Optimizing Games for ATI's IMAGEON™ 2300

Aaftab Munshi

3D Architect
ATI Research

“A 3D hardware solution enables publishers to extend brands to mobile devices while remaining close to original vision of the property”

Stuart Platt
Director of Product Development
THQ Wireless, Inc.
(GDC’04)

Presentation Outline

IMAGEON™ 2300: Integrated Architecture

Optimization Process

Designing Games for Performance

Porting Games from PC & Consoles

IMAGEON™ 2300: Integrated Architecture

IMAGEON™ - Powered 3D Gaming Vision

Hardware 3D Floating Point Pipeline:

- Better use of the main CPU chip for AI and physics
- No more worrying about number conversion errors

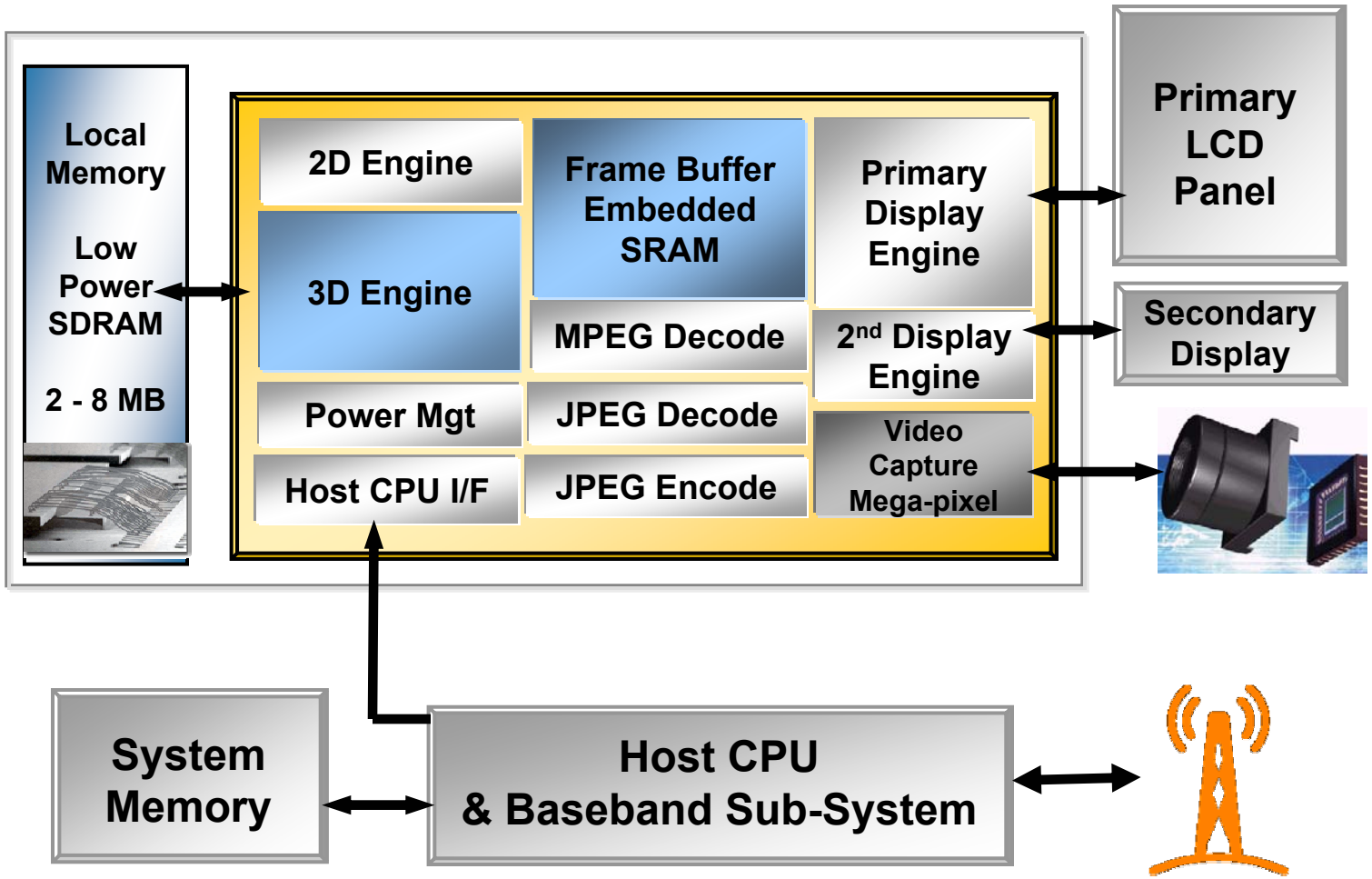
Faster Game Loop

- Higher polygon counts for models and advanced texturing
- No more extra polygon reduction & texturing tweaking steps
- Smooth control and fluid animations
- Very high image quality, immersive environments

