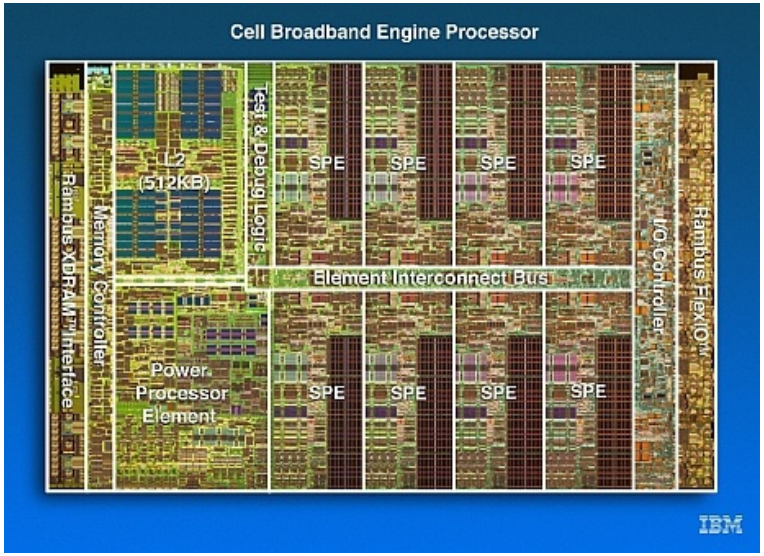
(sólo arquitecturas para realidad virtual, obsoleto en cuanto a arquitecturas gráficas)



Una arquitectura simple



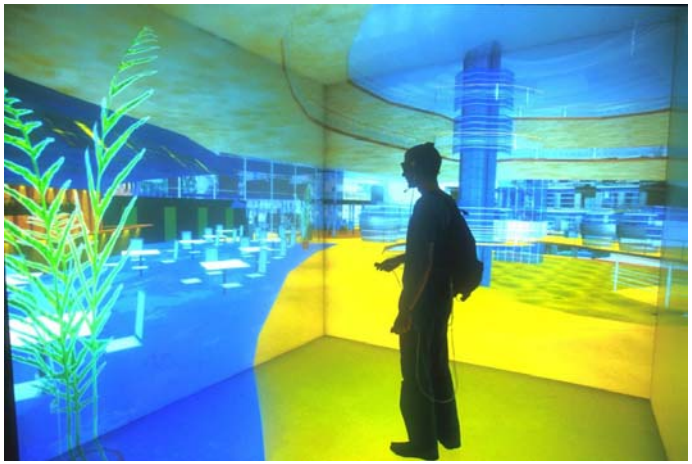
Una arquitectura ¿simple?



CPU multi-core
(IBM cell, Intel Larrabee...)

GPU programable
(NVIDIA Tesla S870)

