SEGA

Dreamcast Developer's Conference

Track 2 / Day 1
March 19, 1999

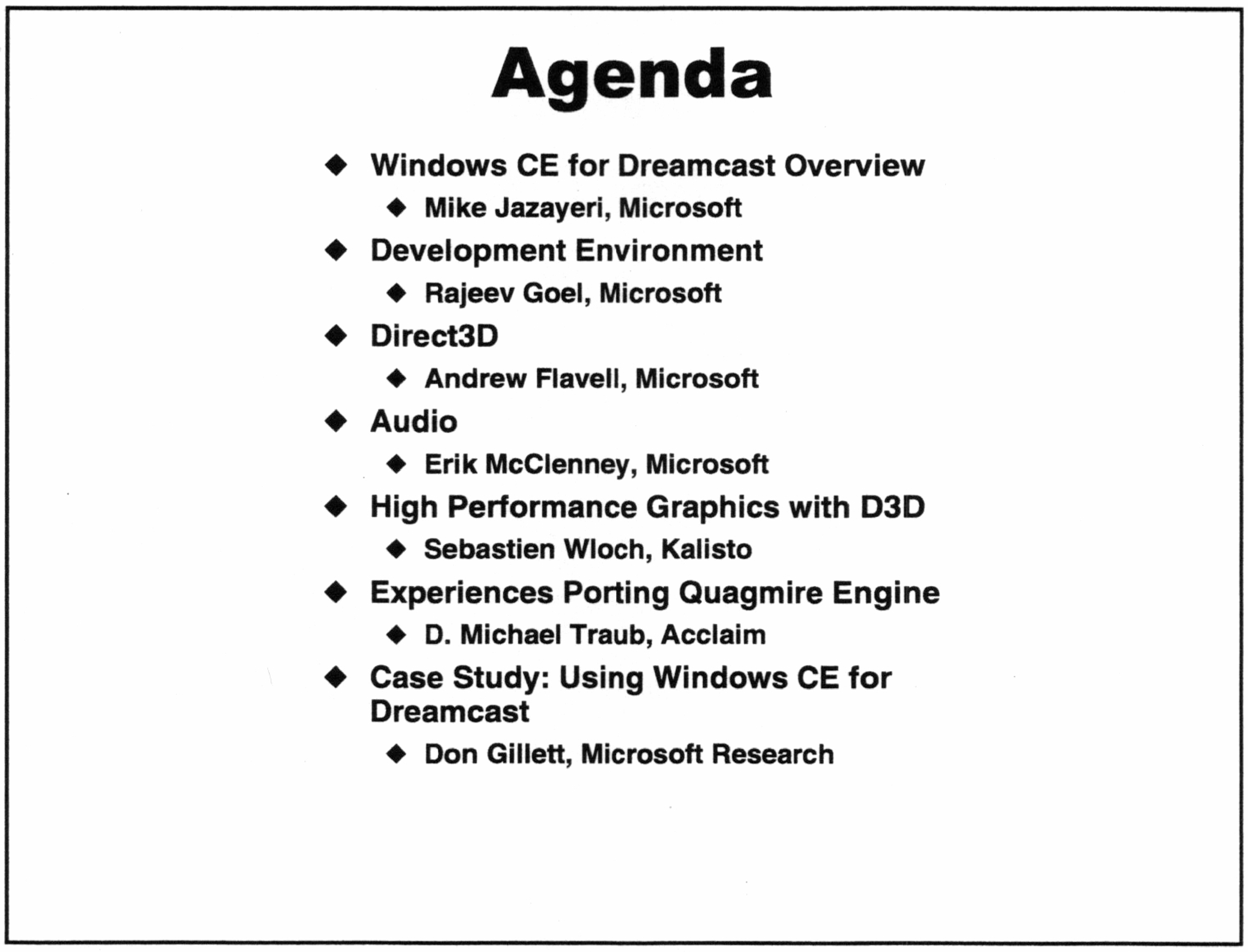
Dreamcast™

SEGA

Windows CE for Dreamcast

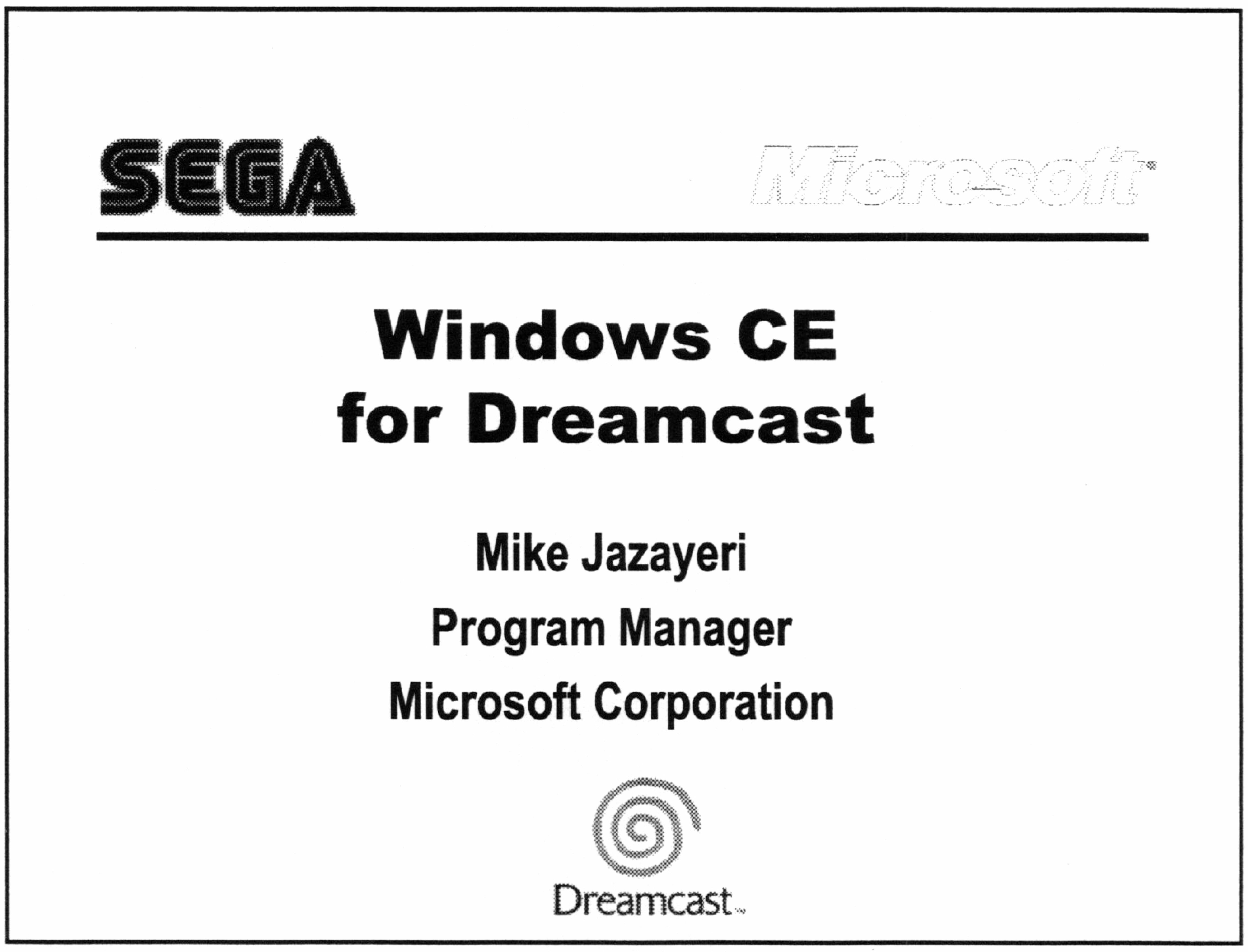
Mike Jazayeri
Program Manager
Microsoft Corporation

Dreamcast™



Agenda

- Windows CE for Dreamcast Overview
  - Mike Jazayeri, Microsoft
- Development Environment
  - Rajeev Goel, Microsoft
- Direct3D
  - Andrew Flavell, Microsoft
- Audio
  - Erik McClenney, Microsoft
- High Performance Graphics with D3D
  - Sebastien Wloch, Kalisto
- Experiences Porting Quagmire Engine
  - D. Michael Traub, Acclaim
- Case Study: Using Windows CE for Dreamcast
  - Don Gillett, Microsoft Research



SEGA Microsoft®

Windows CE for Dreamcast

Mike Jazayeri

Program Manager

Microsoft Corporation

Dreamcast™

What is Windows CE for Dreamcast?

- Optimized OS for Dreamcast platform
  - High performance
  - Light-weight
  - Componentized
- Windows API Compatible
- Win32 and DirectX based
- Internet enabled

Powered by Microsoft Windows CE

Windows API Compatibility

- System supports multithreaded Win32® programming model
  - Familiar .exe/.dll files, processes, threads
- Most of the popular Win32 APIs are supported
  - Sophisticated applications have already been written using this subset
- Applications built with Visual Studio IDE
  - Large base of programmers know how to write Windows CE applications already

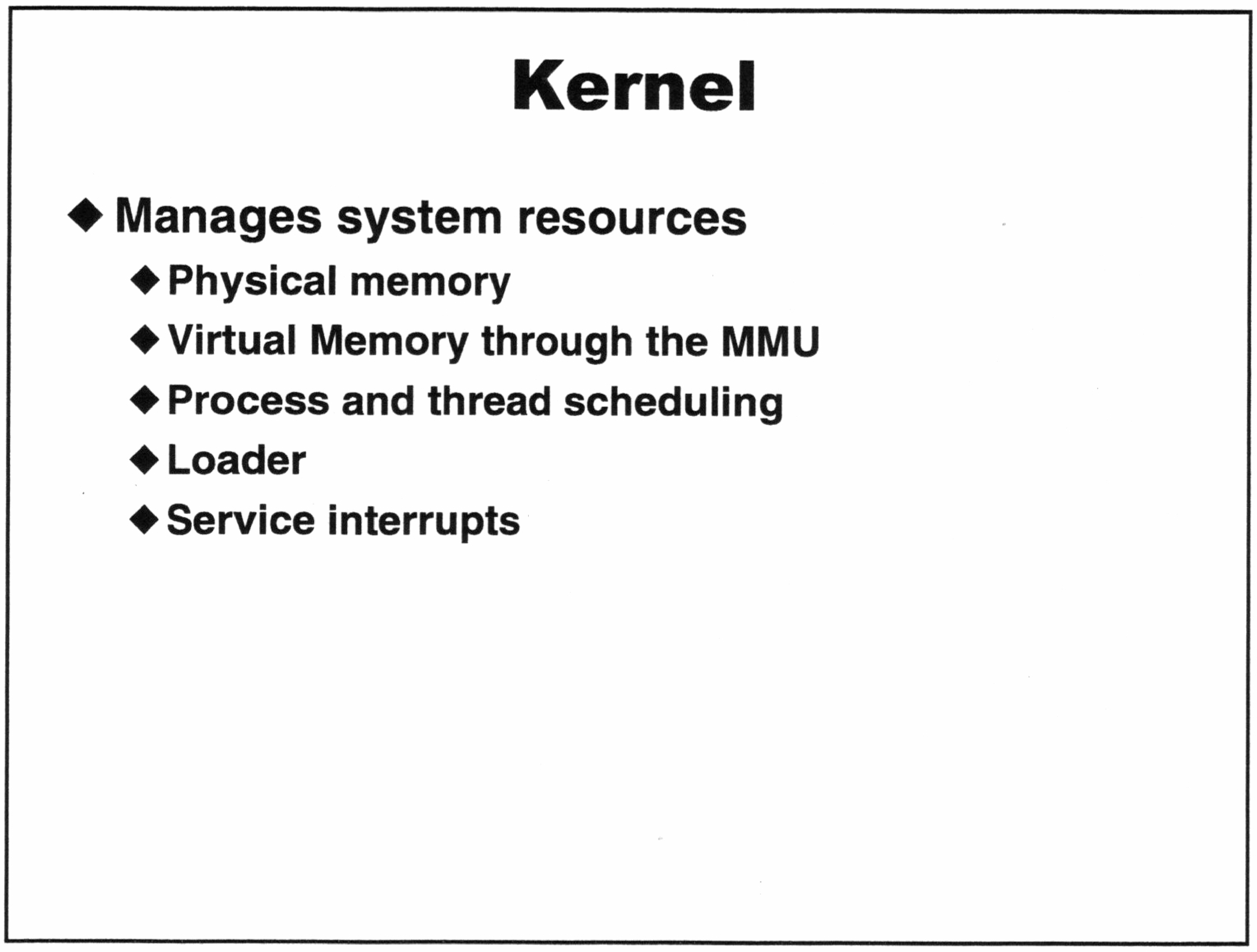
Guiding Design Principles

Small, Fast, Flexible

- Small
  - Provides only APIs required for Games
  - No GUI
  - Tailored implementation for Dreamcast hardware
- Fast
  - Reduced overhead
  - Uses SH4 assembly code for critical loops
  - Uses SH4 and Dreamcast specific features
- Flexible
  - Componentized architecture enables custom configurations

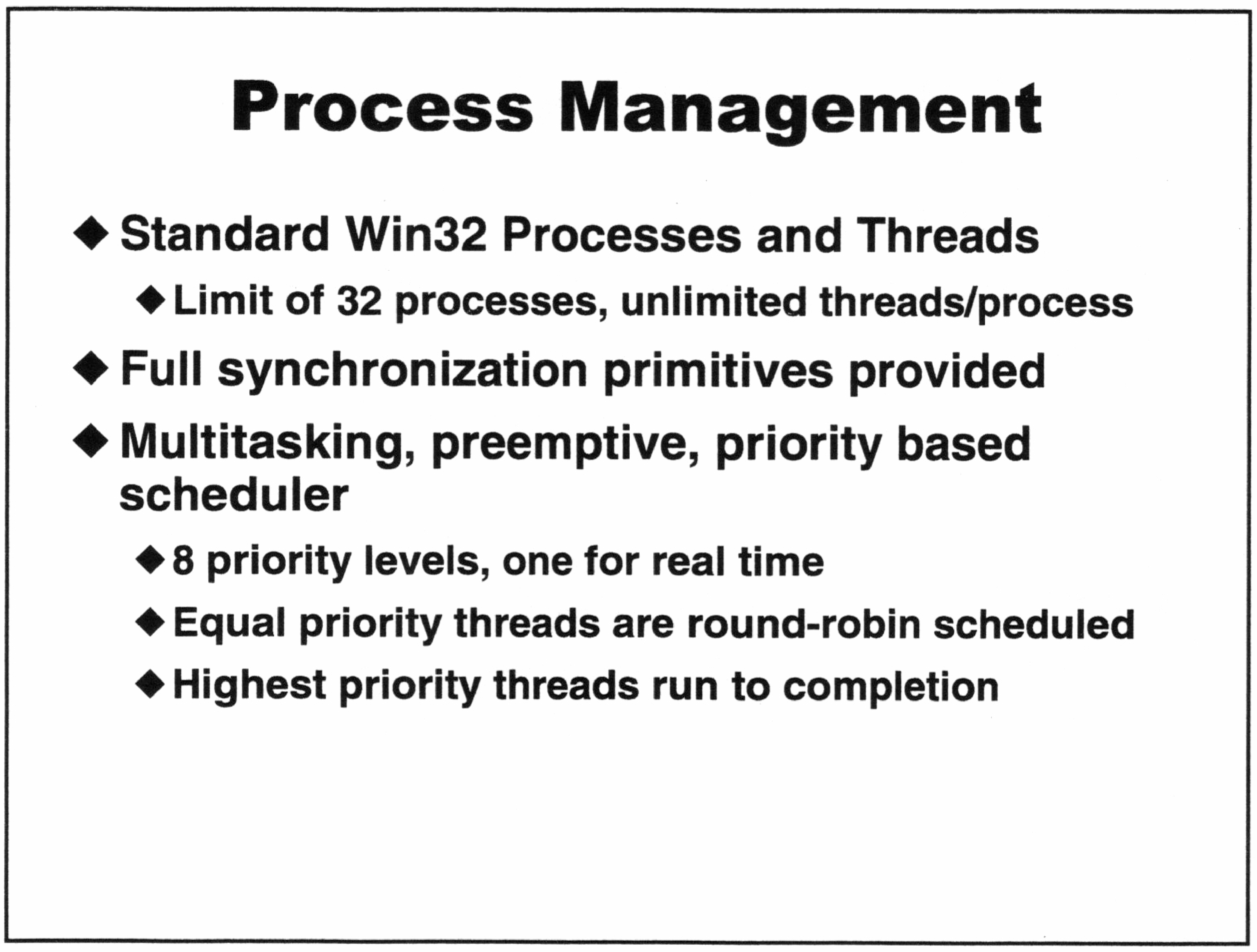
Application Model

- Single Game
  - OS distributed on game CD (*avoids versioning issues*)
- DirectX Full-Screen Exclusive-Mode
- Game provides *ALL* UI
  - No visible windows/controls support
- Game developer chooses components
  - Many of the components are optional
  - Developer are free to mix and match optional components
  - Developers can add custom-built components



Kernel

- ◆ Manages system resources
  - ◆ Physical memory
  - ◆ Virtual Memory through the MMU
  - ◆ Process and thread scheduling
  - ◆ Loader
  - ◆ Service interrupts



Process Management

- ◆ Standard Win32 Processes and Threads
  - ◆ Limit of 32 processes, unlimited threads/process
- ◆ Full synchronization primitives provided
- ◆ Multitasking, preemptive, priority based scheduler
  - ◆ 8 priority levels, one for real time
  - ◆ Equal priority threads are round-robin scheduled
  - ◆ Highest priority threads run to completion

Memory System

- Uses MMU for virtual memory
  - 4KB page size, 64 TLBs
  - Special 64KB and 1MB allocations
- Single virtual address space, shared by all processes.
  - Simplifies kernel and IPC
- Demand paging not supported
  - Except for Memory mapped files

Executable Loading

- Executables *(EXEs and DLLs*) are completely loaded into RAM
- Remain RAM resident throughout usage
- No paging of EXEs/DLLs from CD
  - Paging from CD is too slow and non-deterministic
- DLLs can be loaded explicitly or implicitly

File System

- Provides access to Dreamcast GD-ROM
- Supports:
  - Memory Mapped Files
  - Streaming through use of asynchronous I/O and read-ahead
- Makes maximum use of DMA

Minimal User

User interface APIs

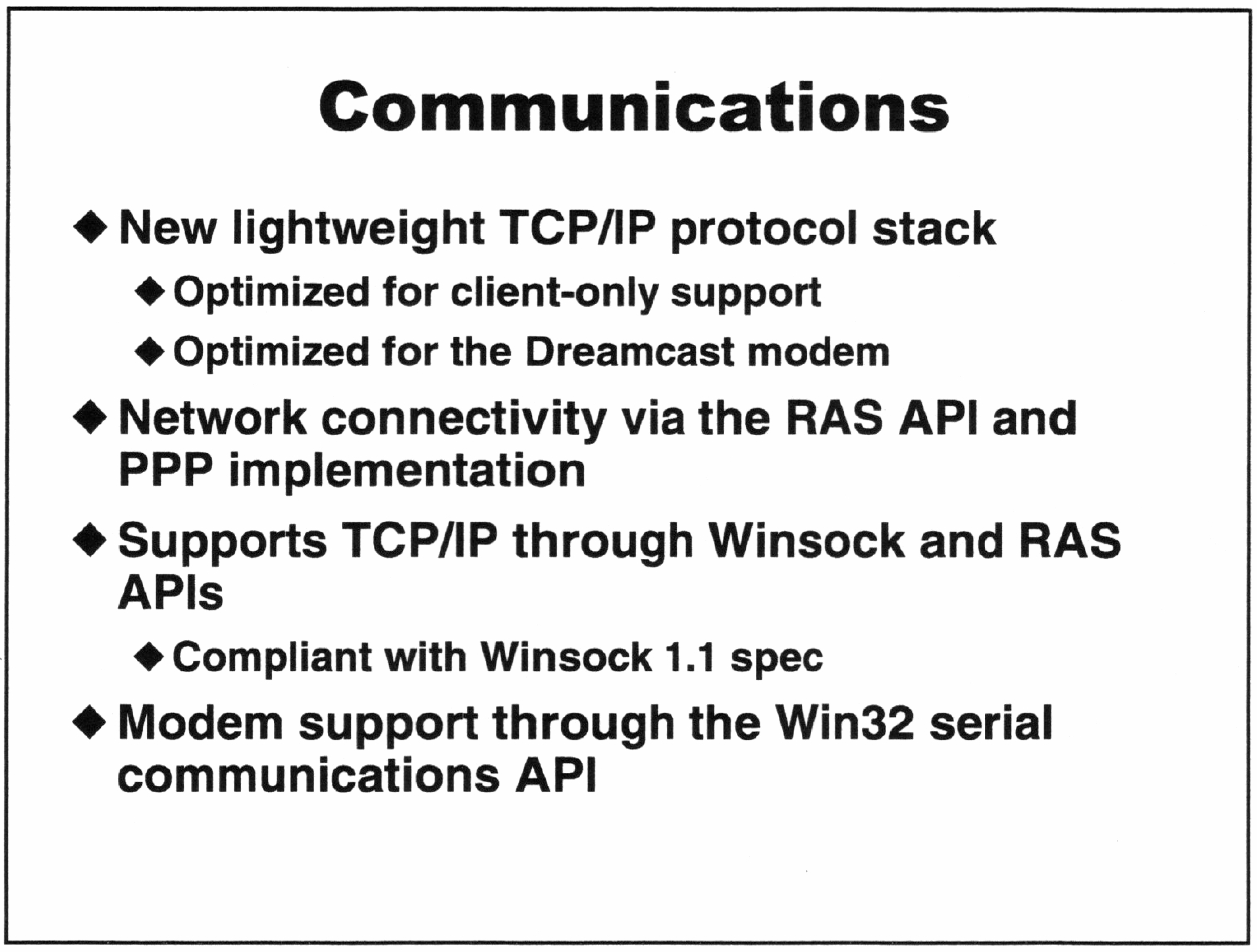
- Handles messages and user input
- Loads resources
- Windows are not visible
  - Serve as targets for messages & input events
  - Game responsible for focus management

GDI: Graphics Device Interface

- Minimal implementation of the desktop GDI
  - Loads fonts
  - Loads bitmaps
  - Copies bitmaps to DirectDraw surfaces
  - Displays text
- Primary graphics libraries are DirectDraw and Direct3D

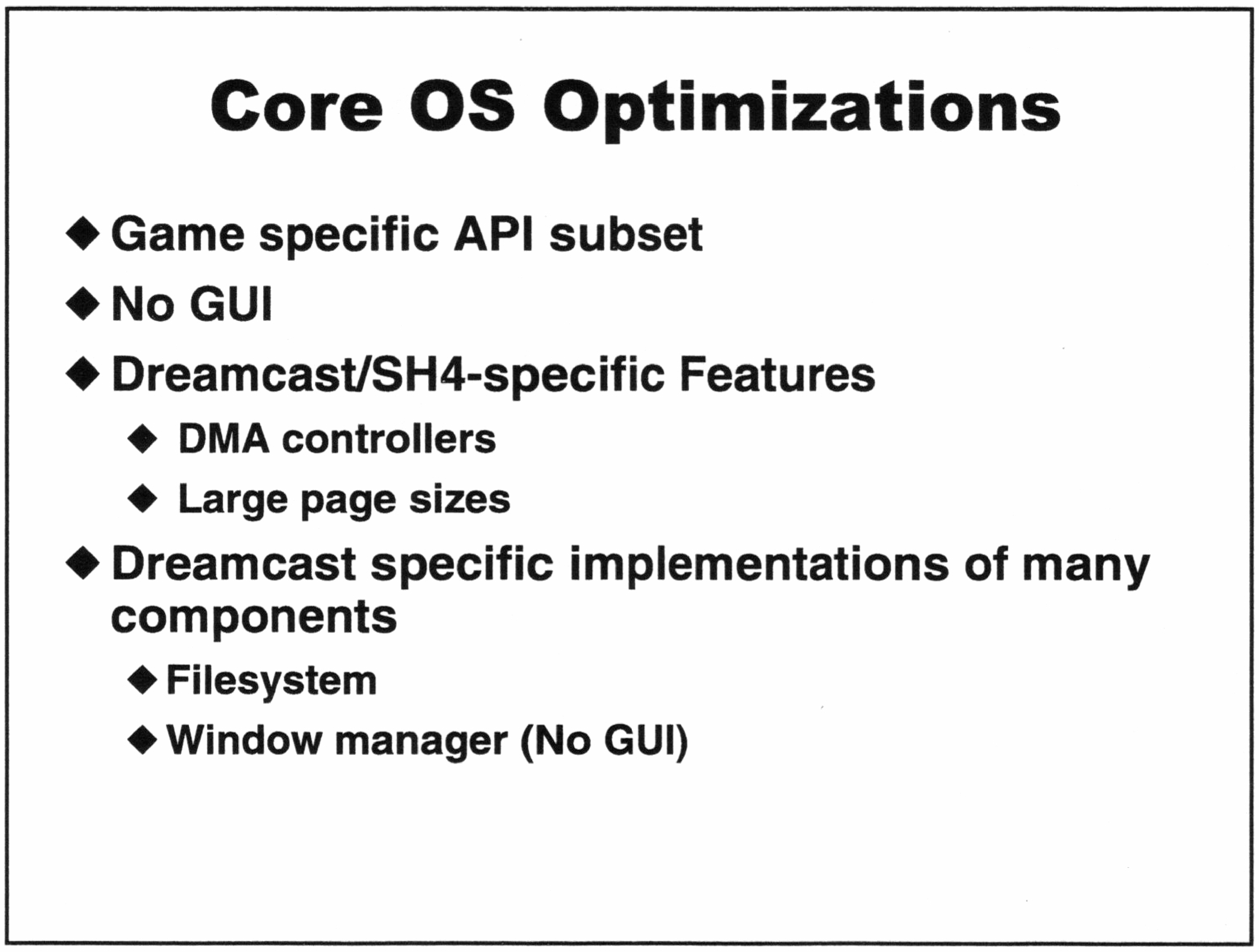
Persistent Storage API

- New API to manage VMS cards
- Block oriented transfers
- Desktop version supplied for emulation



Communications

- ◆ New lightweight TCP/IP protocol stack
  - ◆ Optimized for client-only support
  - ◆ Optimized for the Dreamcast modem
- ◆ Network connectivity via the RAS API and PPP implementation
- ◆ Supports TCP/IP through Winsock and RAS APIs
  - ◆ Compliant with Winsock 1.1 spec
- ◆ Modem support through the Win32 serial communications API



Core OS Optimizations

- ◆ Game specific API subset
- ◆ No GUI
- ◆ Dreamcast/SH4-specific Features
  - ◆ DMA controllers
  - ◆ Large page sizes
- ◆ Dreamcast specific implementations of many components
  - ◆ Filesystem
  - ◆ Window manager (No GUI)

DirectX Components

- DirectDraw
- Direct3D Immediate Mode
- DirectInput
- DirectSound
- DirectPlay
- DirectShow

DirectDraw

- Enables direct manipulation of:
  - Display memory
  - Hardware blitter
  - Hardware overlay support
  - Flipping surface support
- Memory Manager for Direct 3D
- Full-Screen Exclusive-Mode *only*
- Clipper objects are not supported

Direct3D Immediate Mode

- Primary Graphics API for Dreamcast
- Drawing interface for 3D hardware
- Transforms, Lights, and Renders Polygons

DirectInput

- Primary input for Dreamcast
- Manages game controller input
- Manages devices connected to controllers
- Currently released peripherals supported
    - Game pad, Wheel,
- Future devices will be supported
    - Vibration pack, Fishing Pole, Light Gun, etc.

DirectSound

- ◆ Plays and captures digitized audio
- ◆ Interfaces with Sega ARM code to manage Dreamcast sound memory
- ◆ Hardware mixing only
- ◆ 3D Sound implemented through QSound in Sega's DSP code

DirectPlay

- ◆ Simplifies application access to communication services
    - ◆ Applications communicate without knowledge of underlying transport, protocol, or online service
- ◆ Transport Independent Gaming API
    - ◆ Write game independent of network-specific details
- ◆ Provides APIs to:
    - ◆ Send/Receive messages to players, groups, etc. -
    - ◆ Interact with matchmaking lobbies
    - ◆ Chat with other players

DirectShow

- Digital Audio/Video playback and synchronization
- Primarily for cut scenes, intros, etc.
- Game has control of media presentation without knowing details of source file format
- Allows game to render video data onto any DirectDraw surface (e.g. a texture) and control the display of that surface

DirectX Optimizations

- Removed ALL Parameter validations
  - Debug versions are supplied that include parameter validation
- Use SH4 specific features
  - Vector/matrix instructions
  - 1/Sqrt()
  - Sin, Cos approximate
  - Store Queue
- Critical loops coded in SH4 Assembler

Tools Components

- Visual C++
  - VC design environment: provides host for integrated SH4 compiler, linker, and remote debugging support
- Windows CE Toolkit for Visual C++
  - Provides SH4-specific tool set
- WinCE for Dreamcast SDK
  - Provides libs, headers, runtimes, samples & docs for building Dreamcast games
  - Version 1.0 now available!

SH4 Compiler Features

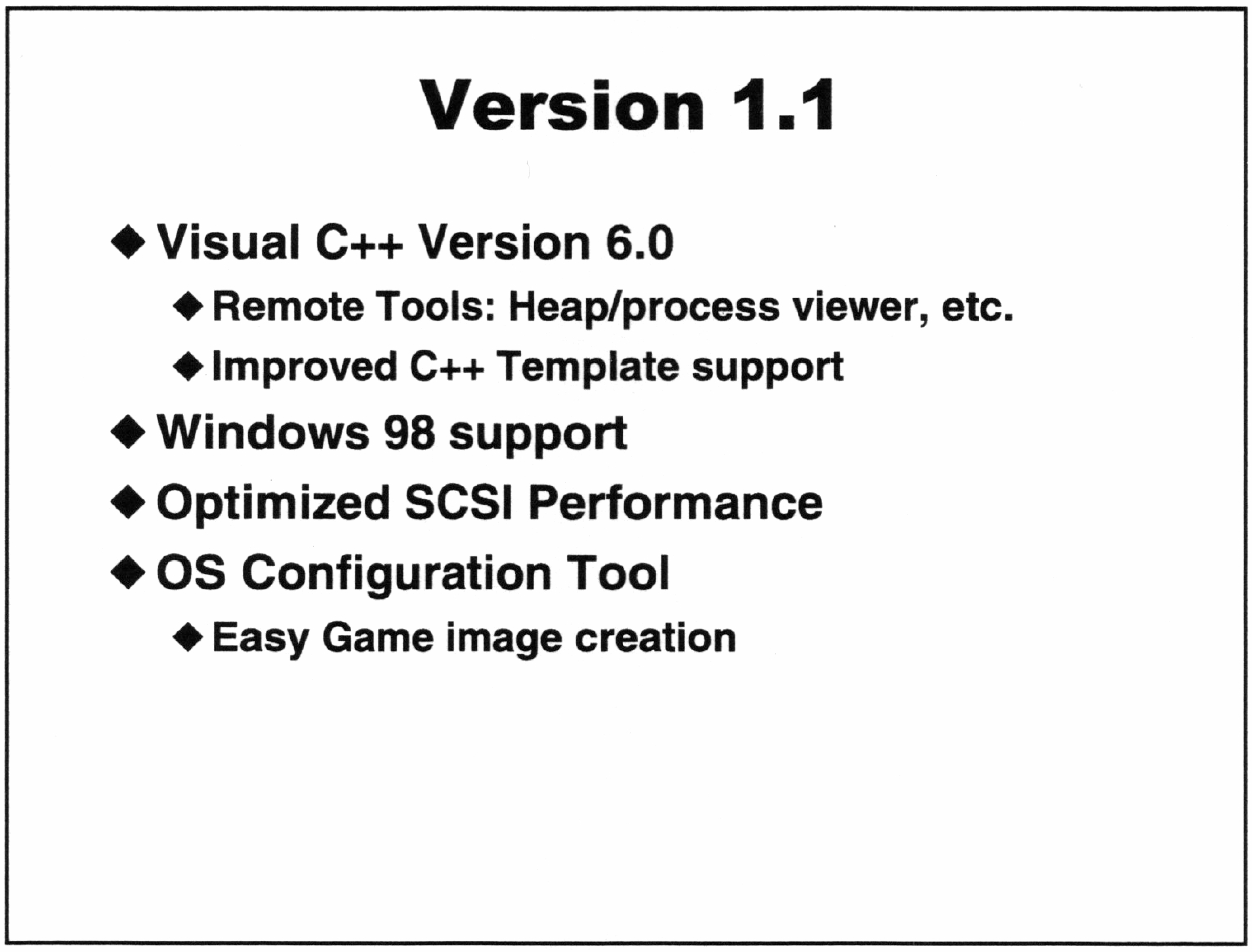
- Builds on extensive Microsoft/Hitachi experience with SHx family on Windows CE
- Integrated as package with Visual C++ IDE
  - Compile & debug within IDE
- Optimized
  - Exploits SH4 features (e.g., dual instruction pipeline, floating point)
  - Intrinsics to take advantage of graphics optimizations
- In-line assembler support

Debugging Support

- ◆ Integrated Visual C++ debugger for applications
  - ◆ Supports remote debugging of Dreamcast
- ◆ Debug APIs exposed to support 3rd party tools
- ◆ WinDbg for kernel-level debugging
  - ◆ Required for driver-level development

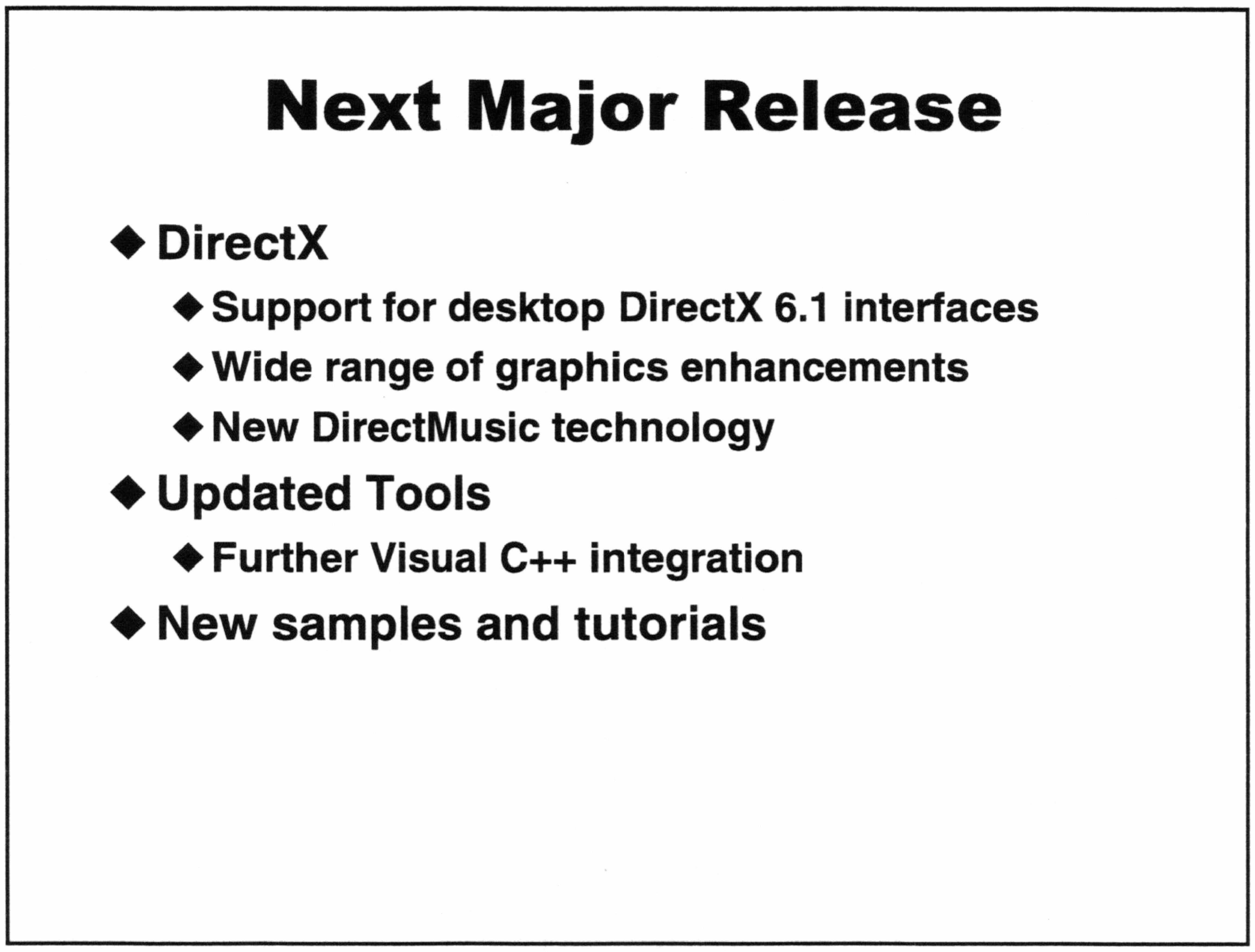
Development Environment

- ◆ Development platform is Windows NT4.0 w/SP3
- ◆ Target execution platforms
  - ◆ Win9x PC *(Win9x emulation platform)*
    - ◆ requires h/w support for Direct3D and DirectSound
  - ◆ Dreamcast Development System
    - ◆ requires WinNT for debugging via Visual C++, CD-ROM emulation, and application development



Version 1.1

- ◆ Visual C++ Version 6.0
  - ◆ Remote Tools: Heap/process viewer, etc.
  - ◆ Improved C++ Template support
- ◆ Windows 98 support
- ◆ Optimized SCSI Performance
- ◆ OS Configuration Tool
  - ◆ Easy Game image creation



Next Major Release

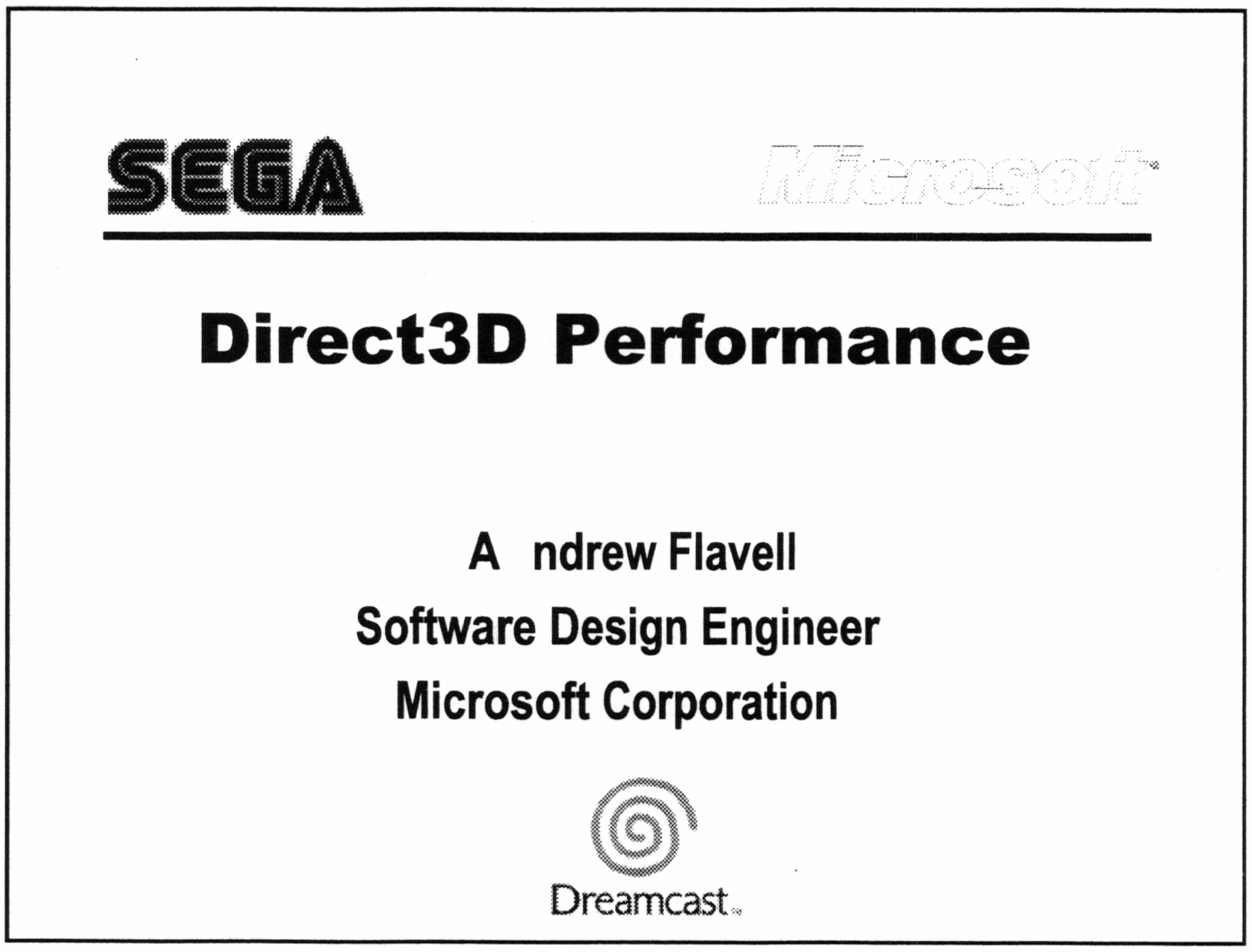
- ◆ DirectX
  - ◆ Support for desktop DirectX 6.1 interfaces
  - ◆ Wide range of graphics enhancements
  - ◆ New DirectMusic technology
- ◆ Updated Tools
  - ◆ Further Visual C++ integration
- ◆ New samples and tutorials