Advanced Electronics

- Very low power consumption
- Small die footprint

Block Diagram



Features

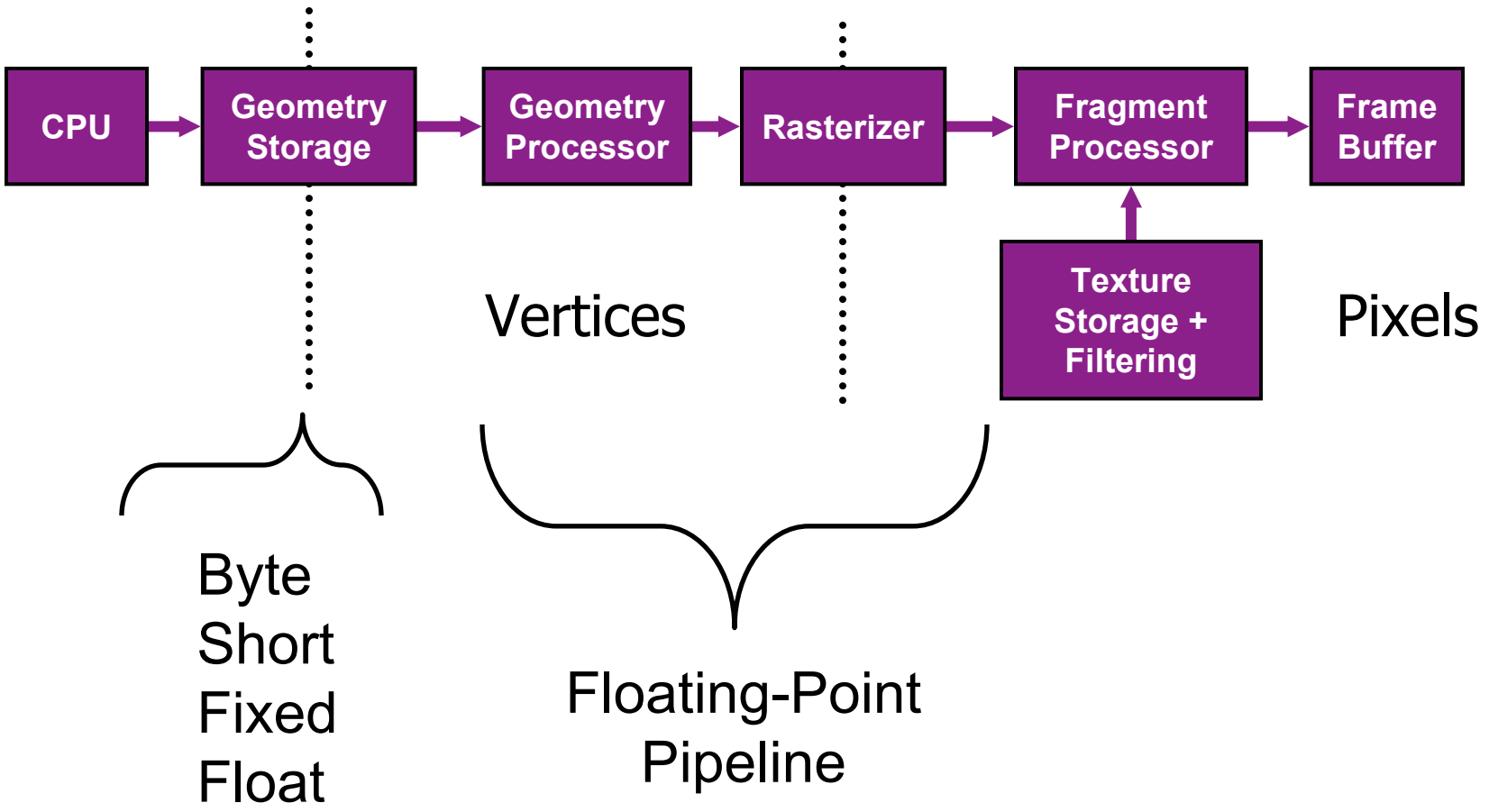
3D Engine

- ***Hardware 3D Floating Point pipeline***
- **Complete geometry process and pixel rendering pipeline**
- **OpenGL ES Texture combine modes, mip-mapping**
- **8/16-bpp destination, 8/16/32-bpp texture, palletized texture**
- **Per-pixel perspective correction**
- **Bi-linear and Tri-linear texture filtering**
- **Dithering, fog, alpha blending**

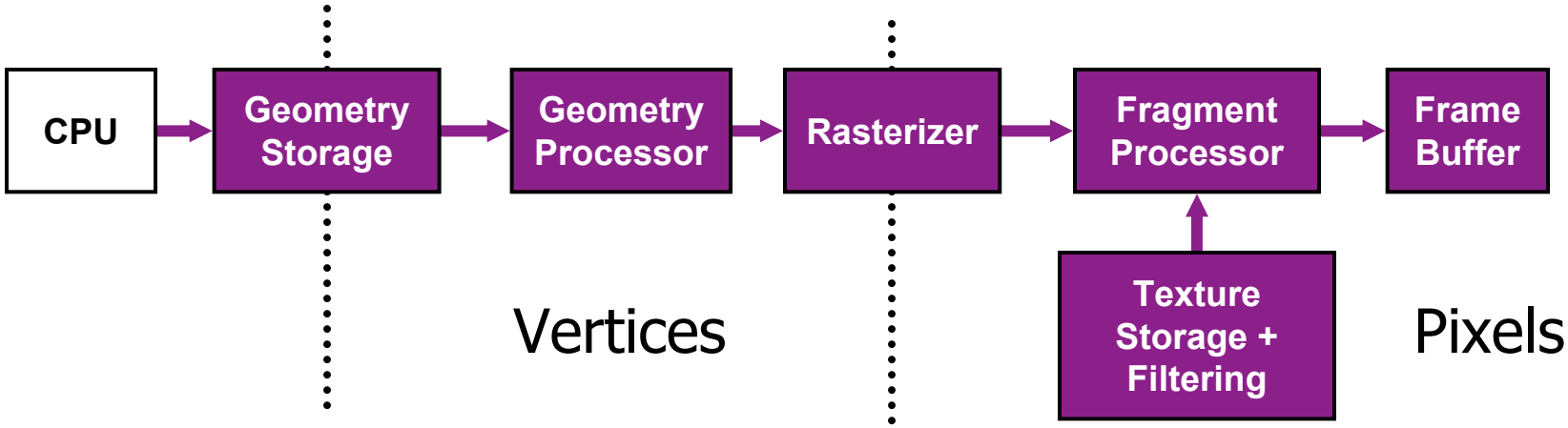
Frame Buffer

- **Double-buffer support for up to QVGA (320x240) resolution**
- **2 - 8MB of frame buffer storage, 384KB of internal storage**

3D Pipeline


CPU
Geometry Storage
Geometry Processor
Rasterizer
Fragment Processor
Frame Buffer
Texture Storage + Filtering
Vertices
Pixels
Byte
Short
Fixed
Float
Floating-Point Pipeline

CPU Interface



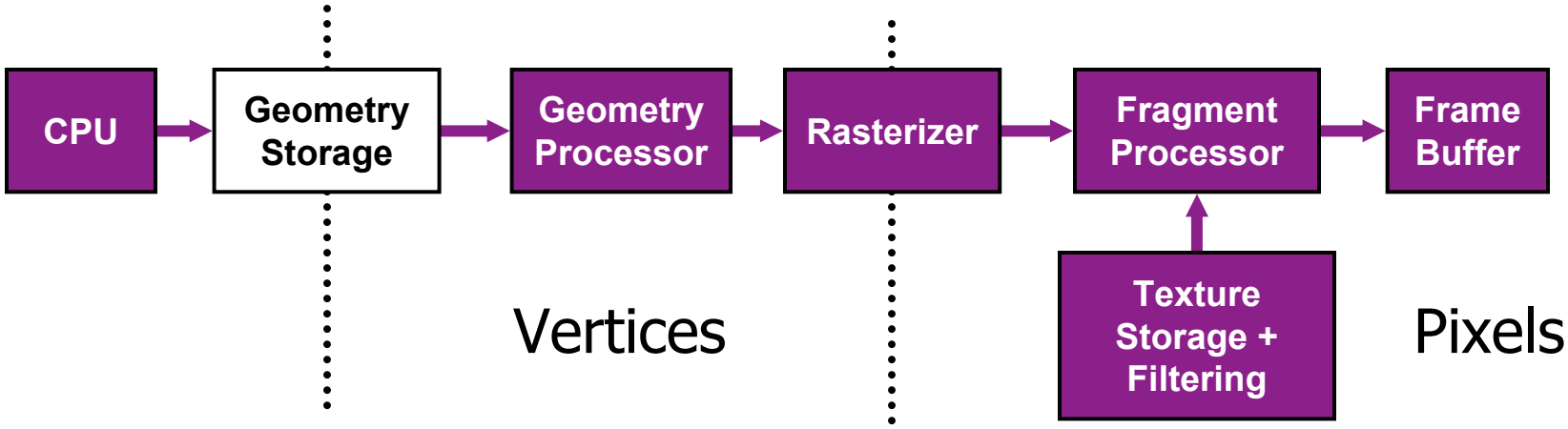
CPU to RAM Bandwidth: 600 MB/sec (32bits @ 150 MHz)

GPU-to-SDRAM bandwidth: 400 MB/sec (32 bits @ 100MHz)

CPU-to-GPU Bandwidth (discrete solution): ~10 MB/sec

-- PLATFORM-DEPENDENT ! --

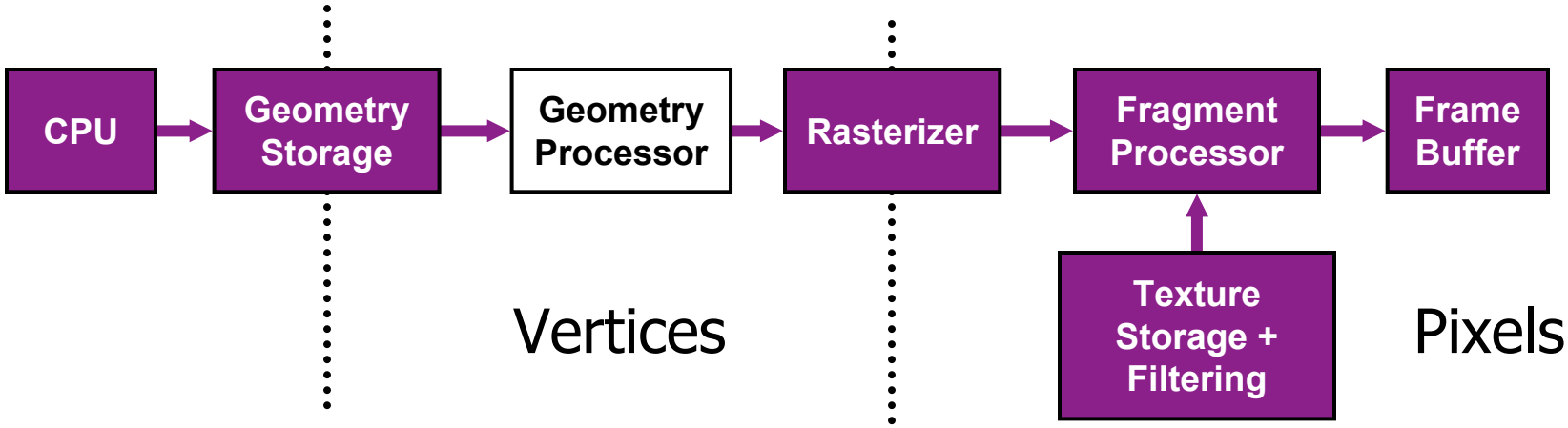
Geometry Storage



Vertex Buffer Objects

- Modified *GL_ARB_vertex_buffer_object*
- Enable caching of geometry in graphics memory
  - vertex data & mesh lists