Múltiples Periféricos



Cueva (CAVE) para realidad virtual

Mosaico de pantallas

Sistema gráfico + háptico

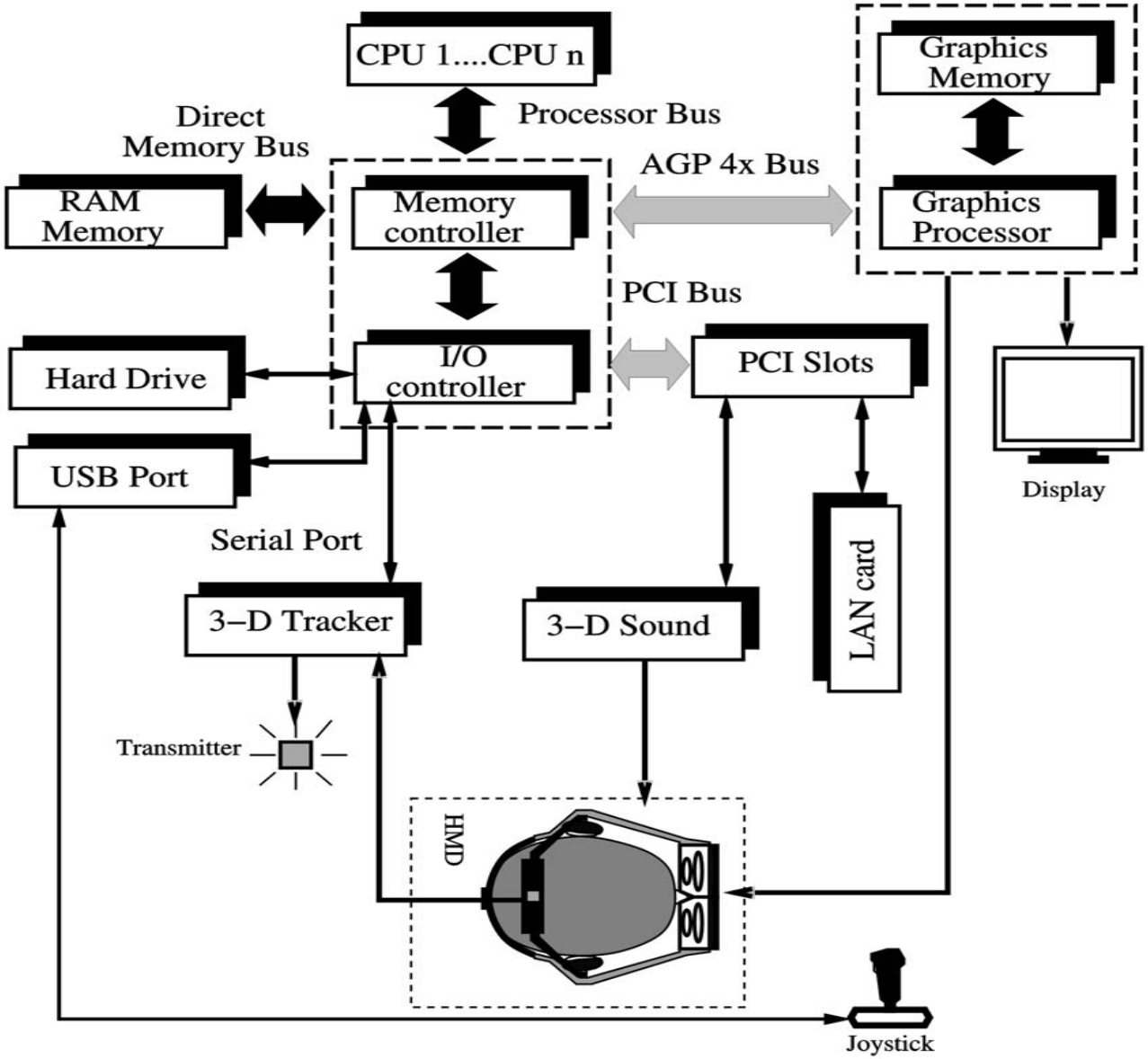
Universidad Rey Juan Carlos

Globalización

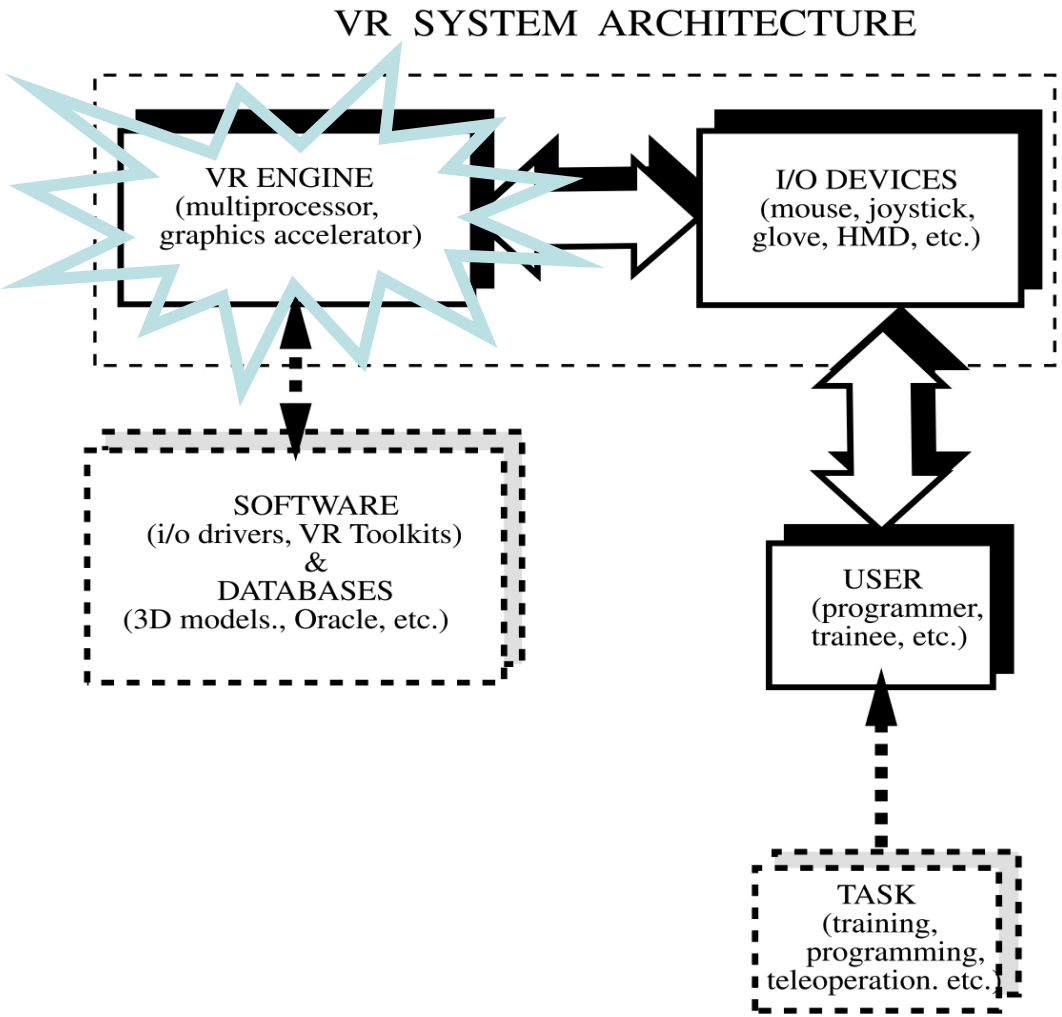
Juegos en red

Second Life

Ejemplo



Introducción



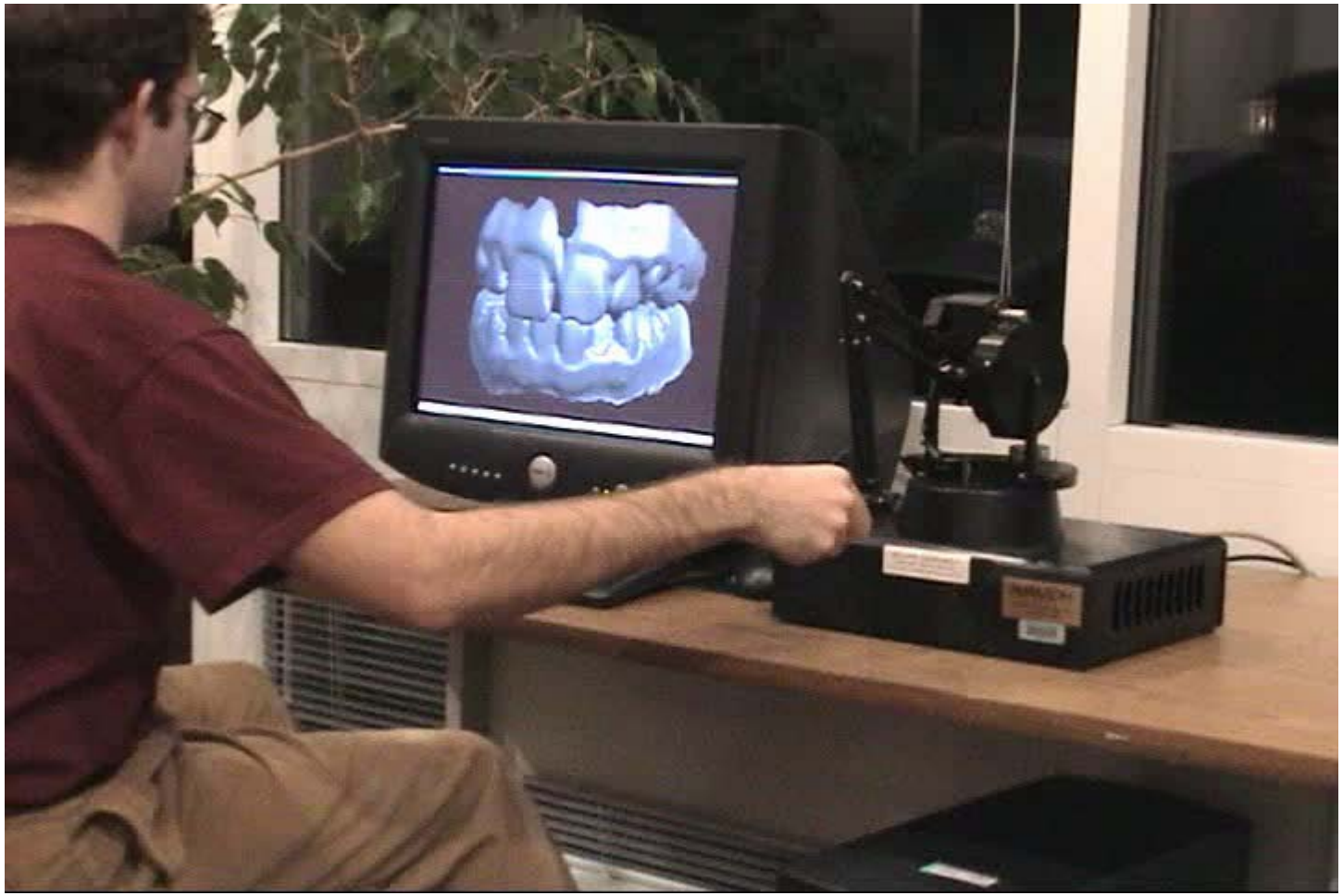
Motor virtual

- Componente clave
- Tarea
  - Lectura de los dispositivos entrada salida
  - Lectura de la base de datos
  - Actualización del estado (base de datos)
  - Escritura en los dispositivos de salida

Motor virtual

- Tiempo real
  - Imposible predecir las acciones de los usuarios
    - El mundo virtual se crea y destruye en tiempo real
  - Requisitos impuestos por los factores humanos
    - Refresco visual 25-30Hz
    - Refresco táctil 500-1000Hz
  - Tan importante es el refresco como la latencia:
    - Latencia: tiempo transcurrido desde que el usuario efectúa una acción hasta que recibe una respuesta
    - Latencia total = latencia del sensor + retraso del bus + cálculo del nuevo estado + retraso del bus + latencia del dispositivo de salida
    - La latencia visual no debe exceder los 100ms

Sistema Gráfico + Háptico



Motor virtual

- Por tanto, los motores de RV:
  - Requieren:
    - Baja latencia
    - Rápida generación de gráficos
    - Hápticos
    - Refresco de audio
    - ...
  - Se tienen que sostener sobre una arquitectura muy potente desde el punto de vista computacional

- Las arquitecturas tradicionalmente se han construido sobre el pipeline de renderizado.

Pipeline

- Definición de pipeline: división de un proceso en etapas, asignando a cada una de ellas distintos recursos.

- Pipelines
  - CPU
  - Gráfico
  - Háptico

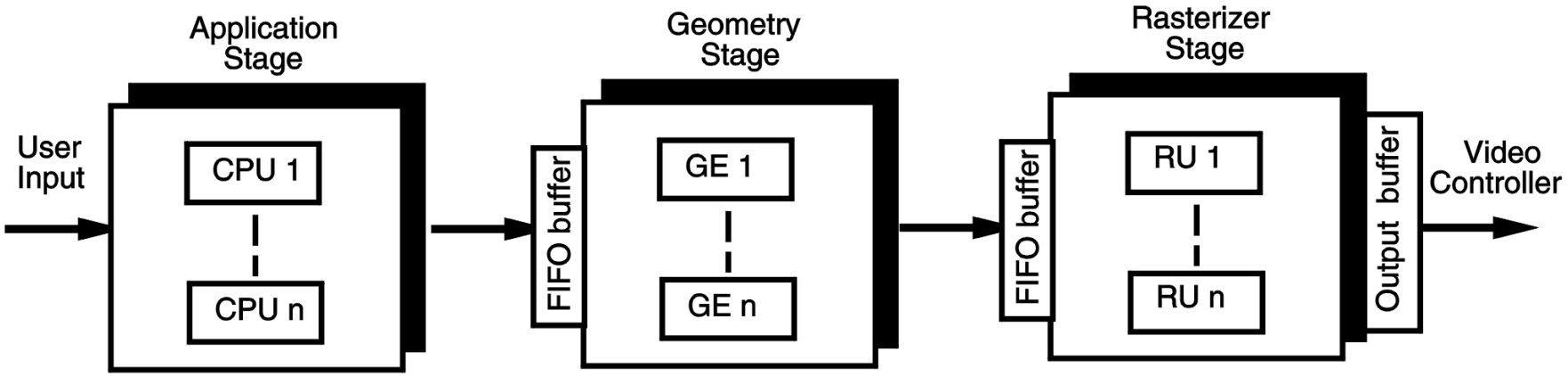
- Render: el proceso de convertir modelos geométricos 3D en escenas 2D para presentarlas a los usuarios
  - También se puede hablar de otros tipos de render (renderizado háptico)