SEGA

Direct3D Performance

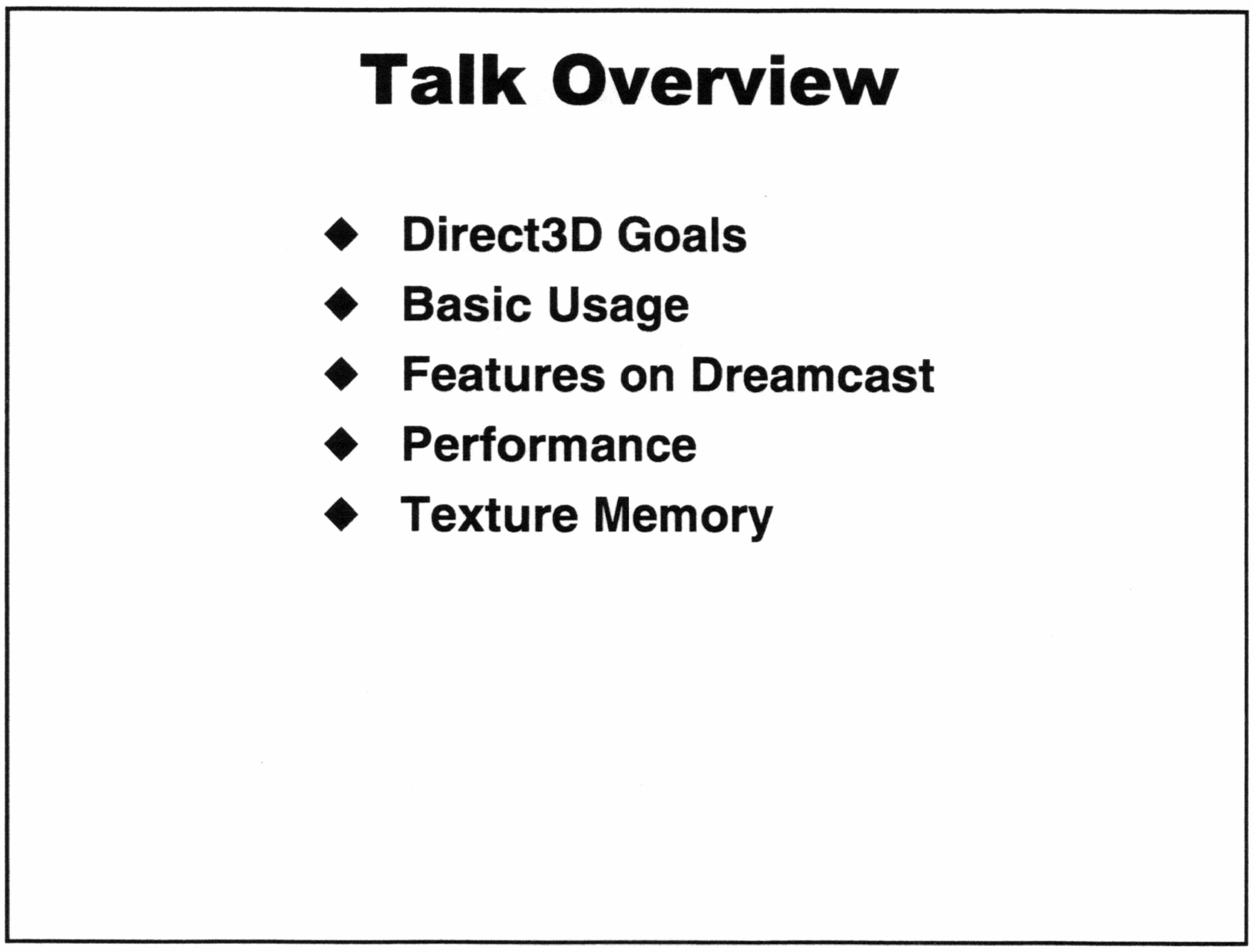
Andrew Flavell
Software Design Engineer
Microsoft Corporation

Dreamcast™



Direct3D Performance

A ndrew Flavell
Software Design Engineer
Microsoft Corporation



Talk Overview

- Direct3D Goals
- Basic Usage
- Features on Dreamcast
- Performance
- Texture Memory

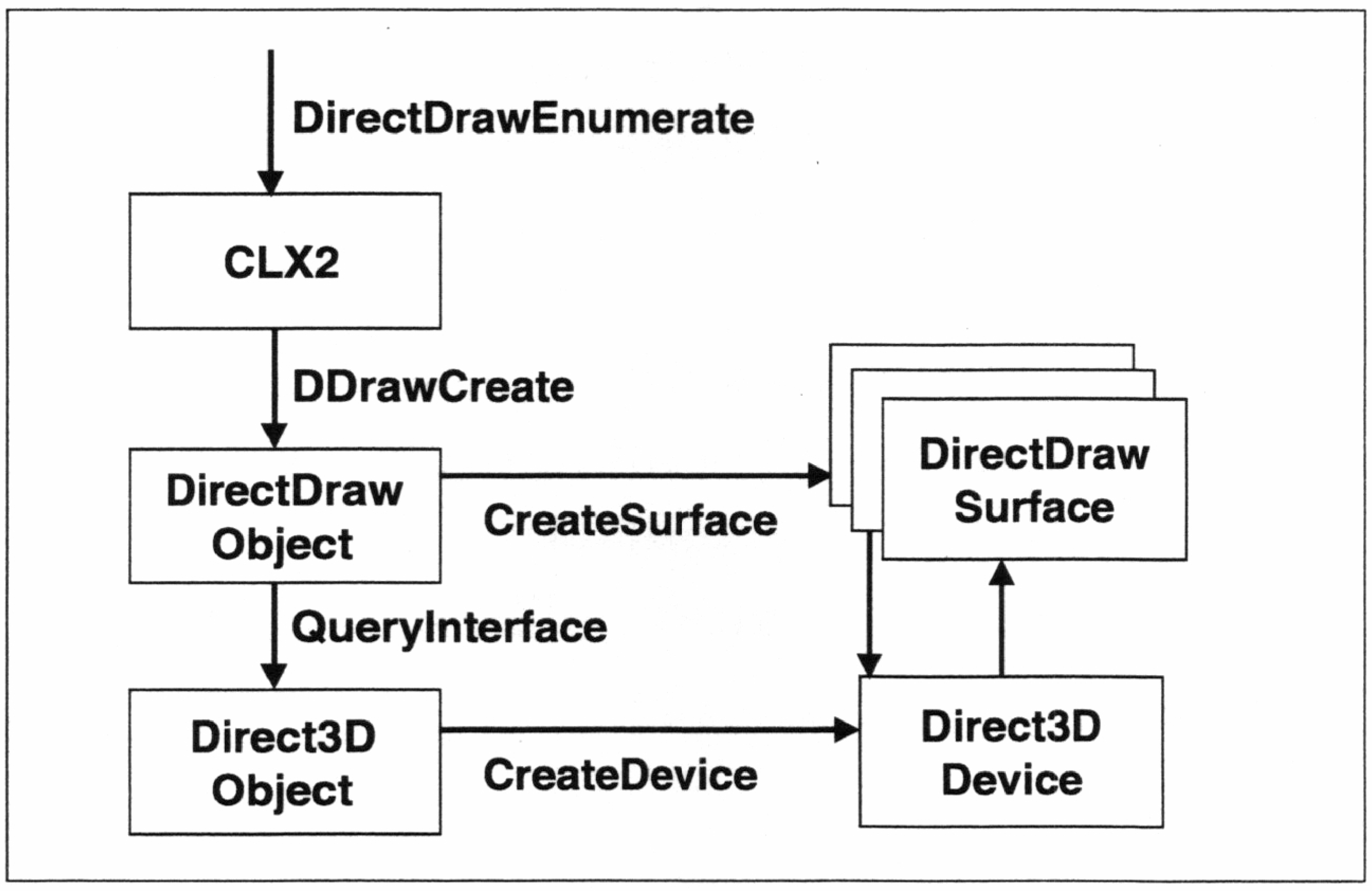
Direct3D Goals

- Dreamcast/PC compatibility
- Provide maximum performance
  - Stay out of the way !
- Highlight hardware features
- Provide high quality development environment
- Provide programmers maximum freedom

Direct3D Basic Usage

- Create DirectDraw & D3D Objects
- Create and load textures
- Main Loop
  - Clear, and BeginScene
  - For each object (mesh):
    - Set Render-states (minimize these)
    - DrawPrimitive / DrawIndexedPrimitive
  - EndScene, and Flip
  - Maintain textures
  - Perform input & game processing

Direct3D Object Model



DirectDrawEnumerate

CLX2

DDrawCreate

DirectDraw Object

CreateSurface

DirectDraw Surface

QueryInterface

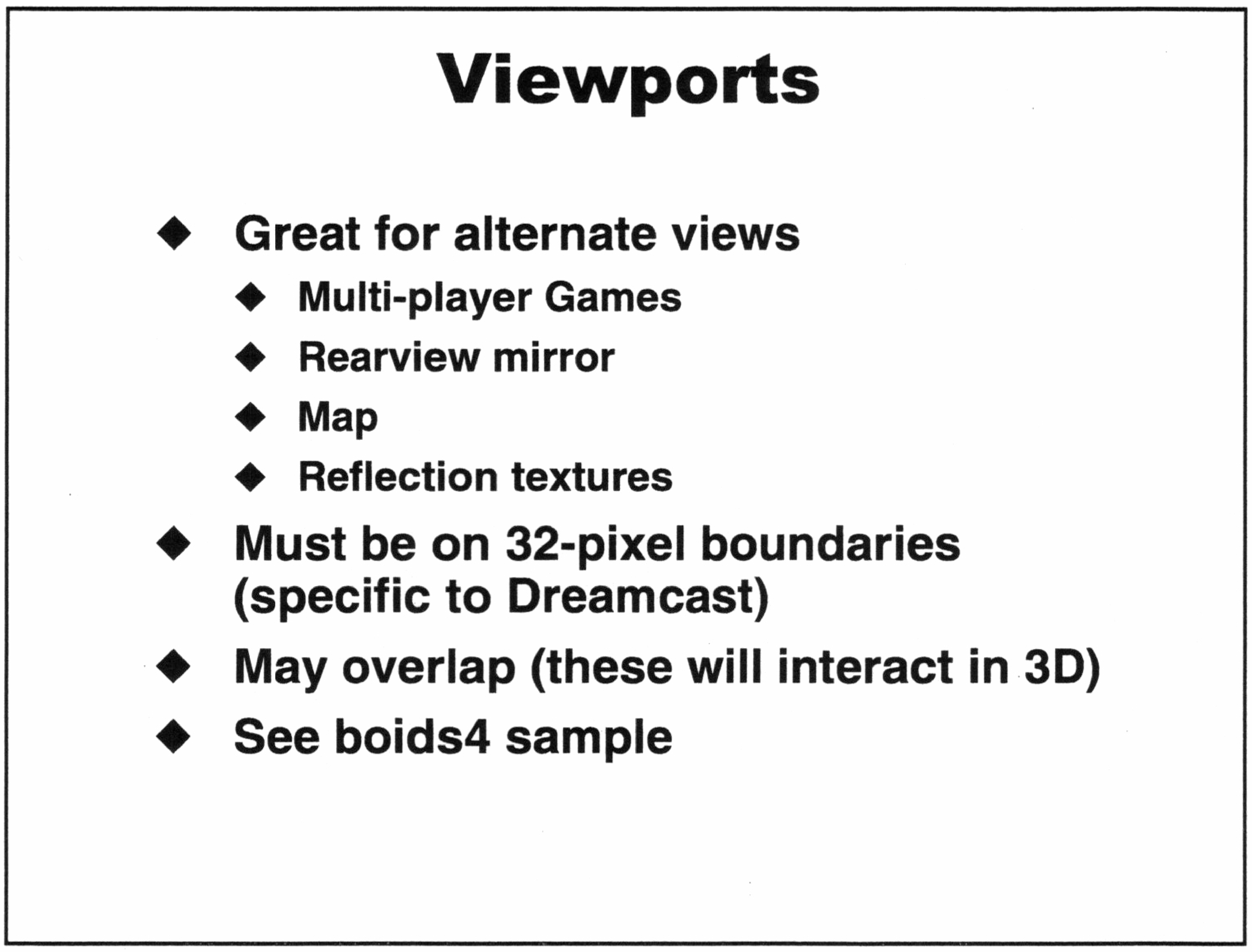
Direct3D Object

CreateDevice

Direct3D Device

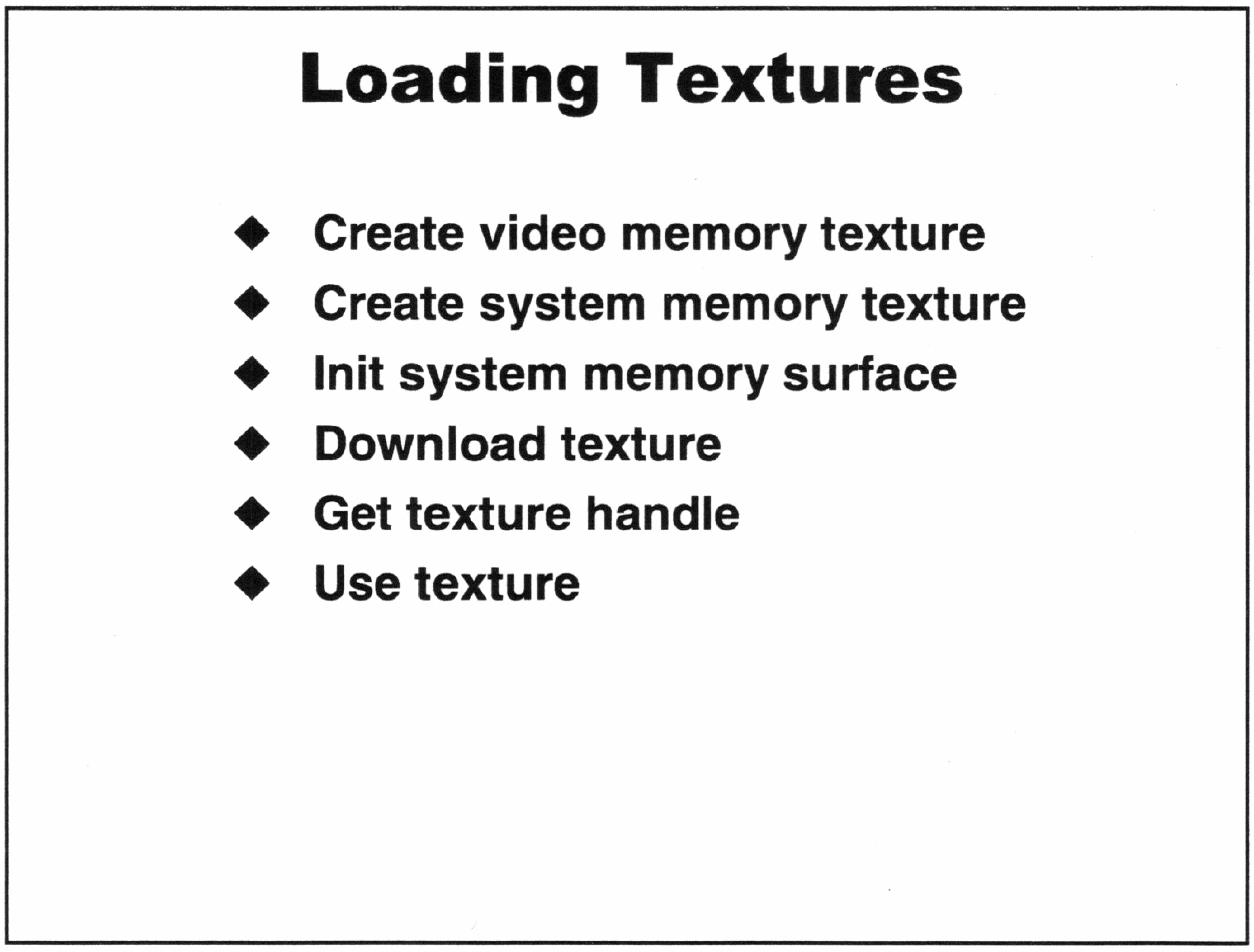
Create 3D Device & Buffers

- Use CreateDevice
  - Instead of QueryInterface
  - Only HAL GUID defined
  - You don't need to use EnumDevices
    - suggested for desktop compatibility
- CreateDevice returns a IDirect3DDevice2
- Create Front buffer
- Create Back buffer
- Z buffer - Dummy on Dreamcast. Necessary for compatibility only



Viewports

- Great for alternate views
  - Multi-player Games
  - Rearview mirror
  - Map
  - Reflection textures
- Must be on 32-pixel boundaries (specific to Dreamcast)
- May overlap (these will interact in 3D)
- See boids4 sample



Loading Textures

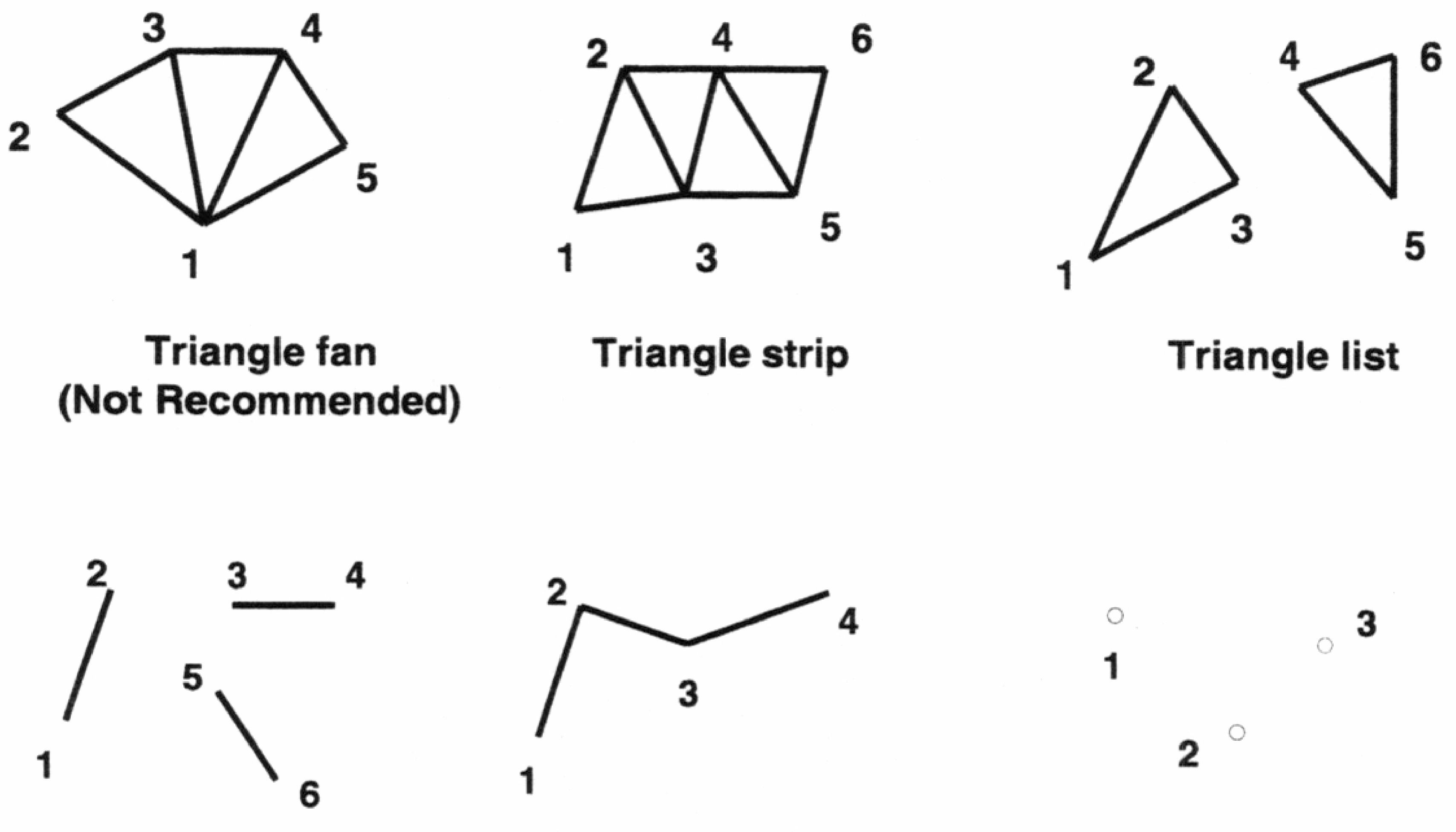
- Create video memory texture
- Create system memory texture
- Init system memory surface
- Download texture
- Get texture handle
- Use texture

State Control

- SetRenderState()
- SetLightState()
- SetTransform()/MultiplyTransform()
- GetXXX() versions of all of the above

Note: states are persistent between frames

Supported Primitives



Triangle fan (Not Recommended)

Triangle strip

Triangle list

Line list

Line strip

Point list

Direct3D Vertex Types

- D3DVERTEX

x	y	z	Nx	Ny	Nz	u	v

- D3DLVERTEX

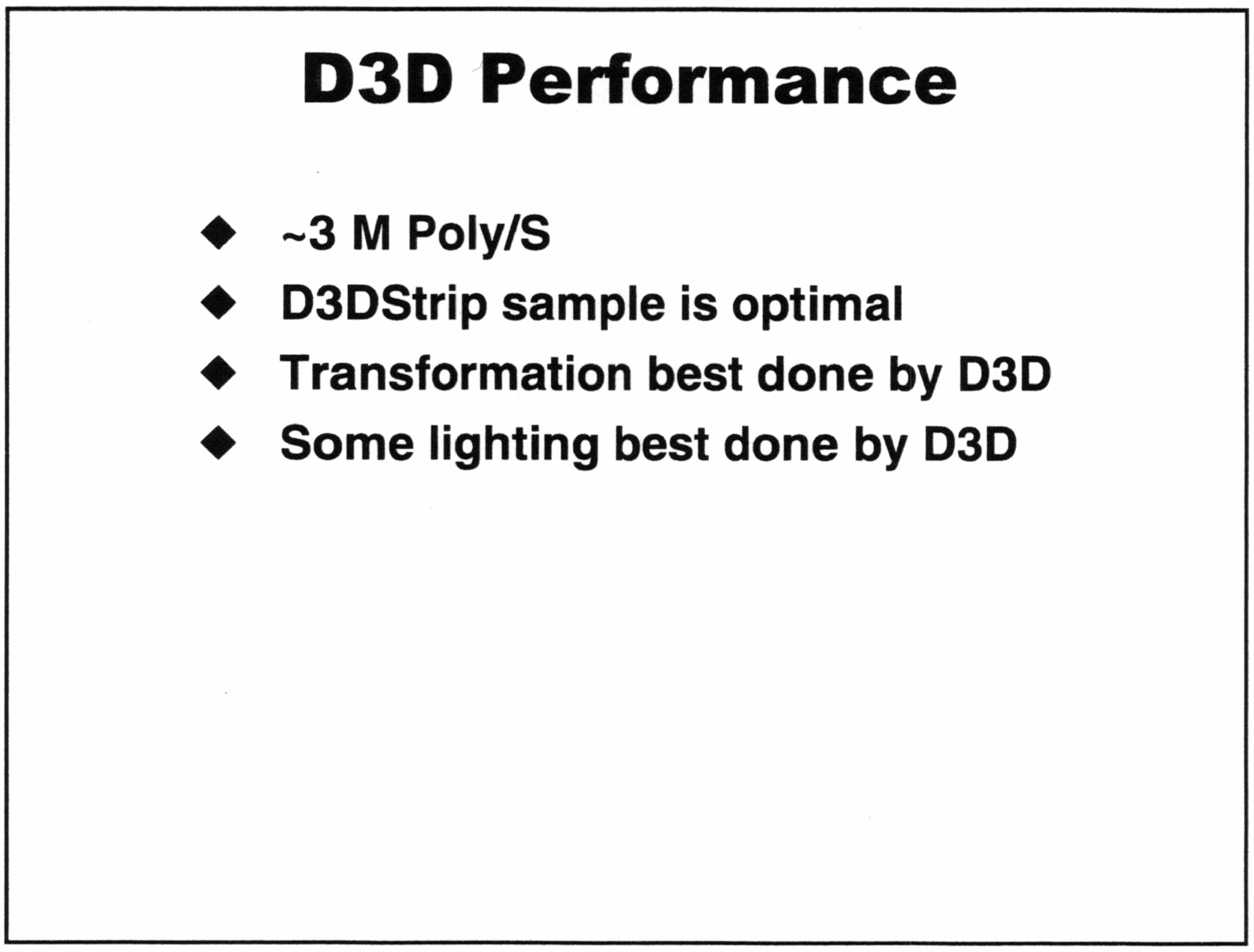
x	y	z	▨	RGBA	RGBF	u	v

- D3DTLVERTEX

sx	sy	sz	rhw	RGBA	RGBF	u	v

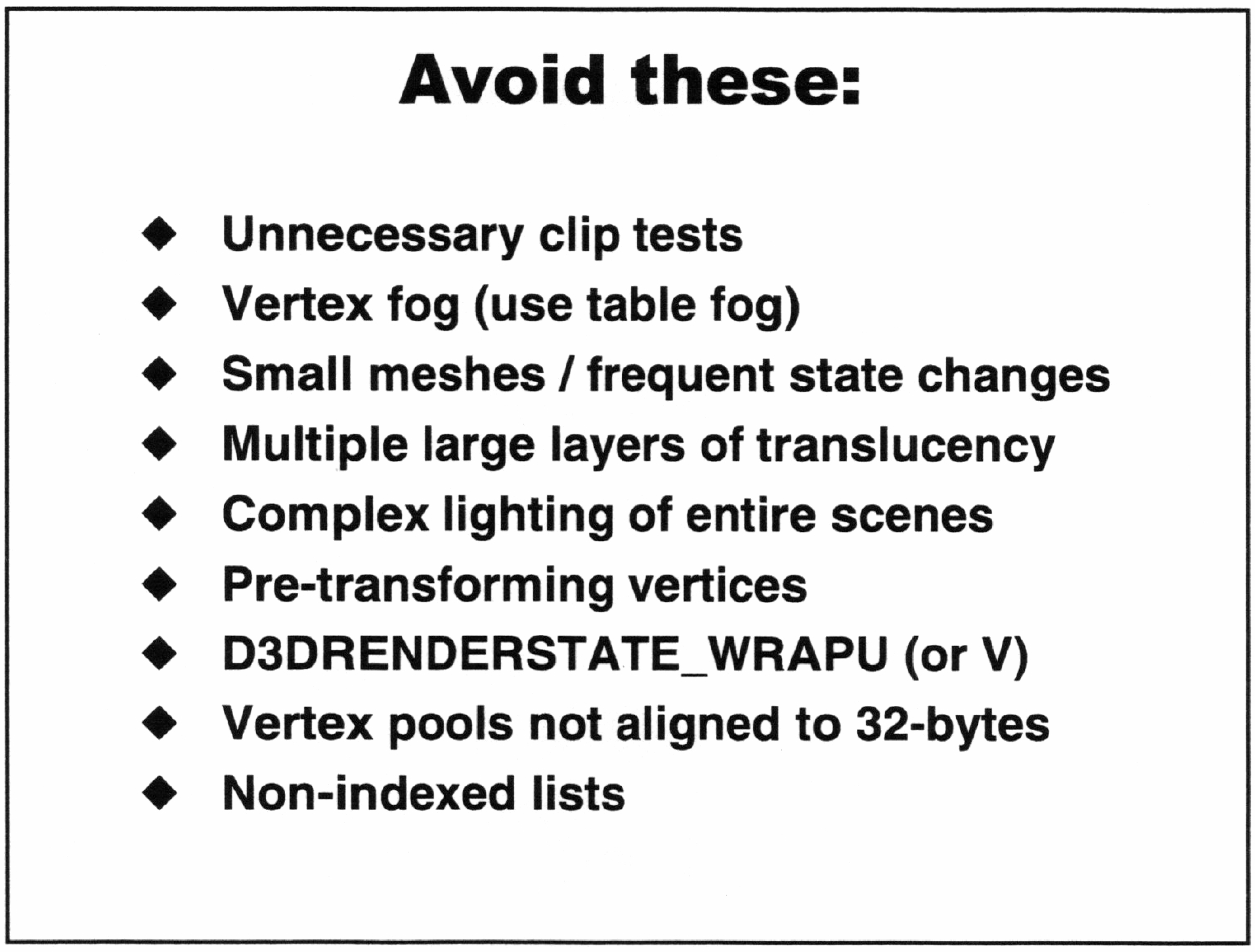
Features on Dreamcast

- Tile-by-tile, each pixel drawn once
- Pixel-accurate translucent sorting
- Punch through pass
- Bump mapping
- Tri-linear MipMapping & Table fog
- Hardware clipping to viewport
- Strip support
- SH4 native matrix operations
- Store Queue to bypass cache



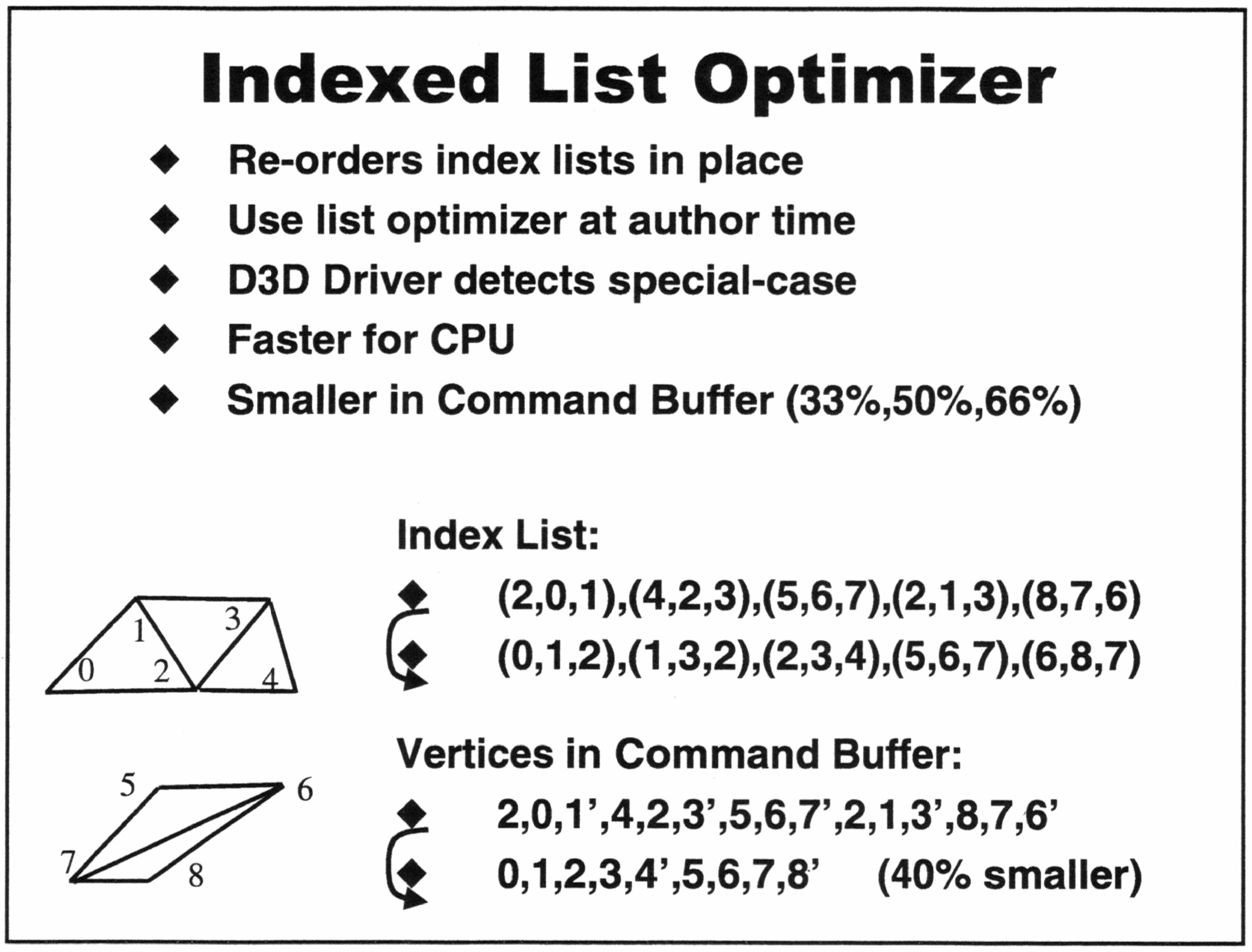
D3D Performance

- ~3 M Poly/S
- D3DStrip sample is optimal
- Transformation best done by D3D
- Some lighting best done by D3D



Avoid these:

- Unnecessary clip tests
- Vertex fog (use table fog)
- Small meshes / frequent state changes
- Multiple large layers of translucency
- Complex lighting of entire scenes
- Pre-transforming vertices
- D3DRENDERSTATE_WRAPU (or V)
- Vertex pools not aligned to 32-bytes
- Non-indexed lists



Indexed List Optimizer

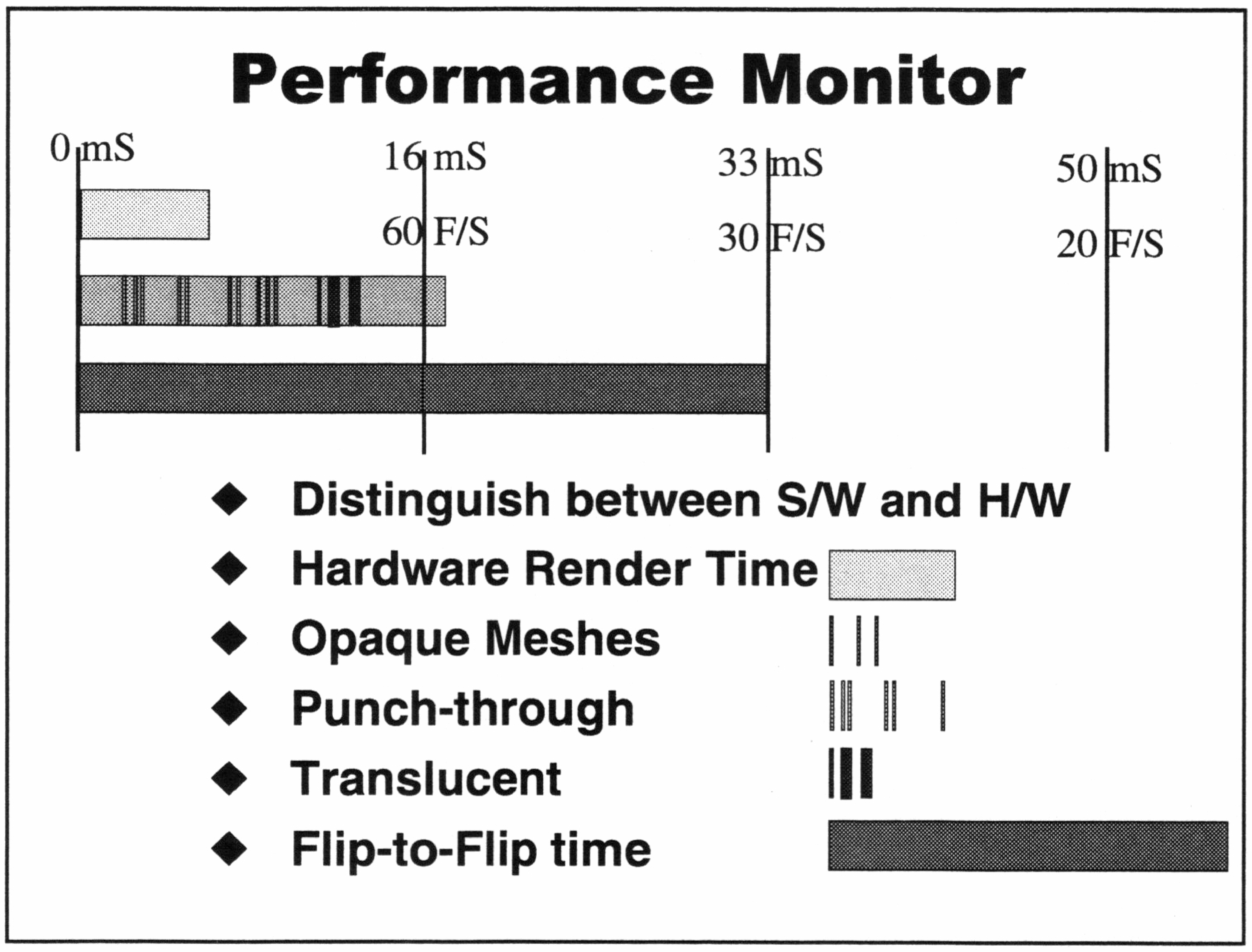
- Re-orders index lists in place
- Use list optimizer at author time
- D3D Driver detects special-case
- Faster for CPU
- Smaller in Command Buffer (33%,50%,66%)

Index List:

- (2,0,1),(4,2,3),(5,6,7),(2,1,3),(8,7,6)
- (0,1,2),(1,3,2),(2,3,4),(5,6,7),(6,8,7)

Vertices in Command Buffer:

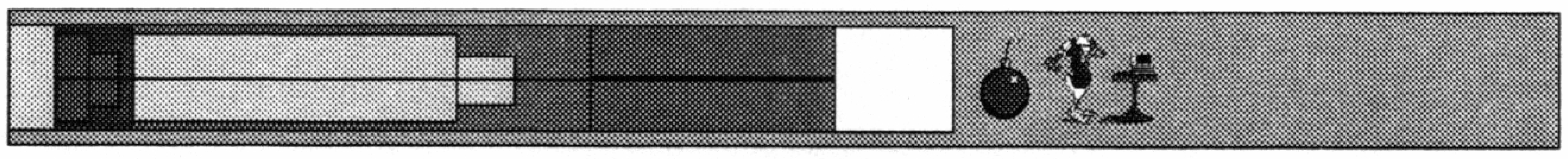
- 2,0,1’,4,2,3’,5,6,7’,2,1,3’,8,7,6’
- 0,1,2,3,4’,5,6,7,8’ (40% smaller)



Performance Monitor

- Distinguish between S/W and H/W
- Hardware Render Time ░
- Opaque Meshes | ||
- Punch-through ||| || |
- Translucent ▌▌▌
- Flip-to-Flip time ▓

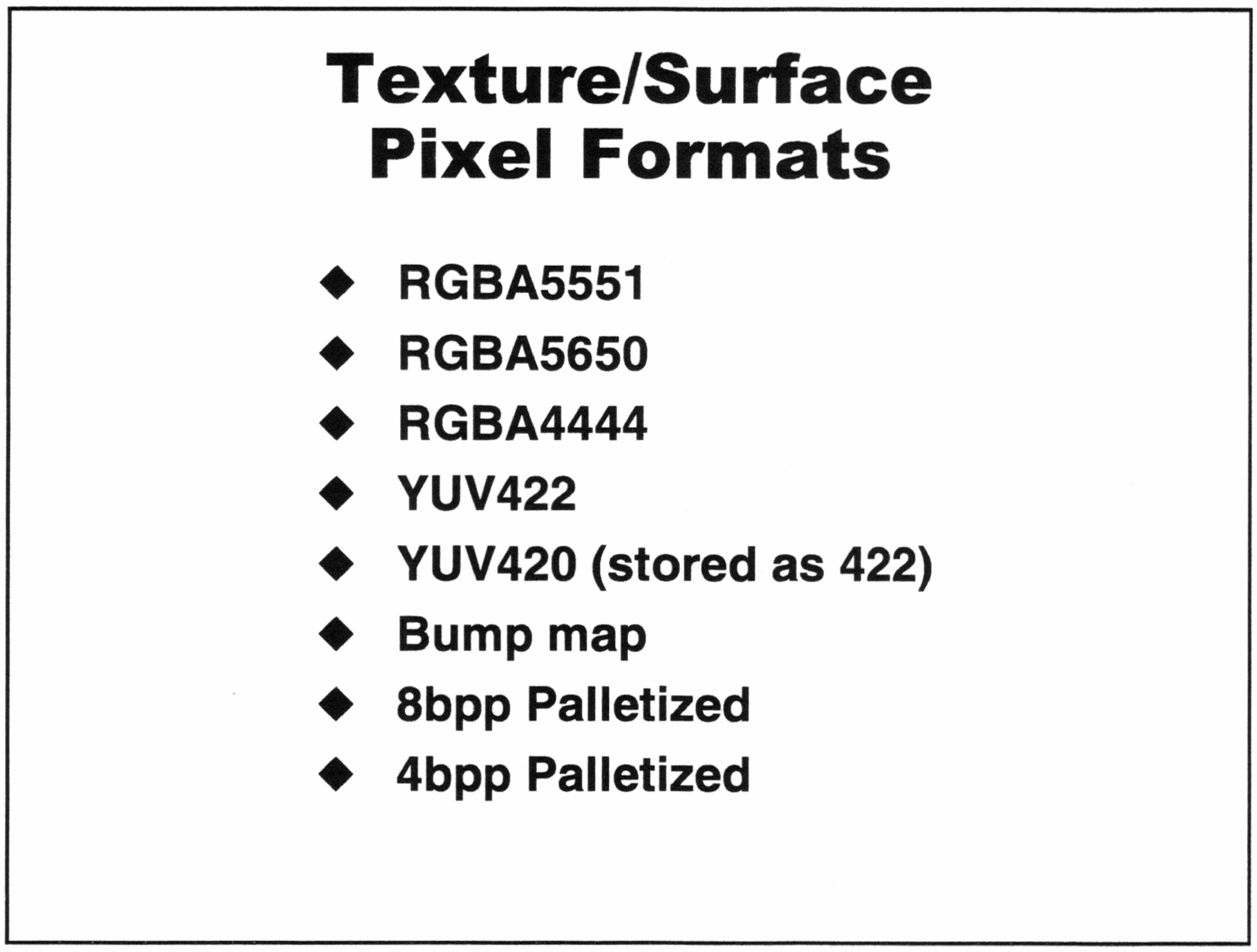
Video Memory Monitor



- Command Buffer (x2) ▨
  - Region Headers ▨
  - Polygon Pointers ▨
  - Vertex Data ▨
- Front & Back Buffers ▨
- Optional Copy Buffer (Autoclear == 0)
- Textures & Off-screen surfaces