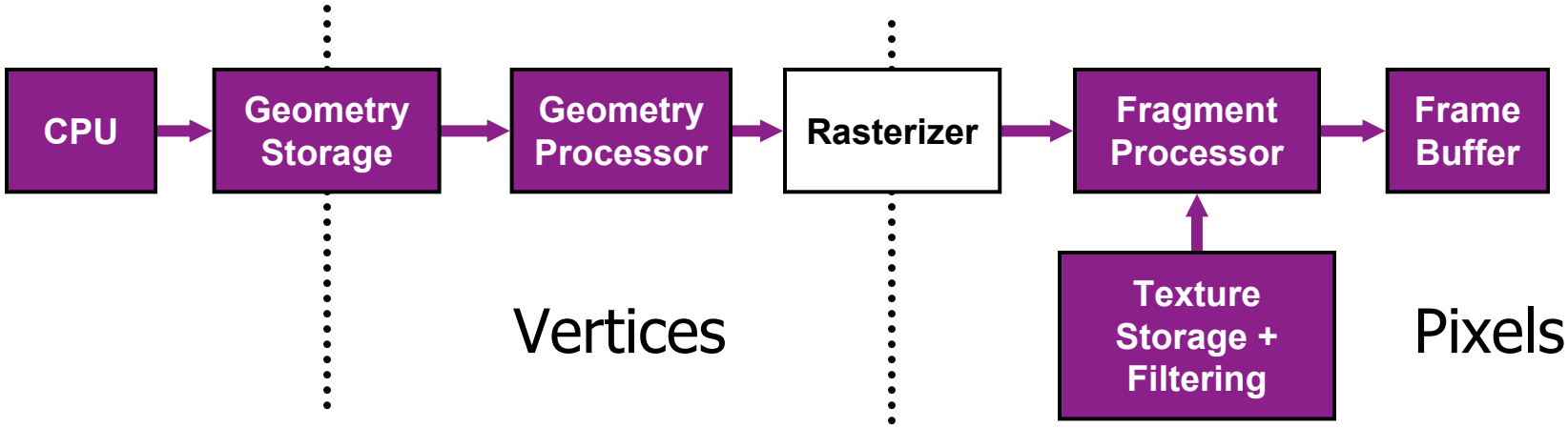
Geometry Processing



Vertex Transform Engine

- Vertex transformation performance: 1M vertices/sec
- Accelerate both vertex and texture coordinate transforms
- Supported vertex and texture coordinate data formats
  - Byte / Short / Fixed / Float

Rasterizer



Triangle setup engine

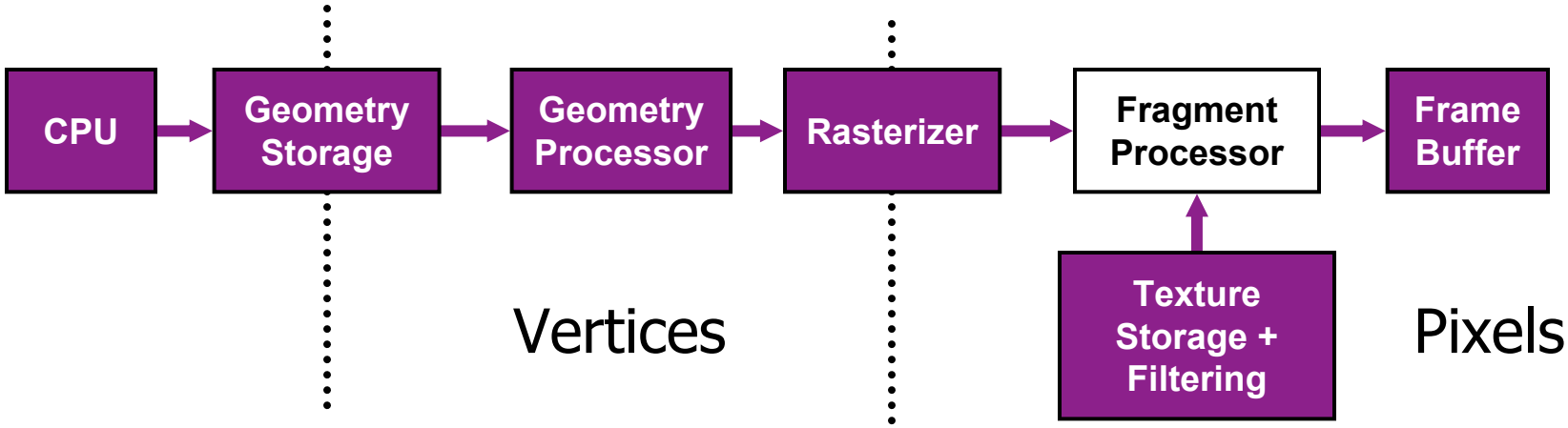
- Performance: 1M Triangles / sec

Flat / Smooth shading models (interpolation)

Per-pixel perspective correction

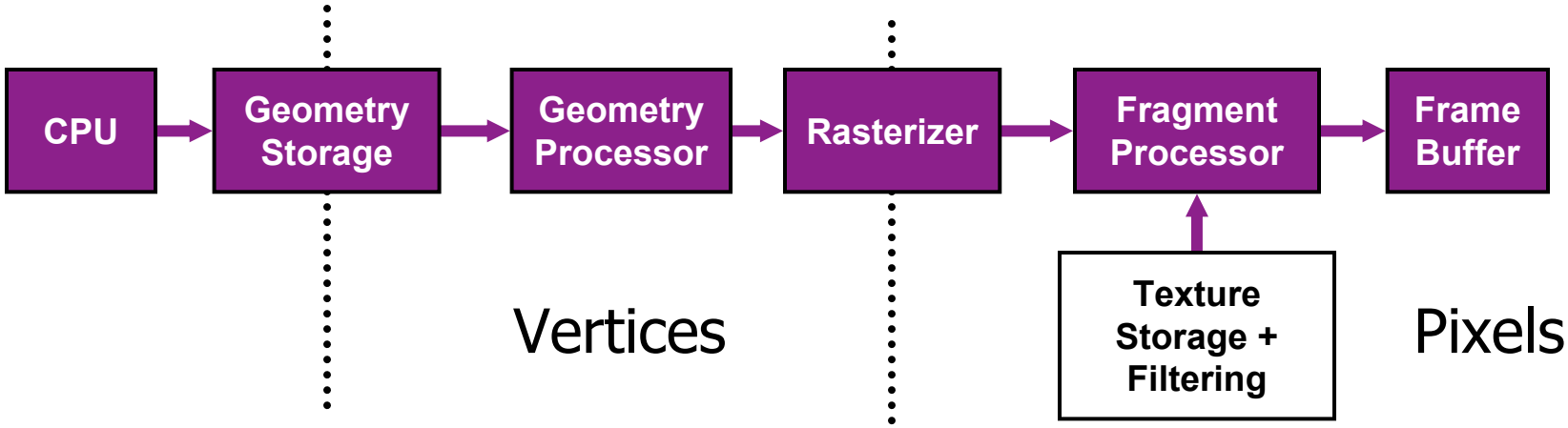
Pixel pipeline : 100 M pixels/sec

Fragment Processor



- 32-bit internal pixel pipeline
- Single texture pipeline
- OpenGL ES 1.0 Texture Combine Modes
- Alpha blending, fog, dither