Pipeline gráfico tradicional

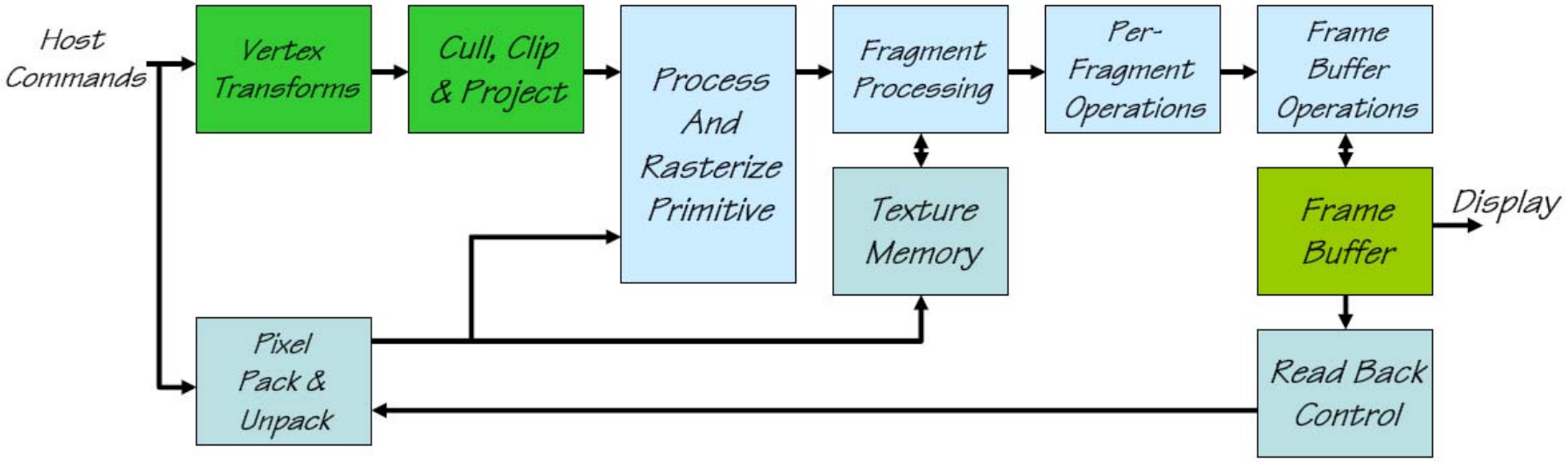
- Etapas
  - Aplicación
    - Lectura de las entradas
    - Lectura y modificación del estado
    - Etapa software sobre la CPU
  - Etapa geométrica
    - Transformaciones
    - Iluminación
    - Efectos atmosféricos
    - Mapeado de texturas
    - Proyecciones
    - Planos de corte
    - Sw o Hw
  - Etapa de rasterizado
    - Transforma la información 3D (vértices, color y textura) en píxeles
    - Controla el buffer de profundidad
    - Antialiasing
    - Conversión de datos en coma flotante a números en coma fija
    - Unidades Hw específicas

Pipeline gráfico tradicional

- Paralelización de las etapas de pipeline



Pipeline Fija

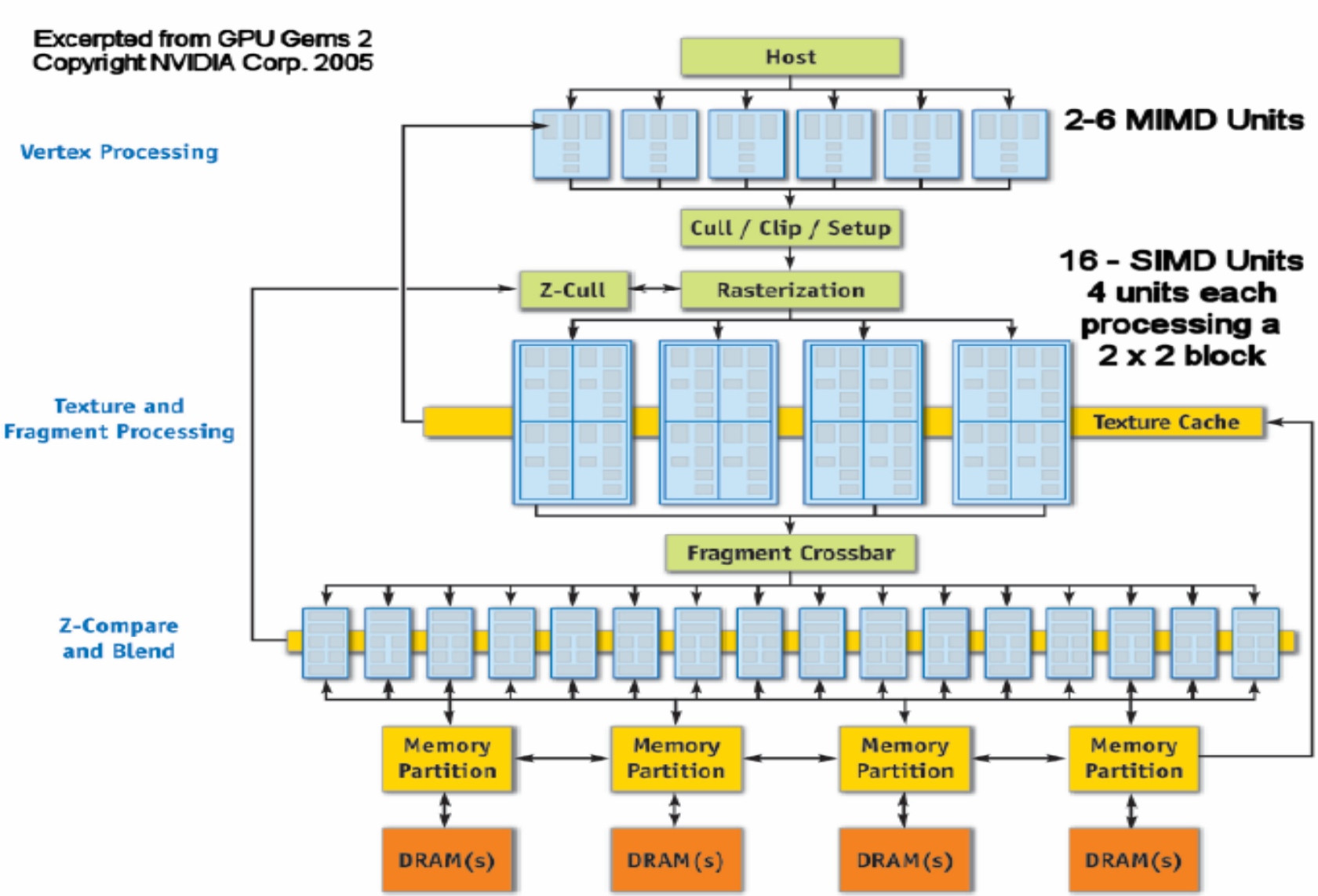


Entradas: primitivas (vértices, polígonos), texturas, comandos

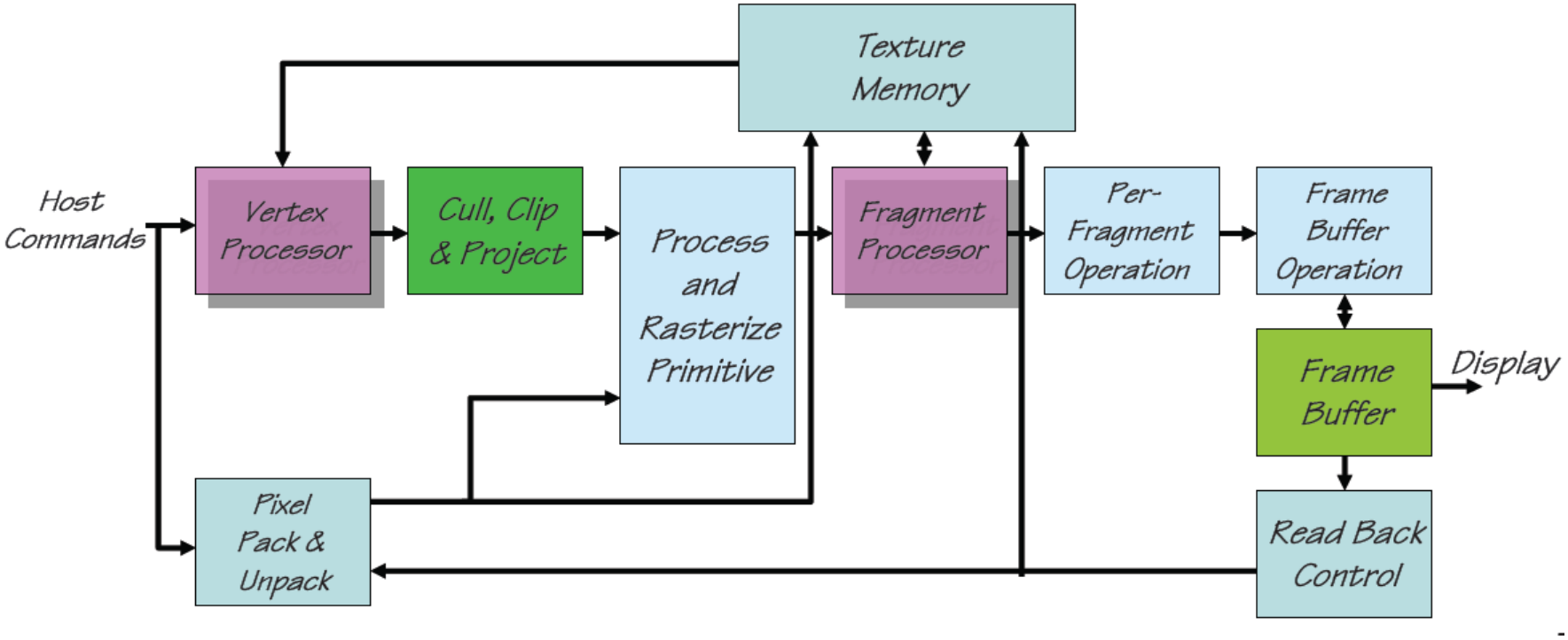
Procesar sólo vértices o fragmentos (pixels). Operaciones fijas.