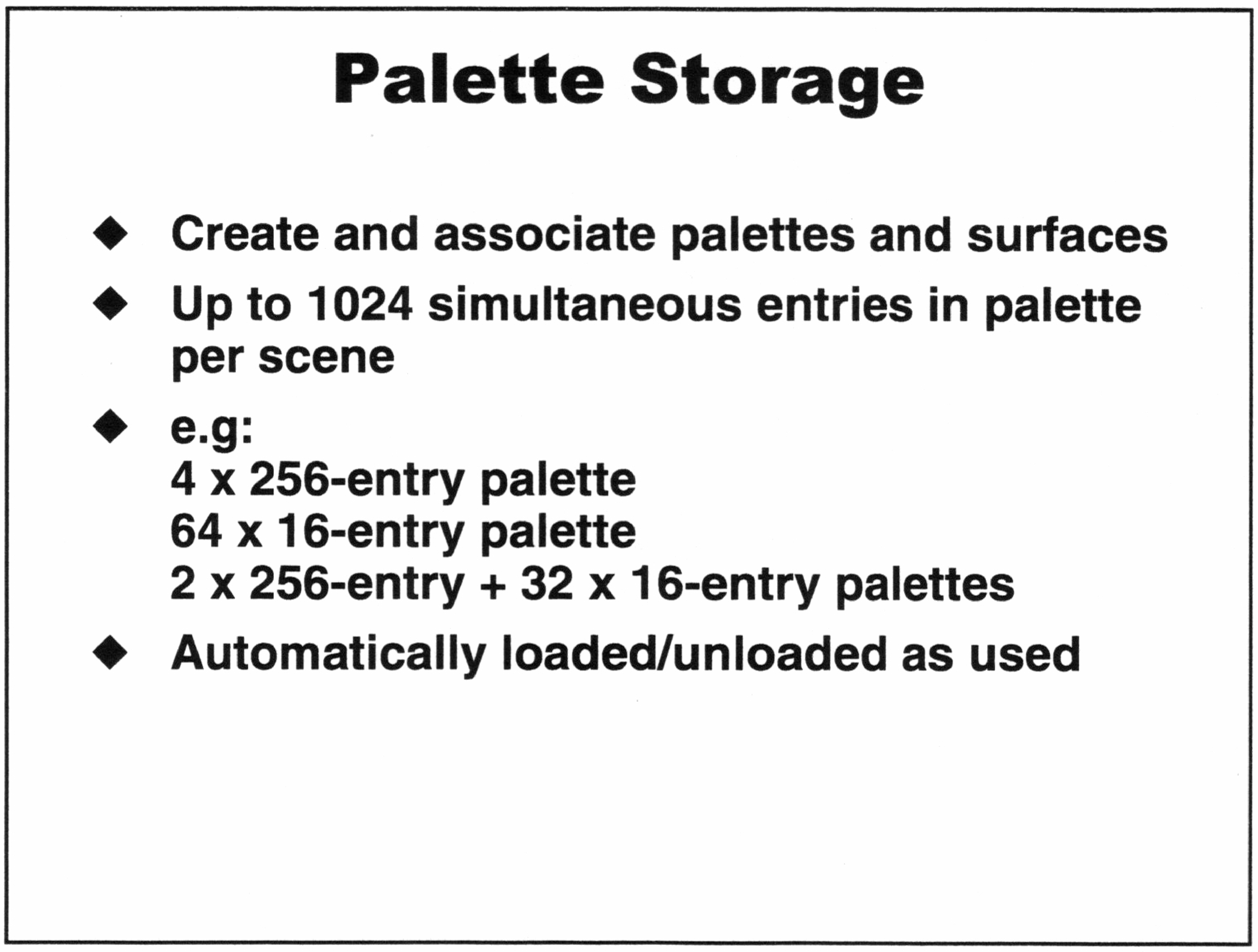
Texture/Surface Storage Flags

- Video memory / System memory
- Twiddled
- MipMap
- Texture
- VQ-compressed



Texture/Surface Pixel Formats

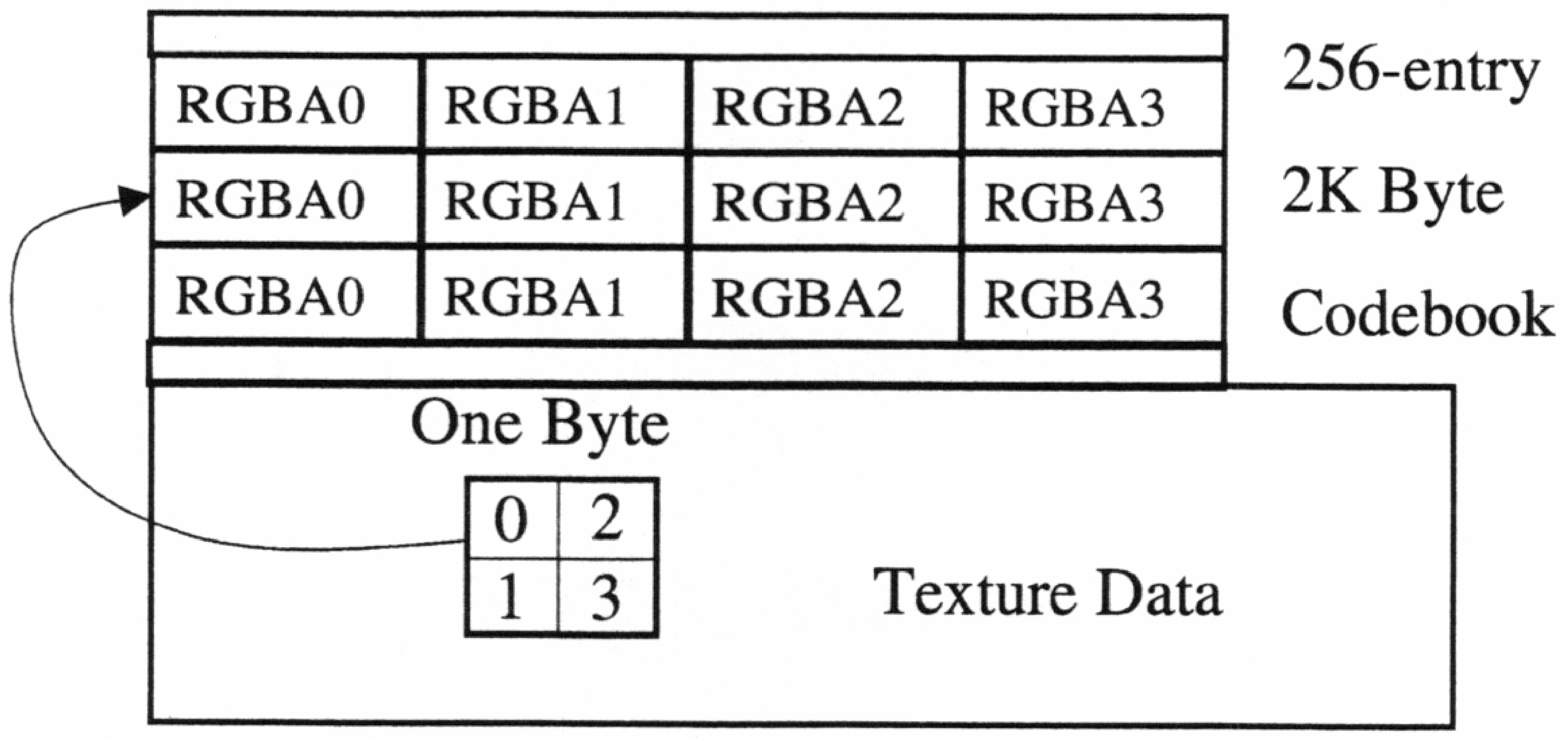
- RGBA5551
- RGBA5650
- RGBA4444
- YUV422
- YUV420 (stored as 422)
- Bump map
- 8bpp Palletized
- 4bpp Palletized



Palette Storage

- Create and associate palettes and surfaces
- Up to 1024 simultaneous entries in palette per scene
- e.g:
  4 x 256-entry palette
  64 x 16-entry palette
  2 x 256-entry + 32 x 16-entry palettes
- Automatically loaded/unloaded as used

VQ Textures



RGBA0	RGBA1	RGBA2	RGBA3
RGBA0	RGBA1	RGBA2	RGBA3
RGBA0	RGBA1	RGBA2	RGBA3

256-entry
2K Byte
Codebook

One Byte

0	2
1	3

Texture Data

- **8:1 compressed + 2KB overhead**
- **Compressed at author time**
- **Compression tool provided**
- **Can compress 16bpp twiddled format**

Bump Map

- **Non-standard D3D**
- **Can achieve excellent results for minimal video memory**
- **May use D3D(T)LVERTEX, or**
- **D3DLIGHTSTATE_BUMPINTENSITY and**
- **D3DLIGHTSTATE_BUMPDIRECTION**
- **Render bump (opaque), then texture (alpha) for opaque objects**
- **Render bump (alpha), then texture (alpha) for translucent objects**

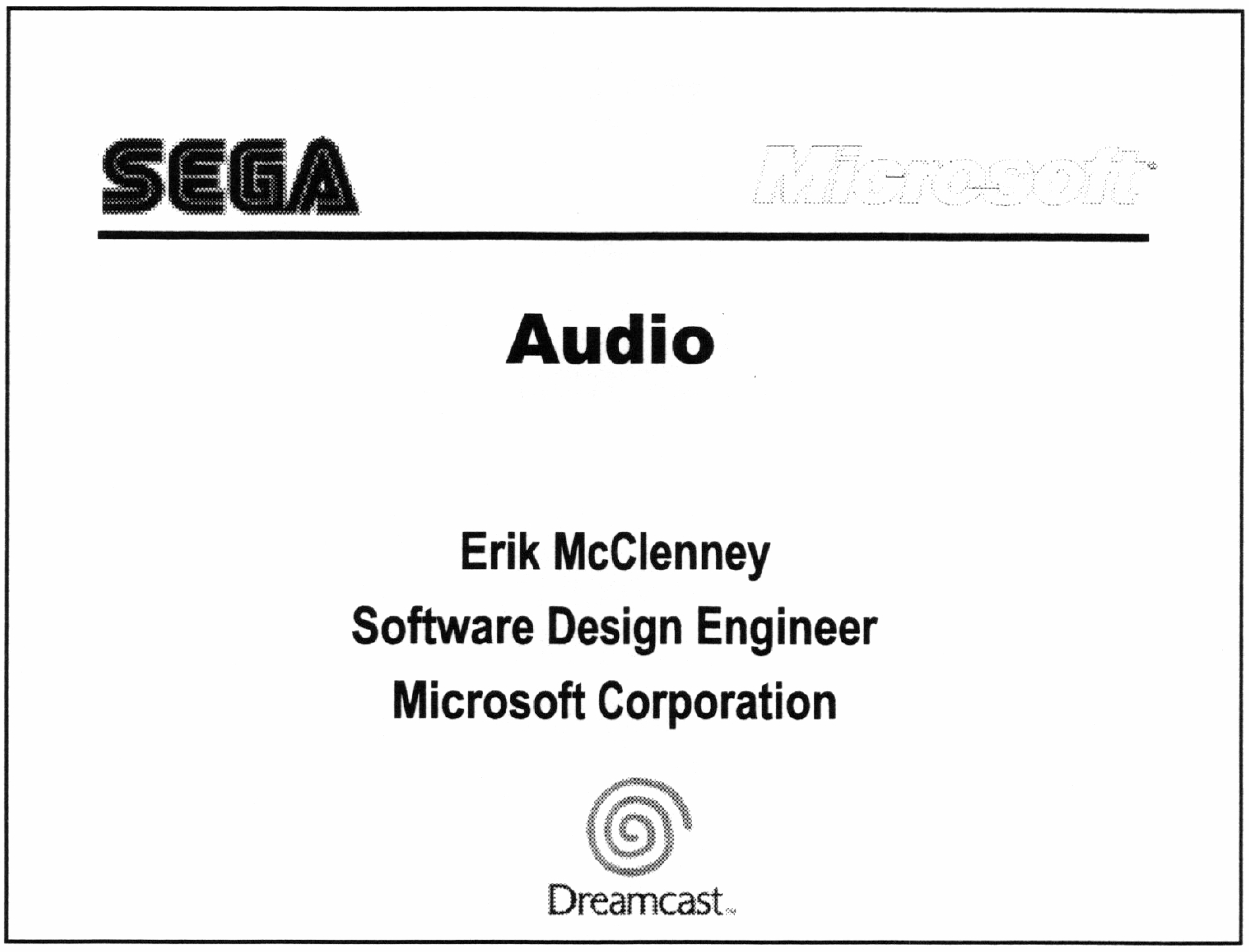
Demonstrations
&
Questions

SEGA

Audio

Erik McClenney
Software Design Engineer
Microsoft Corporation

Dreamcast™

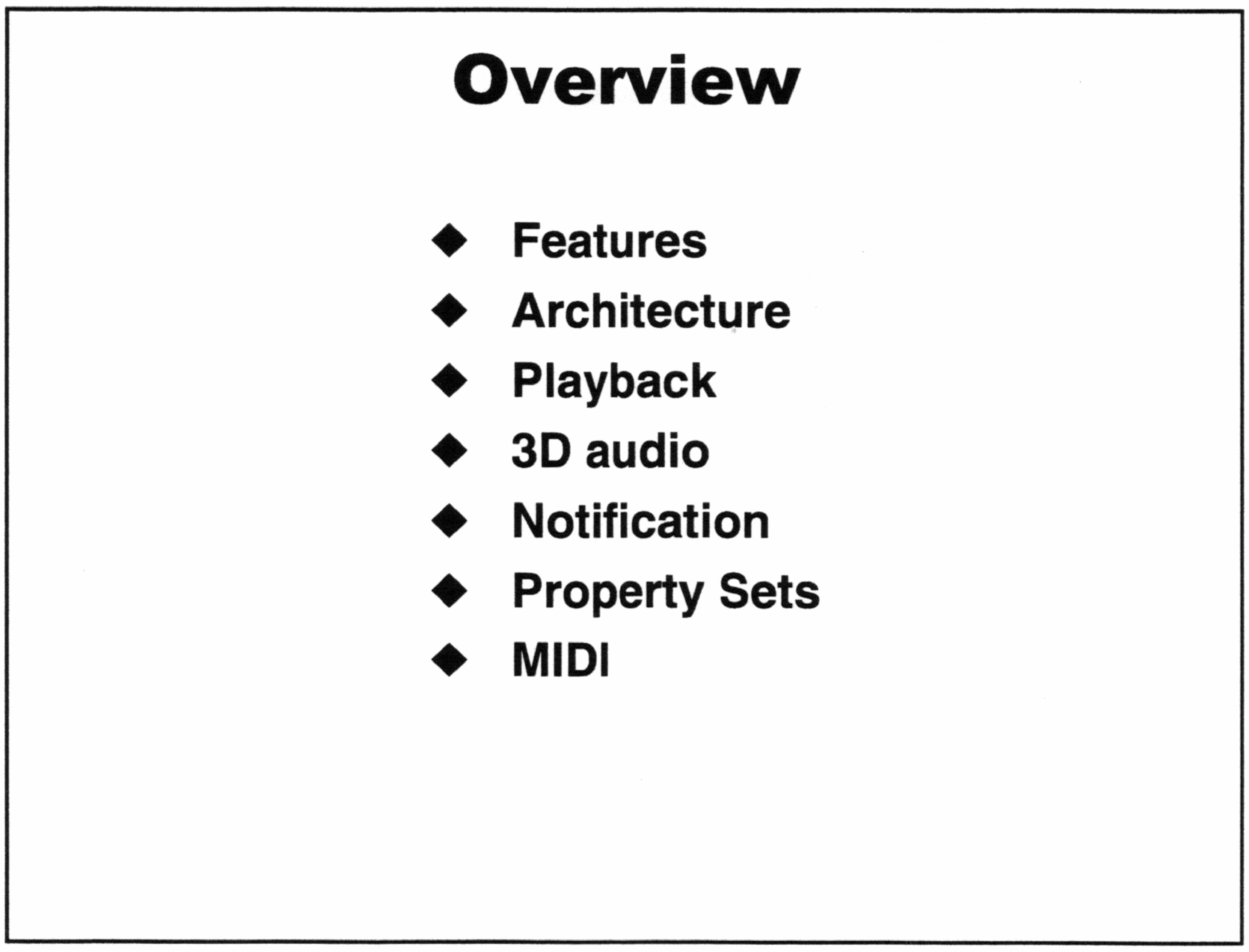


SEGA Microsoft

Audio

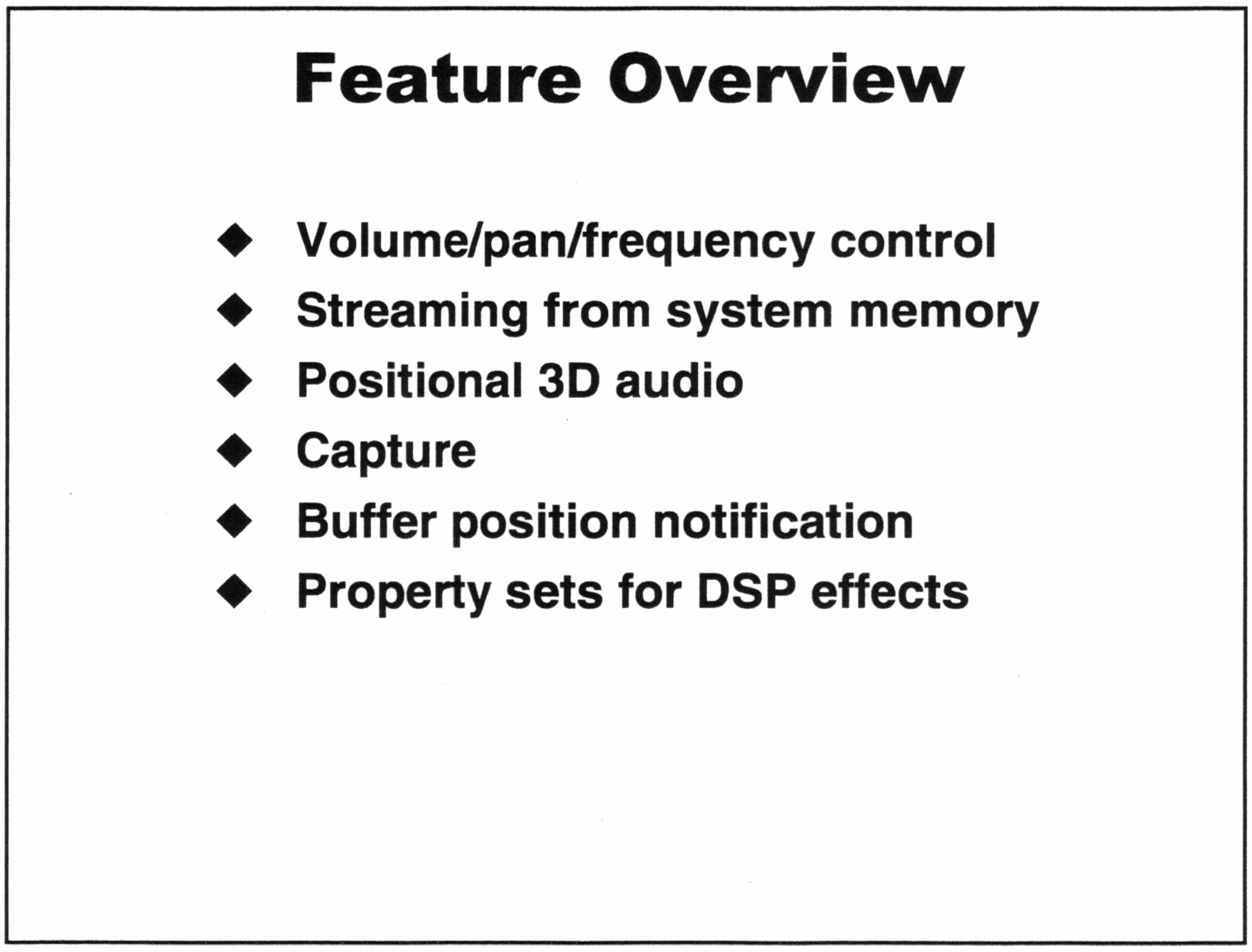
Erik McClenney
Software Design Engineer
Microsoft Corporation

Dreamcast™



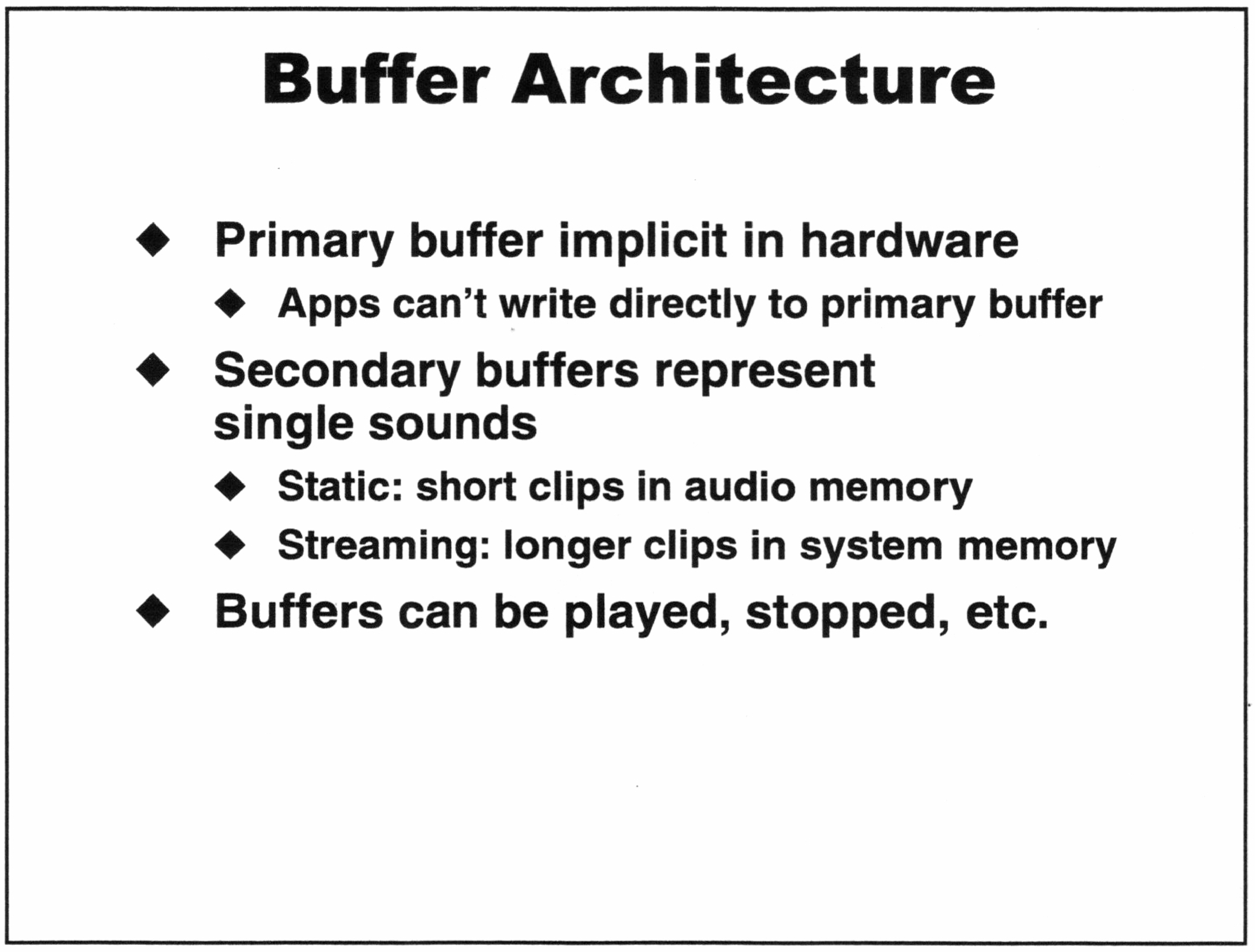
Overview

- Features
- Architecture
- Playback
- 3D audio
- Notification
- Property Sets
- MIDI



Feature Overview

- ◆ Volume/pan/frequency control
- ◆ Streaming from system memory
- ◆ Positional 3D audio
- ◆ Capture
- ◆ Buffer position notification
- ◆ Property sets for DSP effects



Buffer Architecture

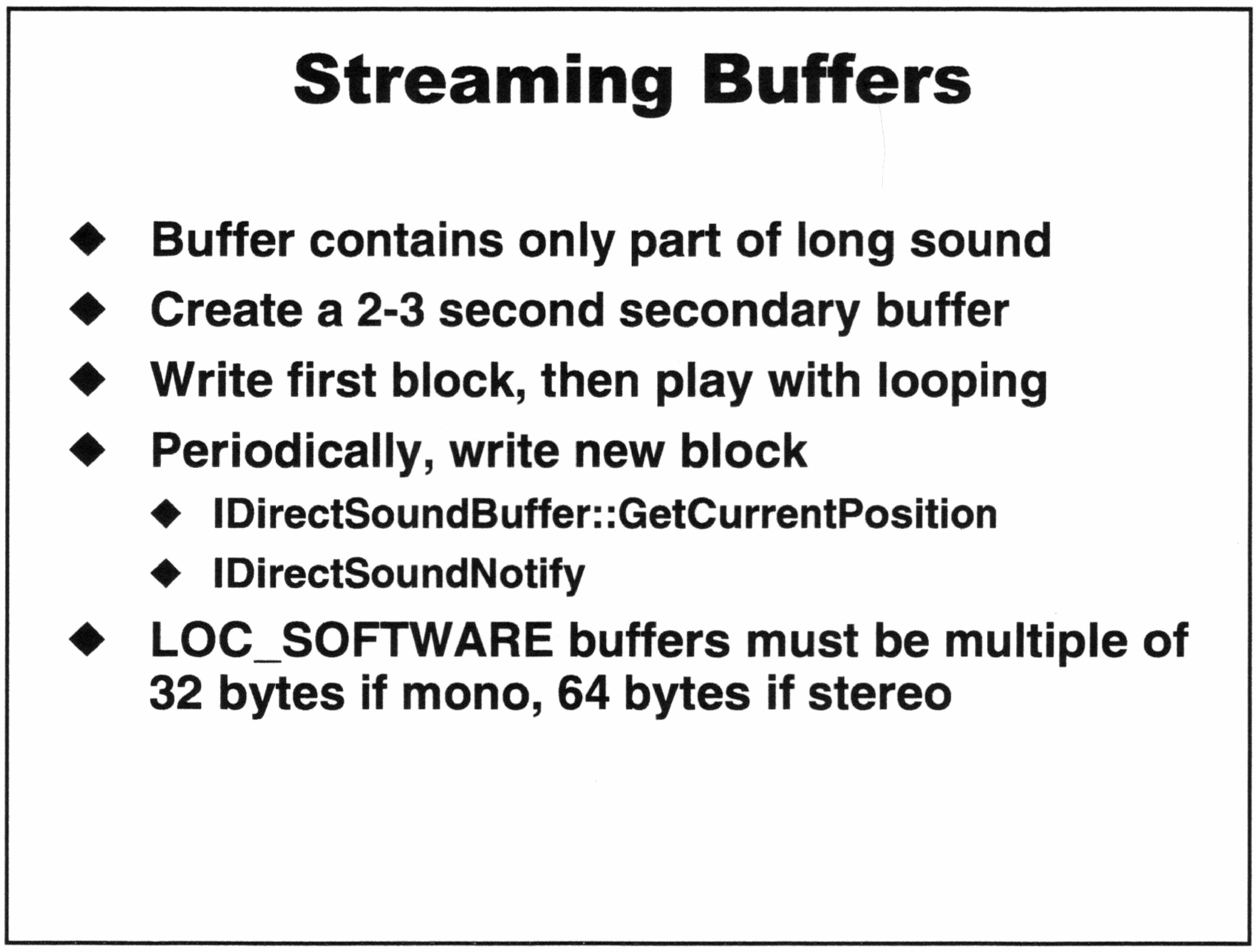
- ◆ Primary buffer implicit in hardware
  - ◆ Apps can't write directly to primary buffer
- ◆ Secondary buffers represent single sounds
  - ◆ Static: short clips in audio memory
  - ◆ Streaming: longer clips in system memory
- ◆ Buffers can be played, stopped, etc.

DirectSound - Basic Usage

- Create DirectSound object
- Set Cooperative Level
- Create secondary buffers for each sound
- Fill secondary buffers with data
  - Lock buffer to obtain write pointer
  - Write sound data into buffer
  - Unlock buffer
- Set notification positions
- Play and Stop secondary buffers

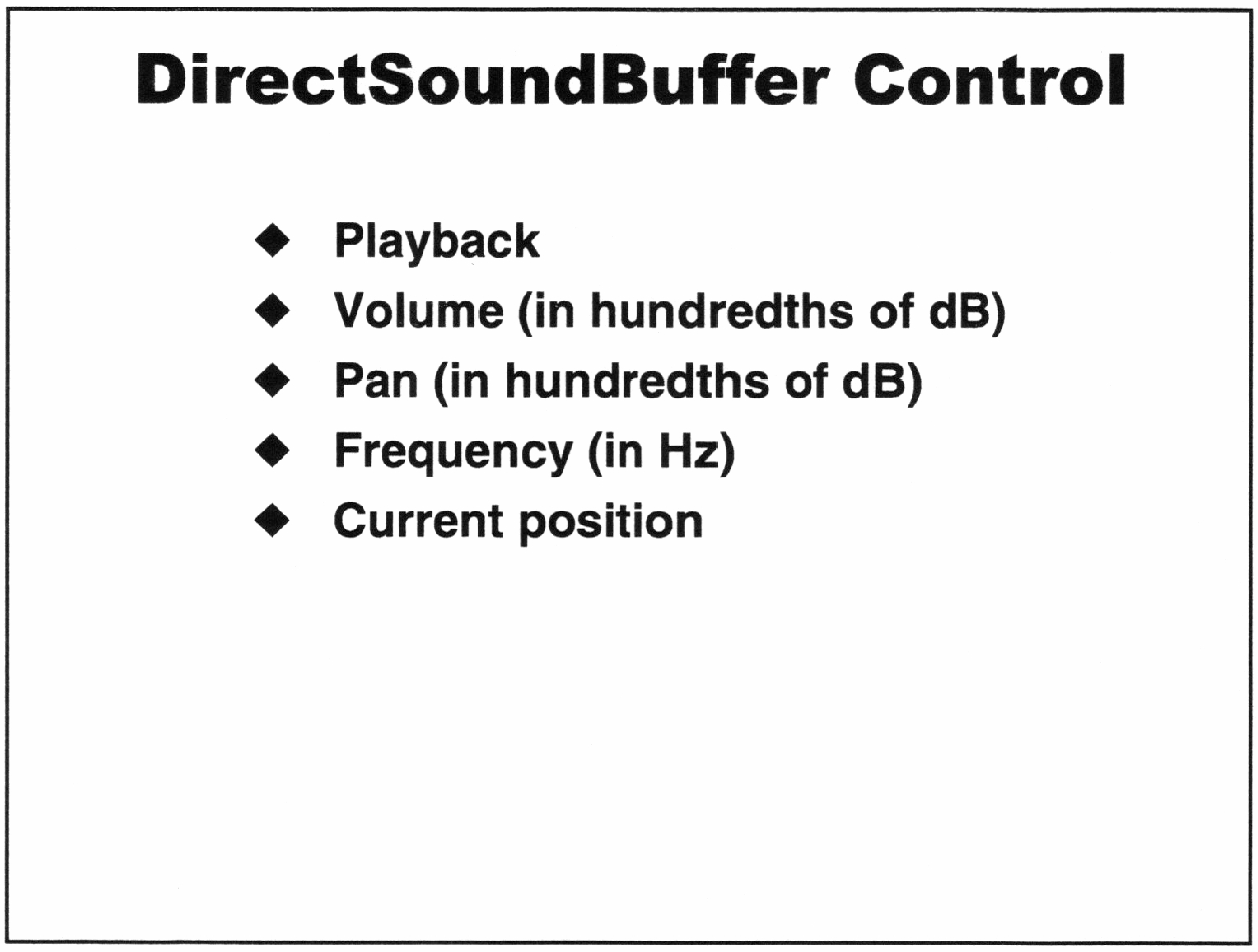
Static Buffers

- Contain complete (short) sounds
- Usually reusable (helicopter, etc.)
- Sound written once, played many times
- DSBCAPS_STATIC - only a hint



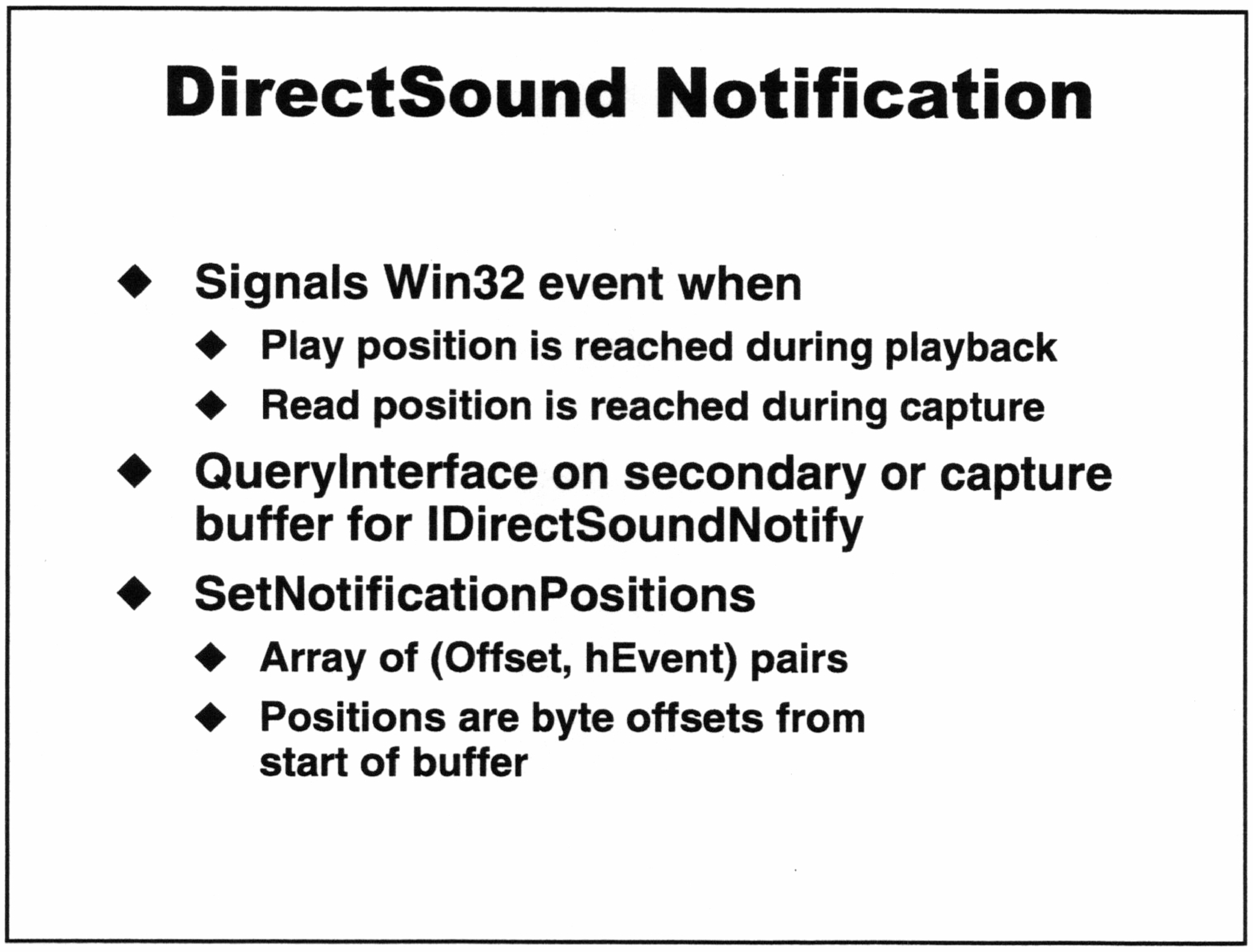
Streaming Buffers

- Buffer contains only part of long sound
- Create a 2-3 second secondary buffer
- Write first block, then play with looping
- Periodically, write new block
  - IDirectSoundBuffer::GetCurrentPosition
  - IDirectSoundNotify
- LOC_SOFTWARE buffers must be multiple of 32 bytes if mono, 64 bytes if stereo



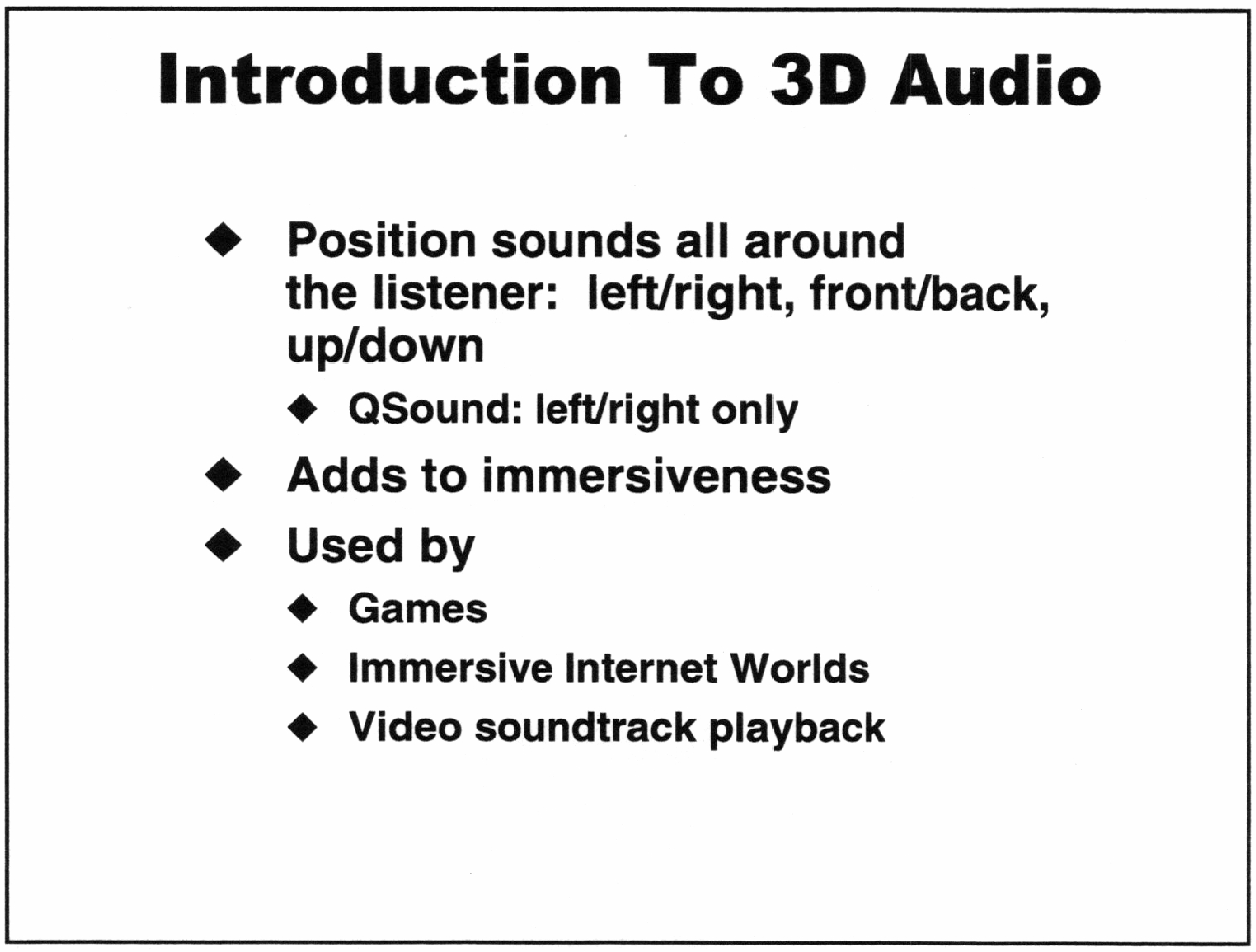
DirectSoundBuffer Control

- Playback
- Volume (in hundredths of dB)
- Pan (in hundredths of dB)
- Frequency (in Hz)
- Current position



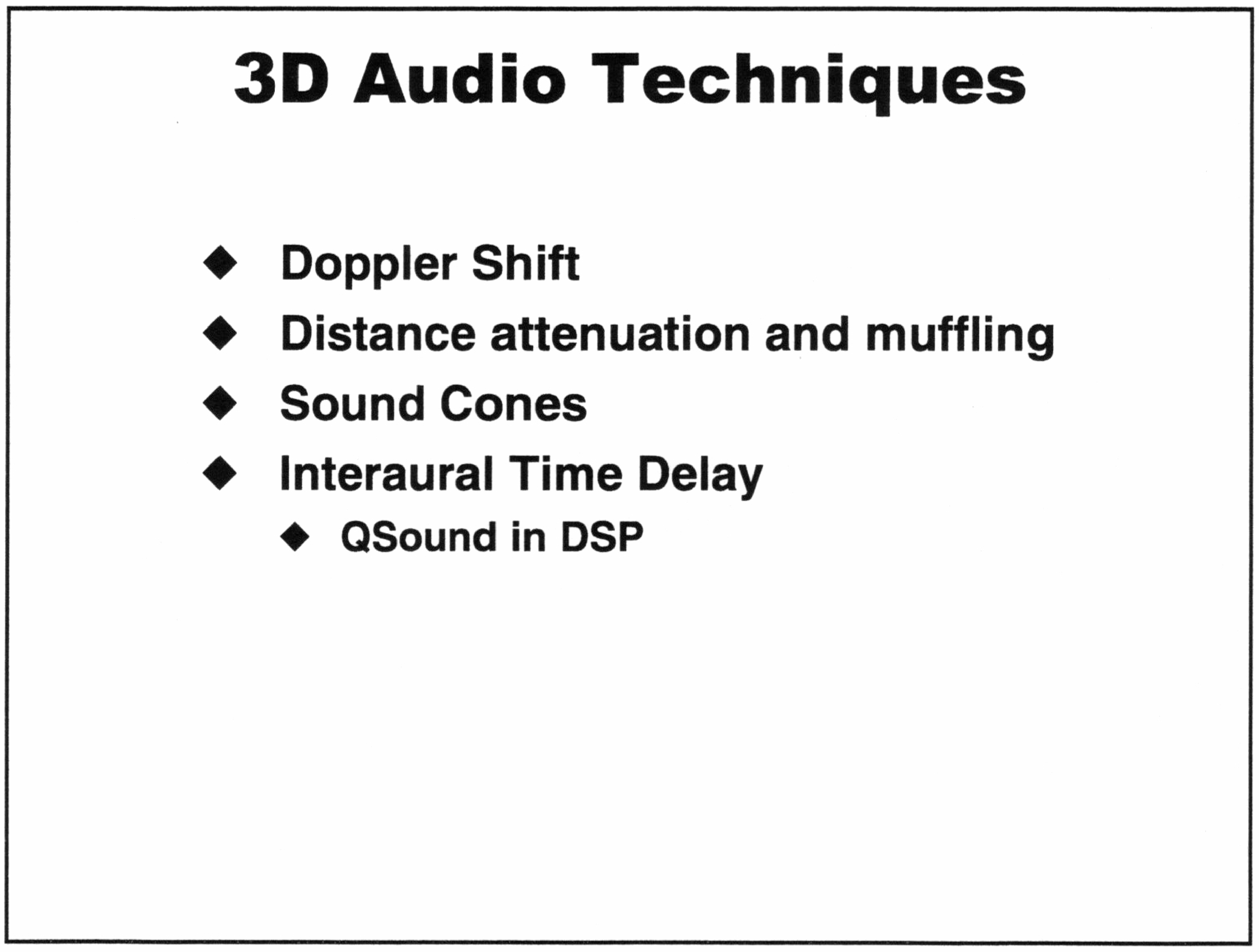
DirectSound Notification

- Signals Win32 event when
  - Play position is reached during playback
  - Read position is reached during capture
- QueryInterface on secondary or capture buffer for IDirectSoundNotify
- SetNotificationPositions
  - Array of (Offset, hEvent) pairs
  - Positions are byte offsets from start of buffer