Texture Storage and Filtering

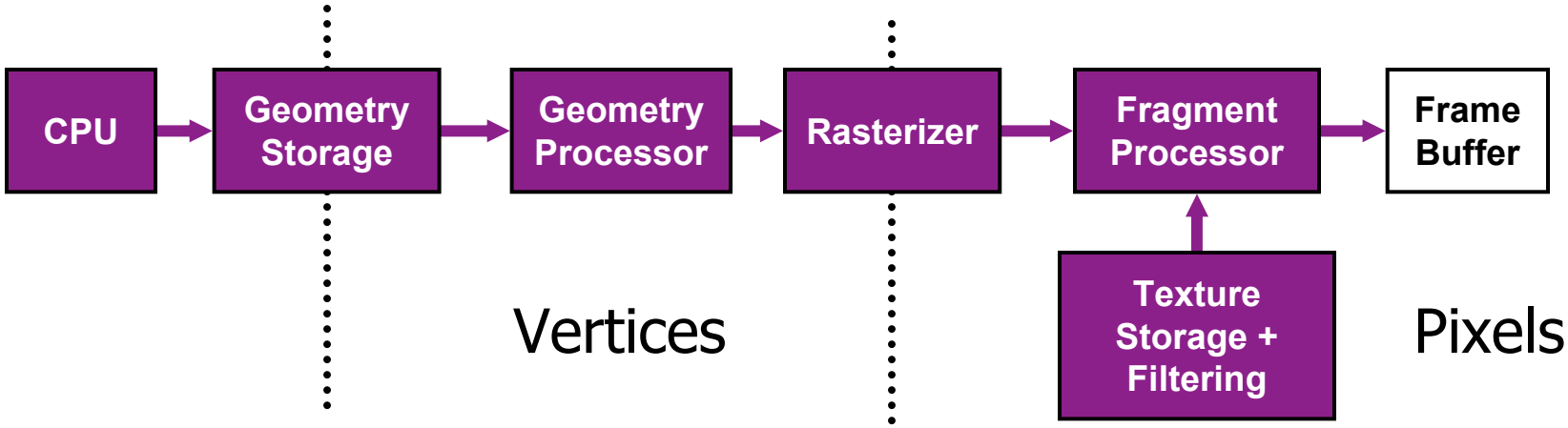


Bi-linear & tri-linear filtering

Mip-mapped picture data

8/16/32 bits per pixel

Frame Buffer



2 - 8MB of double-buffered frame buffer storage

- Plus: 384KB of internal storage
- Shared with VBOs & texture

QVGA size (320 x 240)

16-bit Z, 16-bit pixel

OpenGL ES : ATI Extensions

Modified GL_ARB_vertex_buffer_object

- glBindBufferARB(…);
- glDeleteBuffersARB(…);
- glBufferDataARB(…);
- glBufferDataATI(…);
- glBufferSubDataARB(…);
- glDrawVertexBufferObjectATI(…);

Mesh Lists

- glMeshListATI(…);

Summary: IMAGEON™ 2300

Full hardware 3D floating point pipeline

- Faster game loop: More CPU for AI, physics, …
- Fixed point in ➔ Floating point processing

Comprehensive OpenGL ES implementation

Designed for very low power consumption

Optimization Process

Optimizations

Regardless on how well an application was written, there will always be one bottleneck somewhere

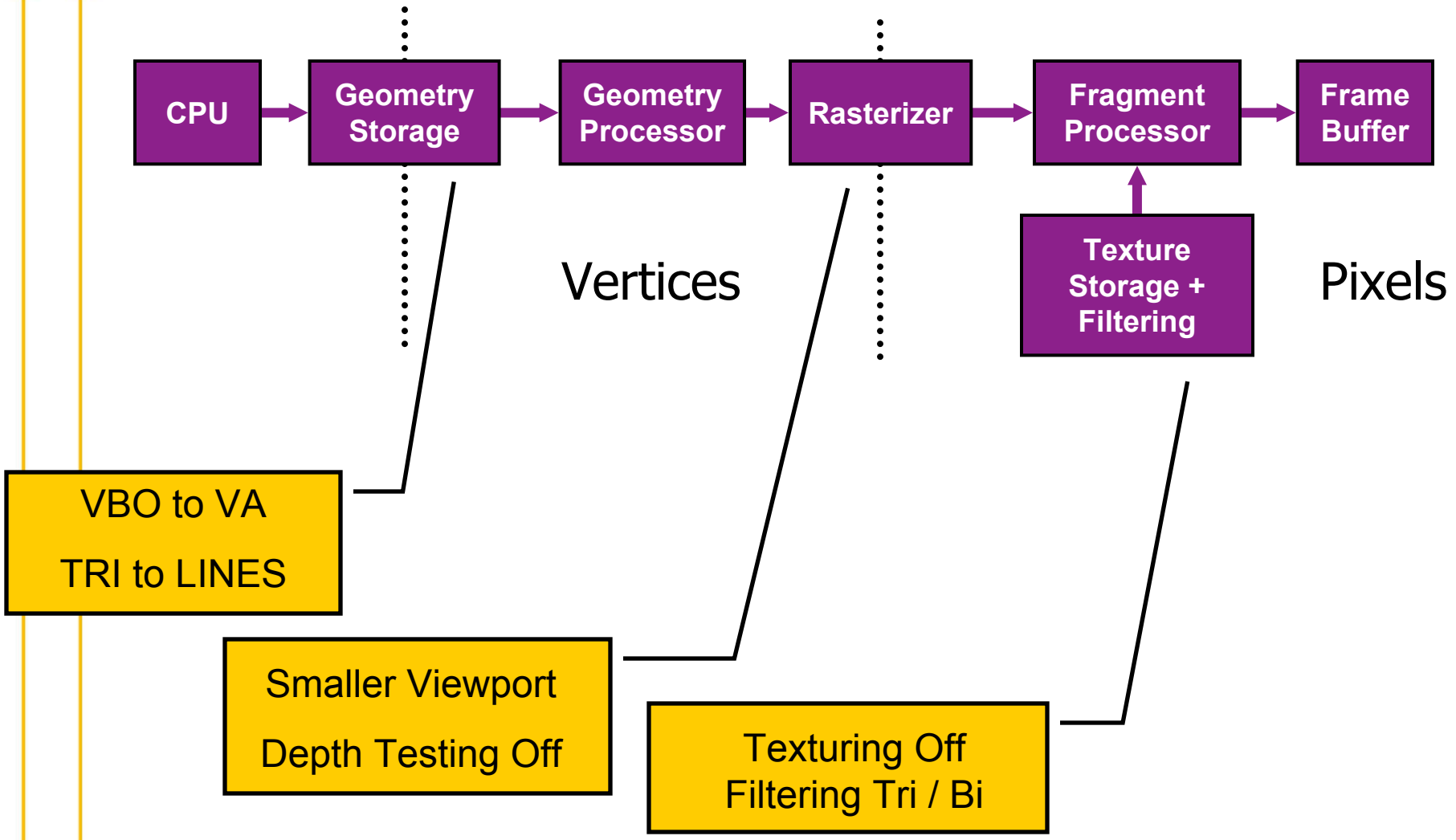
Optimization: identify and solve the bottlenecks