Las operaciones a realizar se determinan mediante el *estado* de OpenGL. El estado se selecciona mediante comandos.

Bloques de la GPU



Pipeline Programable



Expone al usuario el procesamiento de vértices y fragmentos.

Memoria de textura general: accesible también en el procesado de vértices, y se puede escribir a ella desde el frame buffer!

Pipeline Programable

- El pipeline no cambia básicamente, pero se exponen al usuario el procesado de vértices y fragmentos

- Se pueden modificar las funciones, o se pueden diseñar funciones completamente distintas (incluso para aplicaciones no-gráficas!)

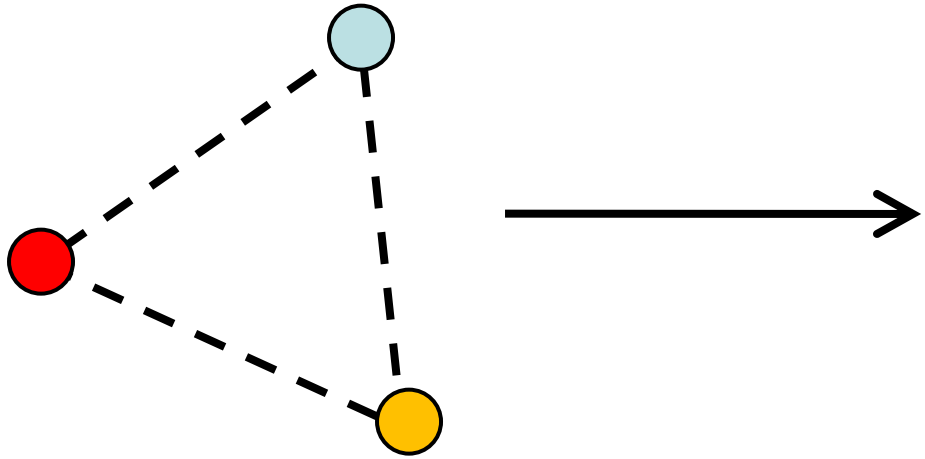
Vertex Processing

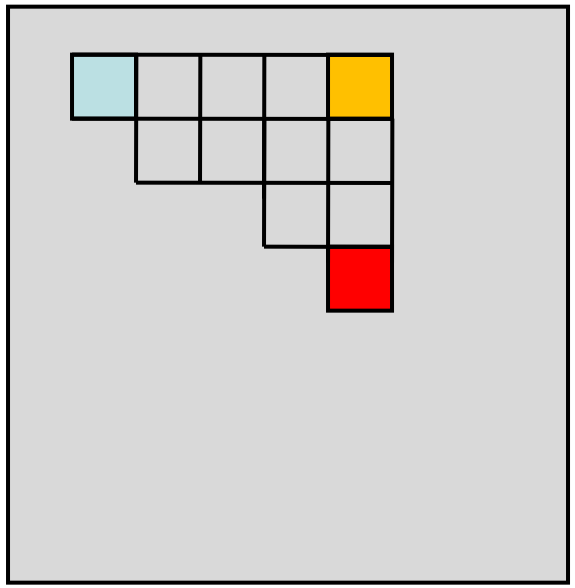
- Funciones que se pueden sustituir/modificar:
  - Tranformación de coordenadas y normales
  - Normalización, escalado
  - Cálculo de iluminación
  - Aplicación de color
  - Generación y transformación de coordenadas de texturas

Fragment Processing

- Funciones que se pueden sustituir/modificar:
  - Acceso y aplicación de texturas
  - Suma y mezcla de colores (blending)
- Funciones que NO se sustituyen:
  - Z-test (profundidad)
  - Posición del fragmento (scan-conversion)
  - ...

Limitaciones





No se puede crear geometría (nuevos vértices)

No se puede cambiar la posición de los fragmentos

Multi-Pass Rendering

- Para crear efectos complejos, realizar varios pases del pipeline, y en cada pase una operación distinta
- El resultado de un pase se puede escribir directamente a la memoria de textura (render to texture)
- En un nuevo pase, se leen los datos escritos a las texturas

Ejemplo 1: Motion Blur (Efectos de Movimiento)

4 pases

16 pases

Ejemplo 2: Depth of Field (Enfoque de la Cámara)

Sin enfoque

Con enfoque

Ejemplo 3: Soft Lighting (Luces con Área)

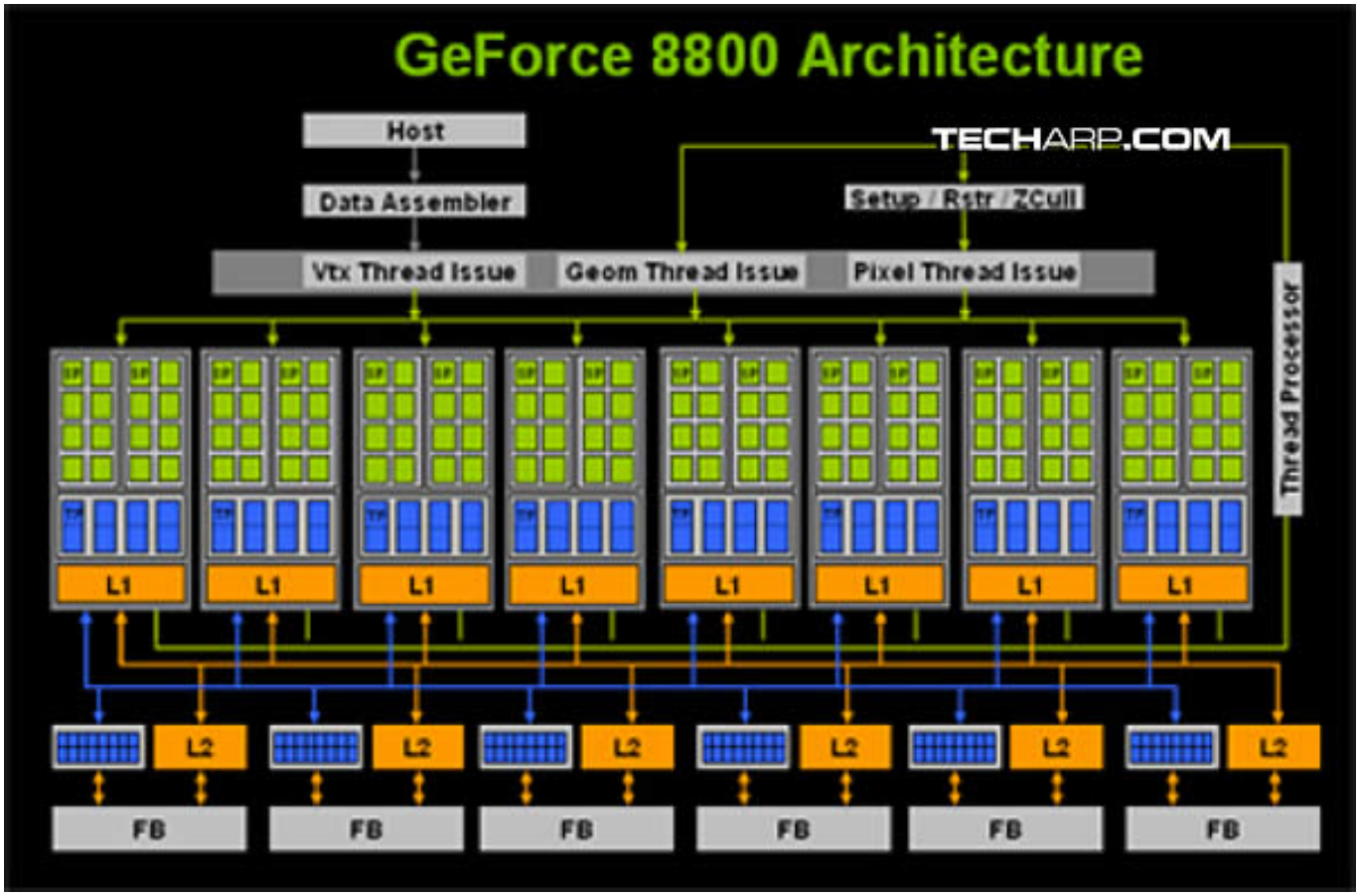
Luz puntual

Luz con área

Lenguajes de Shading

- Cg (NVIDIA): se programa y compila por separado, y se carga desde la aplicación
- GLSL (OpenGL): se programa directamente en la aplicación, y se compila *on-the-fly*; se puede modificar!
- HLSL (DirectX): similar pero de Microsoft