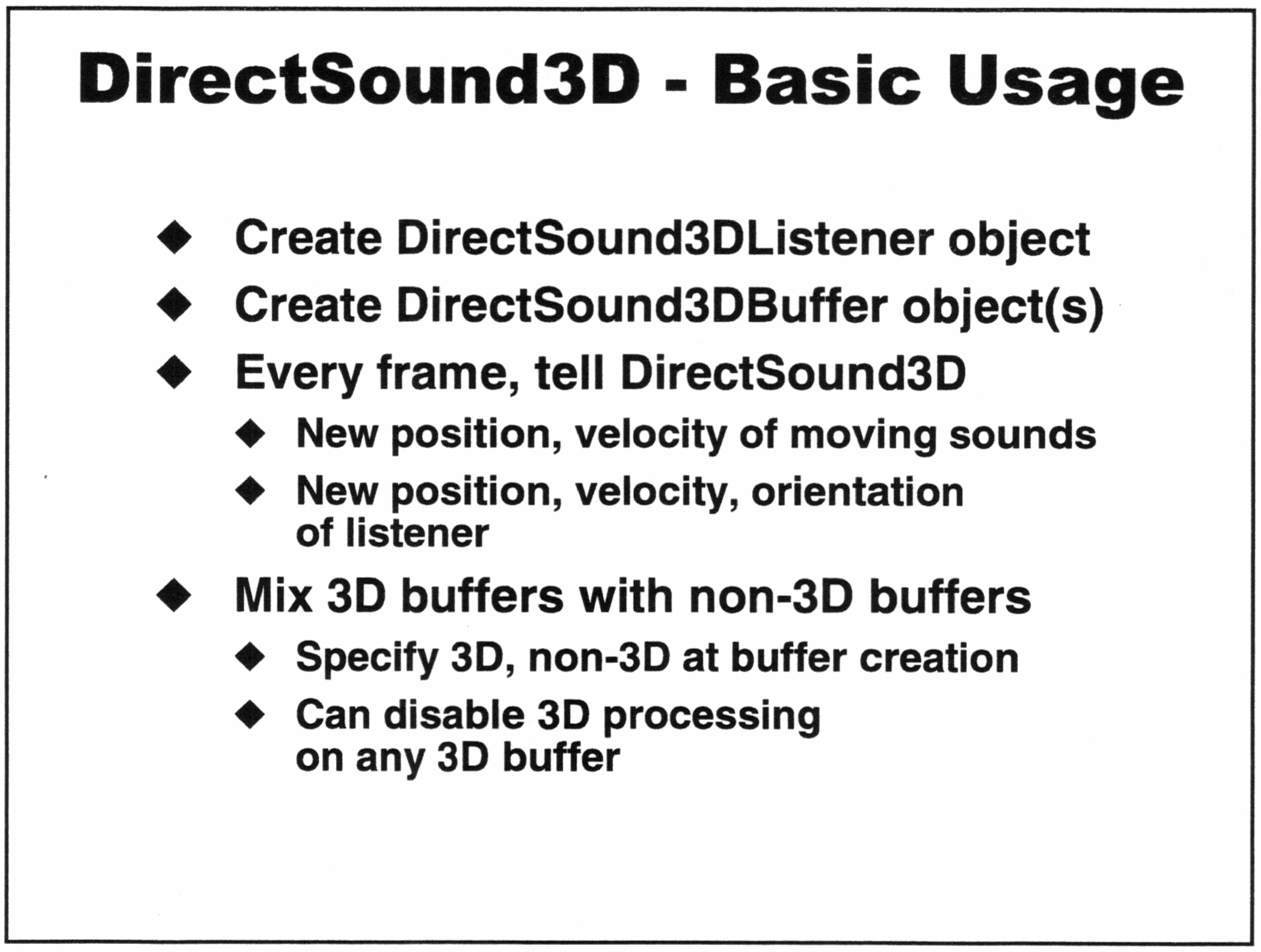
Introduction To 3D Audio

- Position sounds all around the listener: left/right, front/back, up/down
  - QSound: left/right only
- Adds to immersiveness
- Used by
  - Games
  - Immersive Internet Worlds
  - Video soundtrack playback



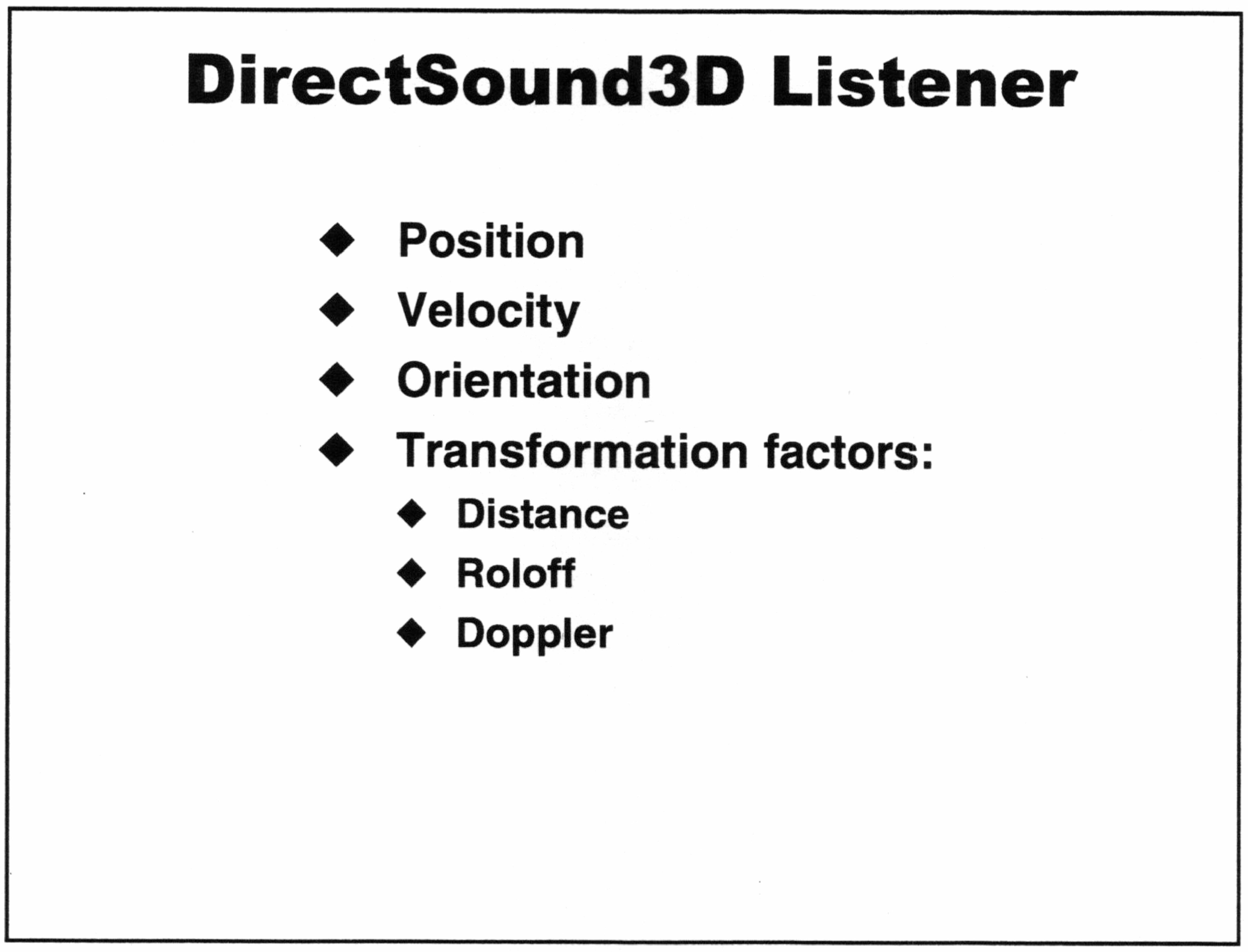
3D Audio Techniques

- Doppler Shift
- Distance attenuation and muffling
- Sound Cones
- Interaural Time Delay
  - QSound in DSP



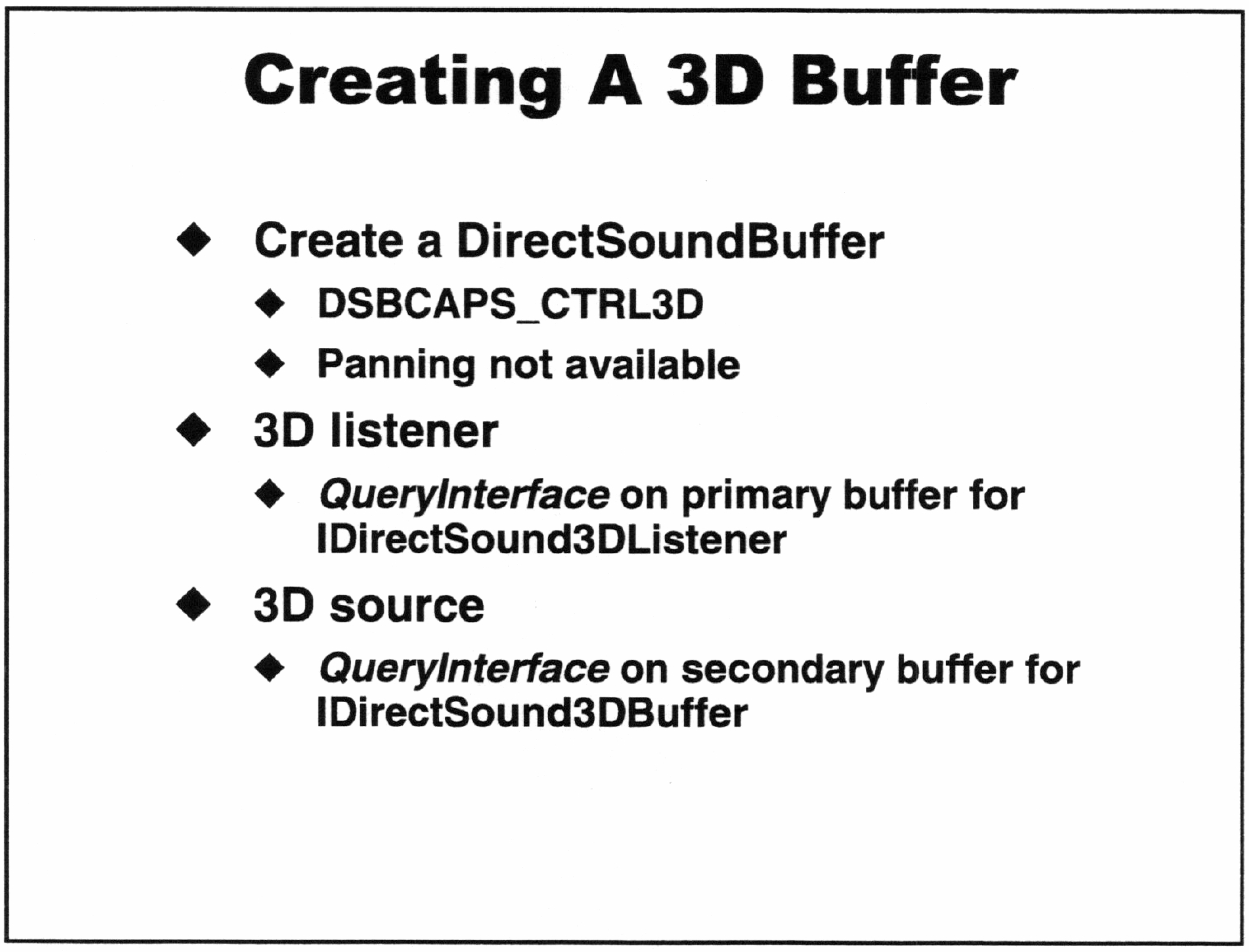
DirectSound3D - Basic Usage

- Create DirectSound3DListener object
- Create DirectSound3DBuffer object(s)
- Every frame, tell DirectSound3D
  - New position, velocity of moving sounds
  - New position, velocity, orientation of listener
- Mix 3D buffers with non-3D buffers
  - Specify 3D, non-3D at buffer creation
  - Can disable 3D processing on any 3D buffer



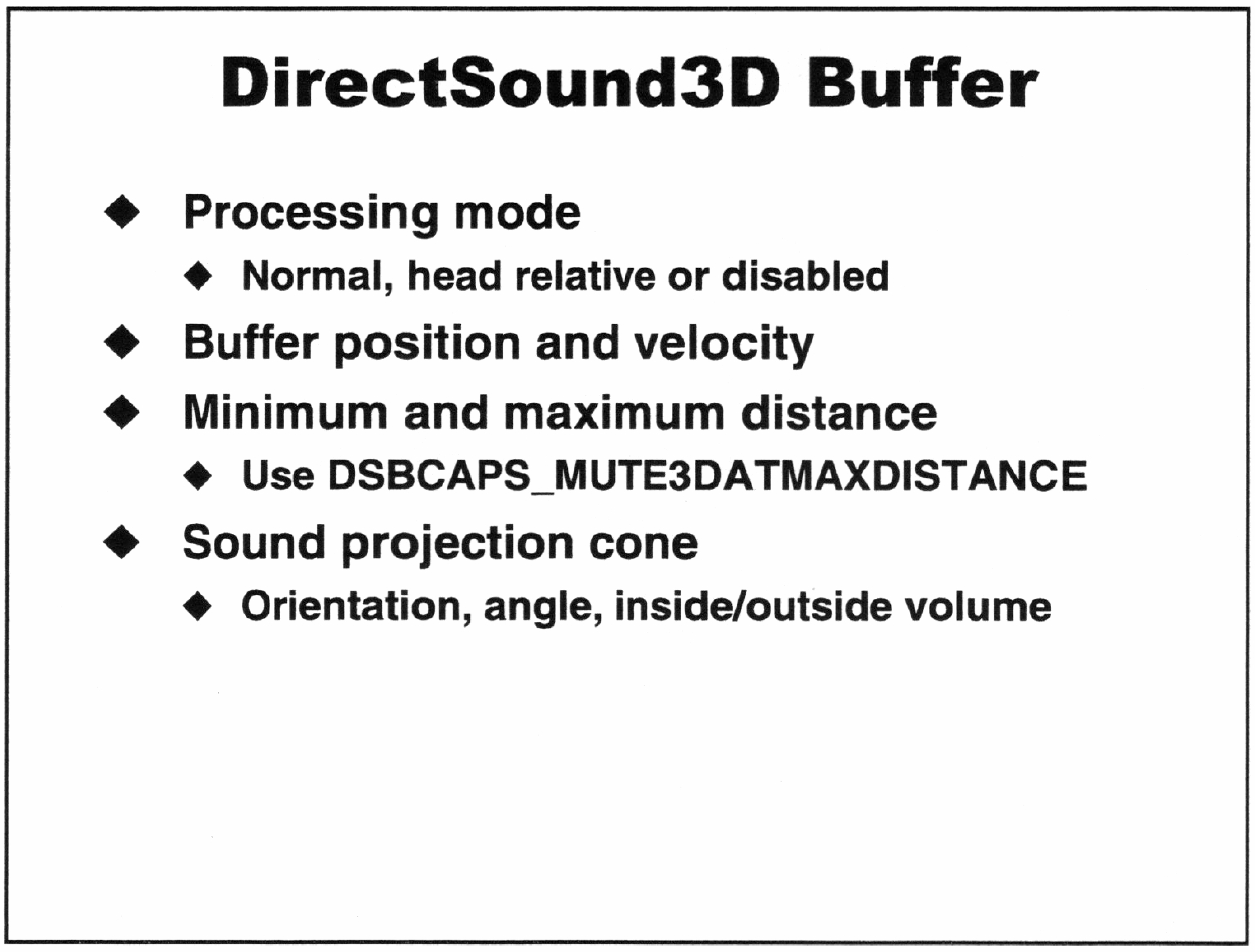
DirectSound3D Listener

- Position
- Velocity
- Orientation
- Transformation factors:
  - Distance
  - Roloff
  - Doppler



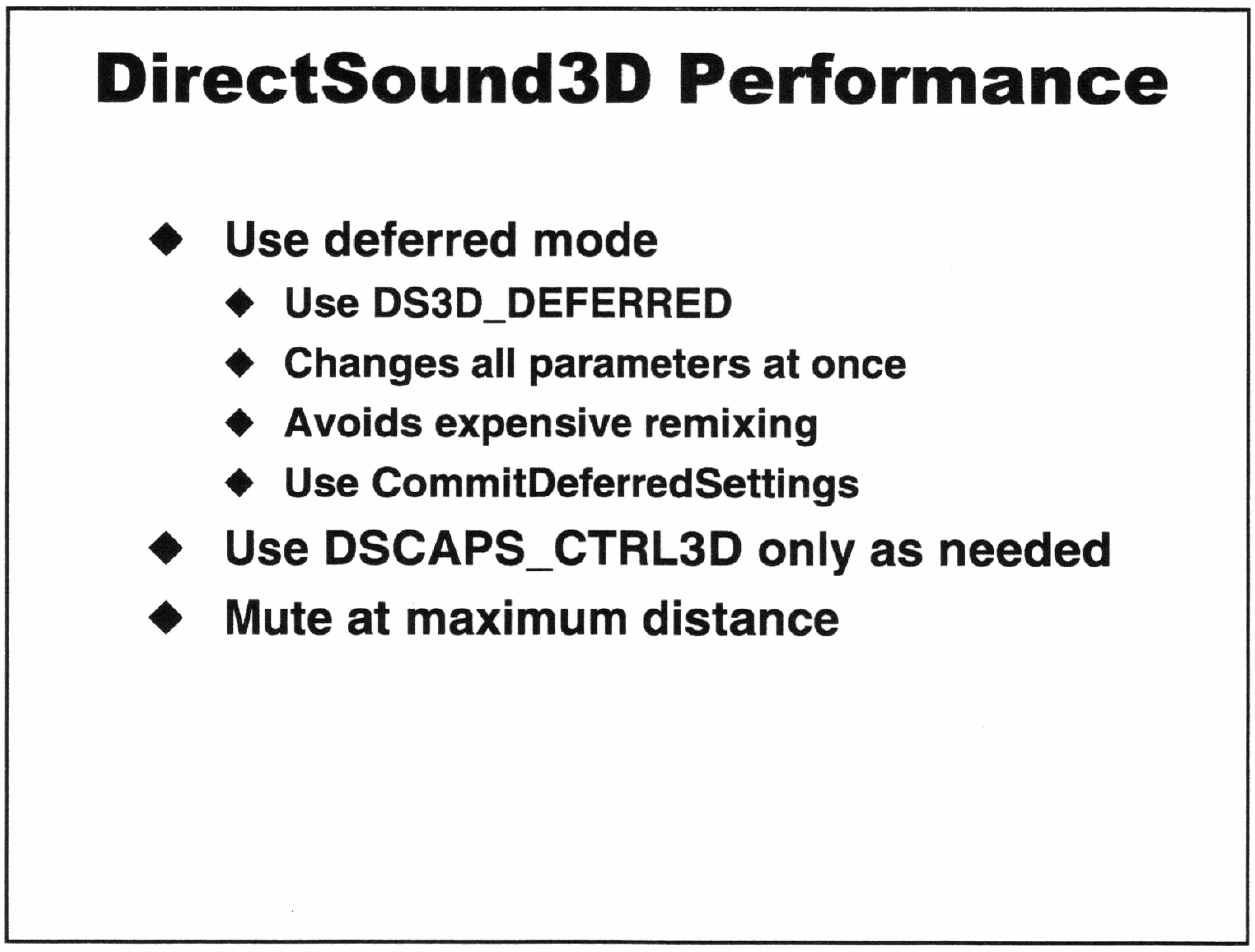
Creating A 3D Buffer

- Create a DirectSoundBuffer
  - DSBCAPS_CTRL3D
  - Panning not available
- 3D listener
  - *QueryInterface* on primary buffer for IDirectSound3DListener
- 3D source
  - *QueryInterface* on secondary buffer for IDirectSound3DBuffer



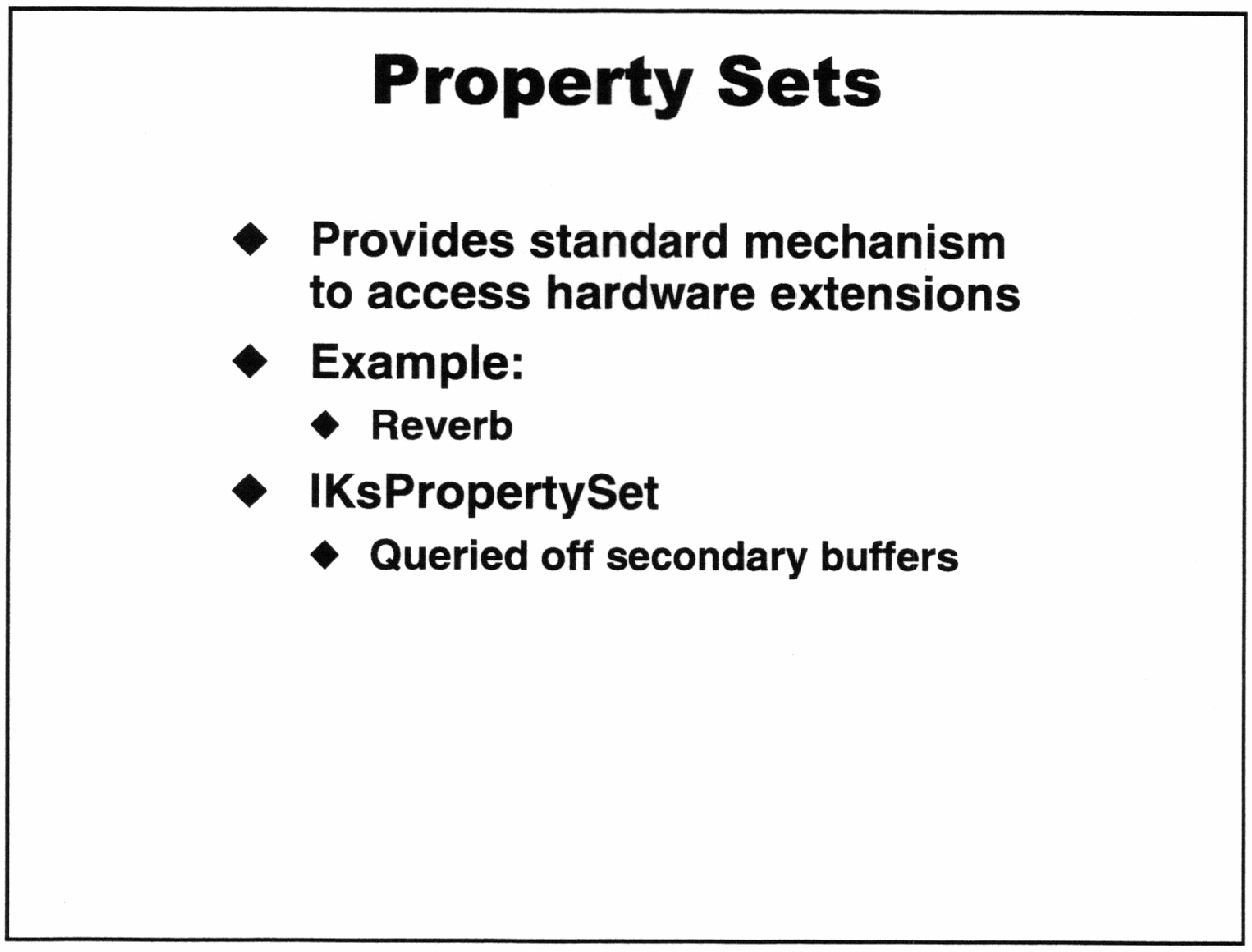
DirectSound3D Buffer

- Processing mode
  - Normal, head relative or disabled
- Buffer position and velocity
- Minimum and maximum distance
  - Use DSBCAPS_MUTE3DATMAXDISTANCE
- Sound projection cone
  - Orientation, angle, inside/outside volume



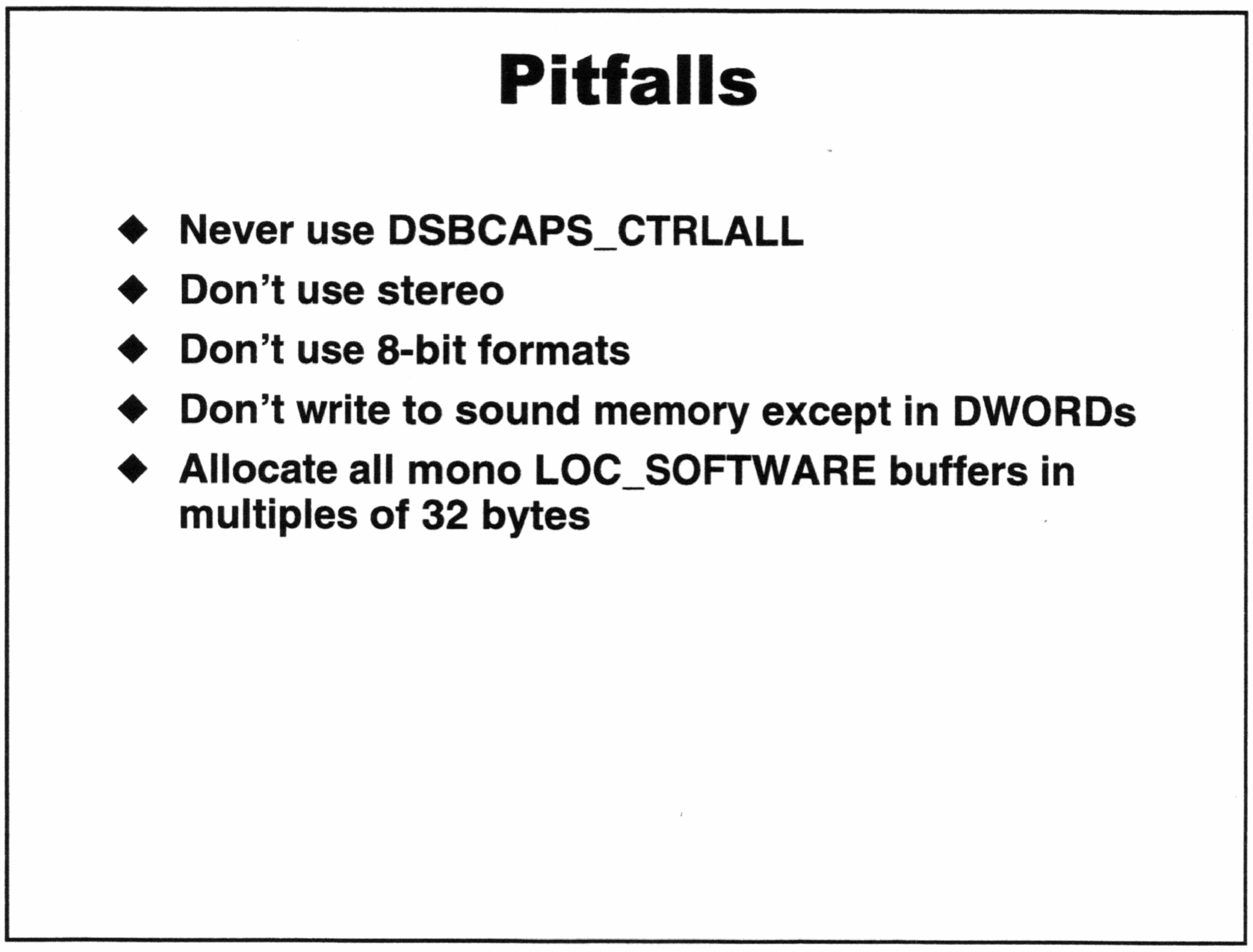
DirectSound3D Performance

- Use deferred mode
  - Use DS3D_DEFERRED
  - Changes all parameters at once
  - Avoids expensive remixing
  - Use CommitDeferredSettings
- Use DSCAPS_CTRL3D only as needed
- Mute at maximum distance



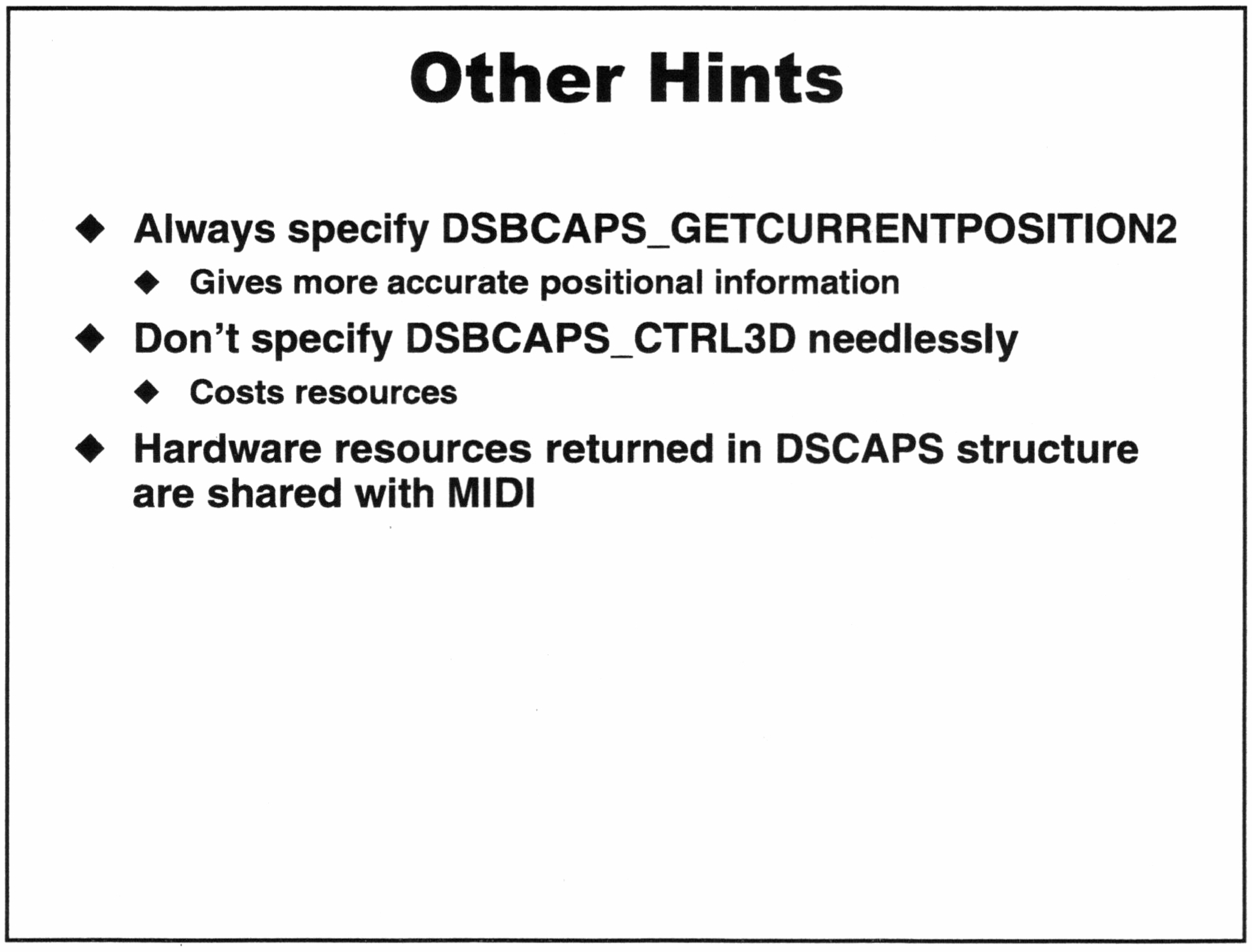
Property Sets

- Provides standard mechanism to access hardware extensions
- Example:
  - Reverb
- IKsPropertySet
  - Queried off secondary buffers



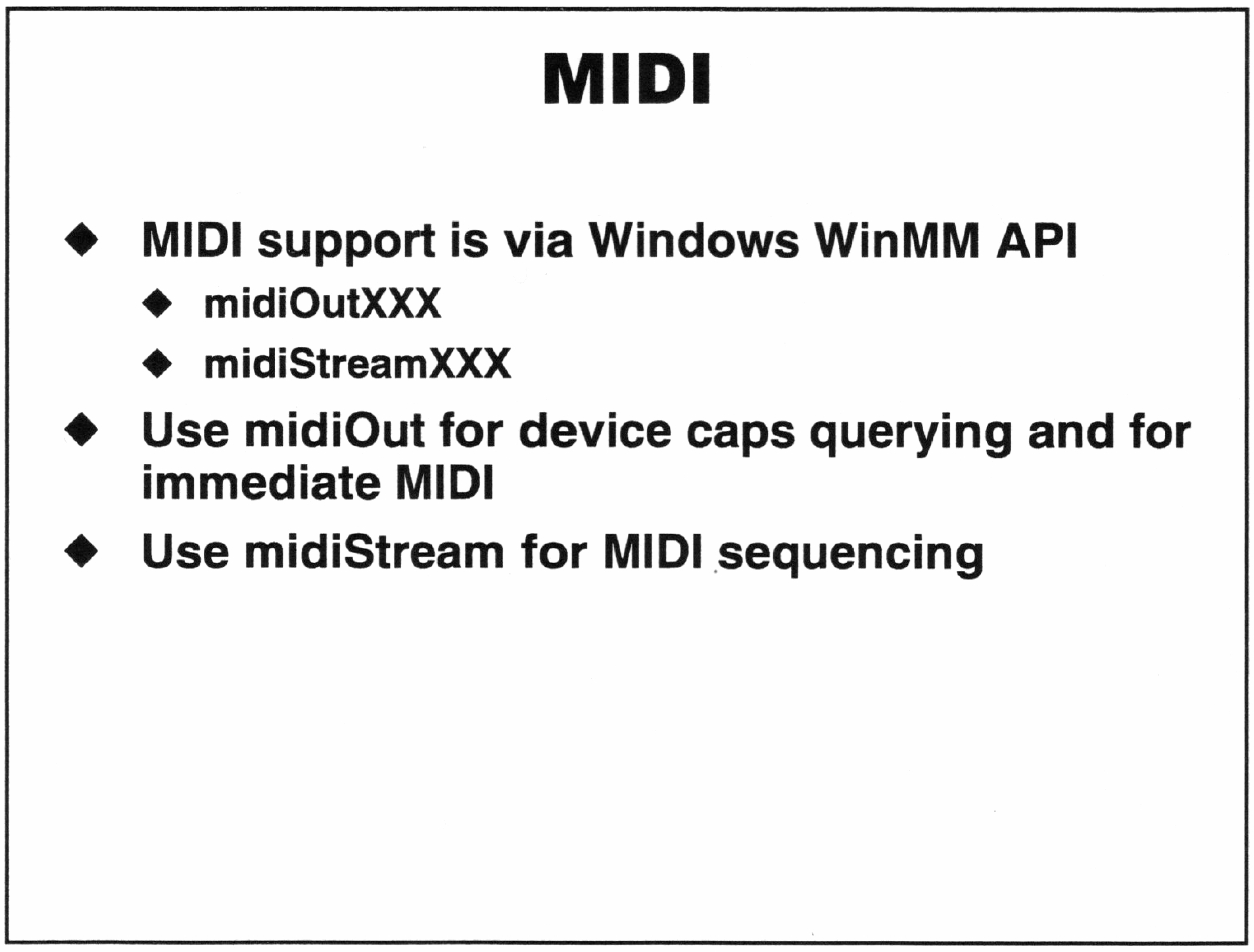
Pitfalls

- Never use DSBCAPS_CTRLALL
- Don’t use stereo
- Don’t use 8-bit formats
- Don’t write to sound memory except in DWORDs
- Allocate all mono LOC_SOFTWARE buffers in multiples of 32 bytes



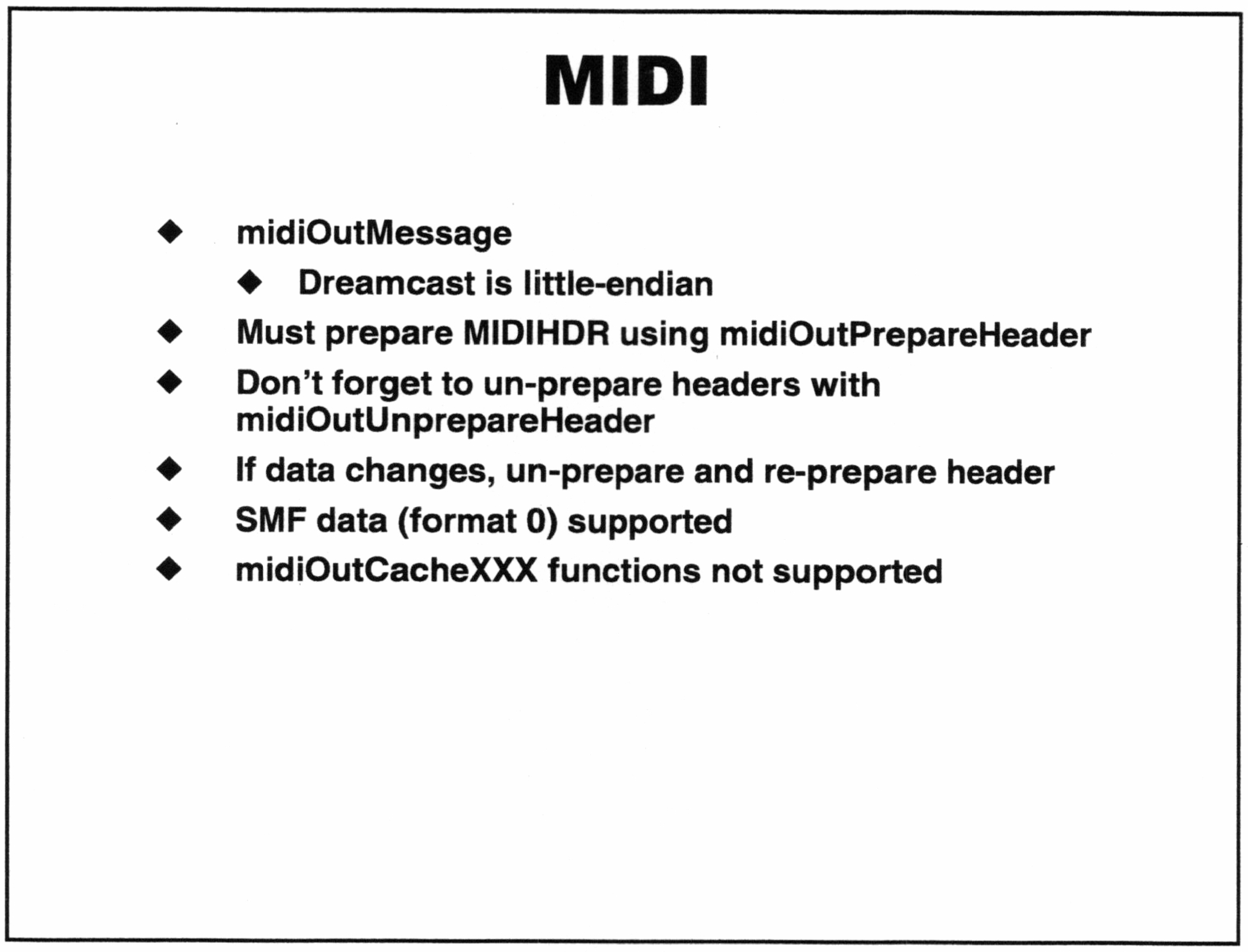
Other Hints

- Always specify DSBCAPS_GETCURRENTPOSITION2
  - Gives more accurate positional information
- Don't specify DSBCAPS_CTRL3D needlessly
  - Costs resources
- Hardware resources returned in DSCAPS structure are shared with MIDI



MIDI

- MIDI support is via Windows WinMM API
  - midiOutXXX
  - midiStreamXXX
- Use midiOut for device caps querying and for immediate MIDI
- Use midiStream for MIDI sequencing



MIDI

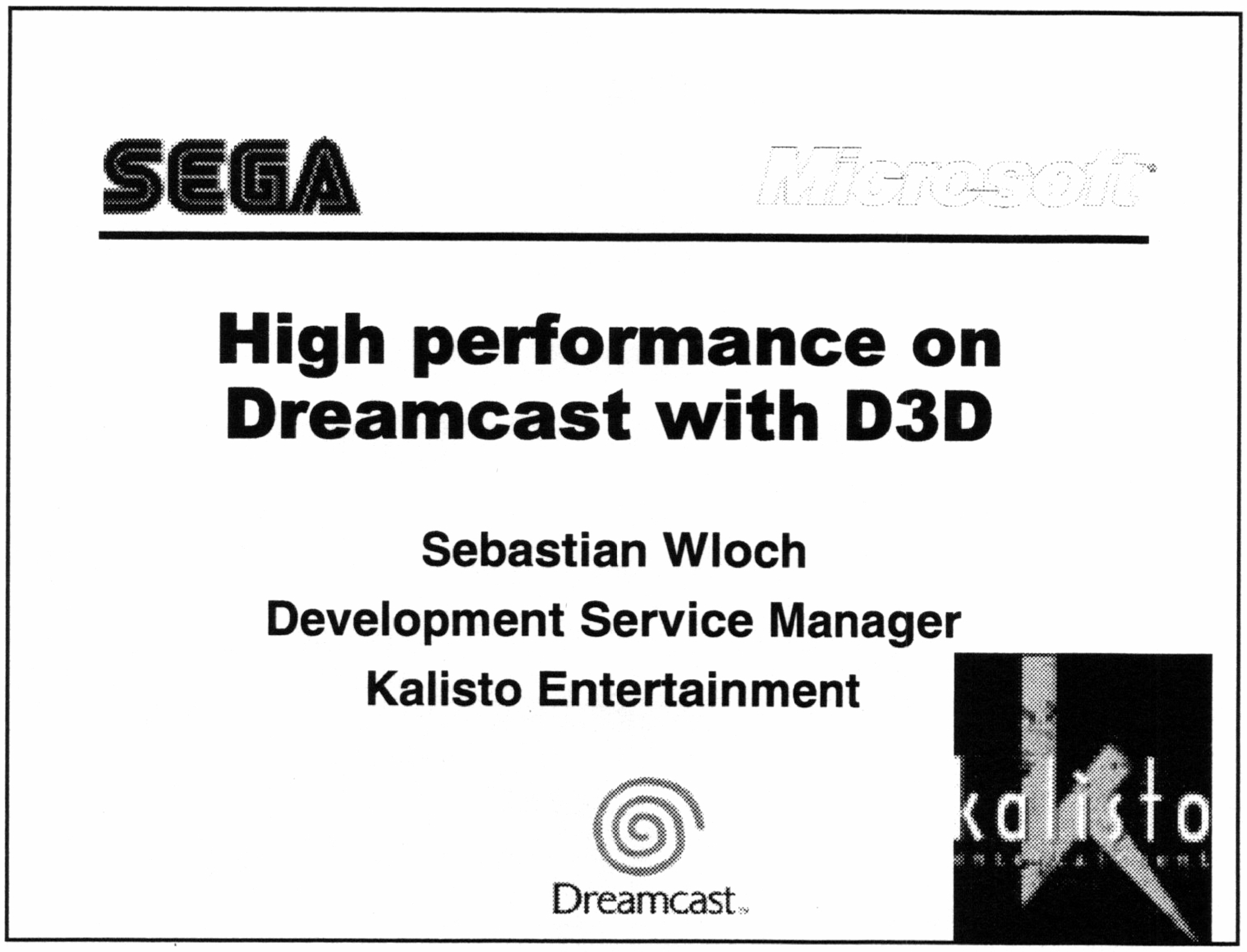
- midiOutMessage
  - Dreamcast is little-endian
- Must prepare MIDIHDR using midiOutPrepareHeader
- Don't forget to un-prepare headers with midiOutUnprepareHeader
- If data changes, un-prepare and re-prepare header
- SMF data (format 0) supported
- midiOutCacheXXX functions not supported