Measure: FPS

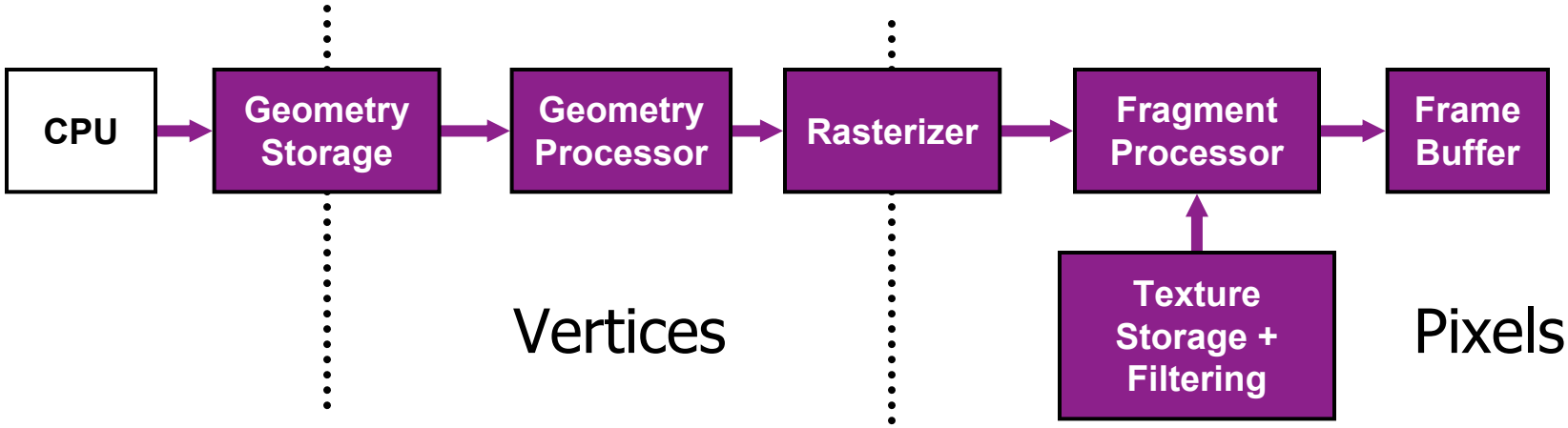
Process: Walk the 3D pipeline

- Start with one-parameter tweaking
- Move to more systematic optimizations where there is a better payback
- Repeat

Start With The 'Easy' Stuff



CPU & Memory Transfer Limits



-- PLATFORM-SPECIFIC! –

- **IMAGEON TIP: Monitor bus bandwidth usage closely !**

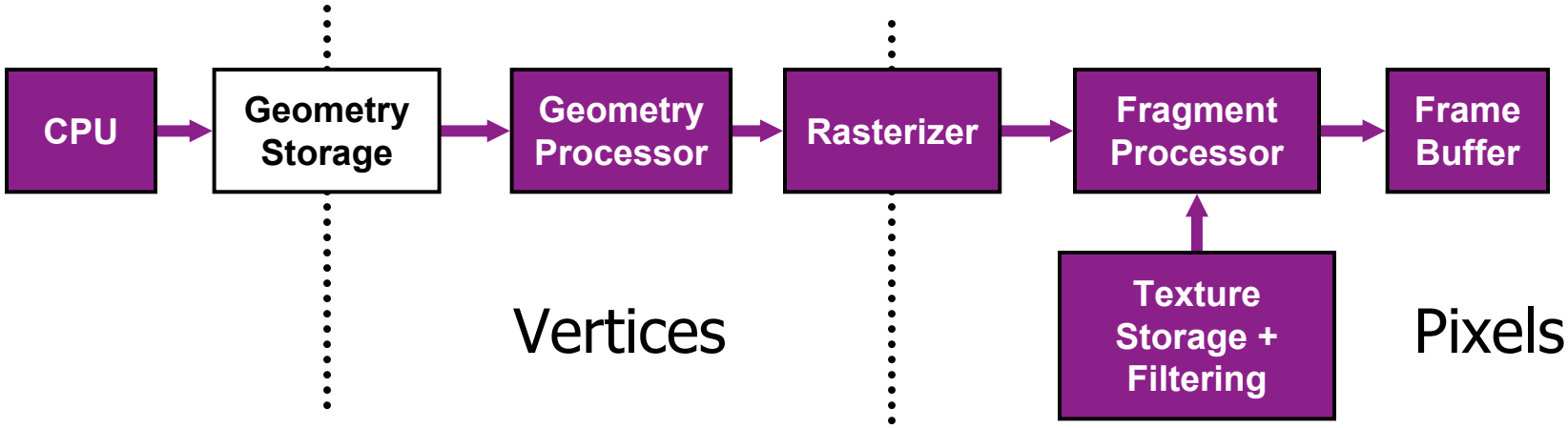
Pitfall: Too many small calls

- **Avoid too many state changes in the driver**
- **Maximize triangle batch size; the more triangles per batch the better**

Pitfall: Too much game assets per frame

- **Use lower-res textures; Reduce model details**

Geometry Storage Limitations



Affected by: Vertex arrays' size & data type

- IMAGEON TIP: Use byte and short whenever possible.

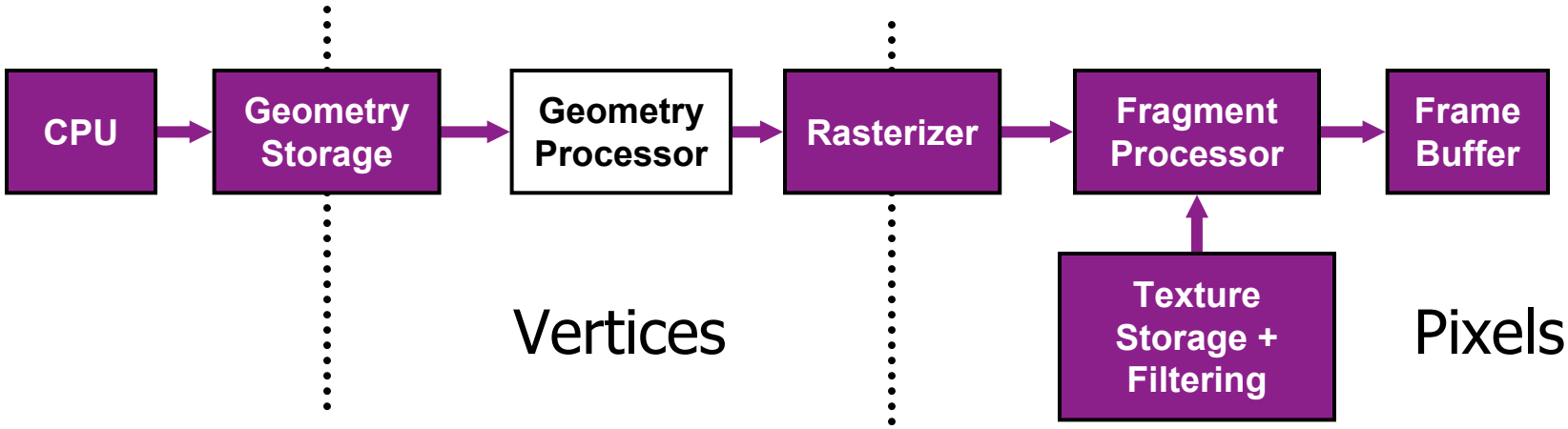
Try to batch as many triangles as possible

- Use strips for saving memory and bandwidth

Pitfalls:

- Vertex Buffer Objects: not used or misused
- VBO memory overflow (reverts to vertex arrays)