Pipeline Unificada

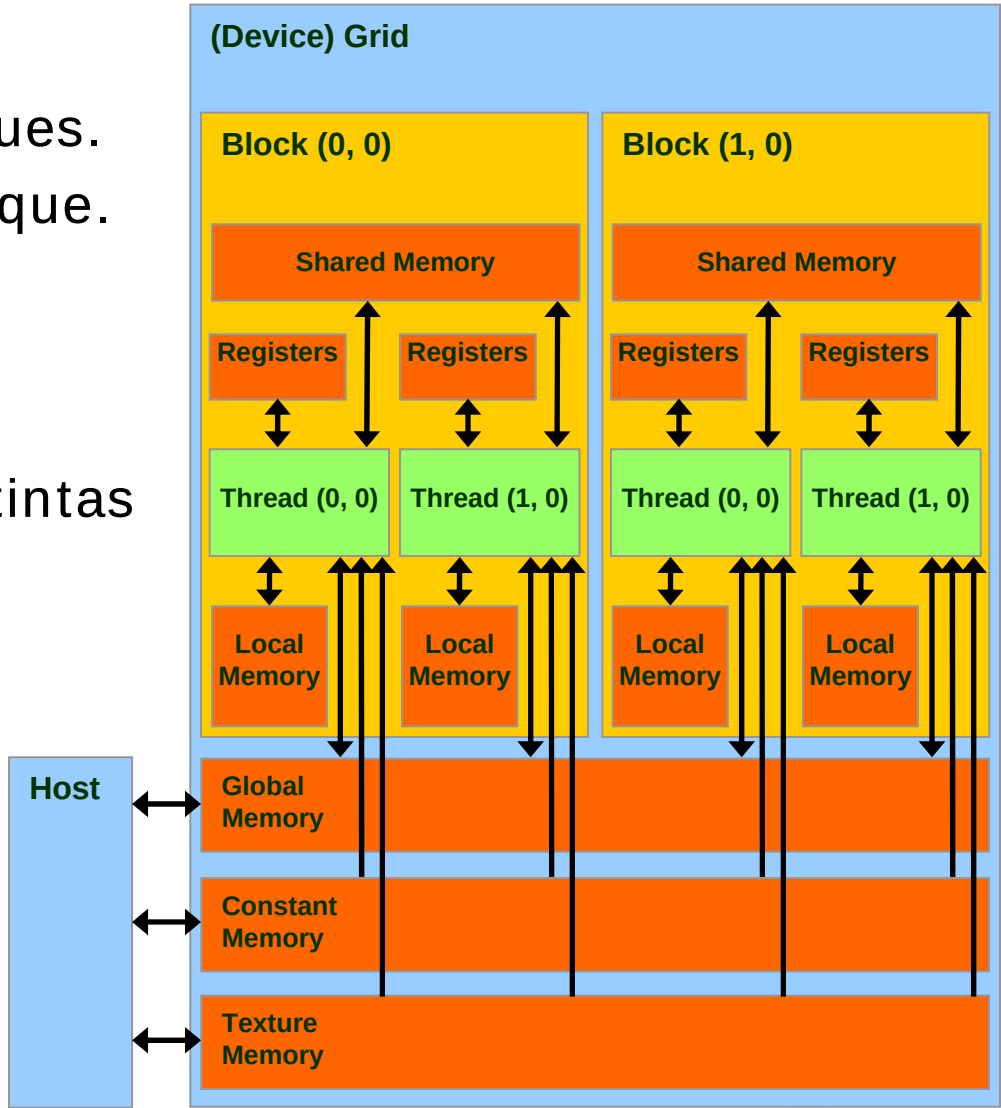


Massive parallel processing: multitud de procesadores ejecutando lo mismo. Desde la CPU, lanzar hebras e indicar cuántas se ejecutan, dónde, y qué memoria utilizan.

Bloques de Hebras

- Hebras distribuidas por bloques.
- Comparten memoria por bloque.
- Identificador de bloque y de hebra.

- 3 tipos de memoria, con distintas propiedades y aplicabilidad



CUDA

- Compute Unified Device Architecture
  - Creado por NVIDIA
  - Compilador + herramientas de desarrollo
  - Funciona en GPUs desde G8X en adelante

Tarjetas gráficas

- Otras características
  - Realizan estéreo (Quadbuffer)
  - Proveen de señales de sincronización para estéreo activo.

Pipeline háptico

- Etapas
  - Detección de colisiones
  - Cálculo de fuerzas
    - Depende de las propiedades de cada objeto
    - Suele utilizarse la ley de Hooke (computacionalmente ligero) o funciones a trozos
  - Suavizado de fuerza
    - Se interpolan las normales de los vértices de la cara que se está tocando
  - Mapeo de la fuerza
  - Texturas hápticas
    - Añaden información de
      - Fricción
      - Suavidad
      - Temperatura

- Control háptico
  - Bucle pesado
  - Bucle ligero

Arquitecturas

- Arquitecturas basadas en PC’s
- Arquitecturas basadas en estaciones de trabajo
- Arquitecturas basadas en sistemas distribuidos

Arquitecturas basadas en PC's

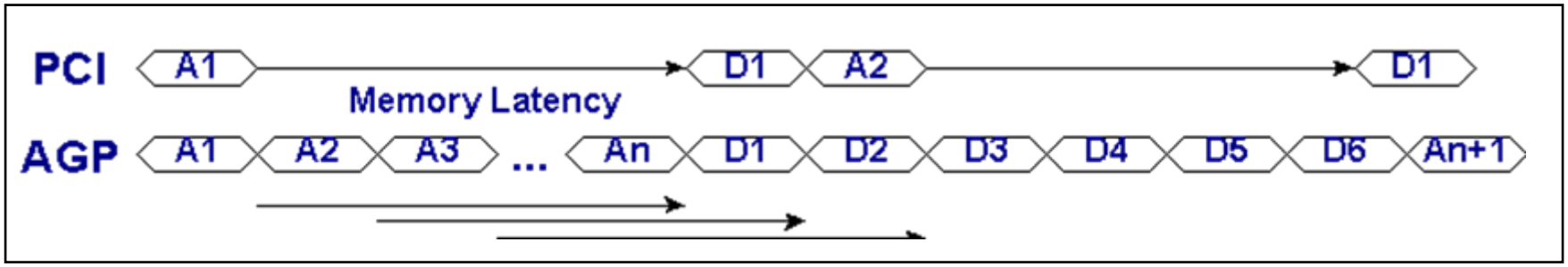
- Es la opción más utilizada hoy
  - Espectacular incremento de las prestaciones
    - A nivel de micros (+3GHz, dual-core, quad-core)
    - A nivel de buses
    - Nuevas tarjetas gráficas
  - Actualmente son sistemas multiusuario
  - Pueden controlar varios dispositivos de salida
  - Precio

AGP

- Accelerate graphic port
- Soluciona el problema del bus
  - Anteriormente los dispositivos gráficos funcionaban utilizando el bus PCI
    - 33MHz
    - La tarjeta gráfica debía competir con otros dispositivos
- Velocidad
  - AGP 1X: velocidad 66 MHz con una tasa de transferencia de 266 MB/s.
  - AGP 2X: velocidad 133 MHz con una tasa de transferencia de 532 MB/s.
  - AGP 4X: velocidad 266 MHz con una tasa de transferencia de 1 GB/s.
  - AGP 8X: velocidad 533 MHz con una tasa de transferencia de 2 GB/s
- Transferencia directa de datos desde memoria

AGP

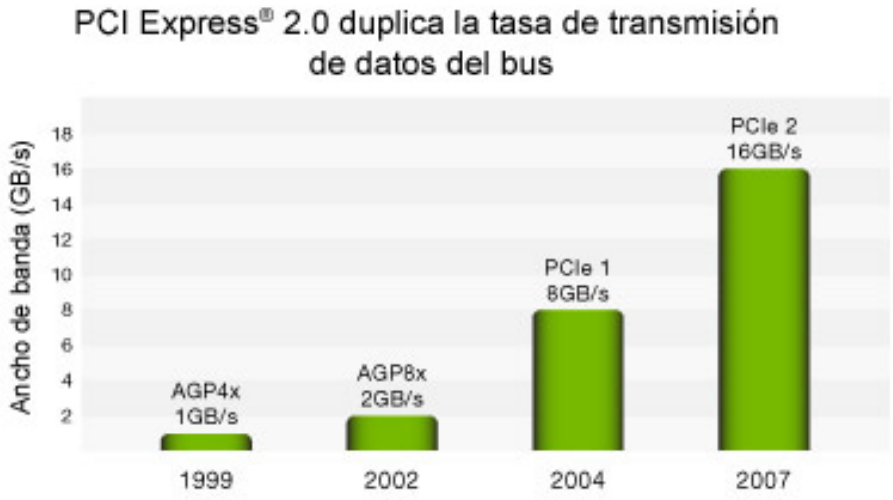
- Reducción de la latencia global
  - Permite enviar nuevas peticiones sin haber recibido los datos anteriores
  - Permite solicitar datos y recibir datos simultáneamente