SEGA

High performance on Dreamcast with D3D

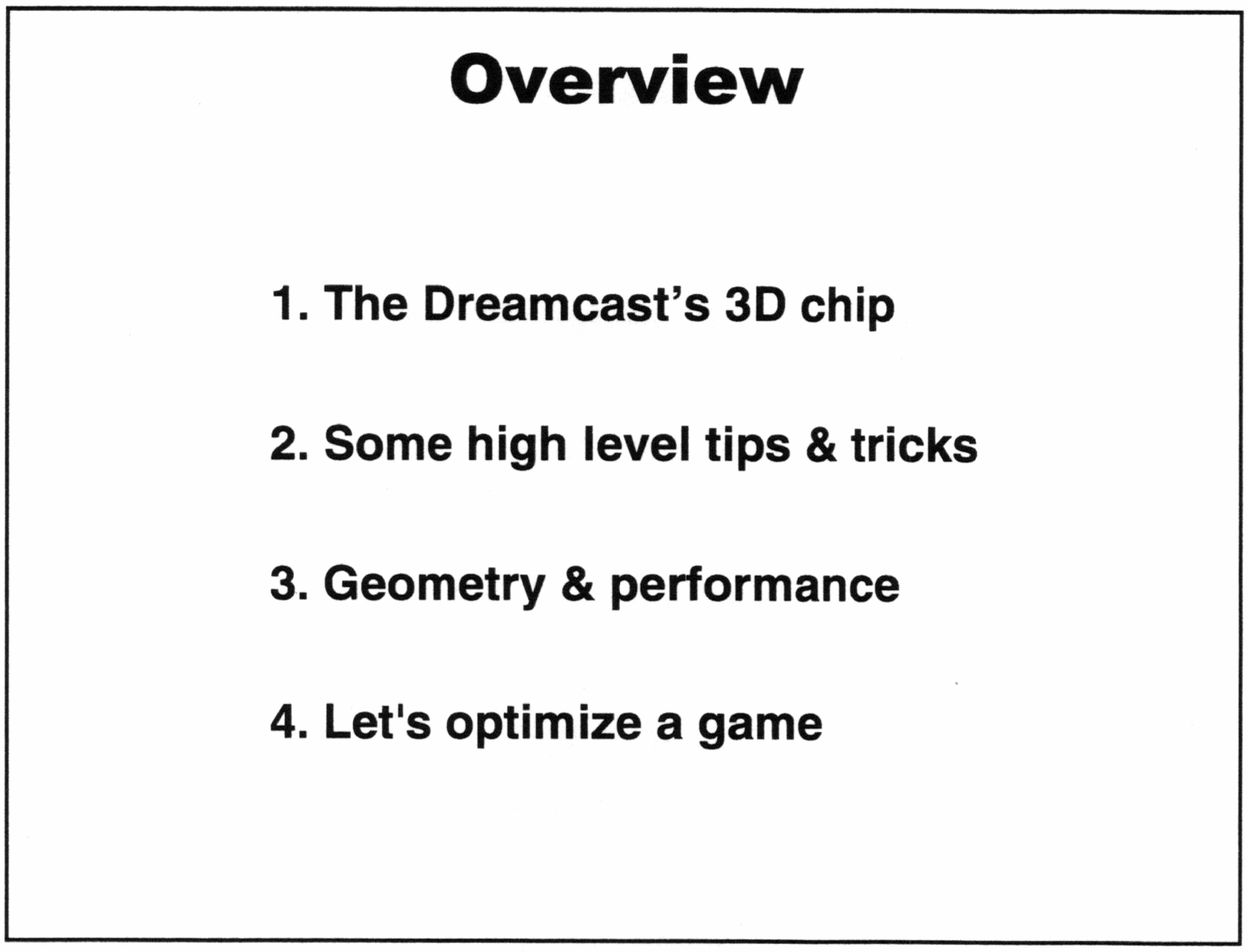
Sebastian Wloch
Development Service Manager
Kalisto Entertainment

Dreamcast™


SEGA
Microsoft
Dreamcast
kalisto

High performance on Dreamcast with D3D

Sebastian Wloch
Development Service Manager
Kalisto Entertainment

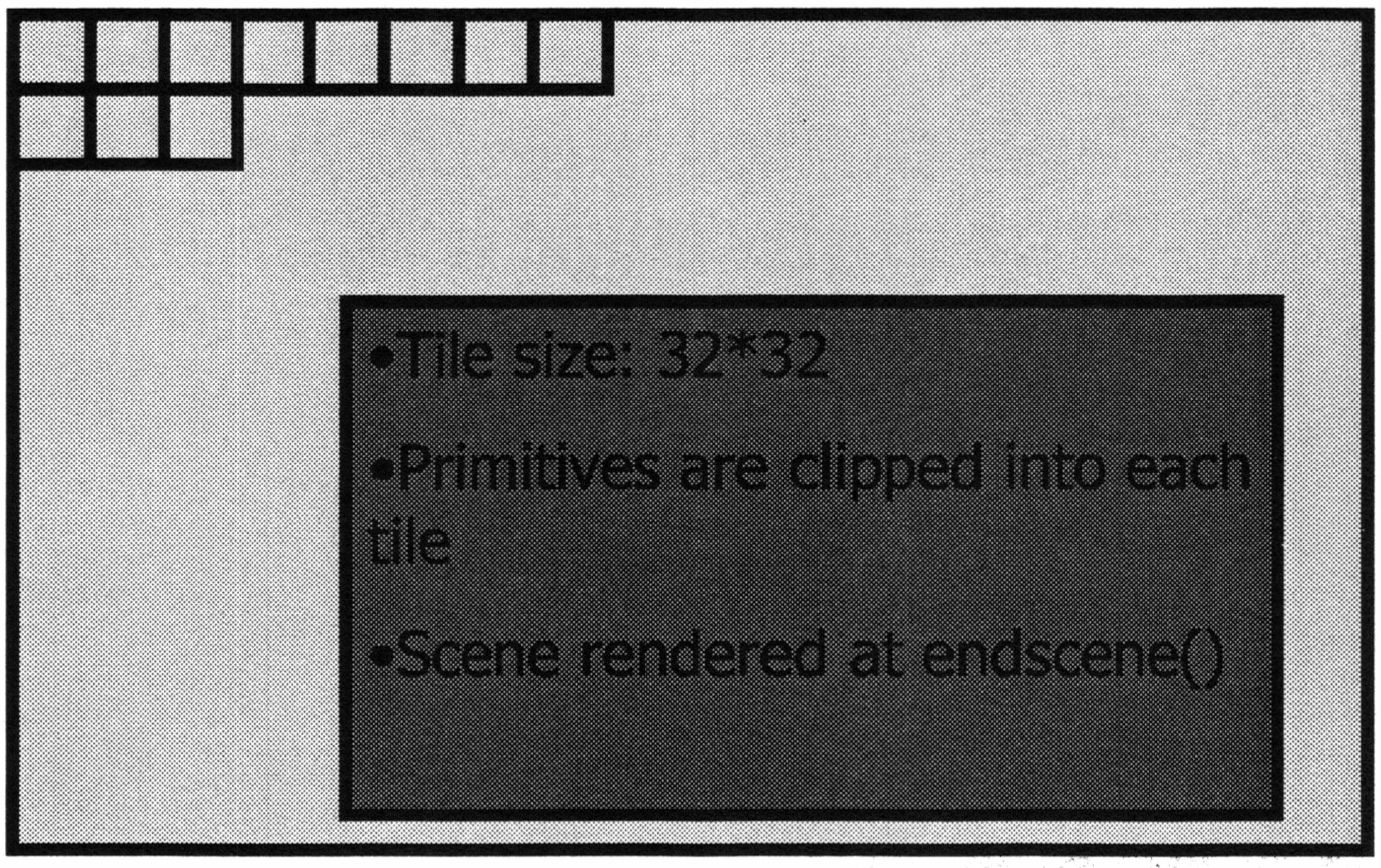


Overview

1. The Dreamcast's 3D chip
2. Some high level tips & tricks
3. Geometry & performance
4. Let's optimize a game

1. The Dreamcast's 3D chip

Tile based hardware



- Tile size: 32*32
- Primitives are clipped into each tile
- Scene rendered at endscene()

Features

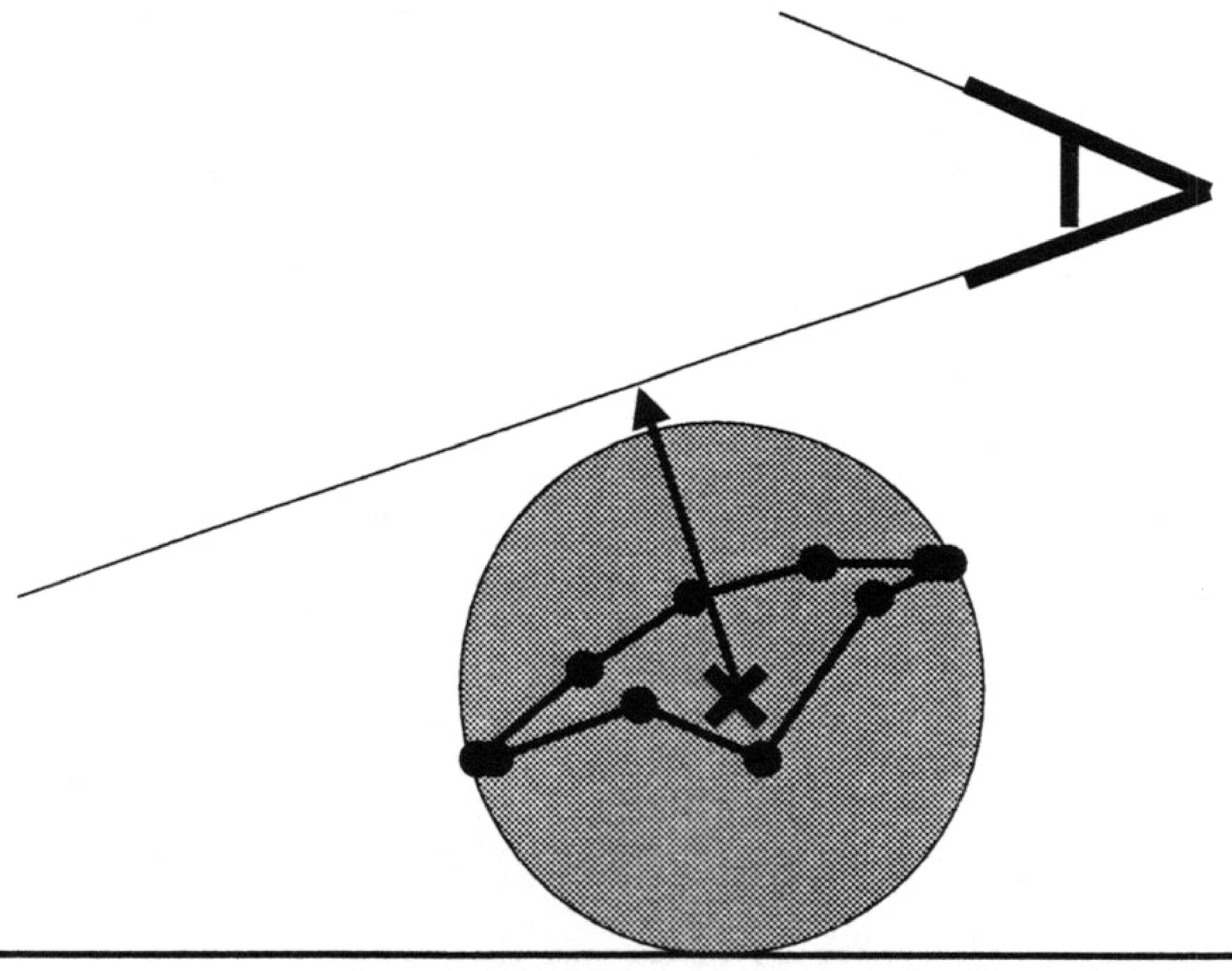
- Fillrate not a problem anymore
- Transparency sorted by hardware
- Viewport clipping done by hardware
- Drawback: Transparency a little slower
- But 5551 mode as fast as opaque mode
- SH4 native operations (intrinsic)
  - void __fsca(float *sin_apprx, float *cos_apprx, int input);
  - float _Dot3dVW0(float *vector1, float *vector2);
  - float _InvSqrtA(float input);

2. Some high level tips & tricks

- Eliminate as many meshes as possible before rendering
- But elimination tests have a cost
- Elimination test algorithm must be very fast
- Only test on potentially eliminated objects
- Existing tests:
  - View frustrum elimination
  - Backface culling
  - Objects hidden by other objects

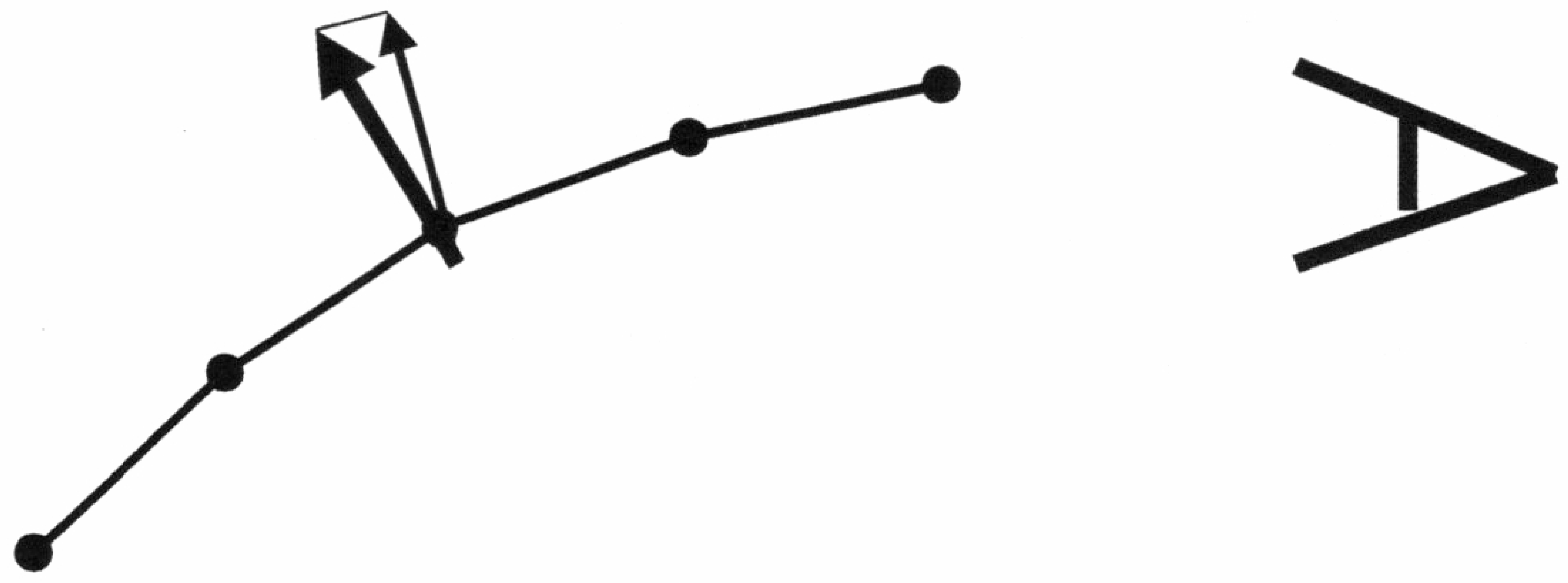
Pyramid removal

- Bspheres faster than Bboxes
  - For all objects
  - For large primitives



Backstrip culling

- Only for large strips (10 triangles or more) use tolerance for curvature
- Make sure you're not visiting individual vertices (D3D can do a better job)
    - Use float _Dot3dVW0(*vector1, *vector2);

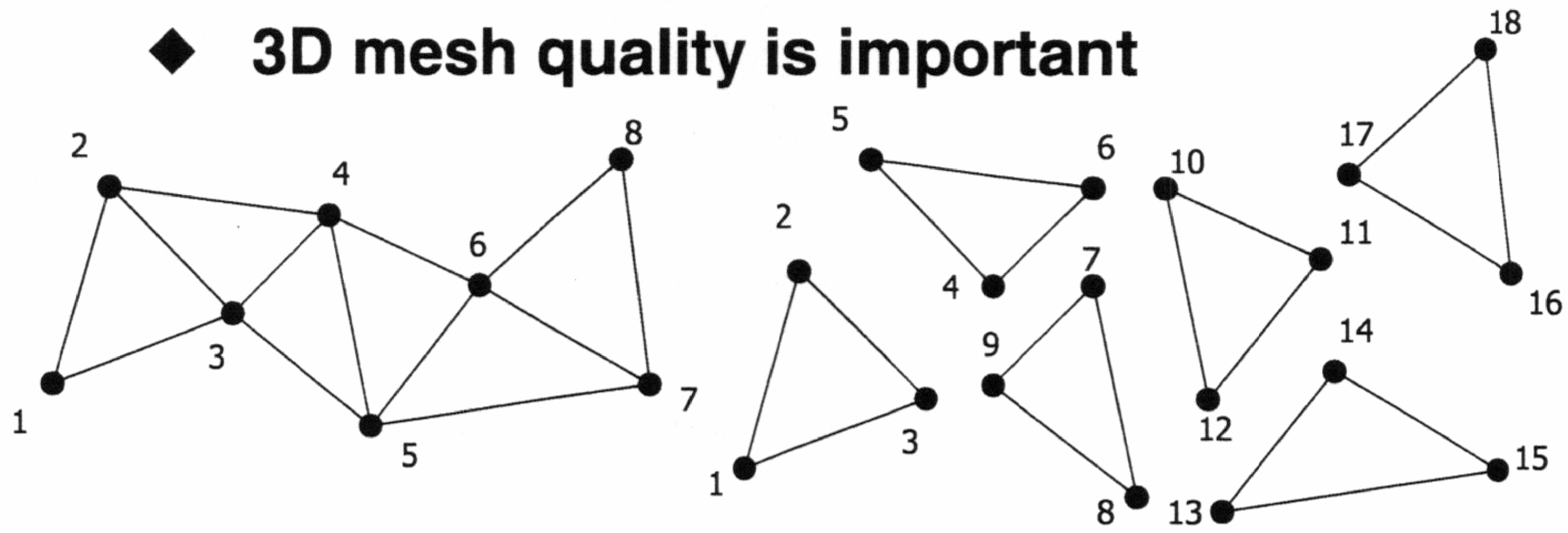


Subdivide very large objects

- Very large objects:
    - Often touch one of the clipping planes
    - You're going to test or to send lots of primitives for nothing
- Subdivide them into smaller objects
- Octree works well for those cases
- Don't create too small objects
    - Object visibility tests are costly

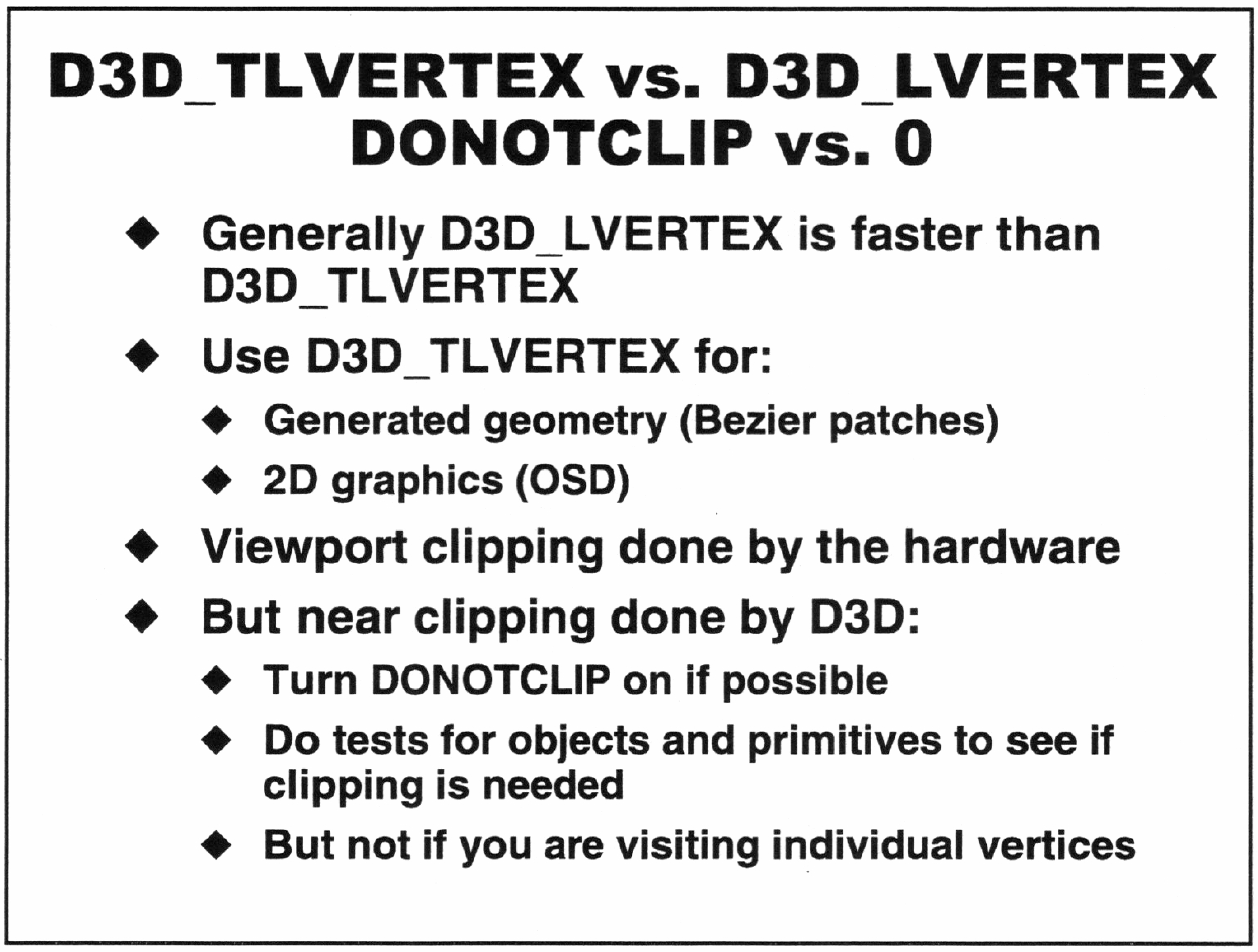
3. Geometry & performace

- Strips have (nbtriangles + 2) vertices
- Up to 3x smaller and FASTER
- But vertices must share everything:
  - States, textures & coordinates
  - Colors, illumination & normal vector
- 3D mesh quality is important



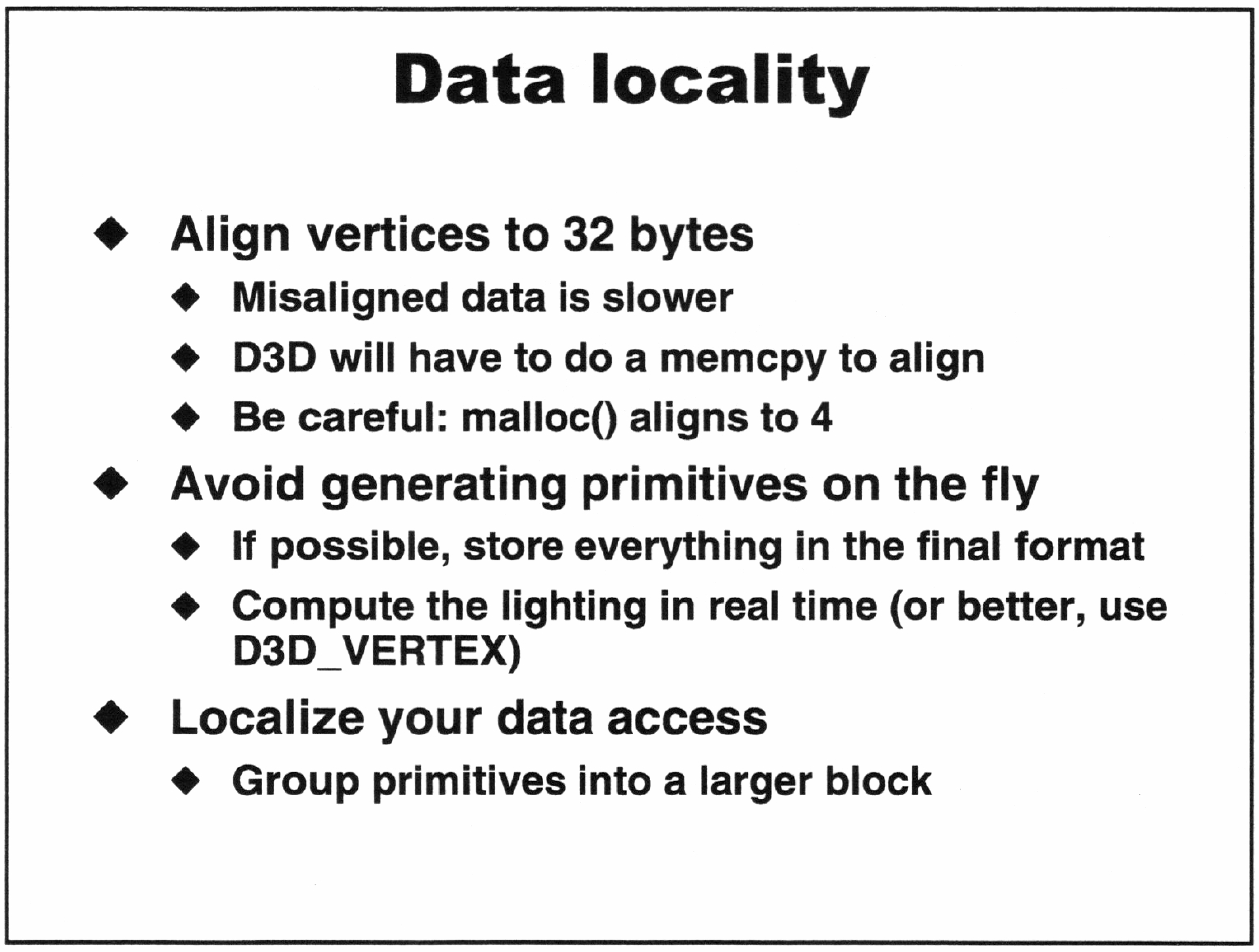
IndexedPrimitive vs. Primitive

- DrawPrimitive:
  - Don't use D3DPT_TRIANGLELIST; switch to DrawIndexedPrimitive
  - Very fast with D3DPT_TRIANGLESTRIP for very large strips
- DrawIndexedPrimitive:
  - Best solution if you don't have very large strips



D3D_TLVERTEX vs. D3D_LVERTEX DONOTCLIP vs. 0

- Generally D3D_LVERTEX is faster than D3D_TLVERTEX
- Use D3D_TLVERTEX for:
  - Generated geometry (Bezier patches)
  - 2D graphics (OSD)
- Viewport clipping done by the hardware
- But near clipping done by D3D:
  - Turn DONOTCLIP on if possible
  - Do tests for objects and primitives to see if clipping is needed
  - But not if you are visiting individual vertices



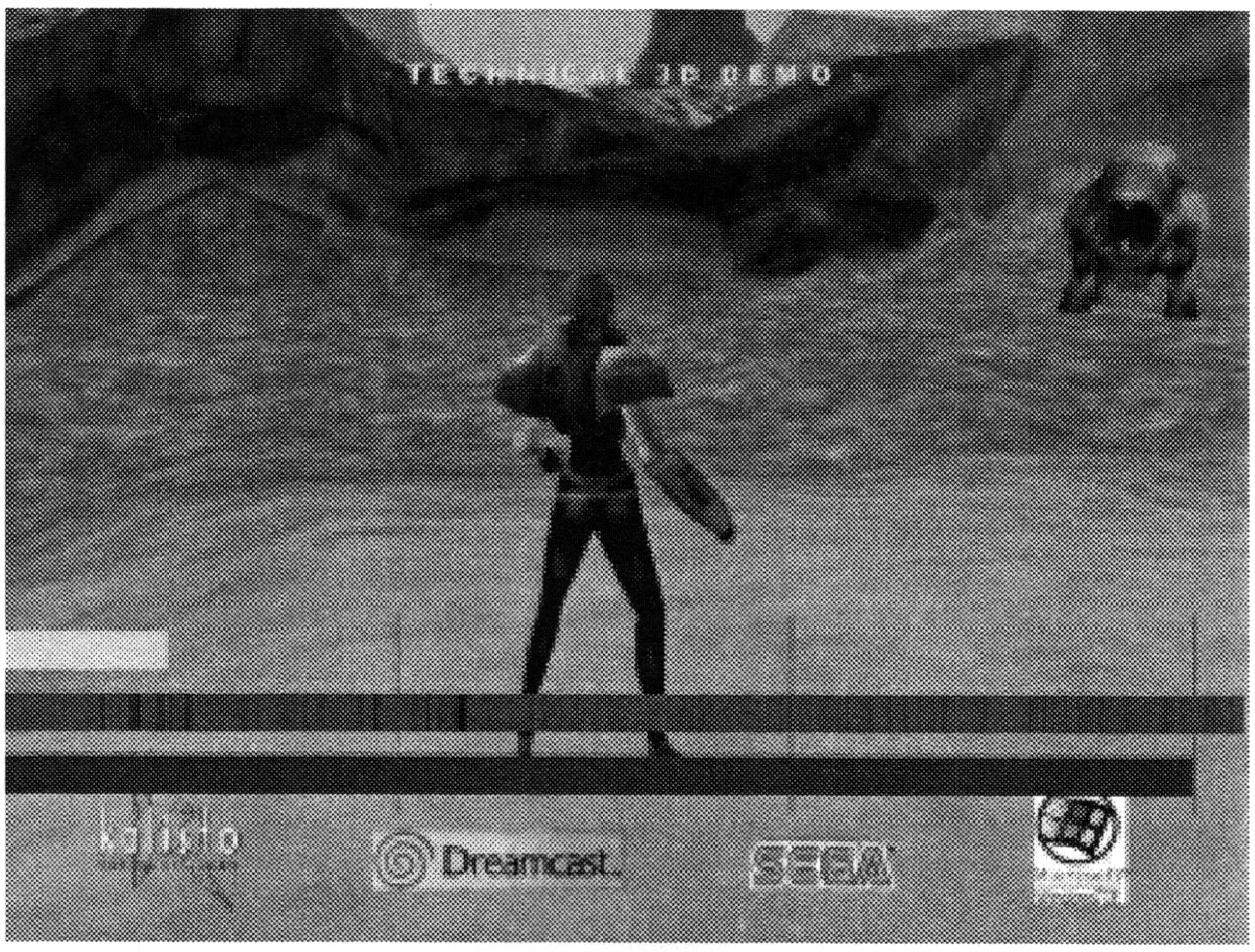
Data locality

- Align vertices to 32 bytes
  - Misaligned data is slower
  - D3D will have to do a memcpy to align
  - Be careful: malloc() aligns to 4
- Avoid generating primitives on the fly
  - If possible, store everything in the final format
  - Compute the lighting in real time (or better, use D3D_VERTEX)
- Localize your data access
  - Group primitives into a larger block

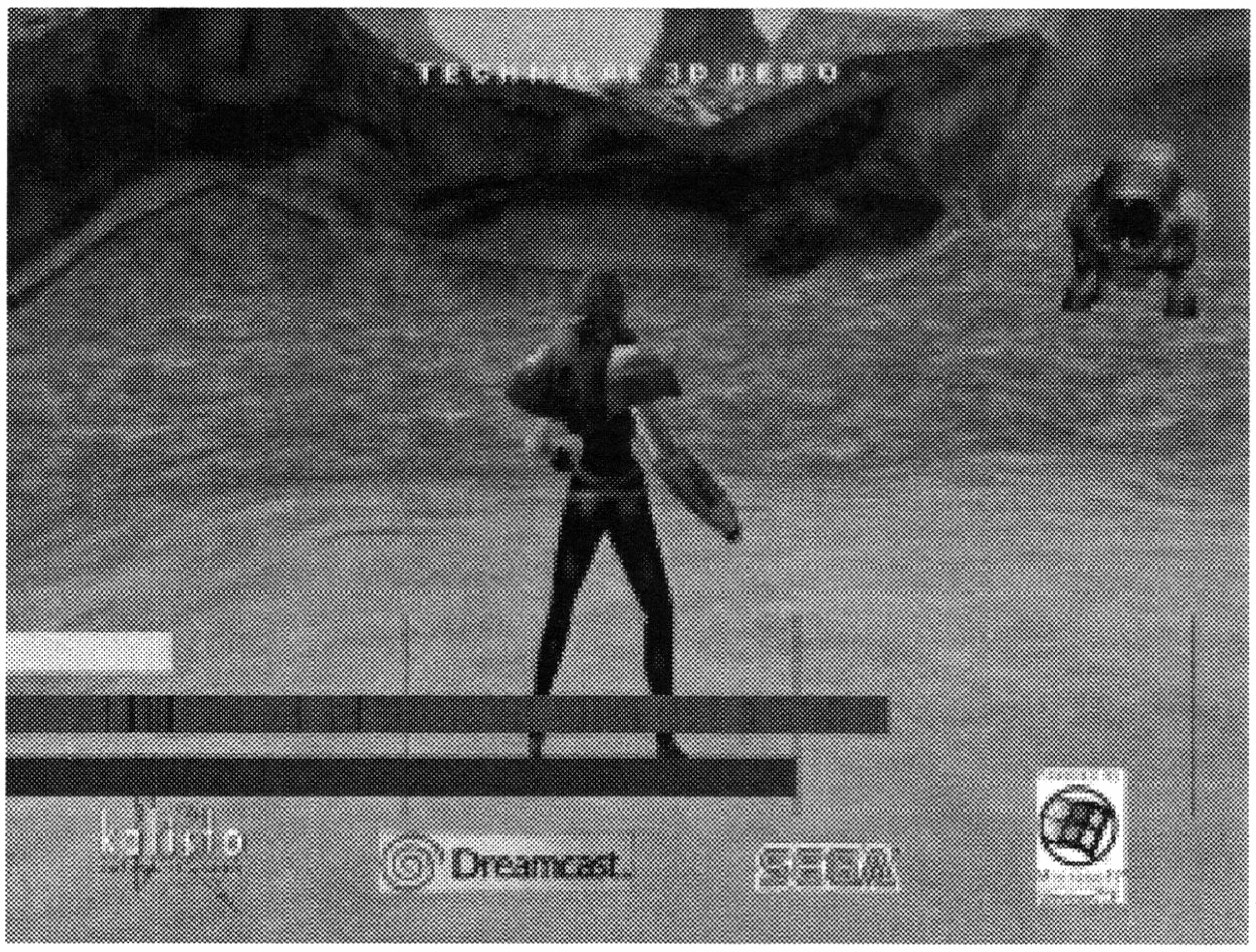
4. Let's optimize a game

- Windows CE performance viewer
- Monitors drop down menu in DCTool to activate once the game launched
- Helps find out exactly what is taking time:
  - Grey: Game code
  - Red: Direct3D, opaque polygons
  - Blue: Direct3D, transparency
  - Green: Direct3D, Punch through

