Lecture 16
Introduction to Game Development
Introduction to Game Development (on the Playstation 3 / Cell)

- Mike Acton
  - Engine Director, Insomniac Games
  - <macton @ insomniacgames.com>
  - Director, CellPerformance.com
  - <macton @ cellperformance.com>
Different Types of Game Development

- Casual
- Console
- PC
- Handheld
- Cellphone
- Single Player
- Multi Player
**Console Development Priorities**

- The code itself is not that important.
- The design of the data affects performance more than the design of the code.
- Ease of programming is either a minor or non-priority.
- Portability is not a concern.
- Performance is still king.
Development Team

- Artists
  - Animation, Shader, Texture, Modeling
  - Environment, Lighting, ...
- Designers
  - Systems, Level, ...
- Writers
- Producers
- Programmers
  - Gameplay, Engine, AI, Special Effects,
  - Sound/Music, ...
What Impacts Game's Technical Design?

- Type of game
- Framerate
- Schedule
- Cost
- Hardware
- Compilers
- ...

- How does this affect code reusability?
- How does this affect cross-platform design?
What are the major game modules?

- Memory management
- Math
- Collision
- Physics
- Static graphics
- Animation
- Procedural graphics
- Lighting

- Loading, streaming
- Scene graph
- AI
- Compression
- Sound, Music
- Special Effects
- State machines
- Scripting
- Motion control
- ...

...
Overview

- How does programming on the Playstation 3 affect the (macro) design of the major systems?
- Overview of design process for a specific system (Animation).
Structure Design (1)

- Conventional structures are (surprisingly?) needed very little in engine-level SPU code.
  - Data is compressed
  - Data is sorted by type (i.e. Fewer flags)
  - Data is organized into blocks or streams
  - Data is accessed only in quadwords
Organize data carefully:
- Prefer fixed (known) size blocks
- Fundamental unit: 128 bytes (Cache line)
- Fundamental unit: 16 bytes (Quadword)
- Prefer uniform data

Minimum working sizes:
- 4 x 2 x 64 bits
- 4 x 4 x 32 bits
- 4 x 8 x 16 bits
- 4 x 16 x 8 bits
- 4 x 128 bits
Basic Math

• e.g. Vector Class
  – Usually the first thing a programmer will make, but consider:
    • SIMD, Altivec vs. SPU instruction set
    • Floats vs. Double vs. Fixed-point
    • SPU floating-point format
    • Component access
  – ... There's no value here.
Memory Manager

• Static allocation is preferred to dynamic
• Most data patterns are known in advance
• When designing allocator, consider:
  – Page sizes
  – LRU is most common, but pretty bad.
  – Hierarchy of allocations
  – Fragmentation is a non-issue for well planned architectures
  – Remember cache line alignment.
  – SPU transfer blocks, 16K
Collision Detection

- Affects high-level design
  - Deferred results
  - Grouped results
- SPU decomposition for:
  - Static geometry in scene
  - Dynamic geometry in scene
Procedural Graphics

- Patch size
- Filter types
- Sync of source reads
- Sync with GPU
- SPU vs. RSX

- Particles
- Cloth
- Fonts
- Textures
- Parametric geometry
- ...

...
Geometry databases

- No scene graph
- Domain information linked by key
- Cache and TLB affect design choices
  - e.g. Static geometry lookup (Octree, BSP, etc.)
- Geometry lookups on SPU
  - Spatially pre-sort
  - Multiple simultaneous lookups
**Game Logic**

- State machines
  - Size affected by SPU
  - Deferred results
  - Logic lines can be deferred
- Scripting
  - Interpreter size
  - Multiple streams to hide memory accesses
- Motion control
  - High-level sync (Animation, AI, Physics)
• **Starting with the basics:**
  - Simple playback, animation channels
    • Related data
    • e.g. Rotation + Translation + Scale = Joint
  - **Euler vs. quaternion**
    • Euler: More compressible
    • Quaternion: Less messy
    • Gimbal lock is manageable in practice.
  - Format, double vs. float vs. half vs. fixed-point
  - Rotations: Degrees, radians or normalized?
Animation (2)

- Animation frame storage
  - Basic 9 channels (raw)
  - Uniform channels
    - Plus uniform channel map
    - Plus uniform channel count
  - X Number of joints
  - Decide on max channels
Animation (3)

- Channel curve fitting
  - Closer to root, tighter fit.
  - e.g. Simple spline
    - Store time values
    - Problem: Looping scalars
    - Problem: Unlimited length
Animation (4)

- e.g. Spline segments
  - Plus storage for time maps
  - Plus segment lookup time
  - Advantage: Can re-order blocks
  - Advantage: Long lengths OK
  - Disadvantage: Less compressable
  - Advantage: Solves scalar loop problem

- Summarize: DMA and transform.
Animation (5)

• e.g. Adding dynamic channel support
  - Add uniform data table
    • Maximum dynamic channels with linkage, or...
    • All uncompressed
  - Add (simple) constraints
    • Max change
    • Max range
    • Max acceleration (impacts storage)
  - Blend information
  - Summarize: DMA and transform.
Animation (6)

• More on mixing:
  – Phase matching
  – Transitions
  – Translation matching

• Drawing animated geometry
  – Single or double buffer joints:
    • Single: Requires more organization
    • Double: More memory, more flexible.
Optimization

- Required for practice
- Impacts design
- NOT the root of all evil