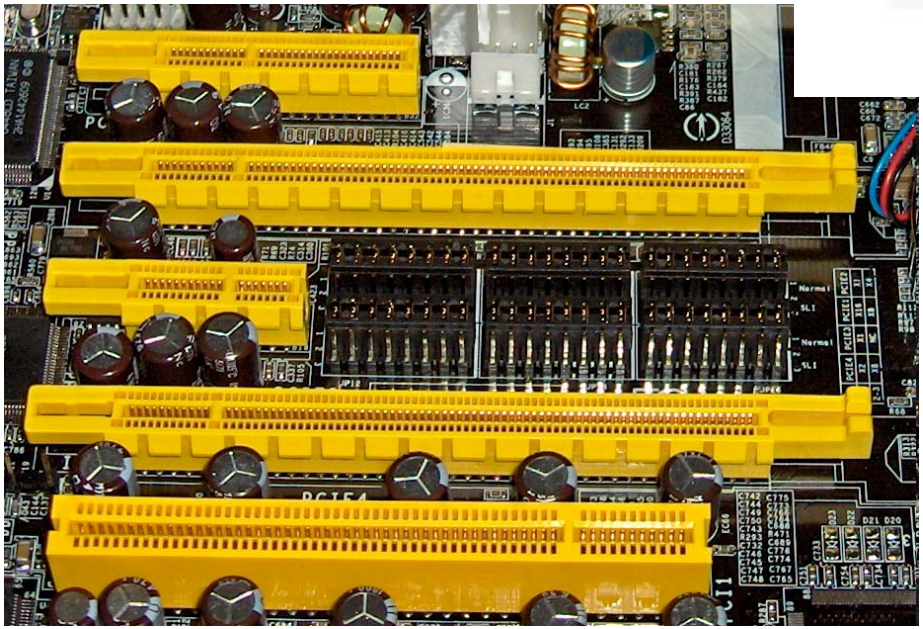


PCI-Express

- Hoy en día el AGP ha quedado superado por el PCI-E
  (Error frecuente: PCI-E ≠ PCI-X)
  - Interfaz serie
  - Estructurado como enlaces punto a punto serie, full-duplex
  - Número de enlaces variable:
    - 1X, 2X, 4X, 8X, 12X, 16X y 32X
  - PCI-E 1:
    - Rendimientos desde 250MB/s hasta 8GB/s
    - Cada enlace 250MB/s. 250 x 32 = 8GB/s
  - 4 veces superior al AGP
  - PCI-E 2.0 dobla la tasa
    - (PCI-E 3.0 ¿en 2010? dobla la tasa otra vez)
  - El más común es el 16X

PCI-Express





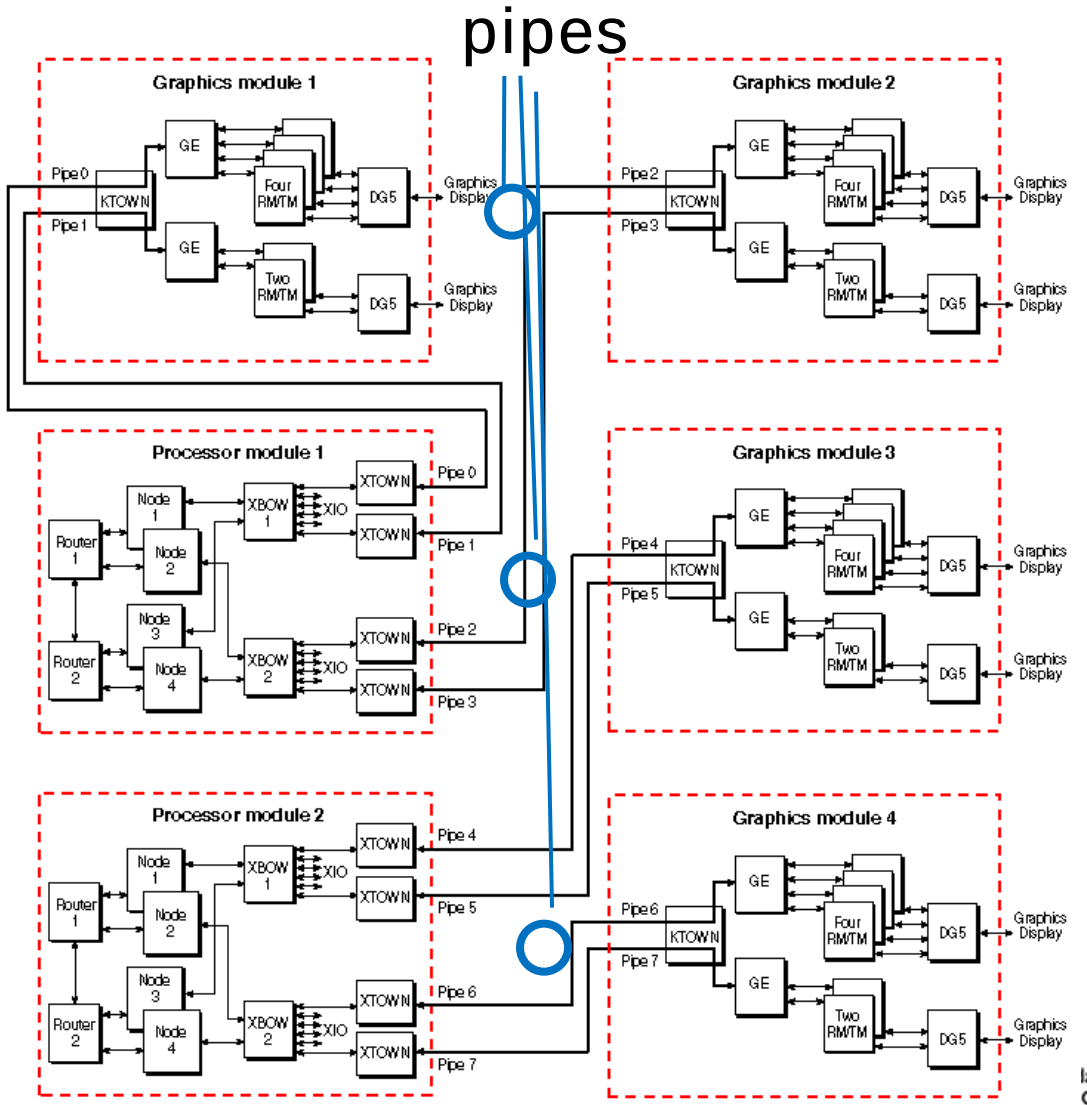
Arquitecturas basadas en workstations

- Después de los PC's es la opción más utilizada
- Mayor capacidad de cómputo
  - Gracias a
    - Arquitecturas superescalables
    - Arquitecturas multiprocesador
    - Mayor velocidad en los buses
  - Tiempo real
- Orientados a la multitarea
  - Requerida por la mayoría de las aplicaciones VR
  - Por lo general usan sistemas operativos UNIX
- Mayor facilidad para soportar varios displays
  - Pueden soportar varias tarjetas gráficas

Un poco de historia

Onyx InfiniteReality

- MIPS
- IRIX (Unix)



SGI Prism

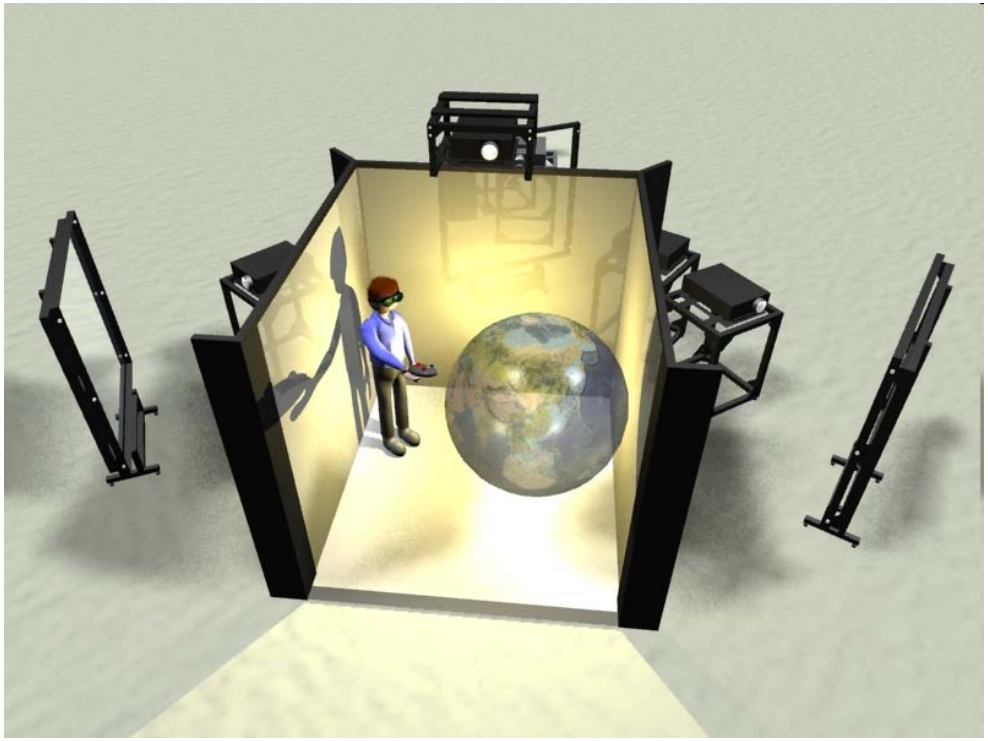
- Basada en Altix 3000 (Itanium2, Linux)
- Multi-pipe
- Tarjetas ATI FireGL 250

Arquitecturas distribuidas

- Sistemas monousuario
  - Control de varios dispositivos
    - Sistemas con varios pipelines
- Sistemas multiusuario
  - Entornos de cooperación

Sistemas con varios pipelines gráficos

- Sistemas que necesitan varios displays
  - Monitores conectados Side-by-side
  - Cuevas
  - Varios proyectores utilizando una misma pantalla
    - Ejemplo: sistemas estéreo pasivo
  - ...