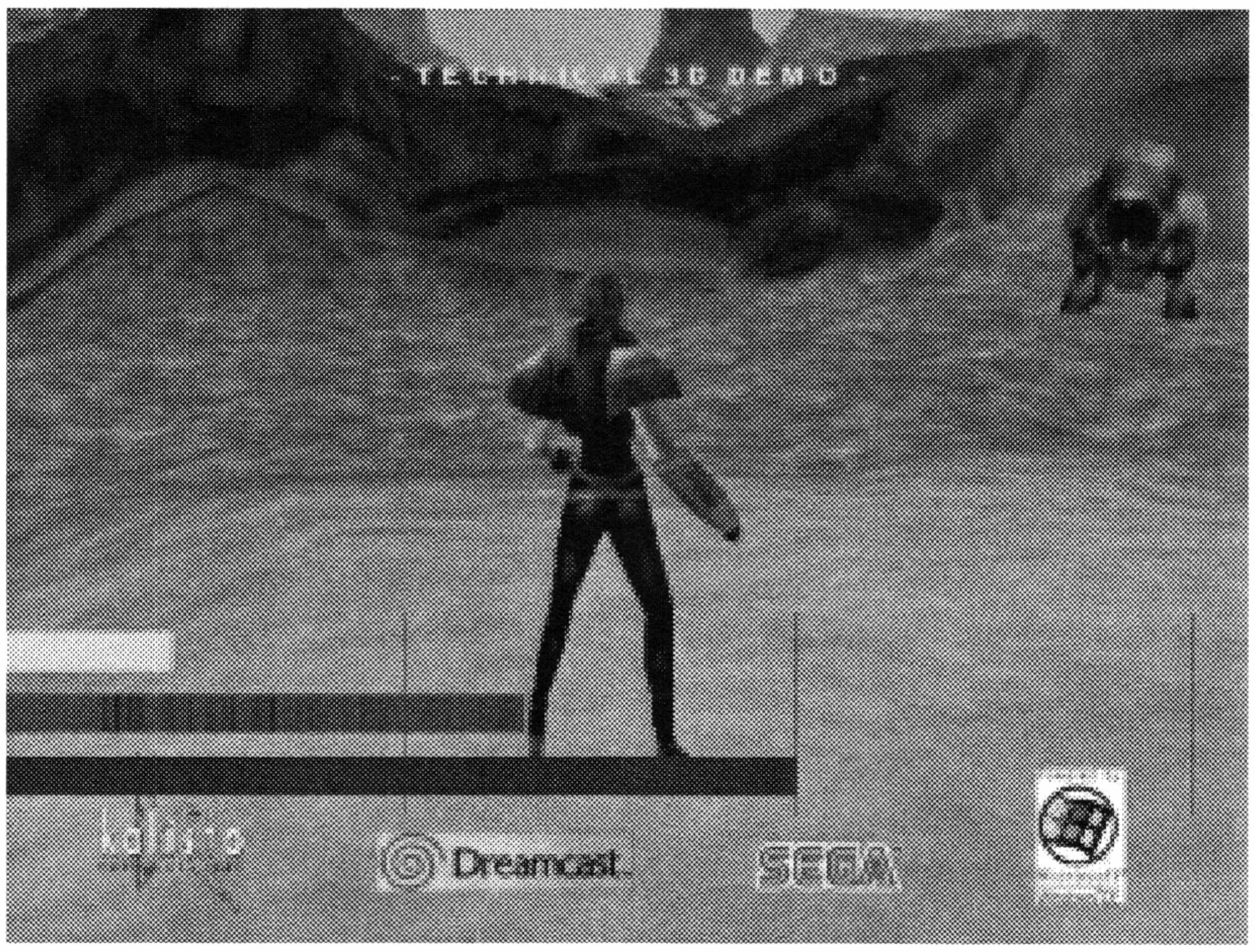
Not optimized game



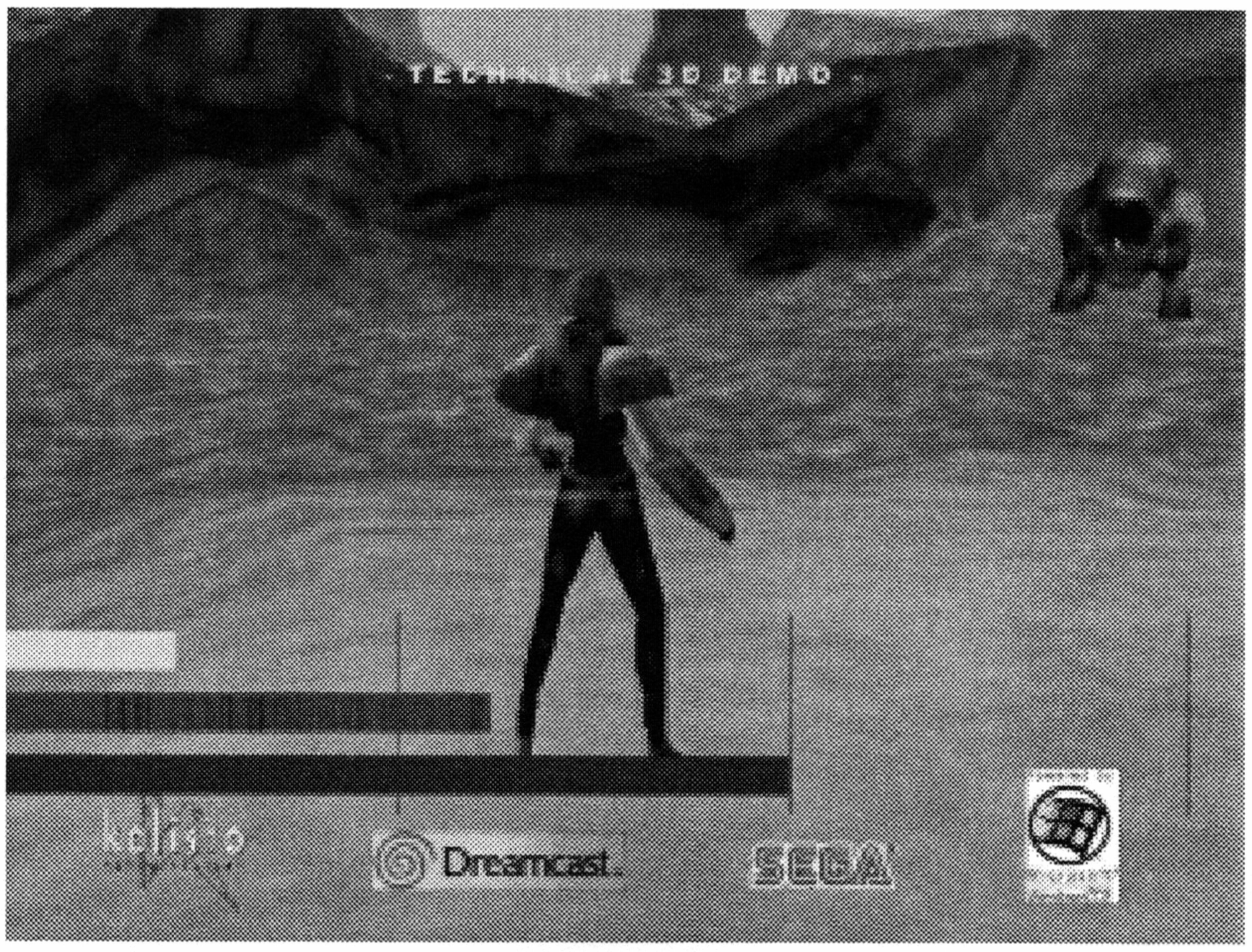
Aligned to 32 and lined up



Use large lists to reduce calls



Generate strips

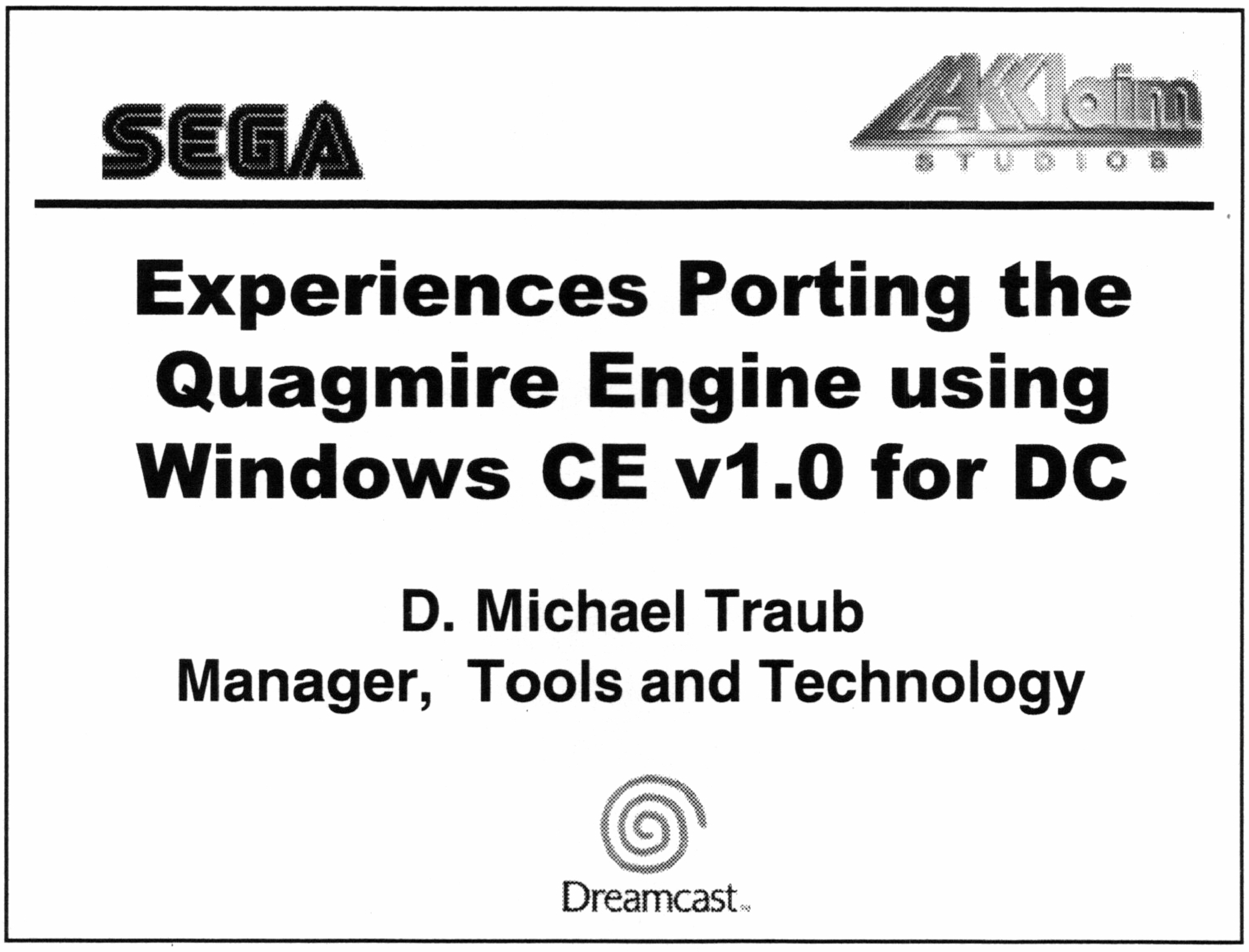


SEGA

Experiences Porting the Quagmire Engine using WindowsCE v1.0 for DC

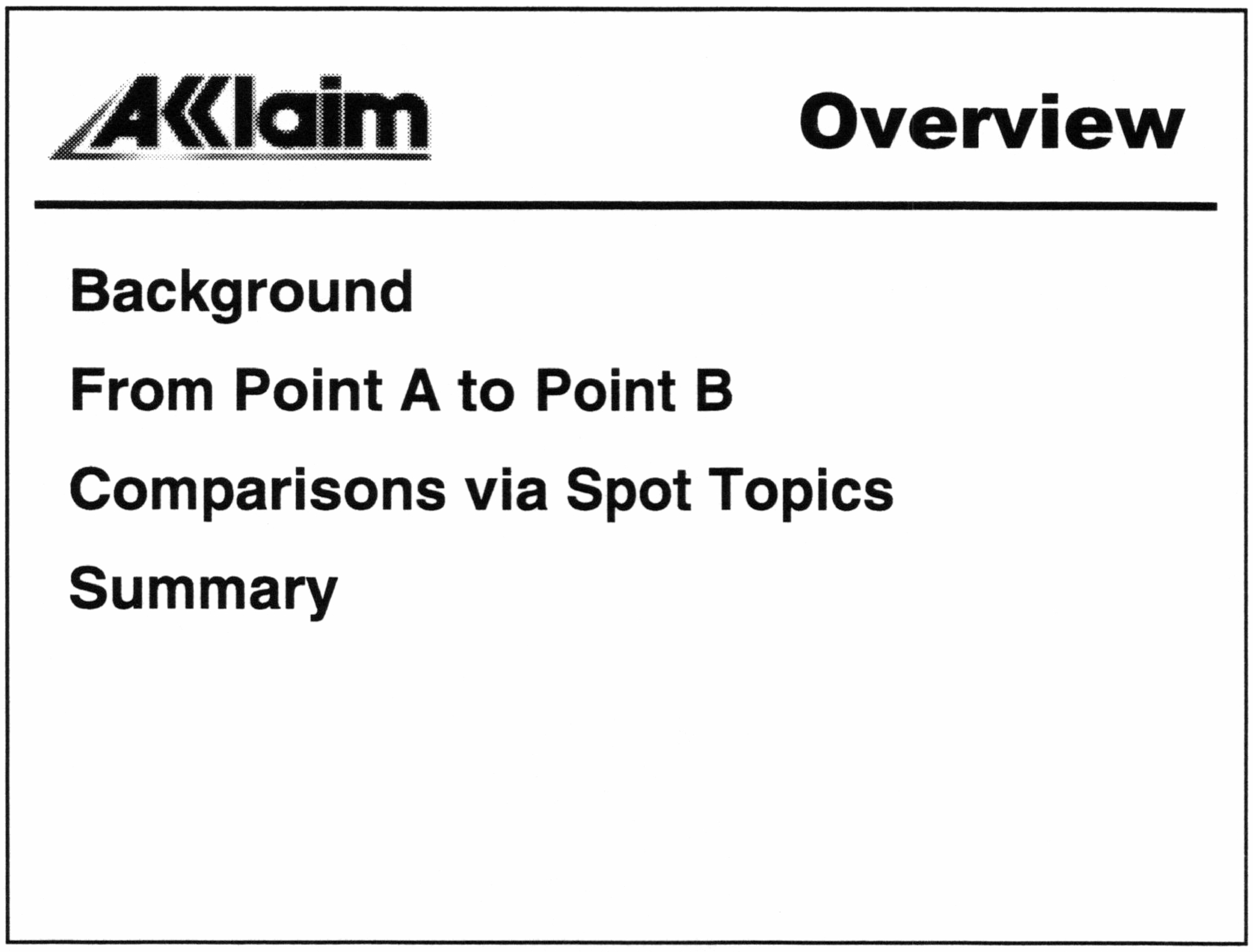
D. Michael Traub
Manager, Tools and Technology
Acclaim

Dreamcast™



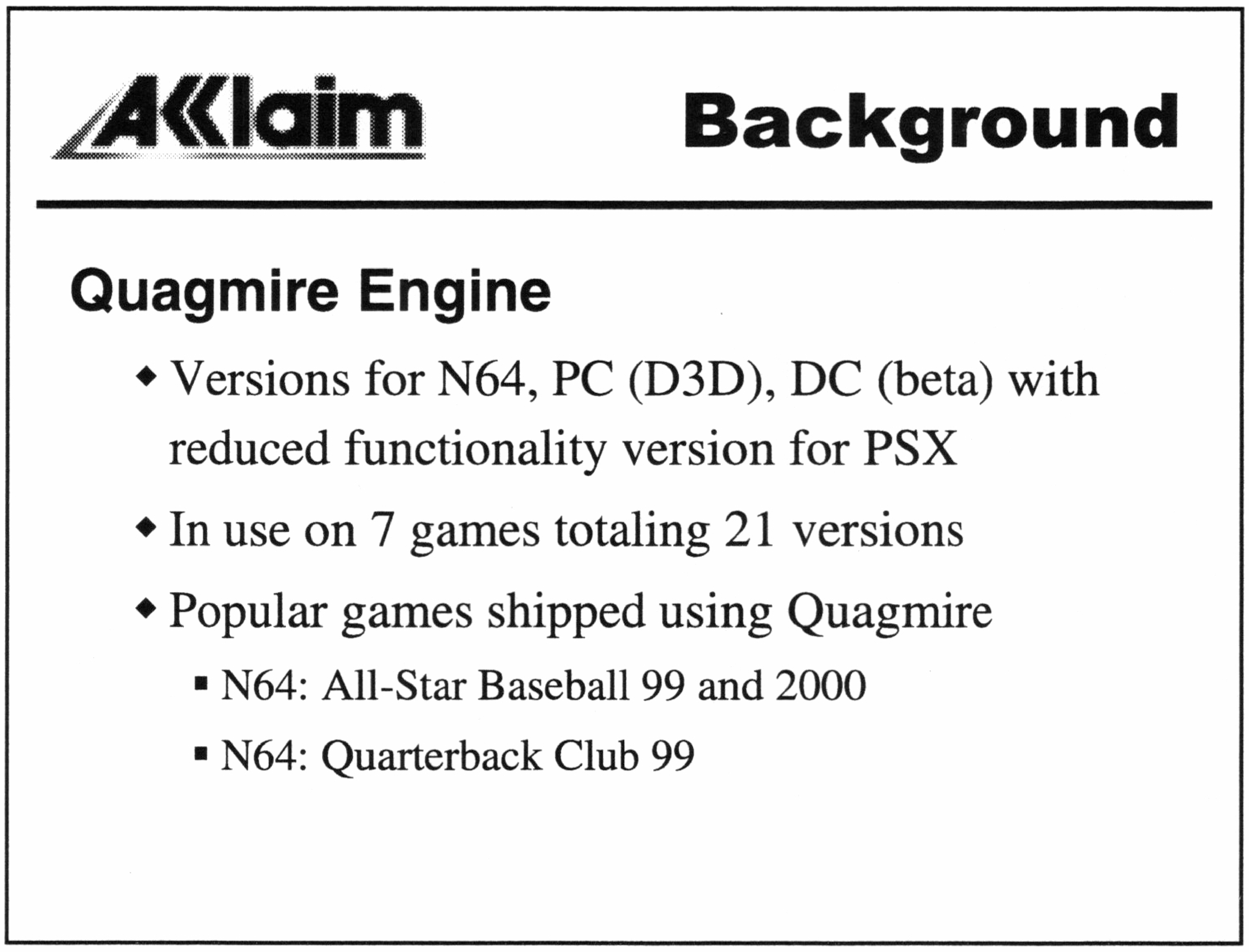
Experiences Porting the Quagmire Engine using Windows CE v1.0 for DC

D. Michael Traub
Manager, Tools and Technology



Overview

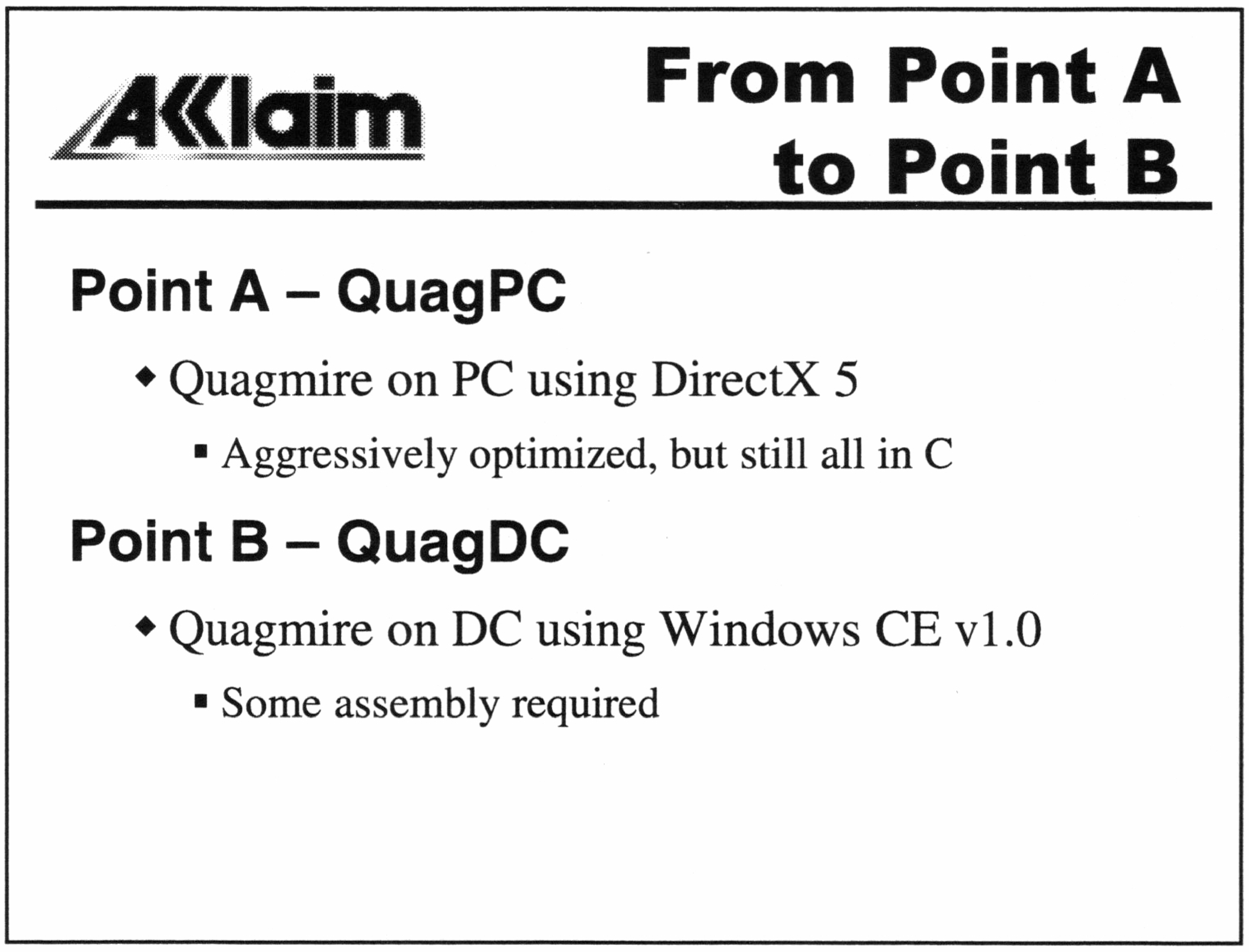
- Background
- From Point A to Point B
- Comparisons via Spot Topics
- Summary



Background

Quagmire Engine

- Versions for N64, PC (D3D), DC (beta) with reduced functionality version for PSX
- In use on 7 games totaling 21 versions
- Popular games shipped using Quagmire
  - N64: All-Star Baseball 99 and 2000
  - N64: Quarterback Club 99



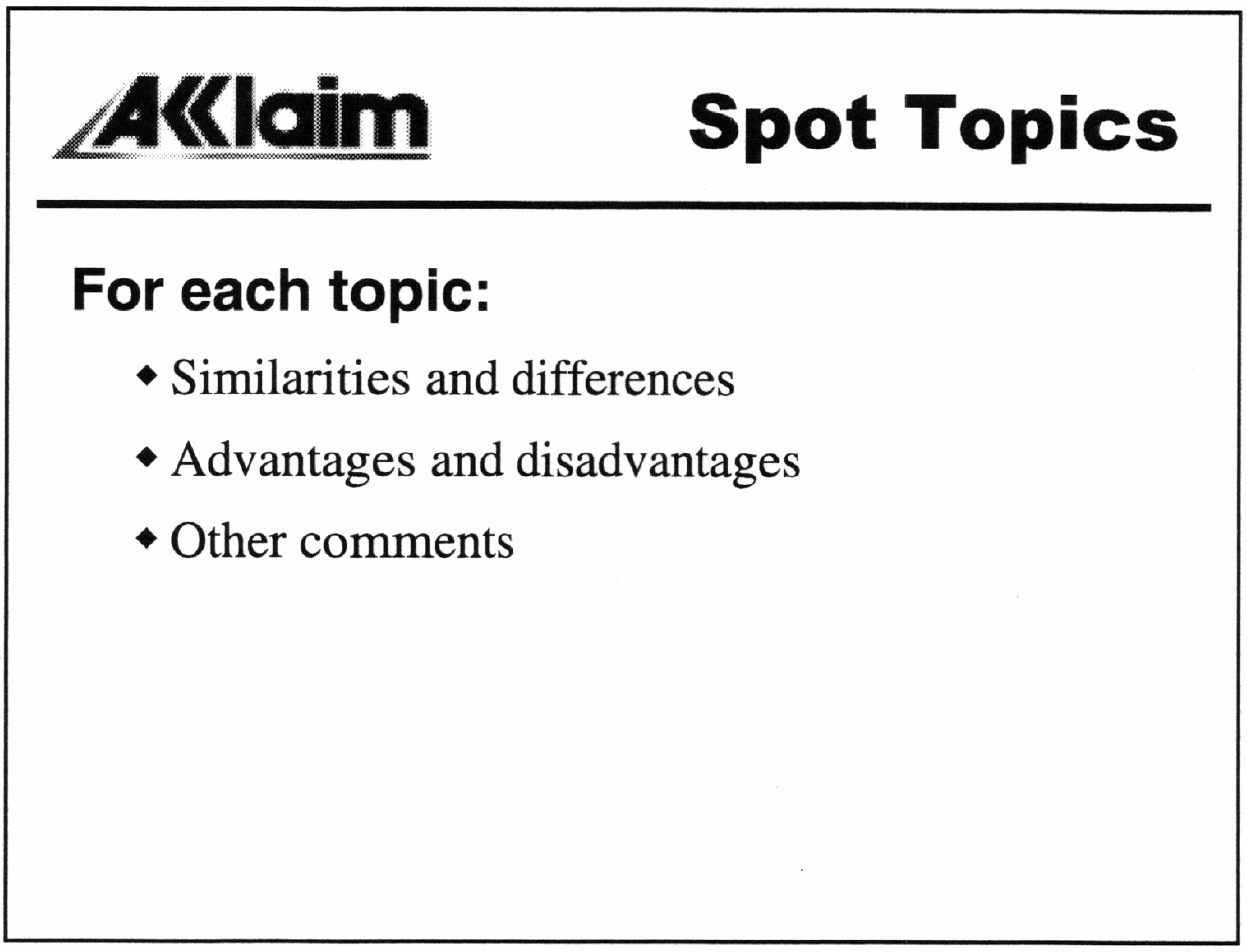
From Point A to Point B

Point A – QuagPC

- Quagmire on PC using DirectX 5
  - Aggressively optimized, but still all in C

Point B – QuagDC

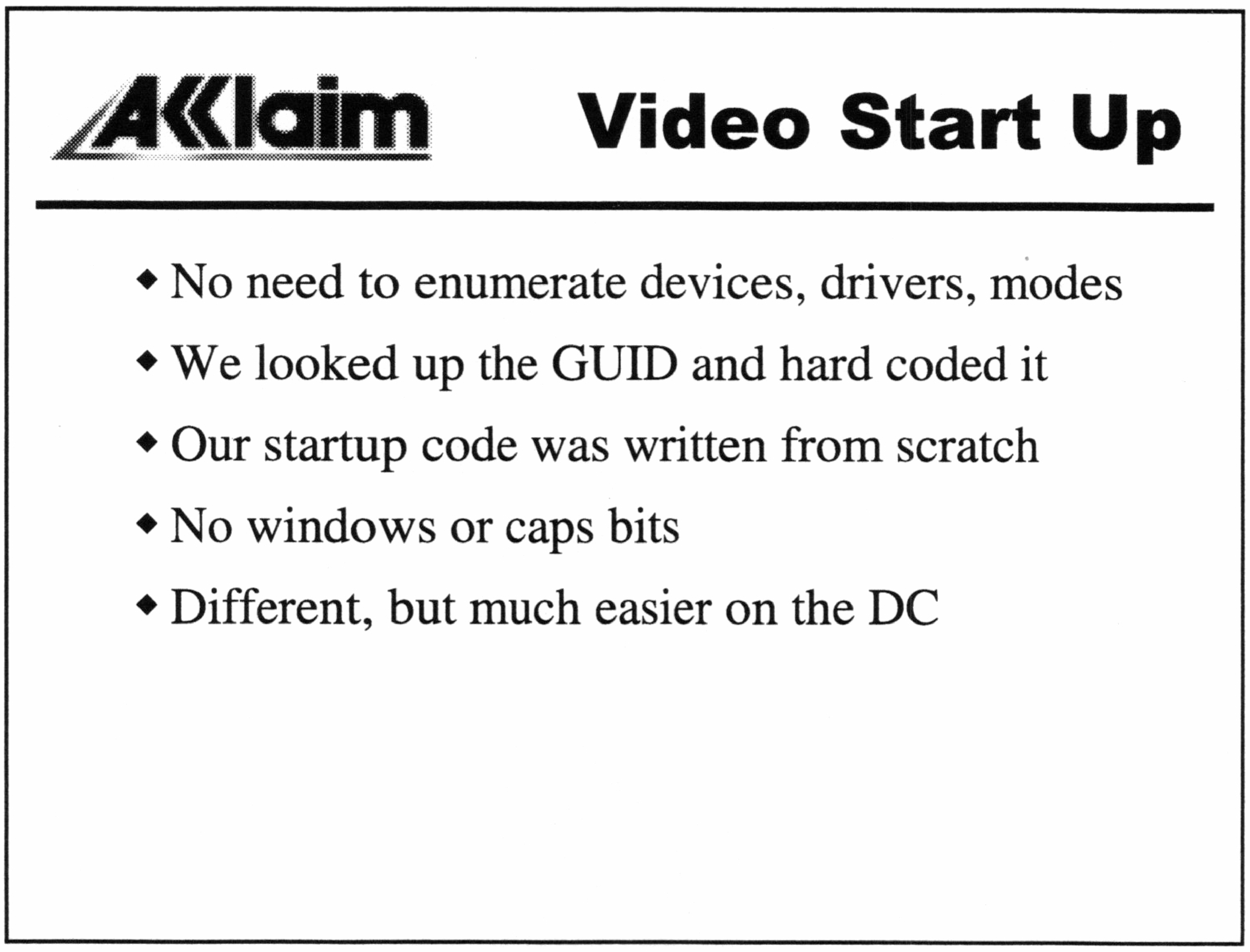
- Quagmire on DC using Windows CE v1.0
  - Some assembly required



Spot Topics

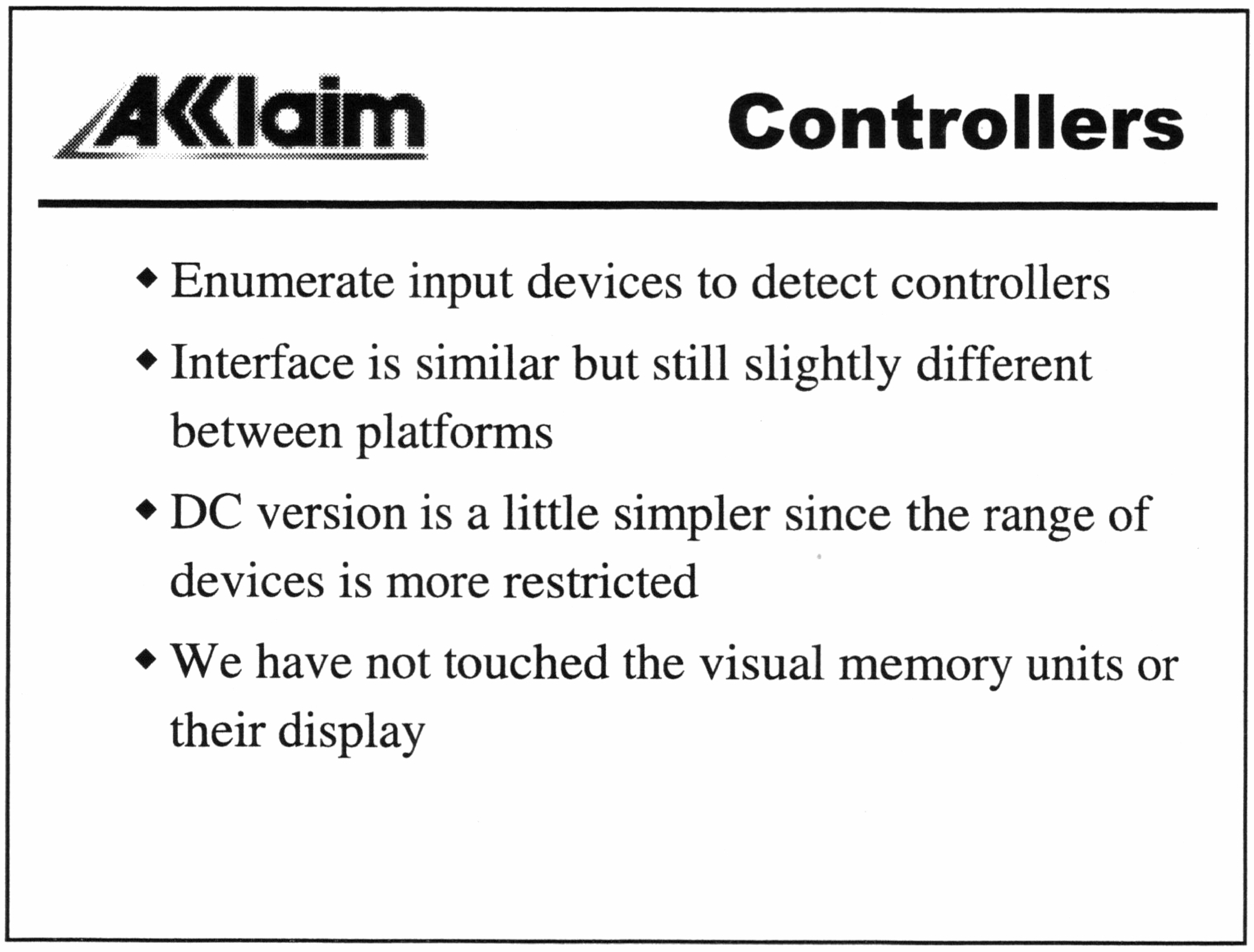
For each topic:

- Similarities and differences
- Advantages and disadvantages
- Other comments



Video Start Up

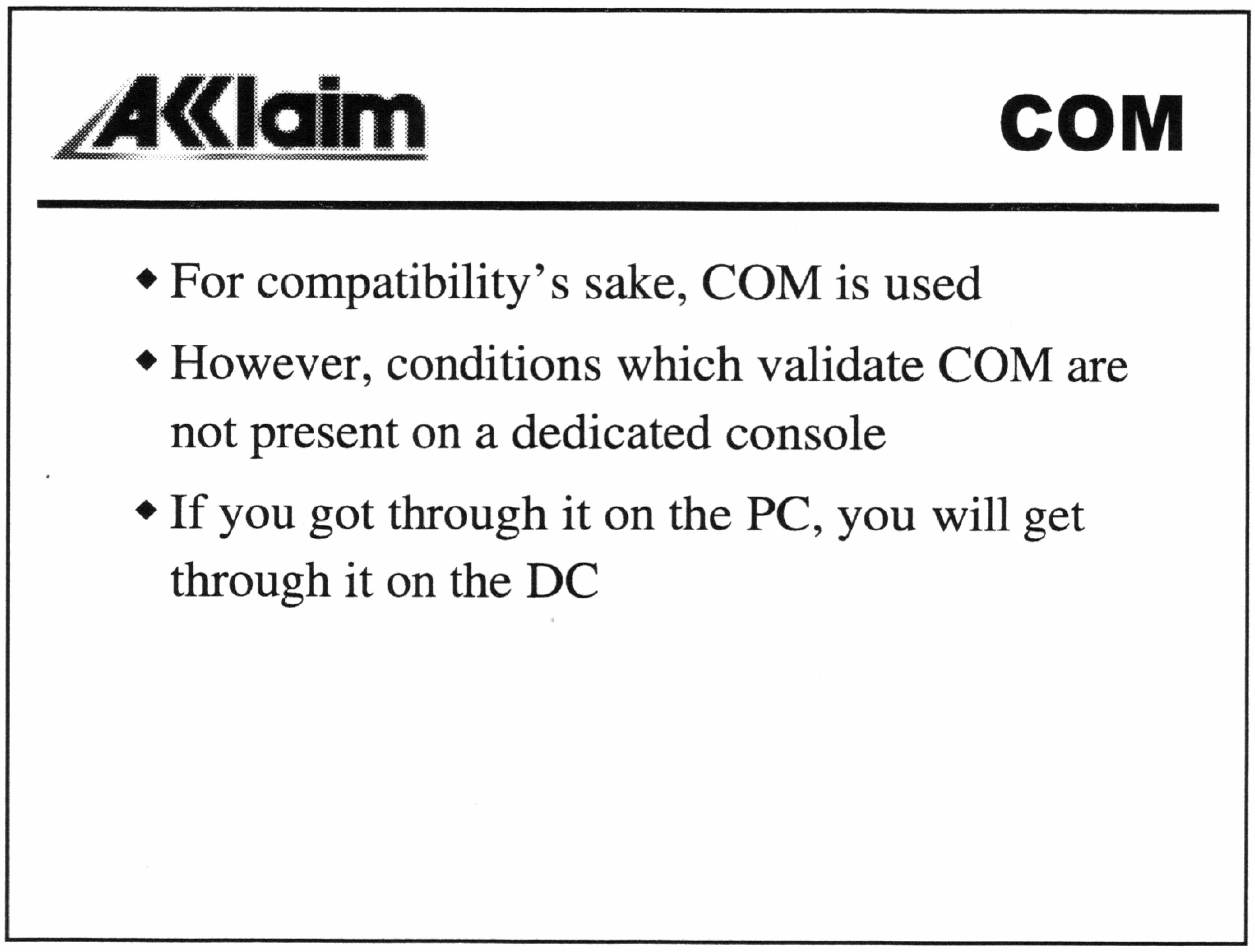
- No need to enumerate devices, drivers, modes
- We looked up the GUID and hard coded it
- Our startup code was written from scratch
- No windows or caps bits
- Different, but much easier on the DC



Acclaim

Controllers

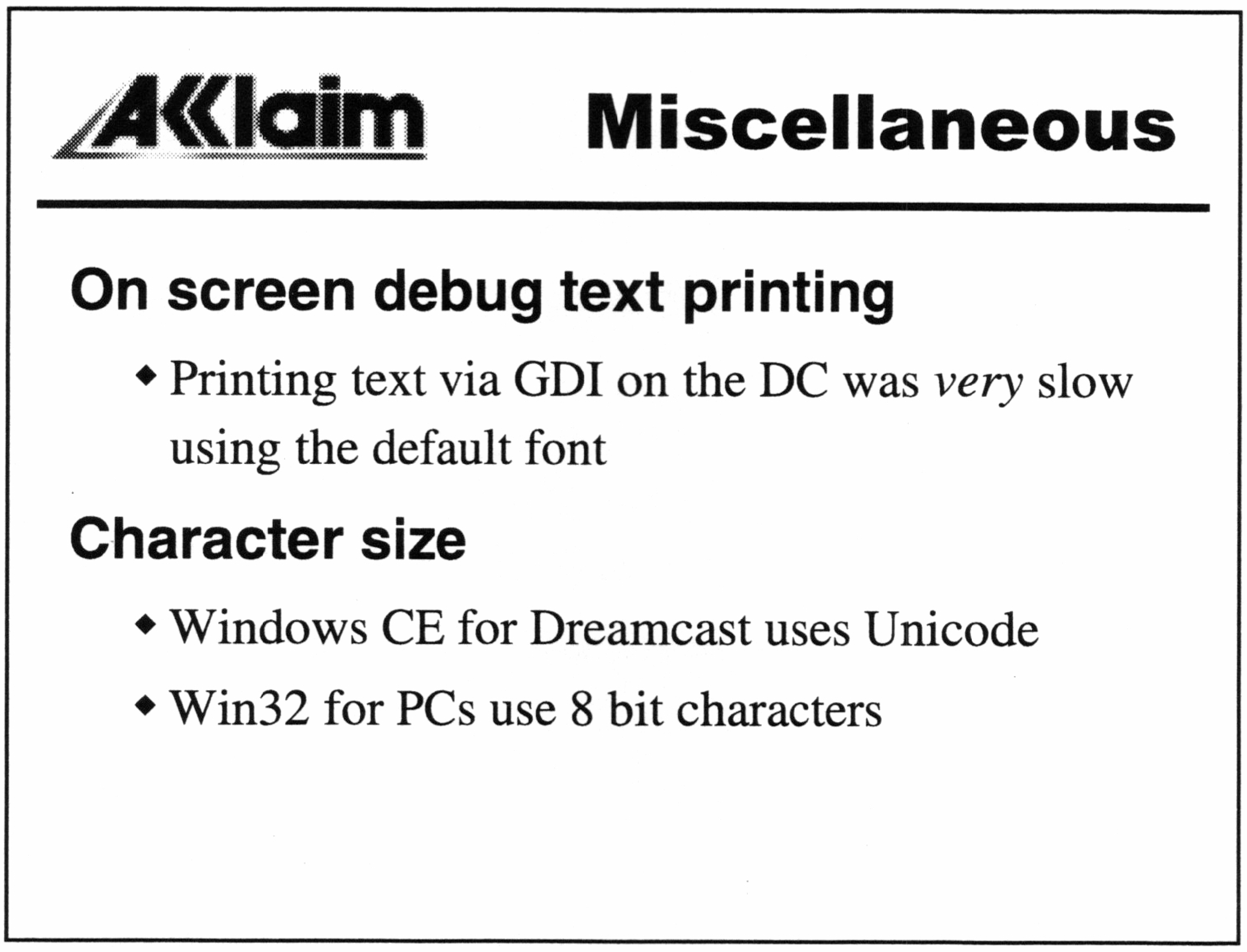
- Enumerate input devices to detect controllers
- Interface is similar but still slightly different between platforms
- DC version is a little simpler since the range of devices is more restricted
- We have not touched the visual memory units or their display



Acclaim

COM

- For compatibility's sake, COM is used
- However, conditions which validate COM are not present on a dedicated console
- If you got through it on the PC, you will get through it on the DC



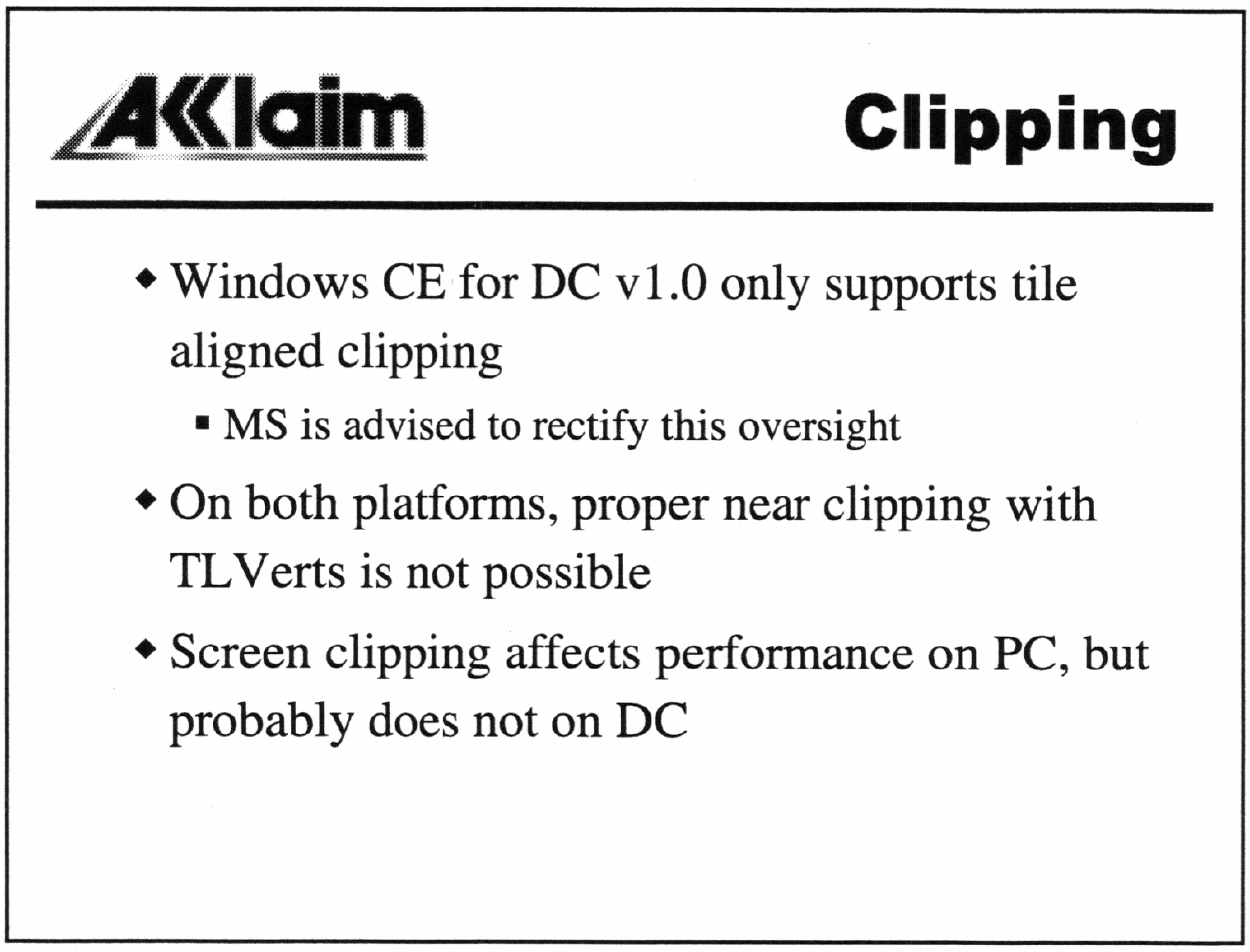
Miscellaneous

On screen debug text printing

- Printing text via GDI on the DC was *very* slow using the default font

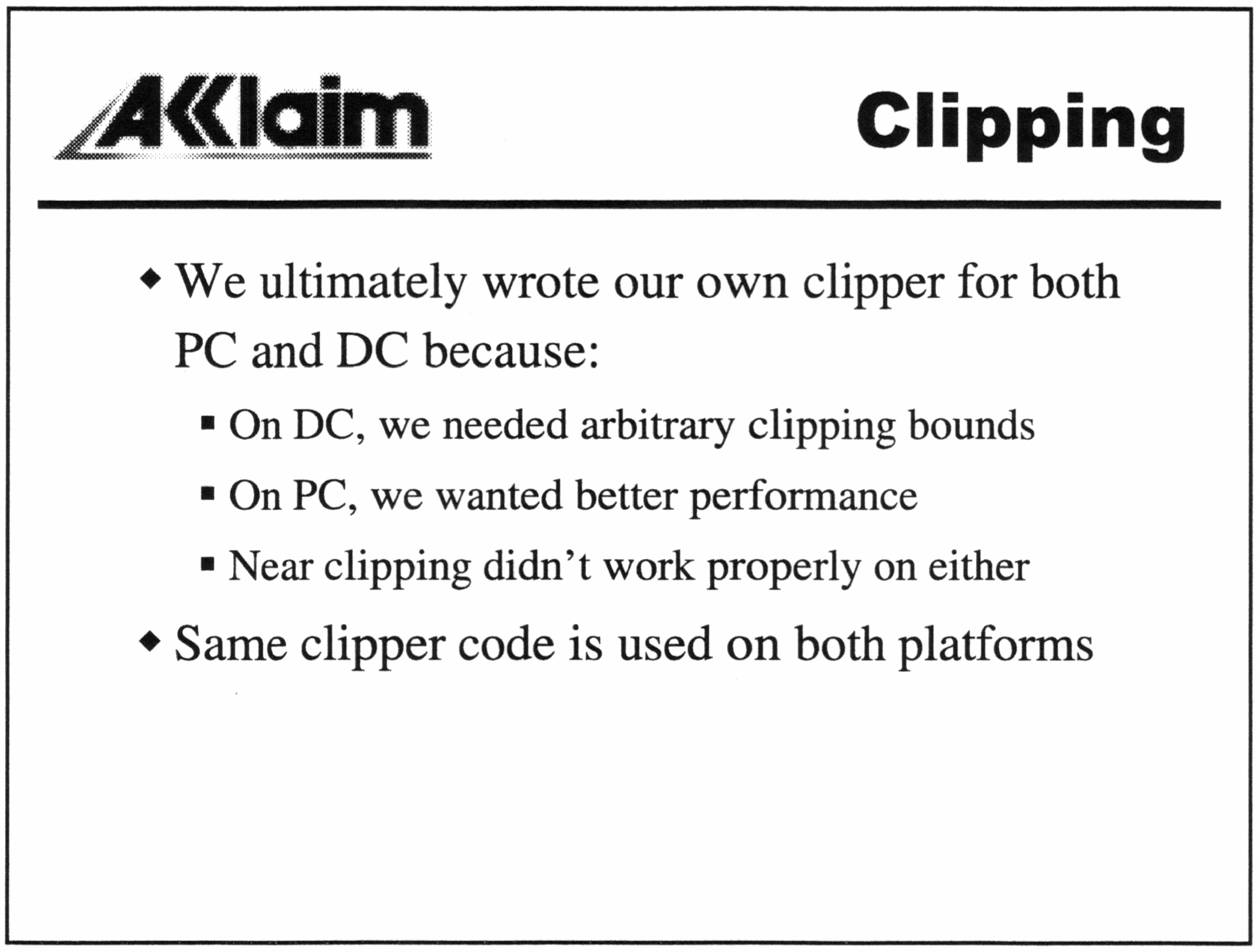
Character size

- Windows CE for Dreamcast uses Unicode
- Win32 for PCs use 8 bit characters



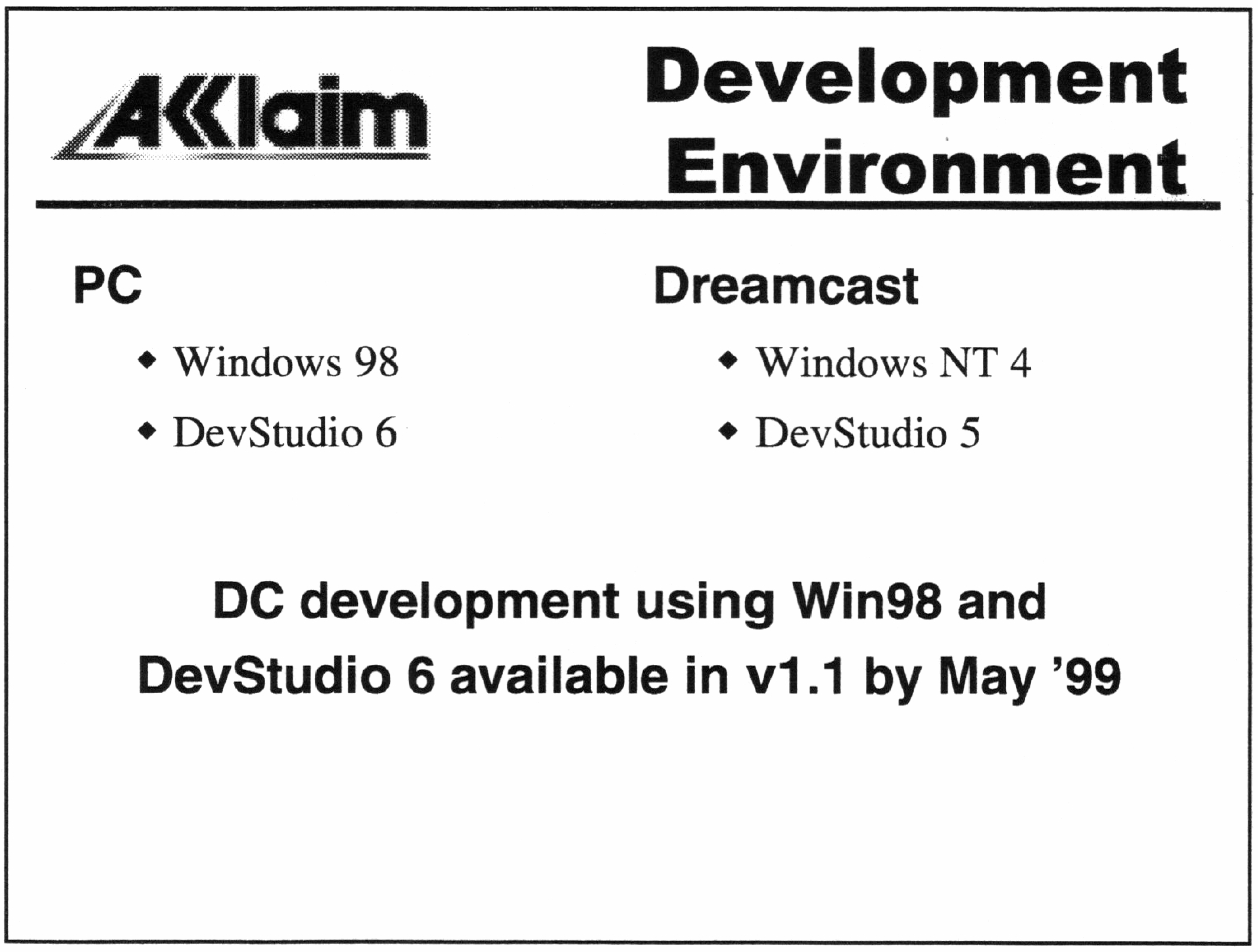
Clipping

- Windows CE for DC v1.0 only supports tile aligned clipping
  - MS is advised to rectify this oversight
- On both platforms, proper near clipping with TLVerts is not possible
- Screen clipping affects performance on PC, but probably does not on DC