Geometry Processing Limits

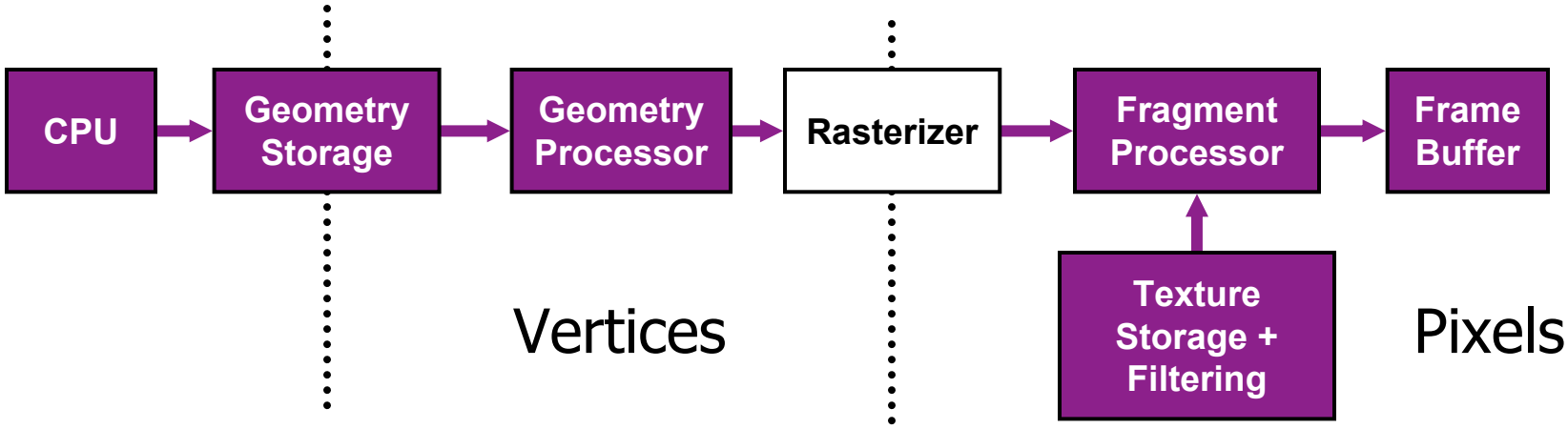


Pitfall: Too many vertices

- Do broad occlusion testing before (PVS)

Avoid deep scene DAGs

Rasterizer / Pixel Fill Limitations



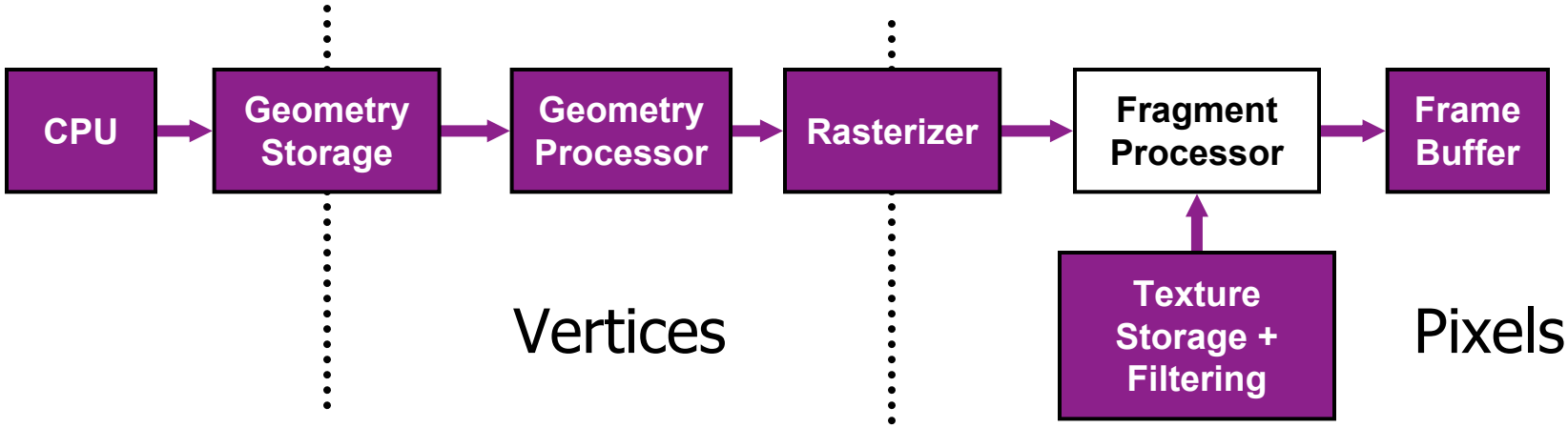
-- Rarely the bottleneck –

- Tuned to the geometry transform performance

Pitfalls:

- Triangles size on screen; Viewport size
- Many depth testing switching (affects triangle setup)

