Multi-monitor Game Development

Thomas Fortier
AMD Graphics Developer Relations
thomas.fortier@amd.com
Topics Covered in this Session

- Eyefinity technology overview.
- Multi-monitor gameplay & impact on game design.
- Developing for Eyefinity solutions.
- Eyefinity certification program.
ATI Eyefinity Technology Overview

- ATI’s latest multi-monitor technology.
- Enables up to 3/6 displays per graphic card.
- Supports multi-monitor gaming.
- Required setup:

  - Radeon 5000 Series
  - Vista / Win7
  - Displays with identical resolutions
ATI Eyefinity Technology Overview

3x1 landscape

3x1 portrait

3x2 landscape
ATI Eyefinity Technology Overview

End user

Windows

Rendering surface

3x display width

1x display height
Multi-monitor Gameplay Considerations

End users expect the following:

1. Eyefinity display modes in game graphics options.
2. Larger field of view.
3. Equal or better gameplay experience.
Developing for Eyefinity Solutions

Support for Eyefinity display modes:

1. Don’t exclude non-standard display modes.
2. Don’t exclude non-standard aspect ratios.

Example display modes for a 3x1 setup:

- Native modes – 1600 x 1200, 1280 x 1024, etc.
- Eyefinity modes – 4800 x 1200, 3840 x 1024, etc.
Developing for Eyefinity Solutions

Support for increased field of view:

1. Adjust projection matrix so that it matches display mode & aspect ratio.
Developing for Eyefinity Solutions

Multi-monitor gameplay experience:

1. Menu & UI element placement considerations.
2. Cut scenes placement.
Querying Eyefinity State Information

- Statically link to:
  - `atigpu.lib`

- Include header file:
  - `atigpud.h`

- Call `AtiGetMultiMonitorConfig()` to retrieve:
  - Eyefinity state information
  - Per display state information
Querying Eyefinity State Information

- Eyefinity state information:
  - On/off, resolution, display grid configuration, etc.
- Per display information:
  - Grid coord, rendering rect, visible rect, etc.
Developing for Eyefinity Solutions

Other gameplay considerations:

1. RTS scrolling.
2. First person shooting crosshair placement.
3. Let us know if you come across any other issues...
Eyefinity Certification Program

*Eyefinity Ready*

1. Support for Eyefinity display modes.
2. Support for expanded field of view.

*Eyefinity Validated*

1. *Eyefinity Ready* with...
2. Proper placement of menu & UI elements.
Key Takeaways

- Multi-monitor solutions matter!
- Test and profile with multi-monitor systems.
  - Don’t hardcode specific multi-monitor configurations.
  - Handle any resolution & aspect ratio
  - Properly handle menu, HUD, & cut scene placement.
- Refer to AMD Eyefinity SDK samples
  - ati.amd.com/developer
Maximizing Multi-GPU Performance

Thomas Fortier
AMD Graphics Developer Relations
thomas.fortier@amd.com
Topics Covered in this Session

- Hardware & driver considerations.
- Impact on game design.
- Profiling & performance gains.
Why Multi-GPU Solutions Matter

- Dual-GPU boards
- Multi-board systems
- Hybrid graphics

Multi-monitor & multi-GPU game development
Why Support Multi-GPU in Your Game

- Growing market share of multi-GPU solutions.
- All game and hw reviews integrate multi-GPU solutions.
- Expectation by gamers is that game framerate should “just scale” with additional GPUs.
- The competition is doing it!

Market trend
Crossfire Technical Overview

Multi-monitor & multi-GPU game development
Crossfire Technical Overview

Frame 1
Frame 3
Frame 5
Frame 7

Frame 2
Frame 4
Frame 6
Frame 8

Multi-monitor & multi-GPU game development
Alternate Frame Rendering

- Alternate frame rendering leads to two types of problems:
  - Interframe dependencies
  - CPU/GPU synchronization points
- In each case, parallelism between CPU and GPUs is lost.
Querying the Number of GPUs

- Statically link to:
  - `atigpu.lib`

- Include header file:
  - `atigpud.h`

- Call this function:
  - `INT count = AtiMultiGPUAdapters();`
  - In windowed mode, set Count to 1
Interframe Dependencies

Frame 1 → Frame 2

Frame 3 ← Frame 2

Frame 3 ← Frame 4

Frame 5 ← Frame 4

Frame 5 ← Frame 6

Frame 7 ← Frame 6

Frame 7 ← Frame 8
Interframe Dependencies

- When are interframe dependencies a problem?
  - Depends on frequency of P2P blits.

- Solutions:
  - Create $n$ copies of the resource triggering P2P blits.
  - Associate each copy of the resource to a specific GPU.
  - $resource[frame\_num \% num\_gpuses]$
  - Repeat resource updates for $n$ frames.
Interframe Dependencies

Frame 1 -> Frame 2
Frame 3 <-> Frame 4
Frame 5 <-> Frame 6
Frame 7 <-> Frame 8
Interframe Dependencies

Frame 1
Frame 3
Frame 5
Frame 7
Frame 2
Frame 4
Frame 6
Frame 8
Interframe Dependencies

- There are many ways to update resources using the GPU:
  - Drawing to Vertex / Index Buffers
  - Stream Out
  - CopyResource()
  - CopySubresourceRegion()
  - GenerateMips()
  - ResolveSubresource()
  - Etc...
CPU/GPU Synchronization Points

Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
CPU/GPU Syncs - Queries

- Having the driver block on a query starves the GPU queues, and limits parallelism.

- Solutions:
  - Don’t block on query results.
  - Don’t have queries straddle across frames.
  - For queries issued every frame, create a query object for each GPU.
  - Pick up query results $n$ frames after it was issued.
CPU/GPU Syncs – CPU Access to GPU Resources

- Triggers pipeline stalls because driver blocks waiting on GPU at `lock/map` call.
- Followed by a P2P blit at `unlock/unmap` call.
- Often results in negative scaling...

- Solutions:
  - DX10/DX11 – Stream to and copy from staging textures.
  - DX9 – Stream to and copy from sysmem textures.
  - DX9 – Never lock static vertex/index buffers, textures.
Multi-GPU Performance Gains

- What kind of performance scaling should you expect from multi-GPU systems?
  - Function of CPU/GPU workload balance.
  - Typical for 2 GPUs is 2X scaling.
  - For 3 & 4 GPUs, varies from game to game.
Crossfire Profiling

- Make sure to be GPU bound.
  - Test framerate scaling with resolution change.
- Test for multi-GPU scaling.
  - Rename app exe to *ForceSingleGPU.exe*.
- Test for texture interframe dependencies.
  - Rename app exe to *AFR-FriendlyD3D.exe*.
- Remove queries.
- Check for CPU locks of GPU resources.
Key Takeaways

- Multi-GPU solutions matter!
- Test and profile with multi-GPU systems.
  - Properly handle interframe dependencies.
  - Check for CPU locks of GPU resources.
  - Don’t block on queries.
- Refer to AMD Crossfire SDK samples
  - ati.amd.com/developer
  - CrossFire Detect & AFR-Friendly projects.
Thank You

Thomas Fortier – thomas.fortier@amd.com