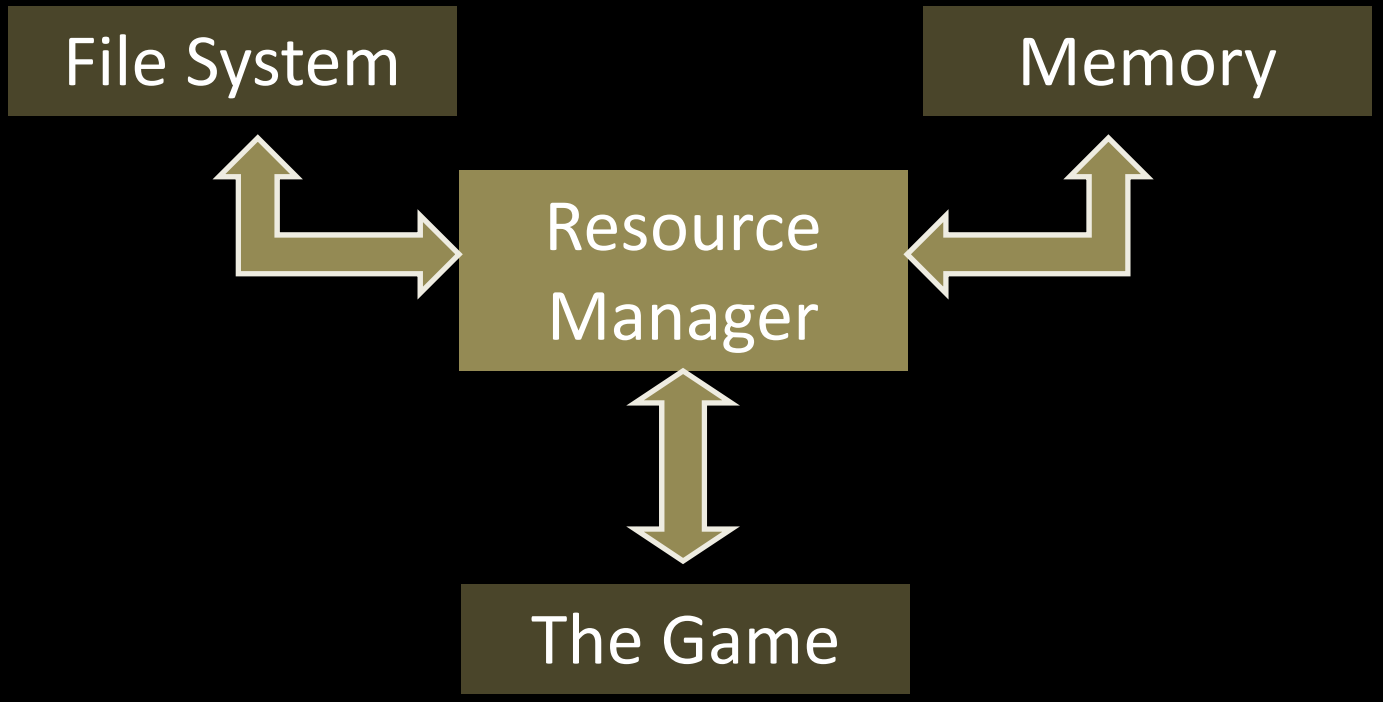
Runtime Resource Management

hannes@ru.is

Overview (again)



Runtime Resource Management

- Responsibilities
  - Ensure one copy of each resource
  - Manage resource lifetime
  - Load composite resources
  - Maintain referential integrity
  - Manage memory use
  - Perform custom post-load processing
  - Provide unified interface
  - Stream

Resource Organization

- Convenient directory structure for developers and artists
  - Engine doesn‘t care...
  - ...but YOU will be glad you‘re organized!
- Support for multiple file formats
  - JPG, BMP, TIF, PNG, 3DS, OBJ, WAV, MP3...
  - Preprocessed binaries from memory images

Resource Organization (cont.)

- What really matters for the engine is:

  Load time = seek time + open time + read time

- Seek Time + Open Time → Single file best
- Read Time → Compression often good

- Ogre3D supports ZIP files of resources
  - Compressable, modular, open format, feels like directory structure

Resource Registry

- Dictionary of GUIDs to memory pointers
- Either resource found or not in memory
- Load when not found?
- **NO**! Either...
  - ...pre-load en masse
  - ...stream in background

Resource Lifetime

- Lifetime Requirements
  - Load-and-stay-resistent (LSR)
  - Game level
  - Short moments
  - Streamed live
- Knowing when to load is relatively easy, what about unloading?

Examples?

Reference Counting

- Procedure for reference counting when switching from one level to the next:
  1. Increase count for all objects required by new level
  2. Decrease count for all objects used by old level
  3. Delete objects now at zero
  4. Load objects that went from zero to one
  5. Leave others intact

Still tricky! Could be counted even if hidden...

Resource Memory Management

- May need to manage various memory locations
  - Not just RAM, also GPU memory, Screen buffers
- Fragmentation a big issue
- May utilize custom memory allocators...
- ...OR stick with general purpose heap allocator
  (OS-es that support Virtual Memory can „cover-up“ fragmentation at the expense of performance)

Custom Memory Allocators

- Stack Based
  - Store LSR first
- Double-ended Stacks
  - BOTTOM: Presistent TOP: Temporary
  - BOTTOM: Uncompressed TOP: Compressed
    Top is then uncompressed into bottom when used
- Pool Based (e.g. data designed around 512 kb chunks)
  + EASY: Chunks associated with game levels
  - WASTE: Unused parts of chunks (allocate also?)

Sectioned Files

- Resource files can be sectioned to improve management

Section 1
    Chunk 1
    Chunk 2
Section 2
    Chunk 1
    Chunk 2
...

Composite Resources

- Represented in
  resource database dependency maps
  - directed graphs
- Both internal and external dependencies

Cross-References

- Pointers when in memory BUT what happens on disk?
  - Use GUIDs on disk, which then are turned into pointers once in memory through a global resource look-up table
  - All pointers are found while in memory, stored in a pointer fix-up table with data on disk, there changed to offsets from file beginning and those changed back to actual pointers when back in memory (easy if contiguous image)

Serialization

Binary Images: Constructors

- When binary images of object instances are placed on disk, how can constructors be called when image is loaded back?
  - Save table of non-PODS on disk as well, with info on associated classes
  - Once binary image has loaded, call all constructors on the predetermined memory locations
    (using the *placement-new* syntax: construction without allocation)

Multi-File Composite Resource

- Key to load ALL interdependent files first
  - Iteratively go though each file and load linked files
- Populate a master look-up table with all loaded objects and their memory locations
- Finally fix-up all pointers using the look-up table

Post-Load Initialization

- Generally best to avoid any processing at load time, better to pre-process as much as possible
- But sometimes
  - UNAVOIDABLE: E.g. transfer data to special buffers
  - CONVENIENCE: Pre-calculation of look-up tables
- Special Init() and Destroy() methods (C++)
  - Init may requeset more memory OR even replace the old data with processed data (change formats)