Fragment Processing Limitation



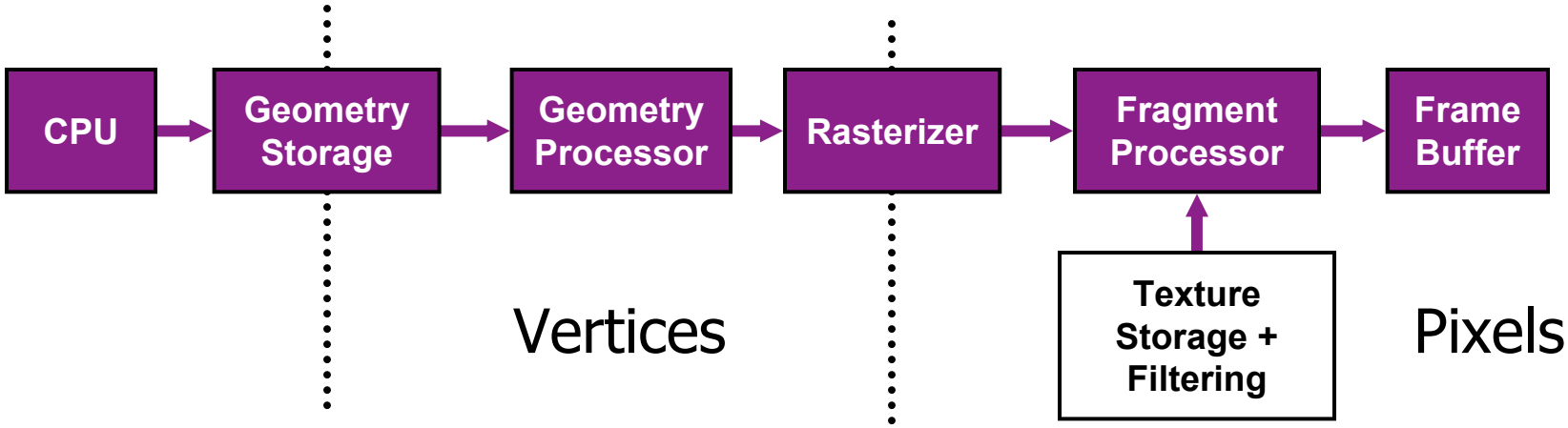
-- Rarely the bottleneck --

- Tuned to the Rasterizer performance

Usual pitfalls

- Too many fragments
- Too much processing per fragment

Texture Limitations



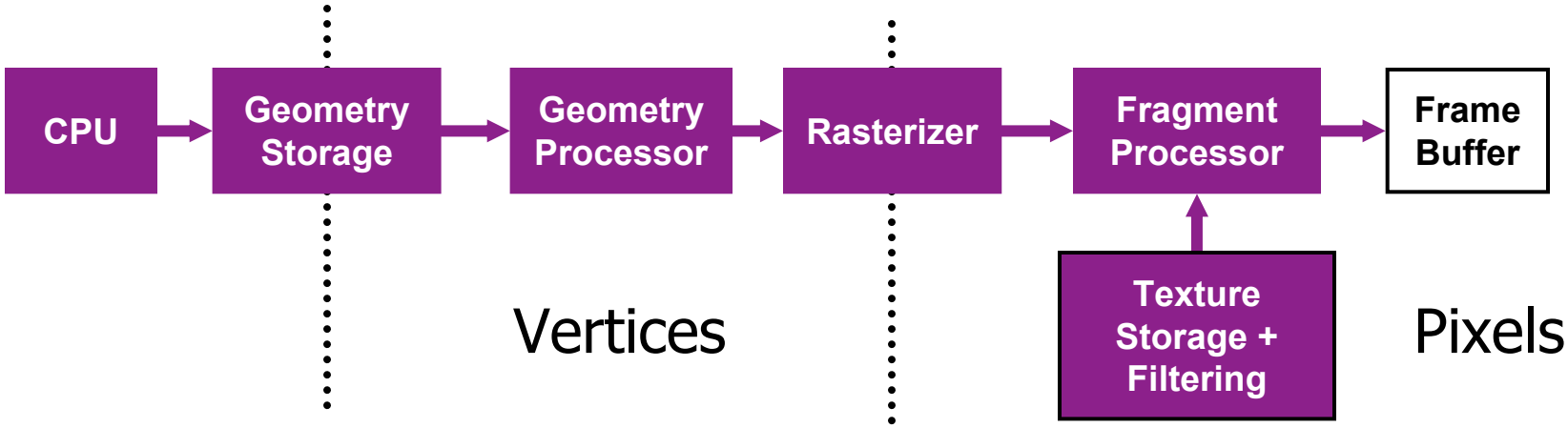
Potential Pitfalls:

- Not enough texture memory
- Excessive texture filtering
- Unbalanced texture resolution
- Memory-hungry internal formats
- No mip-mapping

IMAGEON TIPS:

- Use mipmapping for best performance
- Linear-map-nearest filtering is best mip compromise

Pixel Ops Limitations



Pitfalls:

- **Reading / Writing pixels directly**

Summary: Optimization Process

More than one bottleneck: Iterative process

Start with the problem that have the best ROI, then be systematic

Walk the 3D pipeline; binary search

KNOW THE PLATFORM !!!

Designing Games for Performance

Challenges on Handheld Platforms

No floating point support

- Some computations need be reworked (i.e. physics)
- Fixed point transforms can be used but a game now has to worry about transform accuracy

Limited MIPS

- Software-only solution:
  - Vertex transformation / Rasterizer on CPU
  - Eats into your AI and Physics budgets

Limited system memory size and bandwidth

Challenges on Handheld Platforms

Low CPU to GPU Bandwidth

- Big impact on vertex/poly count which can be sent from CPU to graphics engine
- Data types used to represent pre-transformed vertex and texture coordinate data can mitigate this

Limited Texture / Framebuffer Footprint

- ex: W2300 ships with:
  - 384KBytes of fast on chip SRAM
  - 2 - 8MBytes of dedicated external framebuffer

Reducing the Overall System Workload

Use the W2300 hardware transform engine

- Use fixed point data types for vertices
- Precision issues addressed by using floating point matrices

Reducing the Overall System Workload

Vertex data

- **Vertex buffer objects for reduced bus traffic**
  - **Radical improvements in triangle rate**
- **ATI Extension:**
  - **Modified version of GL_ARB_vertex_buffer_object**
  - **Enable caching of geometry (vertex data and mesh lists) in graphics memory**
- **Vertex data in fixed point, short / byte format**
  - **S & T coordinates can also be in short / byte format**
    - Normalized to [0,1] via matrices
- **Reduces required CPU memory footprint and bus B/W**

Reducing the Overall System Workload

Texturing: Mipmapping

- Higher texture cache utilization
- Improves memory bandwidth efficiency
- Linear-mip-nearest is the “sweet spot”

OpenGL ES

- Use GLbyte and GLshort data types for vertices and texture co-ordinates
- Use the “Common Lite” NOT the “Common” profile

Implement lighting using texturing effects

- Multi pass texturing with alpha blending to achieve multi texturing for sophisticated light effects

Reducing the Overall System Workload

Reduce the number of function calls

- Large triangle batch sizes
- Maximize number of triangles per vertex buffer
  - Amortizes hardware setup and configuration over a larger number of primitives

Avoid redundant or intermediate state settings

- Do not load matrices on each change, only when used

Content Recommendations

High End Cellphone

ARM9 – 400MHz :

- *32bit SDRAM @ 100MHz*
- *240x320 Display*
- *W2300 with 8MB of external framebuffer*

30fps QVGA+ gaming

Highly detailed 3D models

- 8K – 10K polys per frame

Texture Footprint: 4 - 6 MB

Vertex Buffer Object Footprint: 1- 2MB

Main Stream Cellphone

ARM9 – 100-200MHz :

- *16bit/32bit SDRAM @ 75MHz*
- *176x220 -> 240x320 Display*
- *W2302 with 2MB of framebuffer*

30fps QVGA+ gaming

Highly detailed 3D models

- 4K – 5K polys per frame

Texture Footprint: <1.5 MB

Vertex Buffer Object Footprint: 200 – 300KB

Benchmark Case: MotoGP Demo

Microsoft XBOX

ATI IMAGEON™ 2300

Benchmark Case: MotoGP Demo

Benchmark Case: MotoGP Demo

Platform:

- ARM9-based @ 168MHz, 64Mb
- W2300 @ 84MHz
- 8MB framebuffer

Performance:

- 3808Kb of texture
- 7826 triangles / frame avg
- 11862 vertices / frame avg
- Frame Rate: 25 fps avg
- Throughput (avg):
  - ~ 200,000 triangles / sec
  - ~ 300,000 vertices / sec

Benchmark Case: MotoGP Demo

Scene Data:

- 8 bikes: 700 triangles each
- Track: ~ 14,000 triangles

Features Used:

- Mipmapping
- Gouraud Shading
- Alpha Blending
- Single textures
- Fog
- Alpha textures

Benchmark Case: MotoGP Demo

Optimizations Used:

- VBO / Mesh lists
- Mipmapping
- Frustrum culling (CPU)

Summary: Designing Games for Performance

Current solutions

- No floating point support: use fixed point
- Limited MIPS: budget carefully
- Limited system memory size and bandwidth

Reducing the System Workload

- General design-oriented optimizations
- Content recommendations

Benchmark Case

Porting Games

From PC & Consoles

Review: Key Advantages of Hardware 3D

Better visual quality

- Higher polygon count, Texture mip-mapping
- 16bit colors, Alpha textures Antialiasing, Fog

Better gameplay (aka 'Fun Factor'):

- Does transformation and rasterizing in hardware.
- Faster game loop
- No need to spend CPU time on rendering tasks means more advanced game logic, AI, physics etc.

Faster development cycle:

- OpenGL ES support – powerful yet easy to use API.
- Easy to port your existing code to OpenGL ES.

Production pipeline closer to Console and PC.

Choices

Game Content Choices

- User Input, form factor (1 or 2 hands)
- Digital content / storyline / gameplay depth
- Carrier business model
  - Ex: Downloadable content
  - Ex: Multiplayer capabilities

Technical Development Choices

- ***Start from the PC / Console and convert to BREW™***
- Start from the BREW Platform and grow new content