Sistemas con varios pipelines gráficos

- Tipos
  - Tarjetas multipipe
    - Barato
    - No son escalables
    - Comparten el bus AGP/PCI-E, memoria y CPU
  - Uso RACKs con varias tarjetas gráficas
    - Necesitas equipos con más de un bus AGP/PCI-E
    - Son sistemas caros (workstations)
  - Uso de un mismo pipe para varios dispositivos
    - Pérdida de resolución
  - Uso de cluster de PC's cada uno con su propia tarjeta gráfica
    - Conectados por redes de alta velocidad
      - Es el cuello de botella
      - Uso de topologías y protocolos de red optimizados
        - » WireGl → Chromium

WireGl - Chromium

WireGl - Chromium

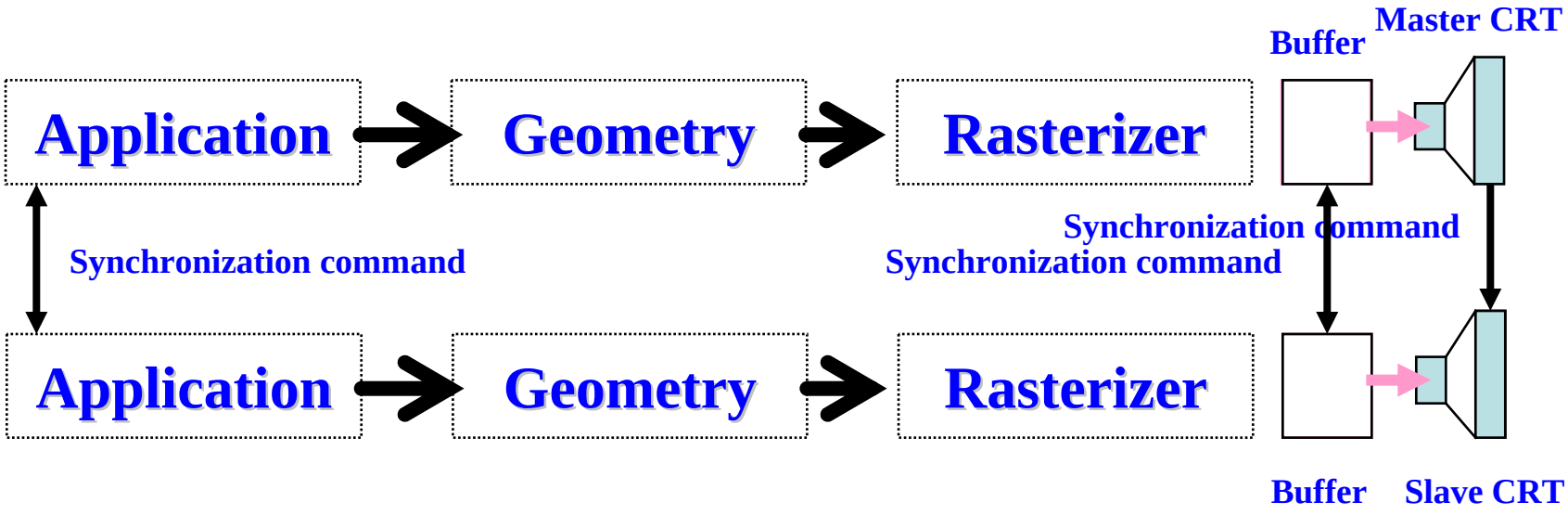
- Protocolo basado en TCP/IP o en protocolos Myrinet
- Se basa en distintos servidores de control cada uno de los cuales supervisa varios servidores de render
- El servidor de control captura los comandos OpenGl y los distribuye entre los distintos servidores de render
- Codifica las instrucciones OpenGl compactando y empaquetando datos y operaciones
- El servidor de control lleva un control del estado de cada nodo de render
  - Solo envía la información que ha cambiado

Sincronización de pipelines gráficos

- Crítica en displays conectados side-by-side
  - Especialmente cuando se usan varios monitores CRT
    - Requieren sincronización pixel a pixel
- En el caso de utilizar imágenes estéreo la sincronización es todavía más crítica
  - Se requiere que todos los dispositivos muestren a la vez el mismo campo de la imagen (izquierdo o derecho)
- La falta de sincronización produce parpadeo y malestar en el usuario

Sincronización de pipelines gráficos

- La sincronización puede darse a 3 niveles
  - Sincronización a nivel de aplicación
  - Sincronización a nivel de cambio de buffers
  - Sincronización a nivel de video



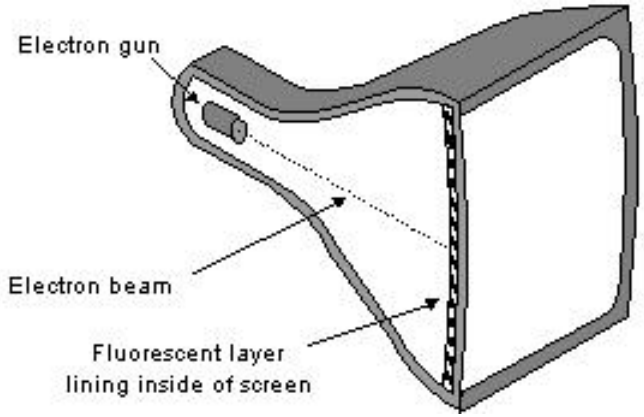
Sincronización de pipelines gráficos

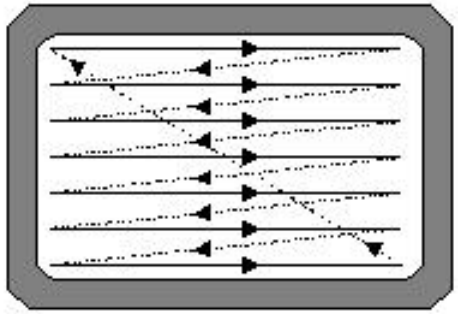
- Sincronización a nivel de aplicación
  - Consiste en comenzar a procesar un nuevo frame a la vez en todos los pipeline
  - Puede que haya pipelines que terminen antes que otros
    - Cuando se llena cada framebuffer se muestra

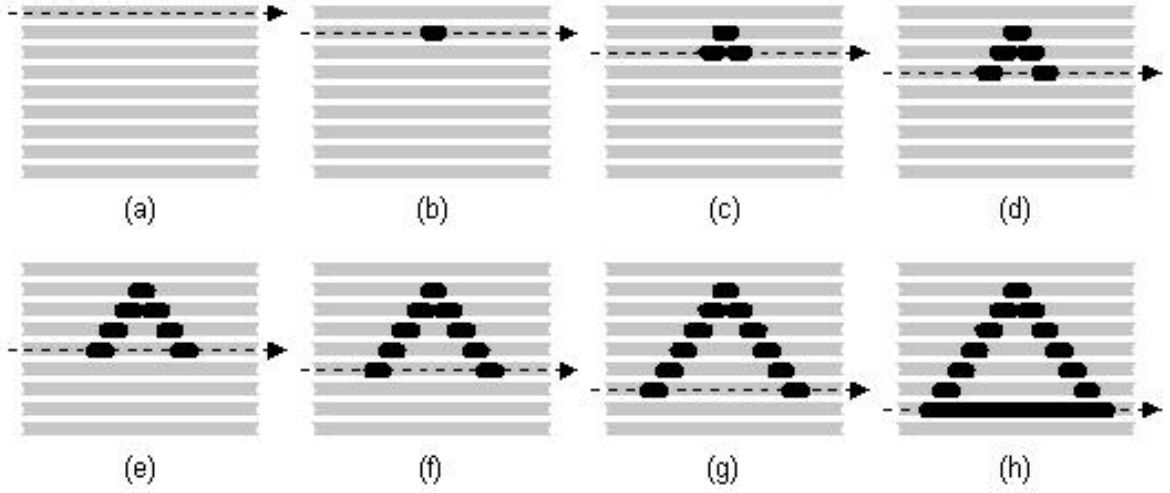
Sincronización de pipelines gráficos

- Sincronización a nivel de cambio de buffers
  - Cuando termina un pipeline espera a que terminen el resto
  - Cuando todos han terminado se intercambian los buffers.
    - glutSwapBuffer() (GLUT)
  - Problema: cada display comienza a pintar en momentos distintos, dependiendo de su ciclo y de su frecuencia de refresco
    - Lo monitores CRT empiezan a pintar cuando su disparador retorna de la esquina inferior derecha a la esquina superior izquierda