Clipping

- We ultimately wrote our own clipper for both PC and DC because:
  - On DC, we needed arbitrary clipping bounds
  - On PC, we wanted better performance
  - Near clipping didn't work properly on either
- Same clipper code is used on both platforms



Development Environment

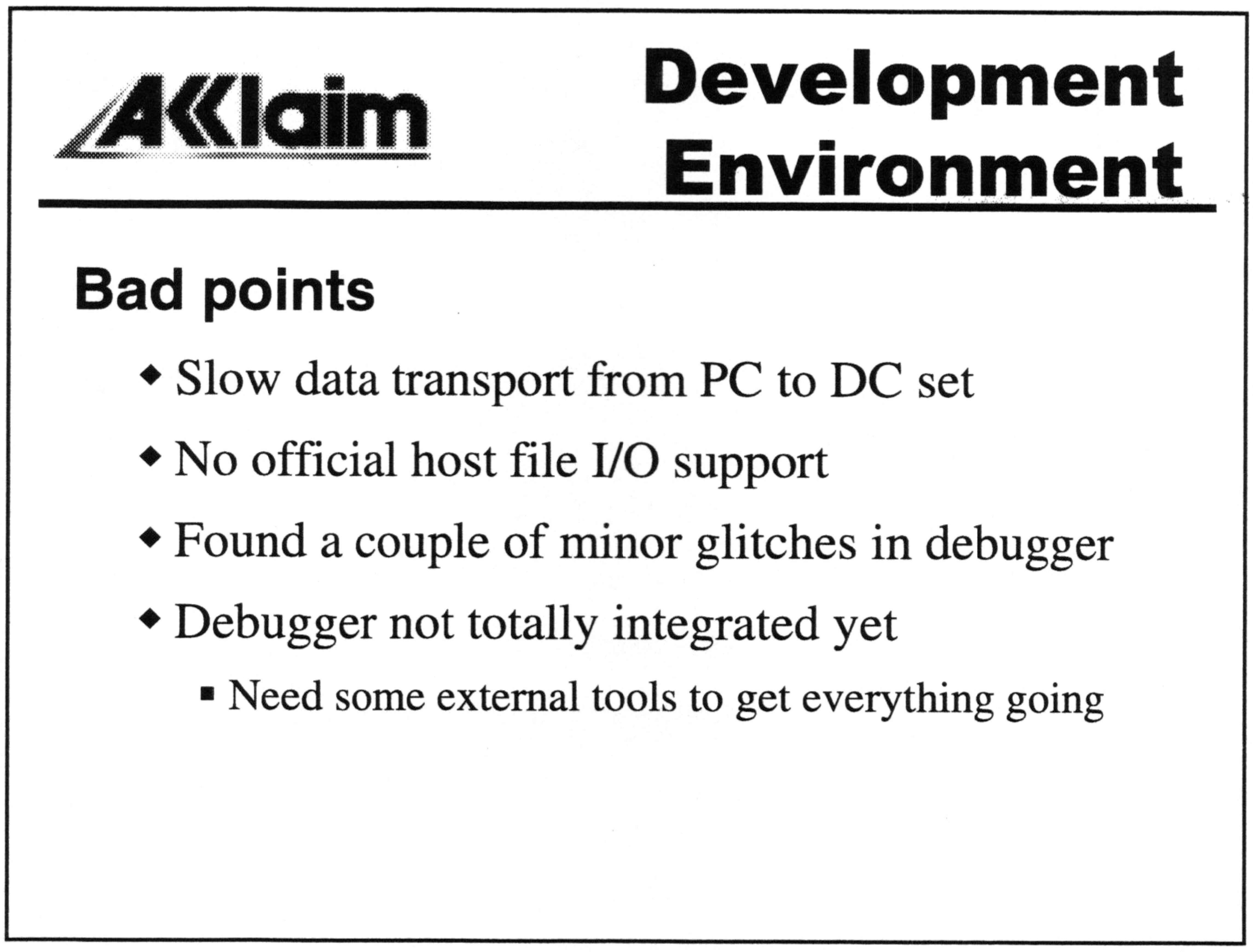
PC

- Windows 98
- DevStudio 6

Dreamcast

- Windows NT 4
- DevStudio 5

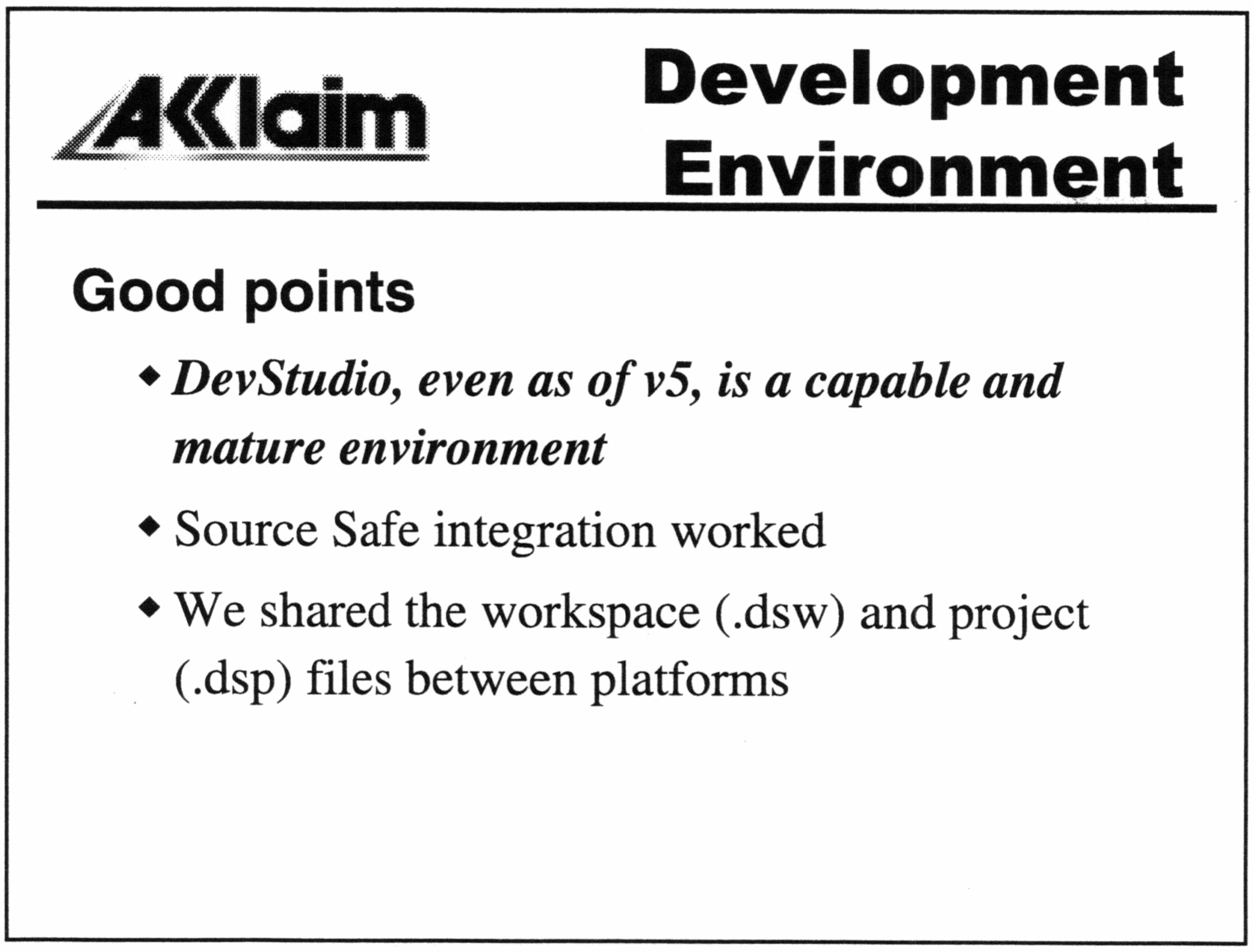
DC development using Win98 and DevStudio 6 available in v1.1 by May '99



Development Environment

Bad points

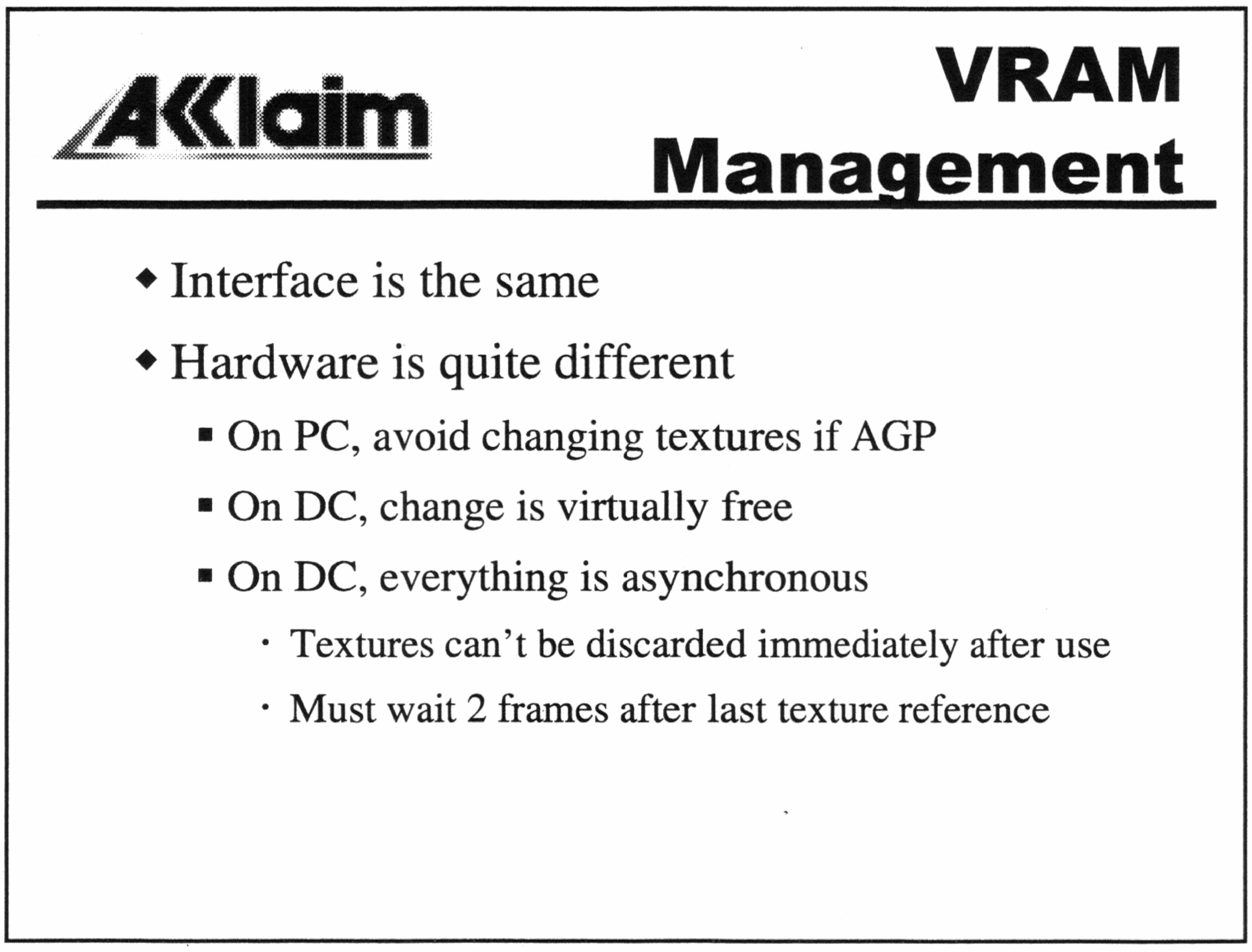
- Slow data transport from PC to DC set
- No official host file I/O support
- Found a couple of minor glitches in debugger
- Debugger not totally integrated yet
  - Need some external tools to get everything going



Development Environment

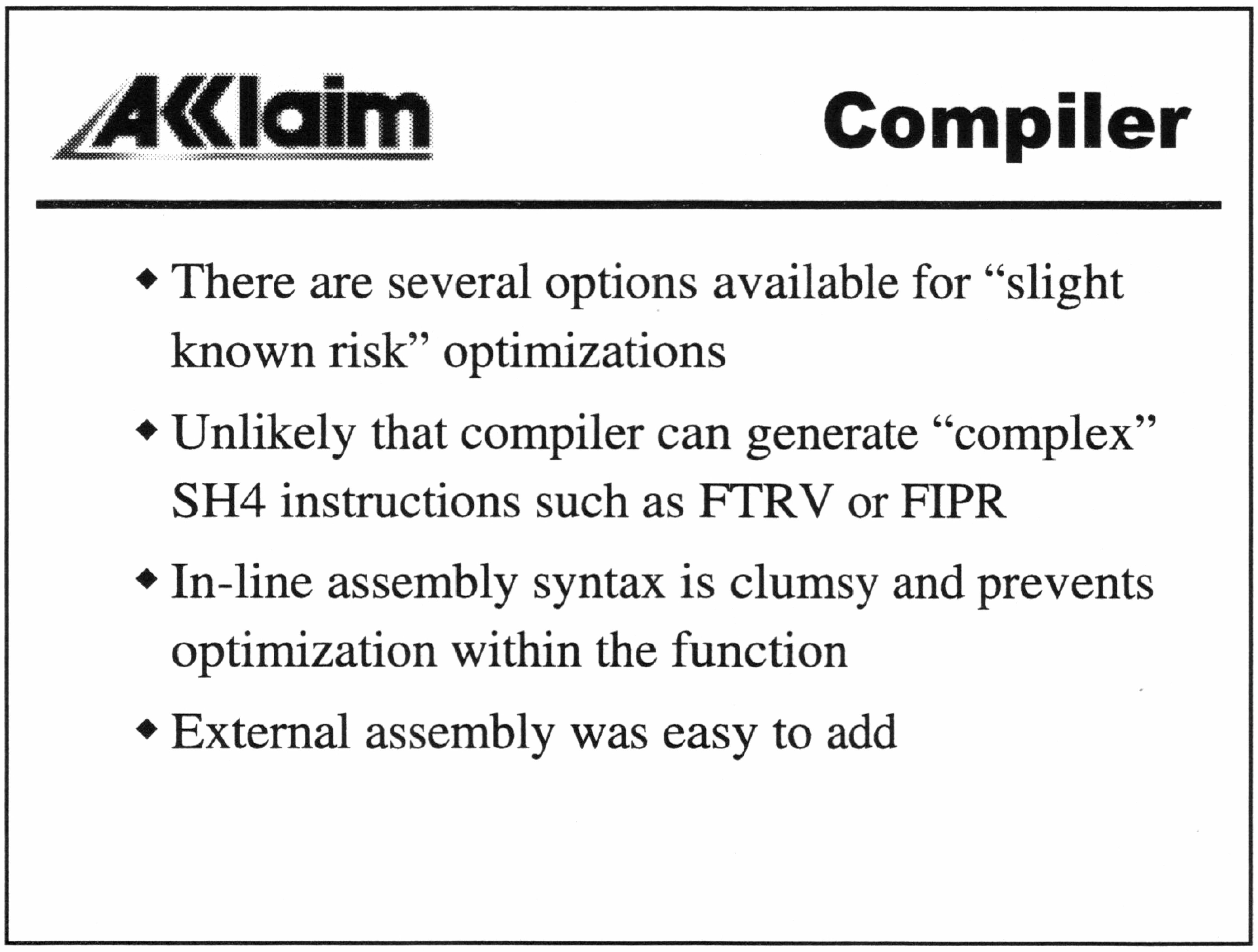
Good points

- ***DevStudio, even as of v5, is a capable and mature environment***
- Source Safe integration worked
- We shared the workspace (.dsw) and project (.dsp) files between platforms



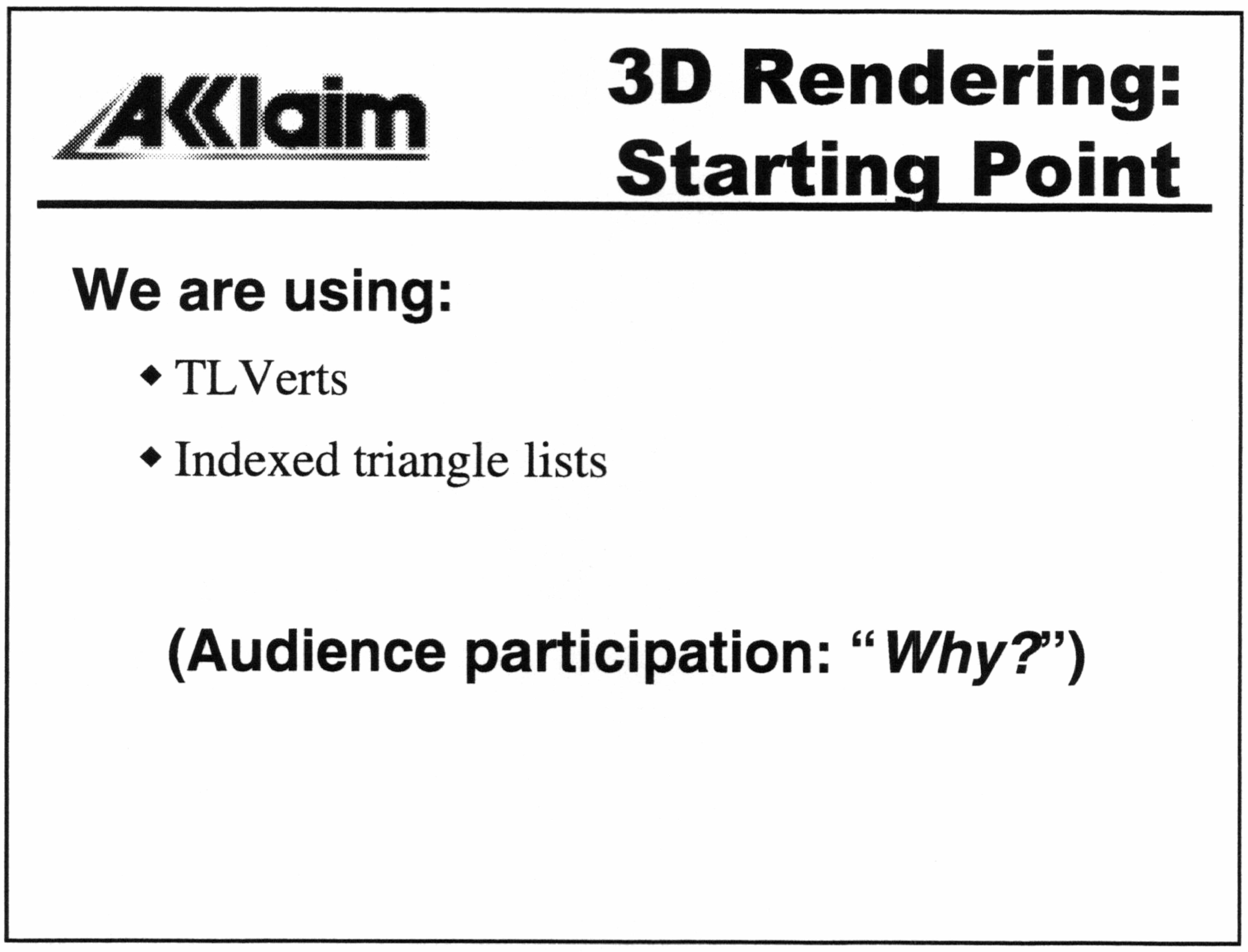
VRAM Management

- Interface is the same
- Hardware is quite different
  - On PC, avoid changing textures if AGP
  - On DC, change is virtually free
  - On DC, everything is asynchronous
    - Textures can’t be discarded immediately after use
    - Must wait 2 frames after last texture reference



Compiler

- There are several options available for “slight known risk” optimizations
- Unlikely that compiler can generate “complex” SH4 instructions such as FTRV or FIPR
- In-line assembly syntax is clumsy and prevents optimization within the function
- External assembly was easy to add

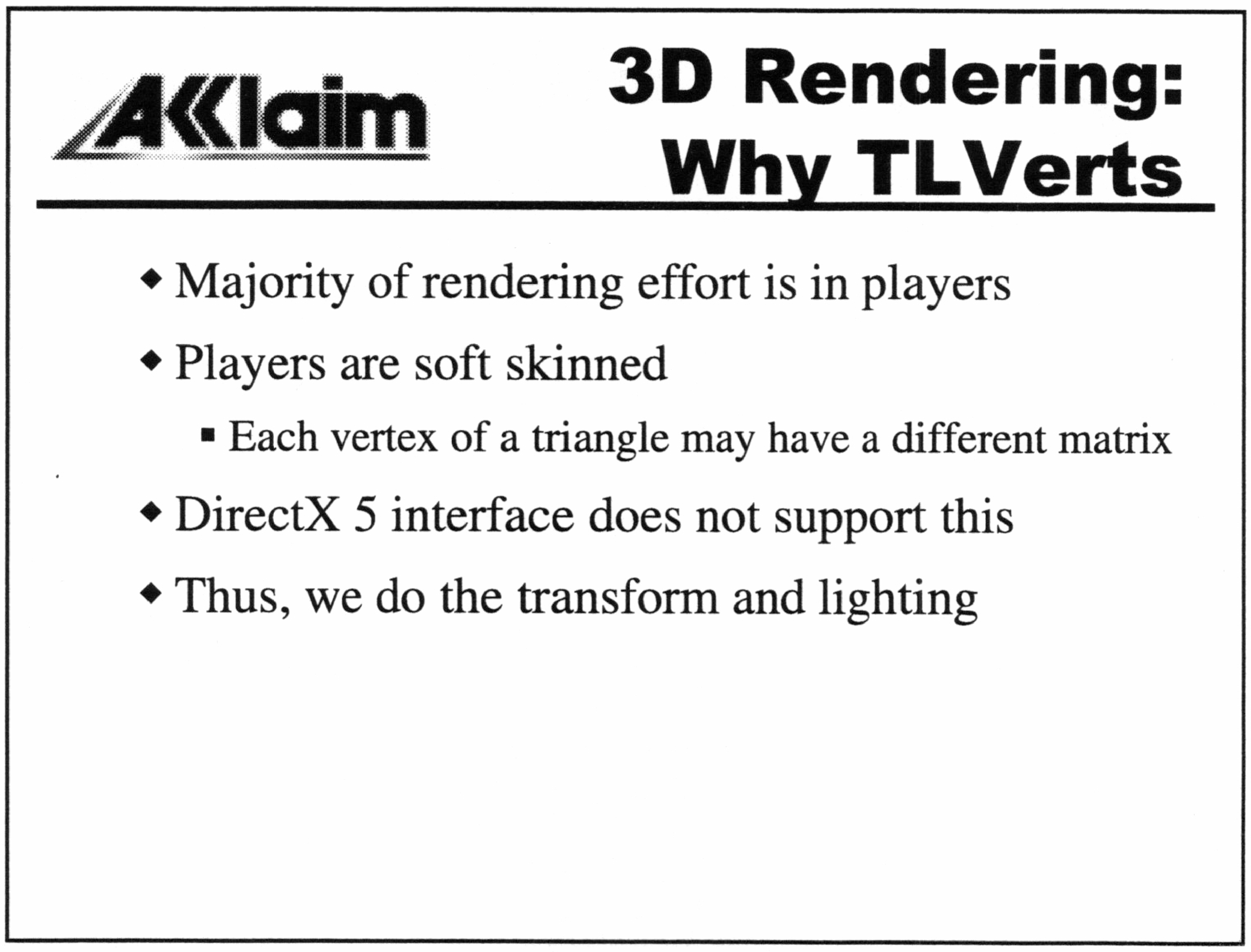


3D Rendering: Starting Point

We are using:

- TLVerts
- Indexed triangle lists

(Audience participation: "*Why?*")



3D Rendering: Why TLVerts

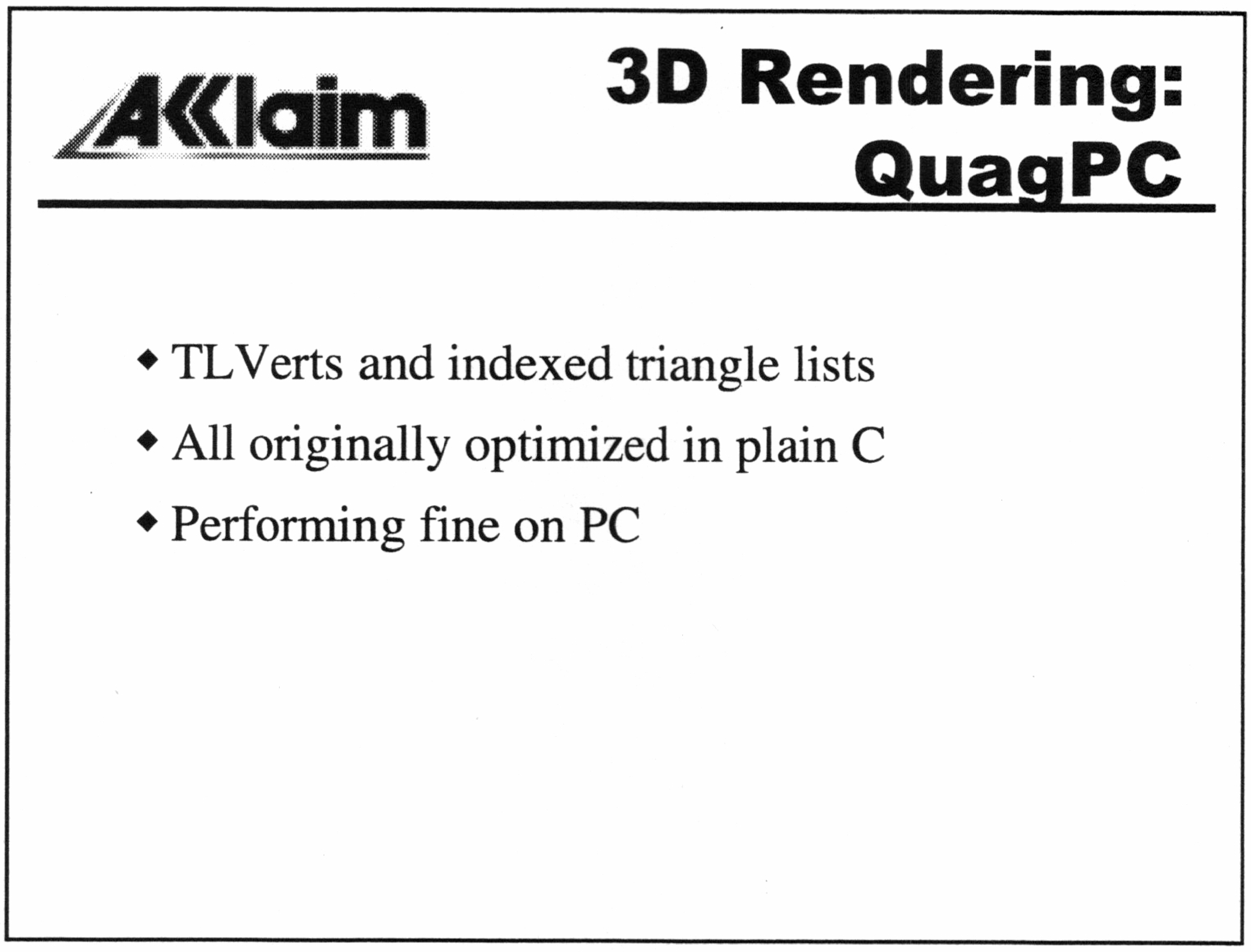
- Majority of rendering effort is in players
- Players are soft skinned
  - Each vertex of a triangle may have a different matrix
- DirectX 5 interface does not support this
- Thus, we do the transform and lighting

3D Rendering: Why I.T.List

- Performance affected by triangle count *and* vertex count
- IMPORTANT: Originally, we were using the clipping provided by D3D
- Collections of triangles having compatible attributes cannot necessarily all be in one strip

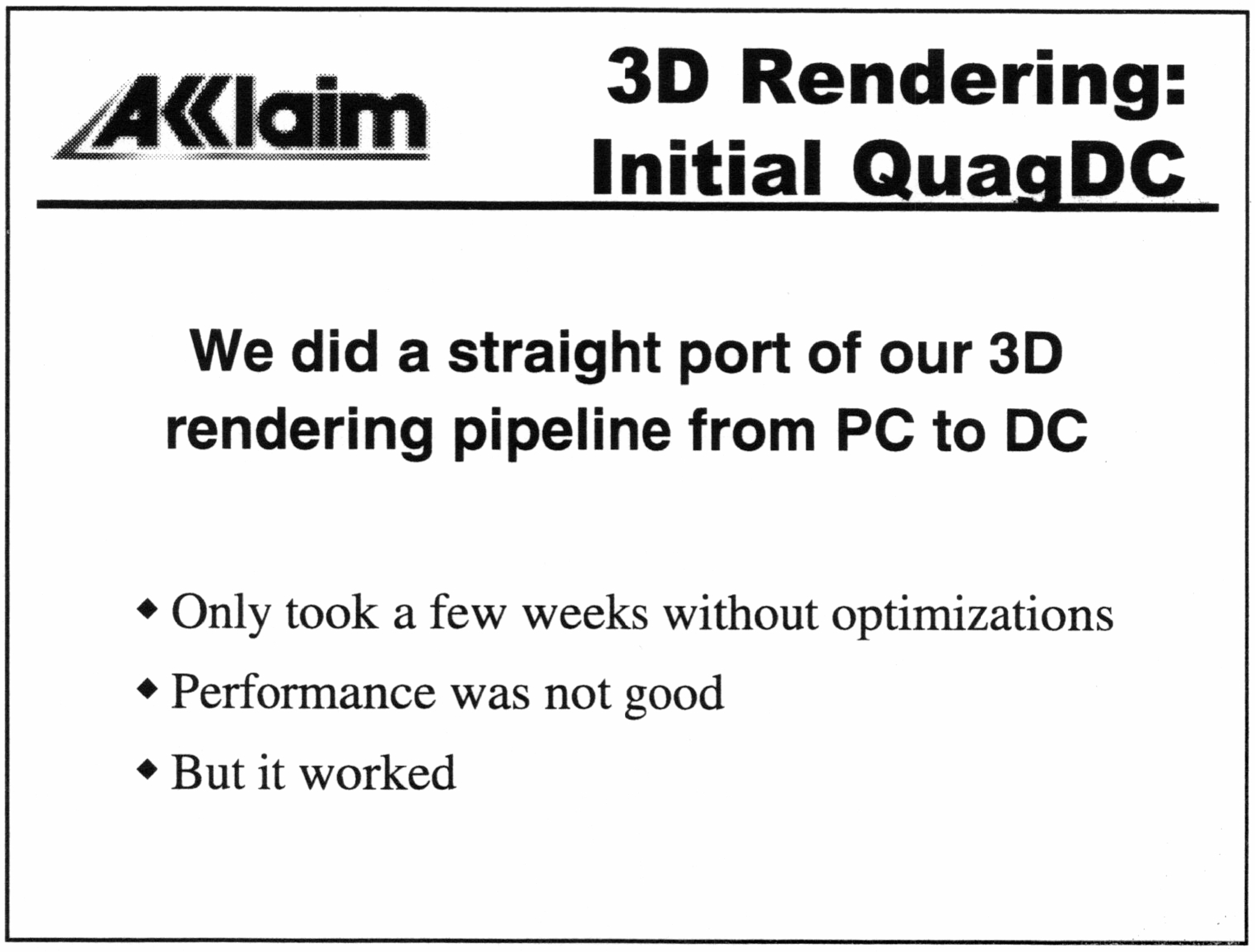
3D Rendering: Why I.T.List

- Multiple strips over a given triangle collection forced multiple processing of some vertices
  - Remember: Vertex count affected performance
- I.T.List renders all triangles and processes each vertex only once
- I.T.List was faster than strips in our situation
- Thus, we use indexed triangle lists



3D Rendering: QuagPC

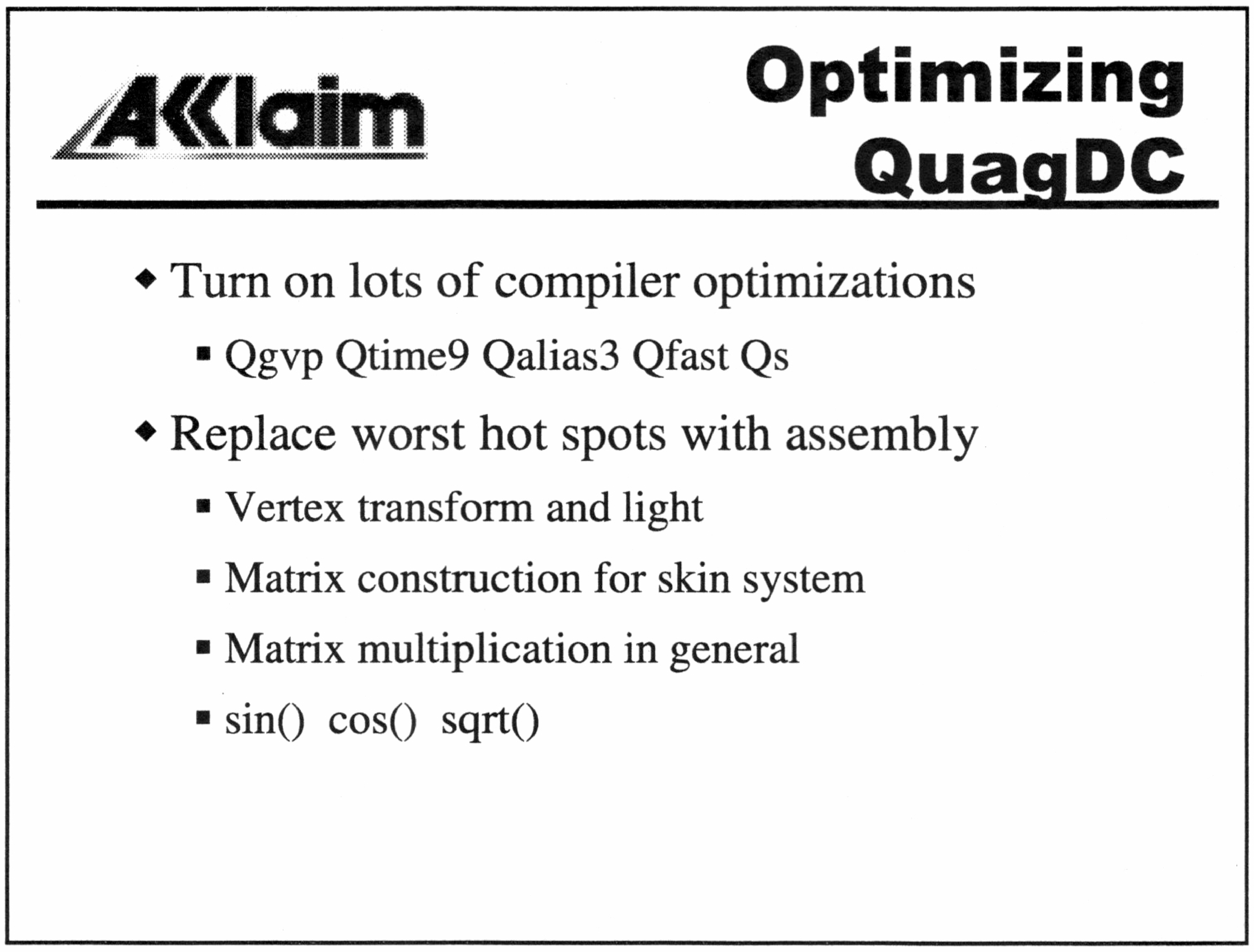
- TLVerts and indexed triangle lists
- All originally optimized in plain C
- Performing fine on PC