- Many commercial solutions & middleware

Converting to BREW™: Steps

Step 1: Attack on the OpenGL ES front

- Convert to OpenGL
- Graphics API trimming

Step 2: Attack on the CPU front

- General math: Fixed point conversion

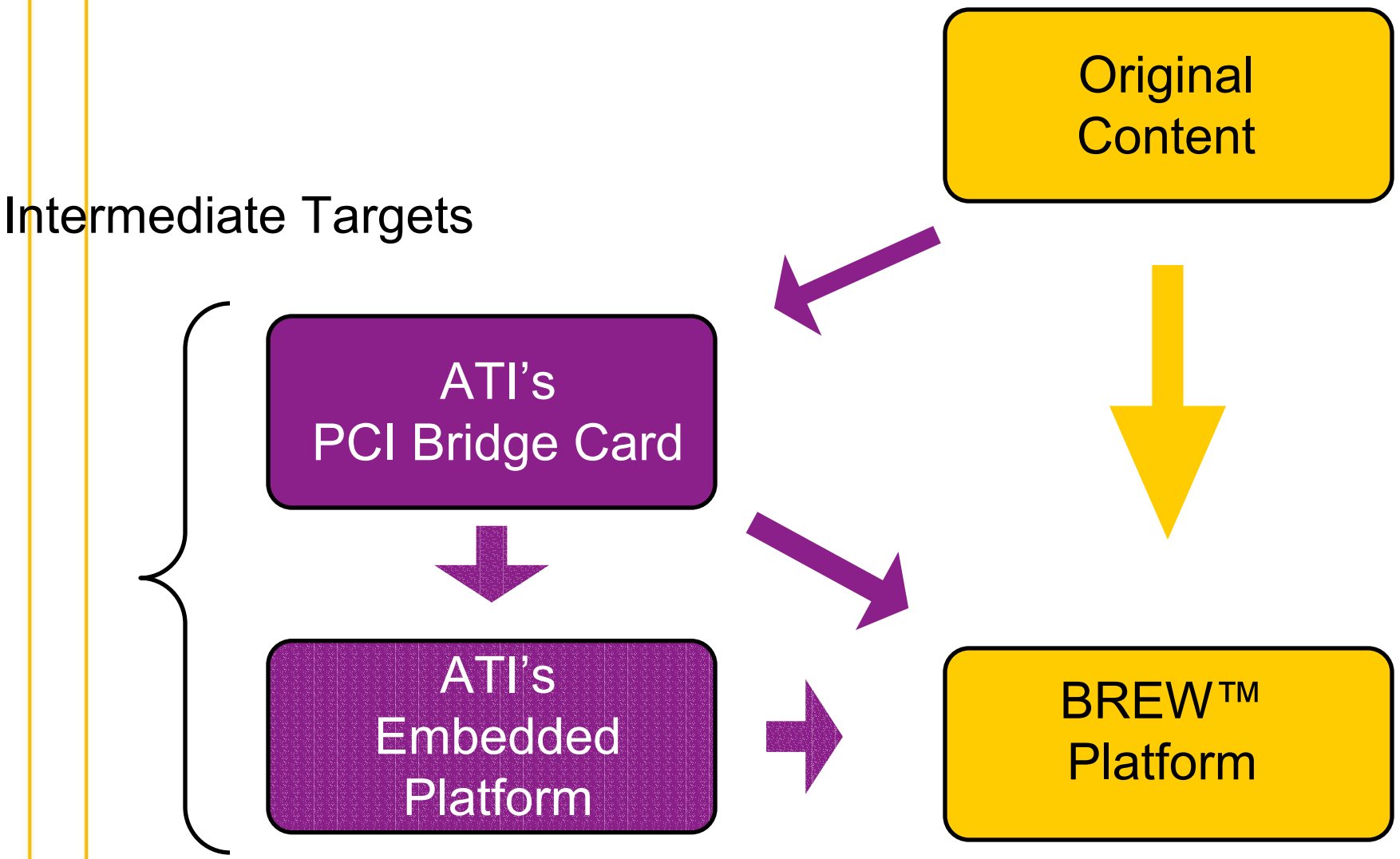
Step 3: Attack on the Memory front

- Trimming down the game assets
- Vertices & texture coords: fixed point conversion
- Texture mipmap & geometry caching

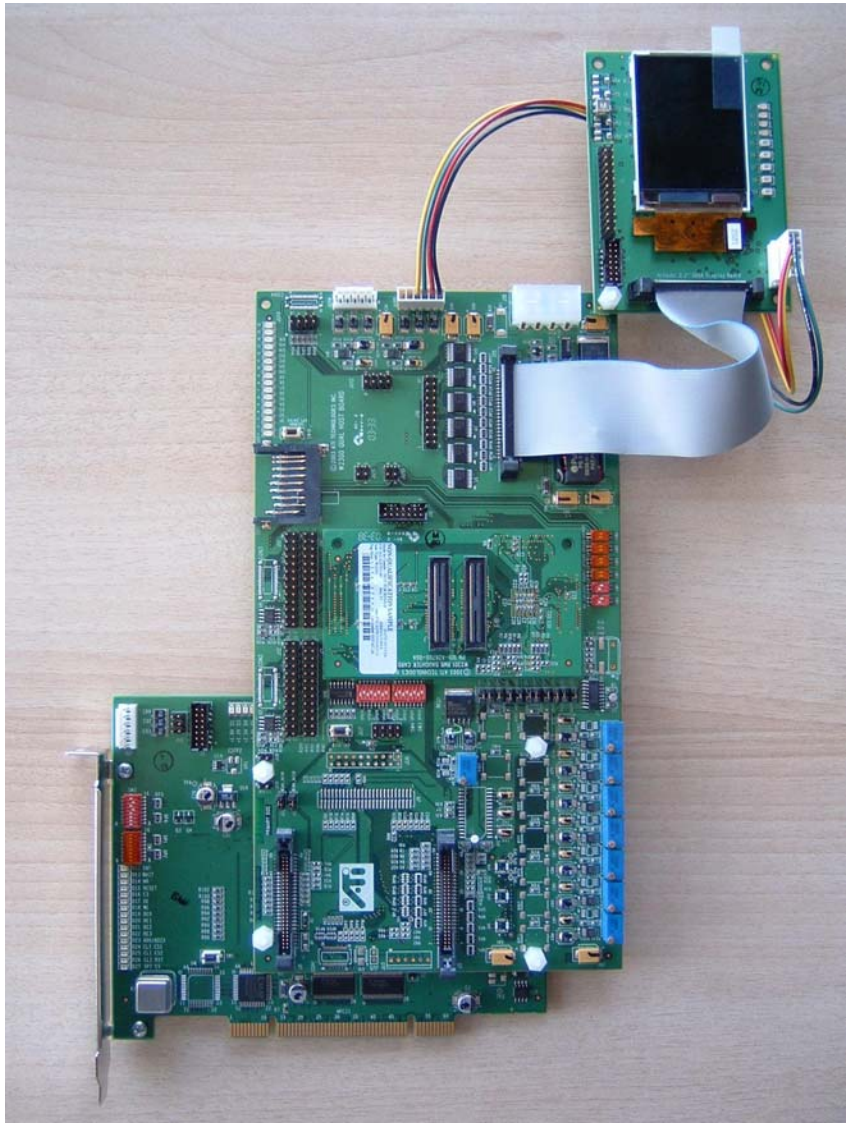
Step 4: Attack on the Platform front

- Convert to the BREW API
- Optimizing for performance

Converting to BREW™: Tools



ATI's PCI Bridge Card



ATI's PCI Bridge Card

Familiar PC development environment

- Game code on PC's CPU, graphics on the IMAGEON
- Windows NT / 2K / XP
- Visual C/C++

Leverages existing toolchain

- In-house tools, 3rd party software
- Significantly reduces development time

Control over the IP's transformation

Summary: PC & Console to BREW™

Choices

- Game content / Gameplay
- Technical

Multi-Step Approach

- Control over IP's transformation
- Risk management

ATI Solutions (Intermediate Targets)

- ATI's PCI Bridge Card
- ATI's Embedded Platform

Presentation Summary

IMAGEON™ 2300: Integrated Architecture

- Comprehensive hardware solution (power, features)
- 3D Floating point pipeline: faster game loop

Optimization Process

- KNOW THE PLATFORM !!!
- Walk the pipeline; iterative process

Designing Games for Performance

- Plan for lack of FPU and for bandwidth limits
- Content recommendation; Benchmark case

Porting Games from PC & Consoles

- Control over the IP's transform; stepwise risk management
- ATI's PCI Bridge card (intermediate target)

QUESTION PERIOD

“3D hardware support such as ATI’s IMAGEON™ chip will allow publishers like THQ to consider bringing high end PC games to mobile devices.”

Stuart Platt
Director of Product Development
THQ Wireless, Inc.
(GDC’04)