Sincronización de pipelines gráficos







Sincronización de pipelines gráficos

- Sincronización a nivel de video
  - Sincronización a nivel Hardware
    - Interconexión entre tarjetas de video
  - Arquitectura maestro-esclavo
  - Asegura que los dispositivos utilicen la misma frecuencia y ciclo de refresco

Genlock

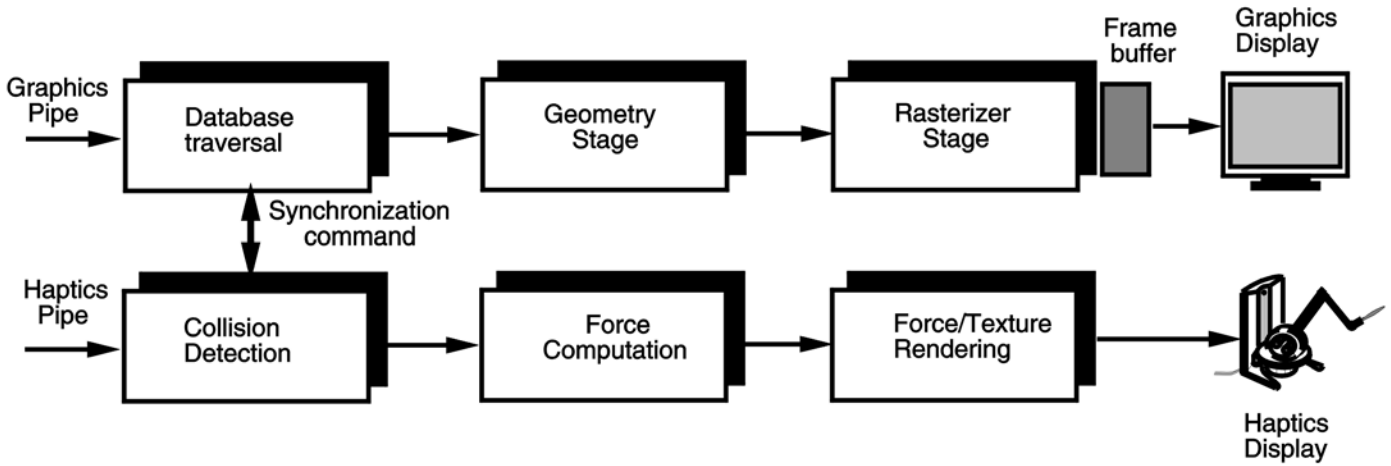
- Señal externa generada por hardware para sincronizar distintos pipelines
- Conexión en de dipositivos en cadena (Daisy-Chain)
- Realiza
  - La sincronización del intercambio de buffers
  - La sincronización a nivel de video
  - La sincronización de campos en imágenes estéreo
  - La sincronización con otros dispositivos (como trackers)
    - Reducción de interferencias

Sincronización de pipelines hápticos

- La sincronización se realiza a nivel de aplicación
  - Después de la detección de colisiones
    - Mas carga en el dispositivo dedicado
    - Mas carga en la red
  - Después del calculo de fuerzas
    - Solo se trasmite el vector de fuerzas
    - Menos carga en el dispositivo dedicado

Sincronización de dispositivos heterogéneos

- La sincronización se realiza a nivel software
- Necesidad de desacoplado
  - Distintas frecuencias de trabajo



Sistemas VR distribuidos multiusuario

- Objetivo generar entornos virtuales distribuidos
  - Varios usuarios separados físicamente comparten un mismo entorno virtual
- Todos los usuarios pueden modificar su entorno
  - Entornos de colaboración
    - Sólo un usuario puede manipular un objeto al mismo tiempo
  - Entornos cooperativos
    - Varios usuarios pueden manipular el mismo objeto a la vez
- A través de red
  - LAN
  - WAN

Topología de la red

- Factores
  - Tipo de aplicación
    - Entornos colaboración
    - Entornos cooperativos
  - Número máximo de usuarios
  - Otros
    - Latencia máxima admitida
    - Escalabilidad
    - Tolerancia a fallos

Topología de la red

- La clave: reducción del tráfico
  - Todos los equipos mantienen un copia de la escena
  - Sólo transmiten las modificaciones que hacen
  - Utilizan un buffer de transmisión para mantener la coherencia en caso de fallo de la red
- Tipos de redes
  - Conexión punto a punto
  - Topologías con un único servidor
  - Topologías con varios servidores
  - Protocolos peer-to-peer

Topología de la red

- Conexión punto a punto
  - Sólo permiten interconectar dos equipos
  - Buffer por si errores y retransmitir
  - Funcionan bien sobre TCP/IP
    - Protocolo orientado a conexión

Topología de la red

- Topologías con un único servidor
  - Unicast sobre TCP/IP
  - Los usuarios cuando modifican la escena se lo notifican al servidor
  - El servidor transmite los cambios en la escena sólo a los nodos que les pueda interesar
  - El número máximo de usuarios lo determinan la capacidad del servidor y la capacidad de la red
  - Si la red es rápida el servidor es el cuello de botella.

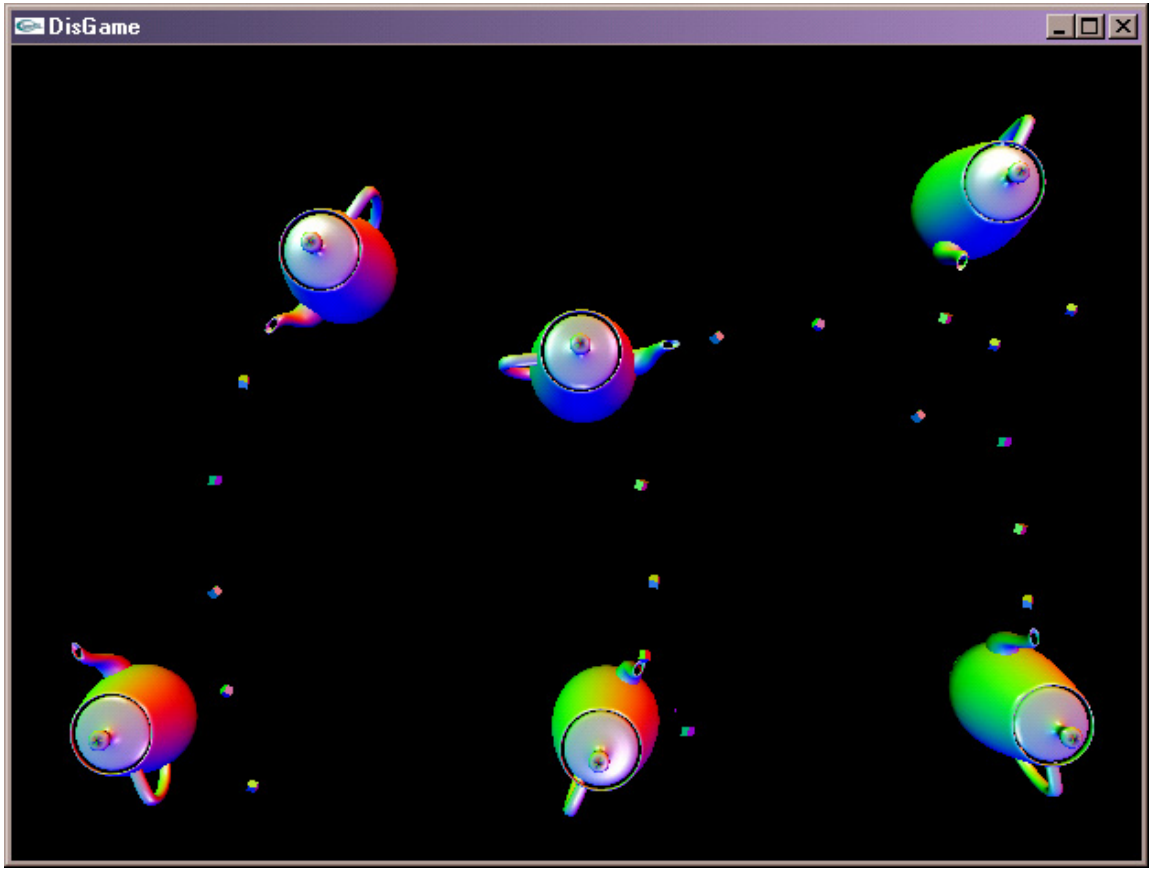
Topología de la red

- Topologías con varios servidores
  - Unicast sobre TCP/IP
  - Cada nodo lleva asociado un servidor
  - Los usuarios cuando modifican la escena se lo notifican al servidor correspondiente
  - El servidor trasmite los cambios en la escena al resto de servidores y a los nodos que tiene asociados (sólo si les pudiera interesar)
  - Aumenta el tráfico por la red
  - El número máximo de usuarios lo determina la capacidad de la red

Topología de la red

- Protocolos peer-to-peer
  - Direccionamiento multicast, sobre UDP/IP
  - Los nodos cuando modifican la escena se lo notifican al resto mediante mensajes multicast
  - Un nodo procesa los mensajes sólo cuando le interesa
  - Si el sistema se queda sin usuarios se pierde el entorno
  - Un nodo al conectarse descarga el entorno de otro nodo mediante TCP-IP

Ejemplo: Juego Distribuido



Dueling Teapots

http://www.scheib.net/school/243/index.html