3D Rendering: Initial QuagDC

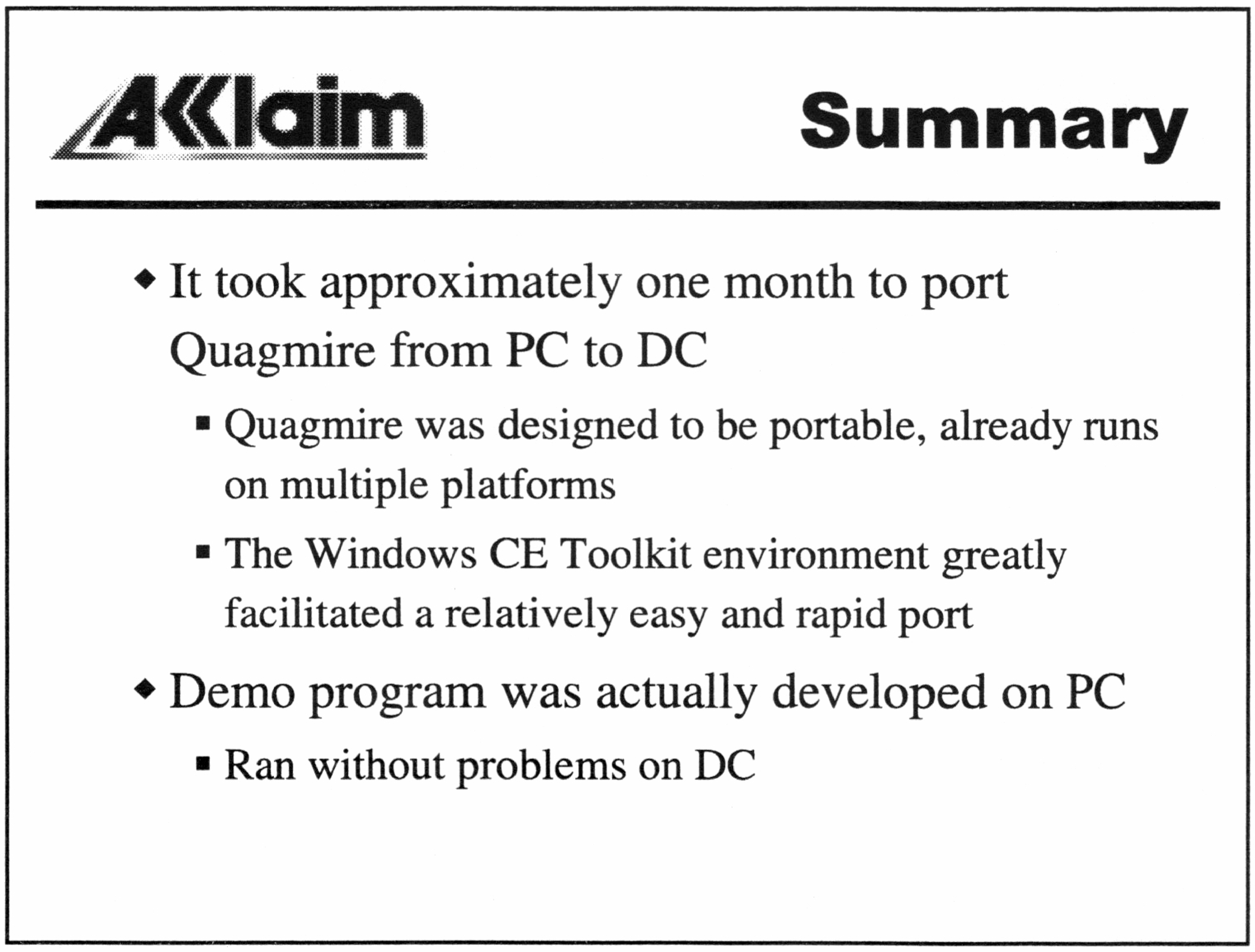
We did a straight port of our 3D rendering pipeline from PC to DC

- Only took a few weeks without optimizations
- Performance was not good
- But it worked



Optimizing QuagDC

- Turn on lots of compiler optimizations
  - Qgvp Qtime9 Qalias3 Qfast Qs
- Replace worst hot spots with assembly
  - Vertex transform and light
  - Matrix construction for skin system
  - Matrix multiplication in general
  - sin() cos() sqrt()



Summary

- It took approximately one month to port Quagmire from PC to DC
  - Quagmire was designed to be portable, already runs on multiple platforms
  - The Windows CE Toolkit environment greatly facilitated a relatively easy and rapid port
- Demo program was actually developed on PC
  - Ran without problems on DC

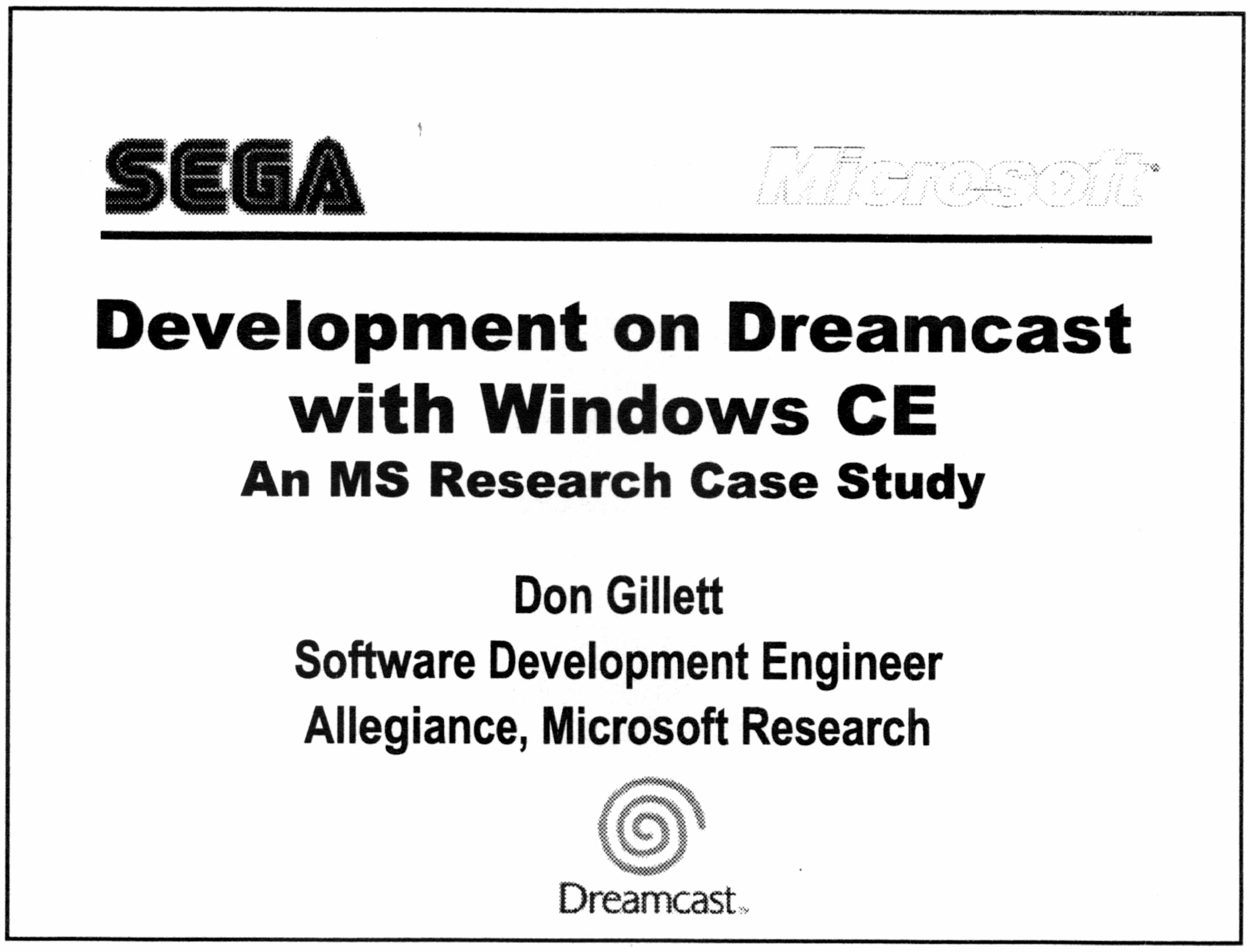
SEGA

Development on Dreamcast with WindowsCE

An MS Research Case Study

Don Gillett
Software Development Engineer
Allegiance, Microsoft Research

Dreamcast™



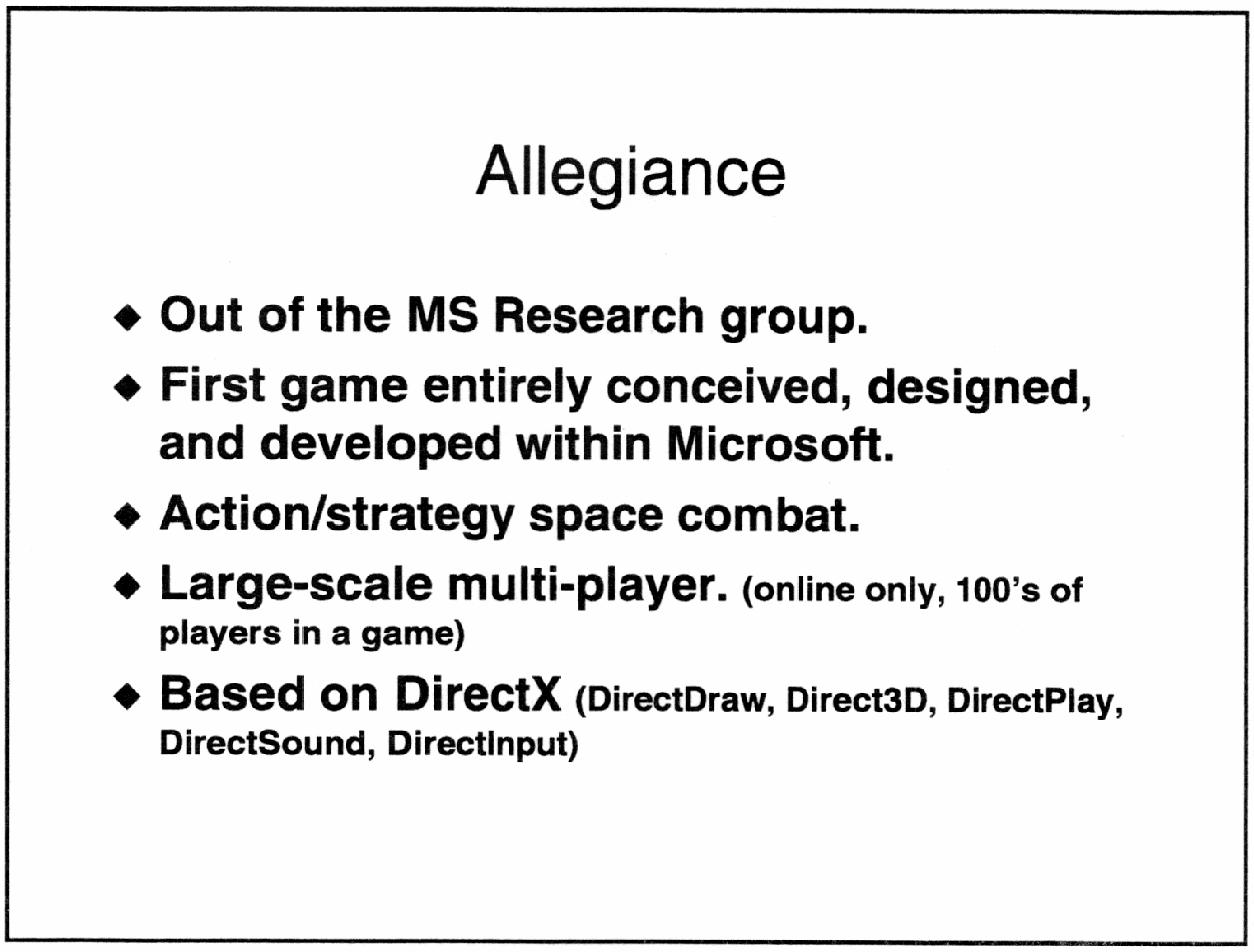
SEGA Microsoft

Development on Dreamcast with Windows CE

An MS Research Case Study

Don Gillett
Software Development Engineer
Allegiance, Microsoft Research

Dreamcast



Allegiance

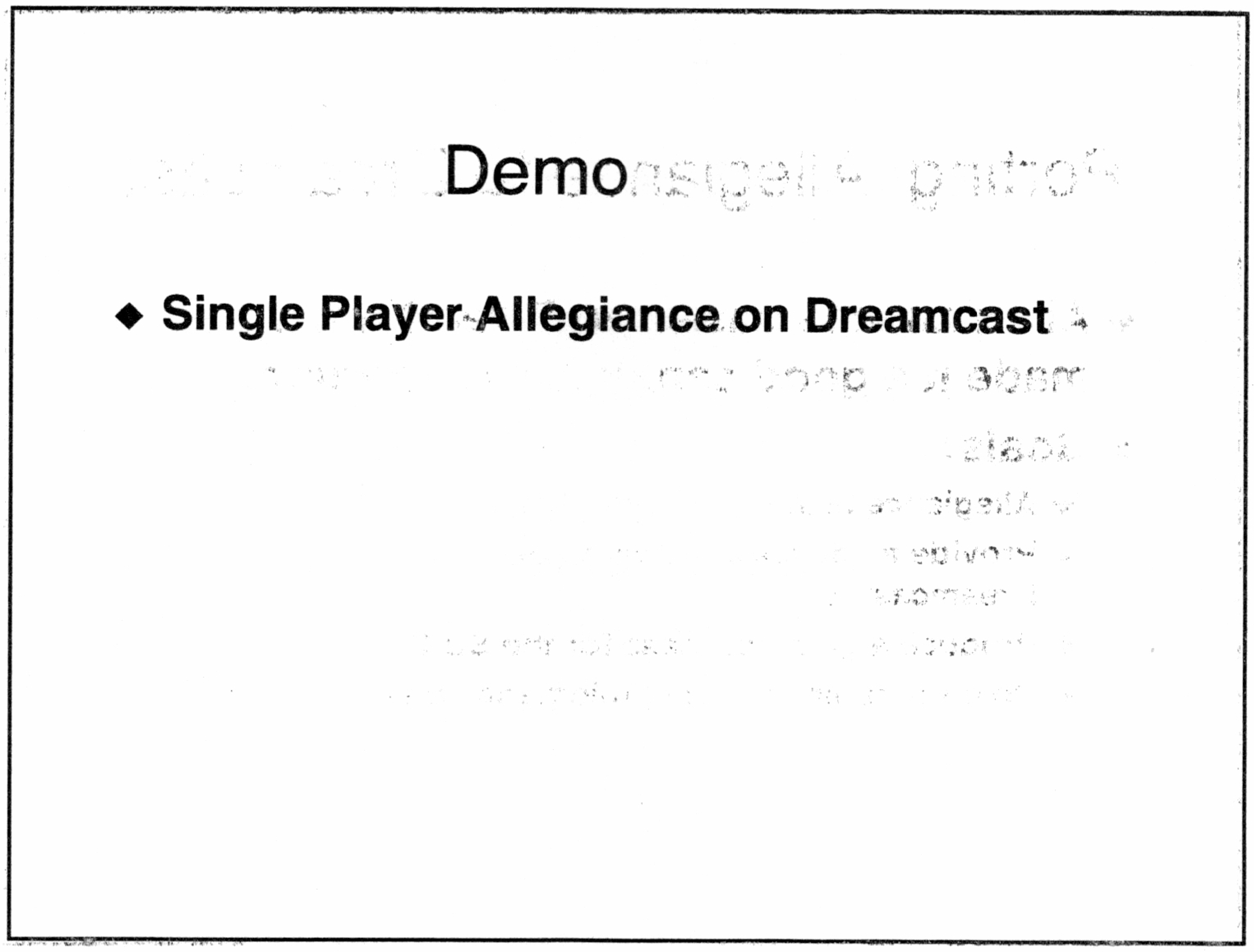
- Out of the MS Research group.
- First game entirely conceived, designed, and developed within Microsoft.
- Action/strategy space combat.
- Large-scale multi-player. (online only, 100's of players in a game)
- Based on DirectX (DirectDraw, Direct3D, DirectPlay, DirectSound, DirectInput)

Porting Allegiance to Dreamcast

- **Allegiance is based on DirectX which made it a good candidate for porting.**
- **Goals:**
  - **Allegiance team to learn the Dreamcast platform.**
  - **Provide feedback and help to the Windows CE SDK for Dreamcast teams.**
  - **Produce a great test case for the SDK.**
  - **No current plans to ship Allegiance on Dreamcast.**

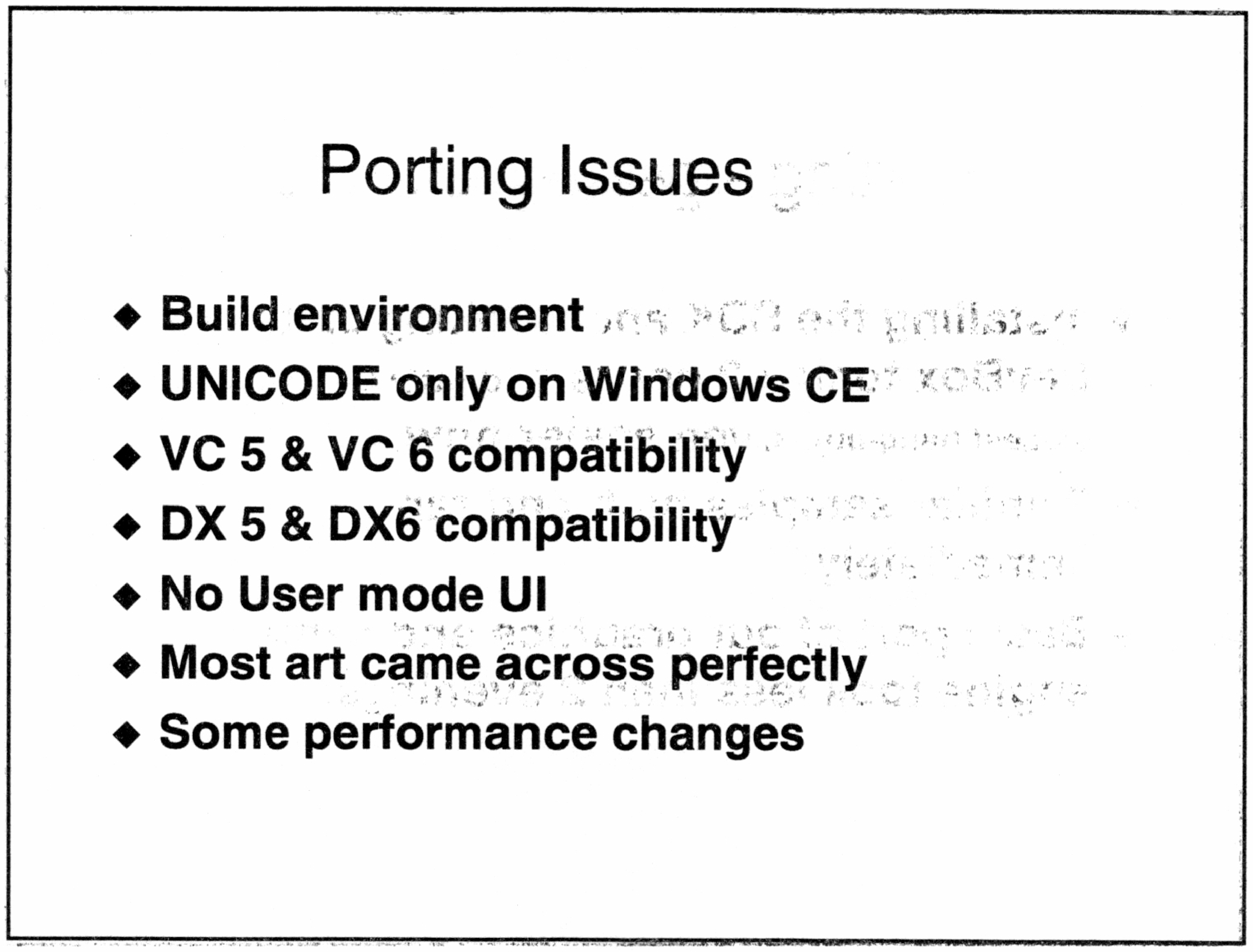
Porting - getting started

- **Installing the SDK and hooking up the DevBox took < 2 hours (2nd SCSI Card was biggest hang-up). Even easier now.**
- **Familiar samples built and ran immediately.**
- **Basic port of our graphics and game engine took less than 2 evenings.**



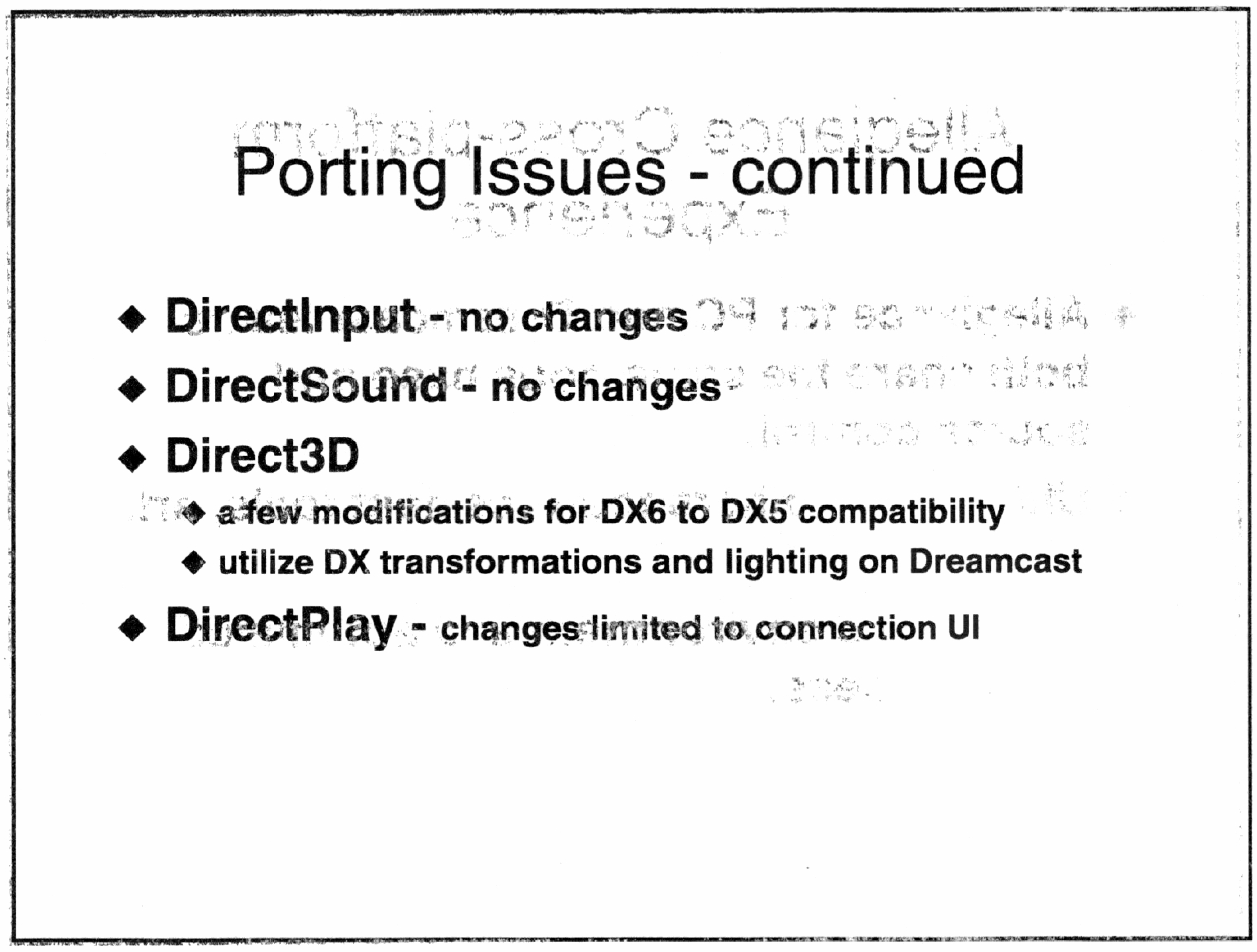
Demo

- Single Player Allegiance on Dreamcast



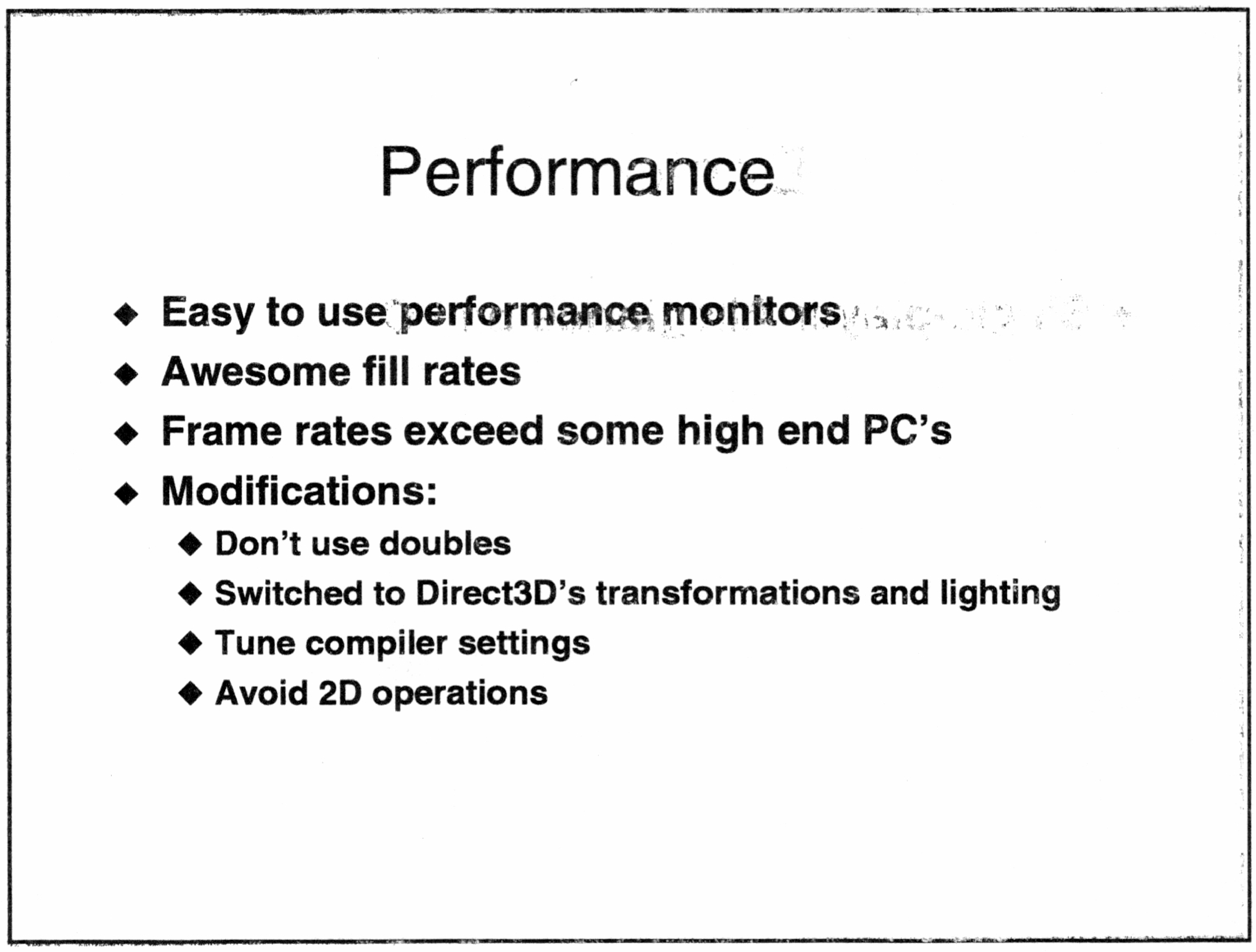
Porting Issues

- Build environment
- UNICODE only on Windows CE
- VC 5 & VC 6 compatibility
- DX 5 & DX6 compatibility
- No User mode UI
- Most art came across perfectly
- Some performance changes



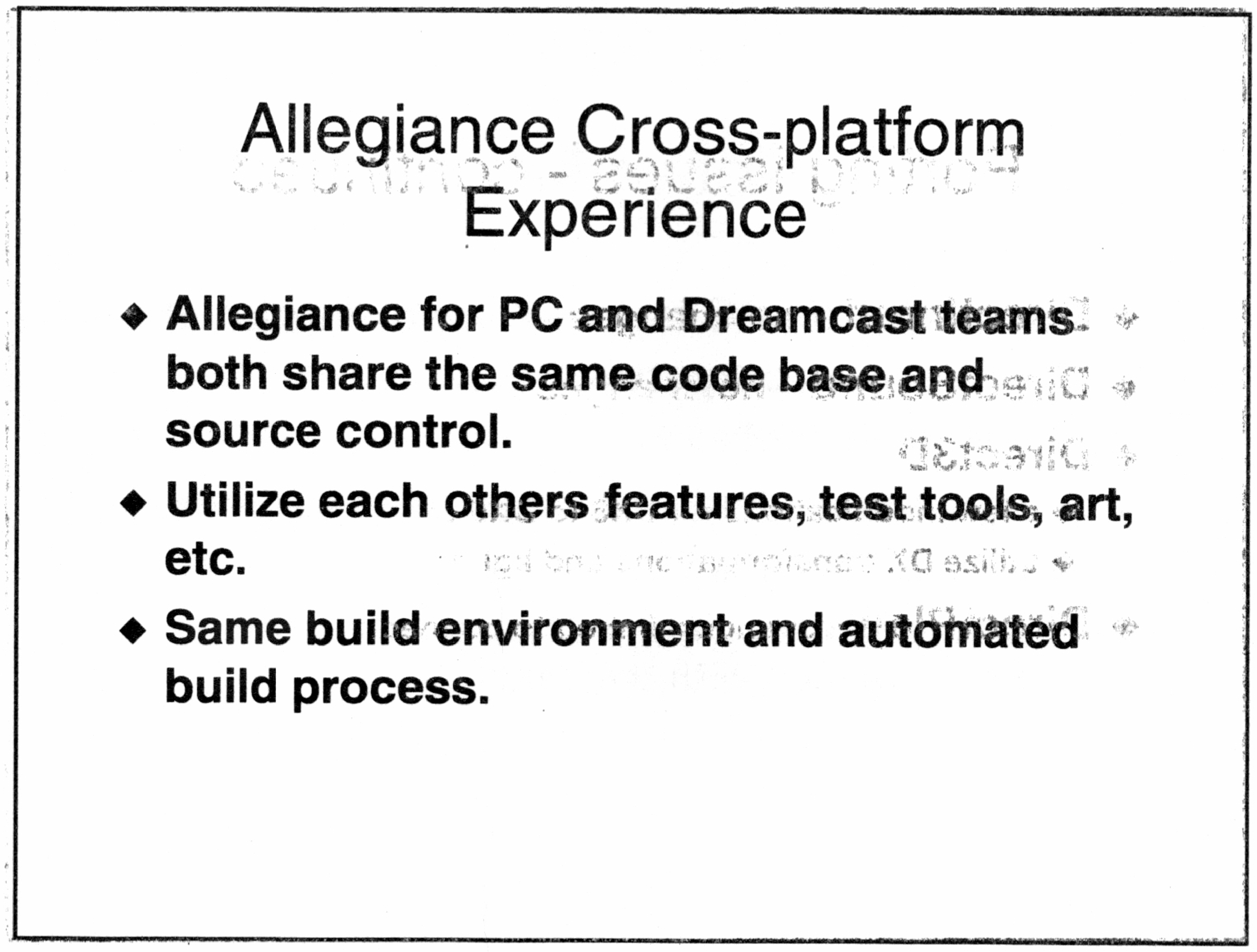
Porting Issues - continued

- **DirectInput - no changes**
- **DirectSound - no changes**
- **Direct3D**
  - **a few modifications for DX6 to DX5 compatibility**
  - **utilize DX transformations and lighting on Dreamcast**
- **DirectPlay - changes limited to connection UI**



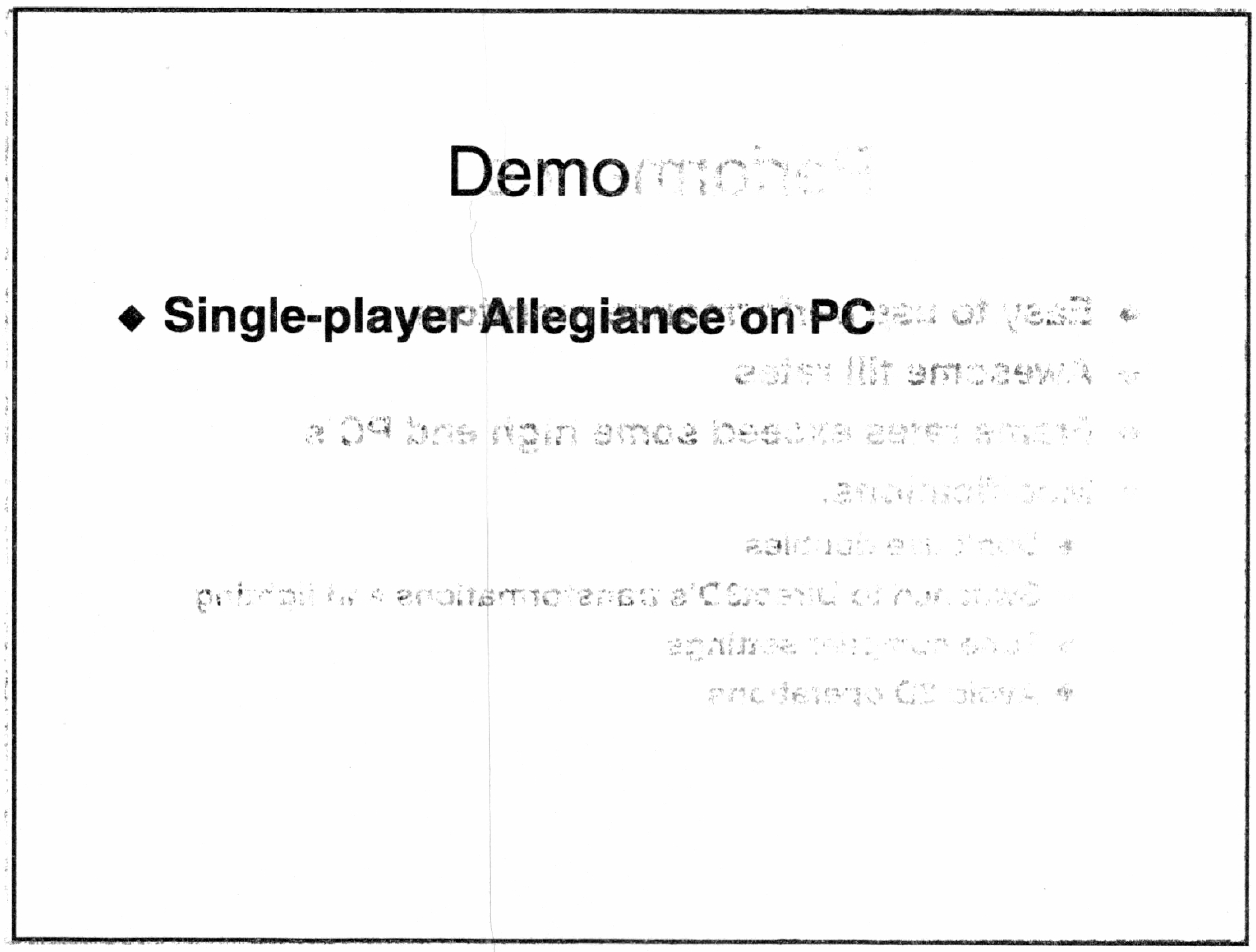
Performance

- **Easy to use performance monitors**
- **Awesome fill rates**
- **Frame rates exceed some high end PC's**
- **Modifications:**
  - **Don't use doubles**
  - **Switched to Direct3D's transformations and lighting**
  - **Tune compiler settings**
  - **Avoid 2D operations**



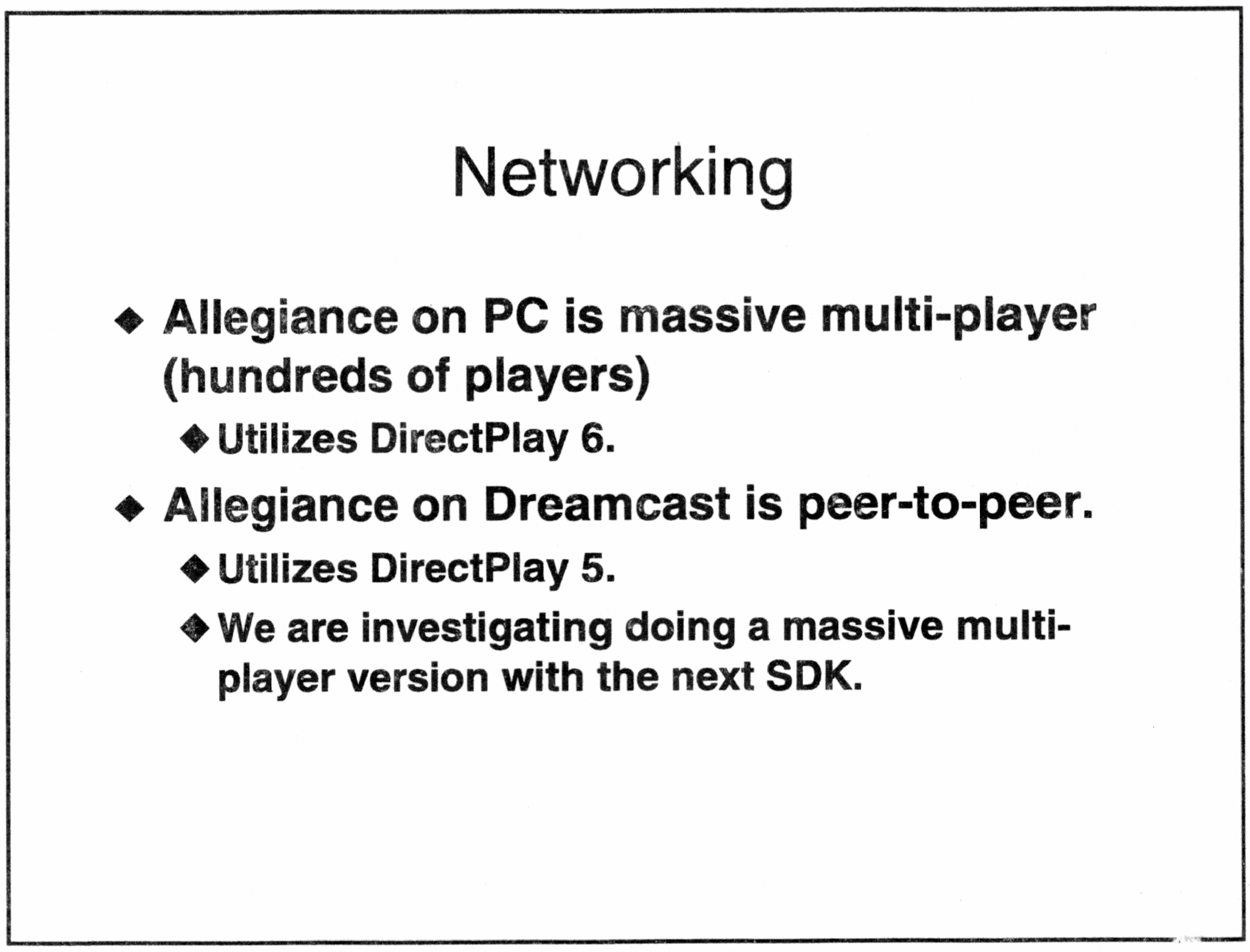
Allegiance Cross-platform Experience

- **Allegiance for PC and Dreamcast teams both share the same code base and source control.**
- **Utilize each others features, test tools, art, etc.**
- **Same build environment and automated build process.**



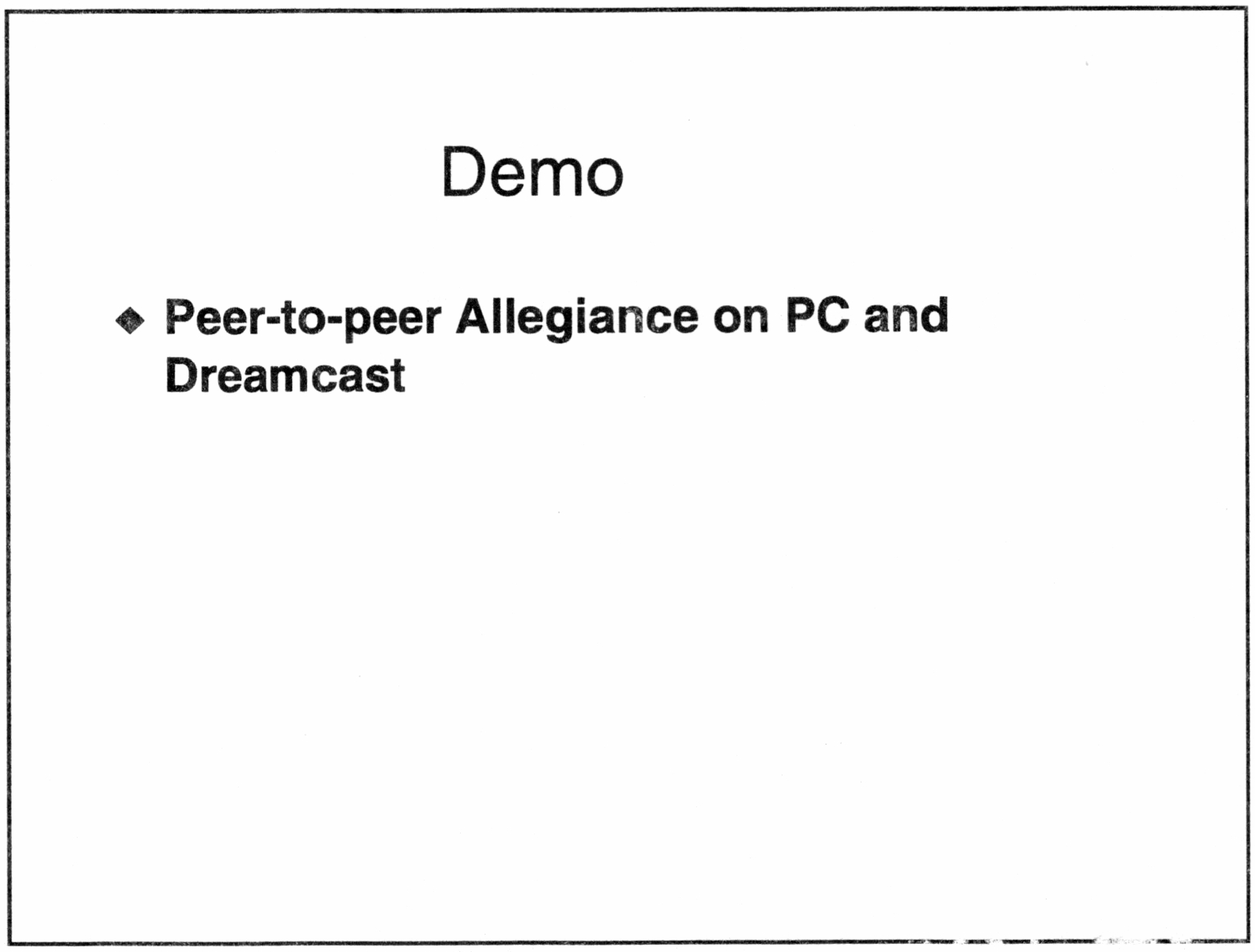
Demo

- **Single-player Allegiance on PC**



Networking

- Allegiance on PC is massive multi-player (hundreds of players)
  - Utilizes DirectPlay 6.
- Allegiance on Dreamcast is peer-to-peer.
  - Utilizes DirectPlay 5.
  - We are investigating doing a massive multi-player version with the next SDK.



Demo

- Peer-to-peer Allegiance on PC and Dreamcast