SharpEye

A parallel, portable, selective rendering system based on RADIANCE

3rd International RADIANCE workshop
Ecole d'ingénieurs et d'architectes de Fribourg
Switzerland - 12 October 2004

Bristol Computer Graphics Group
Francisco Pereira

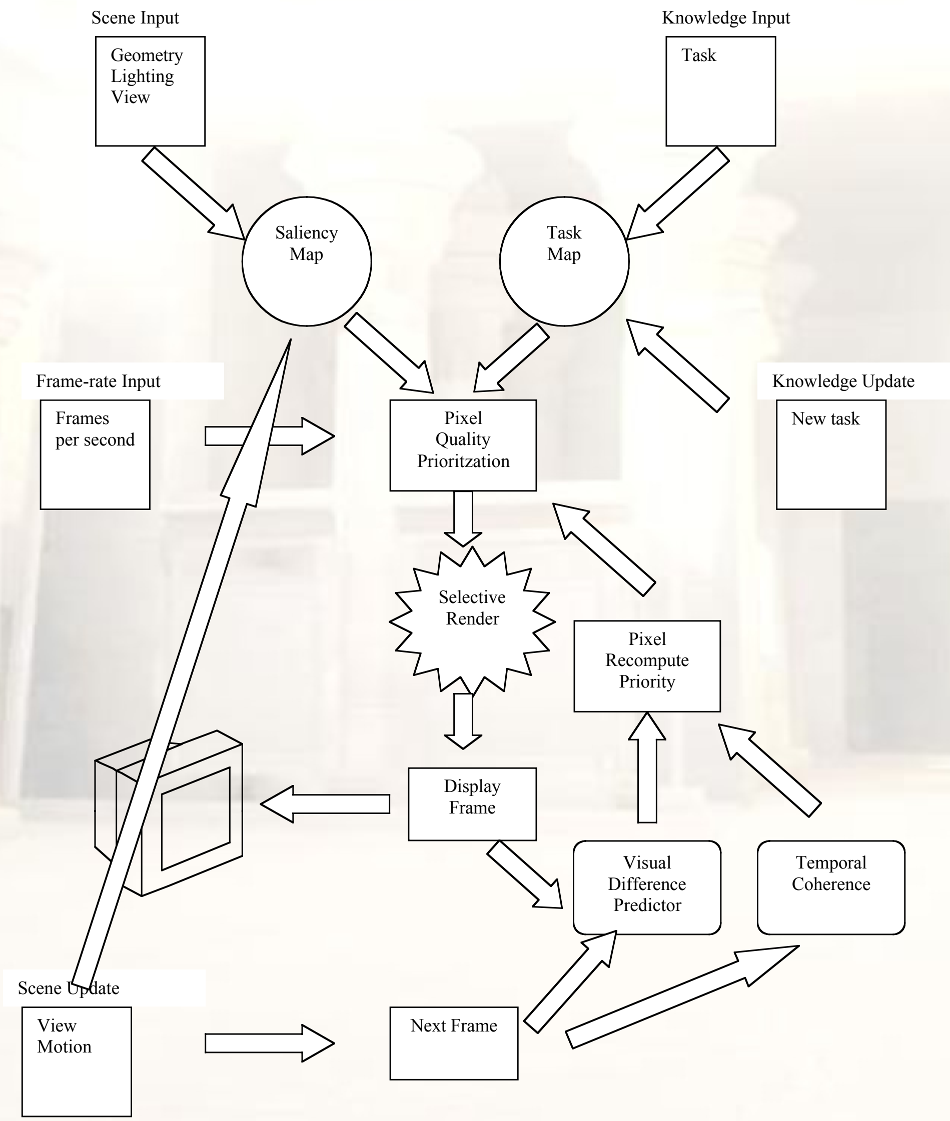
PRESENTATION OUTLINE

- Goals
- Overall level division
  - Maps Construction (and basic Rendering)
  - Parallel Render (data/task management)
  - Hardware Abstraction and Adaptation
- Summary

GOALS

- RoD – Rendering on Demand (Real time RADIANCE)
- Develop a interactive rendering system for high fidelity rendered images
- Ranging from interactive walkthroughs to final full animation production
- Exploit all the computer power (resources) available in a standard computer or cluster of computers
- Provide what's really is important to the viewer and enable fast interaction between modelling and visualisation of effects

SharpEye FRAMEWORK



PREMISES

- Maximise compatibility with standard RADIANCE package (inputs/output files and parameters)
- Modular and layered design
- Hardware and topology independent (high level abstraction)
- Maximise computational power for each resource
- Selectively enable/disable the map’s influence on the rendering process
- Code division into distinct groups (basic primitives, creation of maps, general management, general rendering)

Platform Independent Parallelism

Commercial Modellers — Maya, 3ds Max

↓ Plug-in

Joint Importance Maps

↓ Pixel Priority List

Selective Rendering — System level parallelism

↓ P-Code level

Hardware Resource Allocation — Node level parallelism

MAPS LEVEL

Compute an "importance" for each pixel in the animation for each type of map

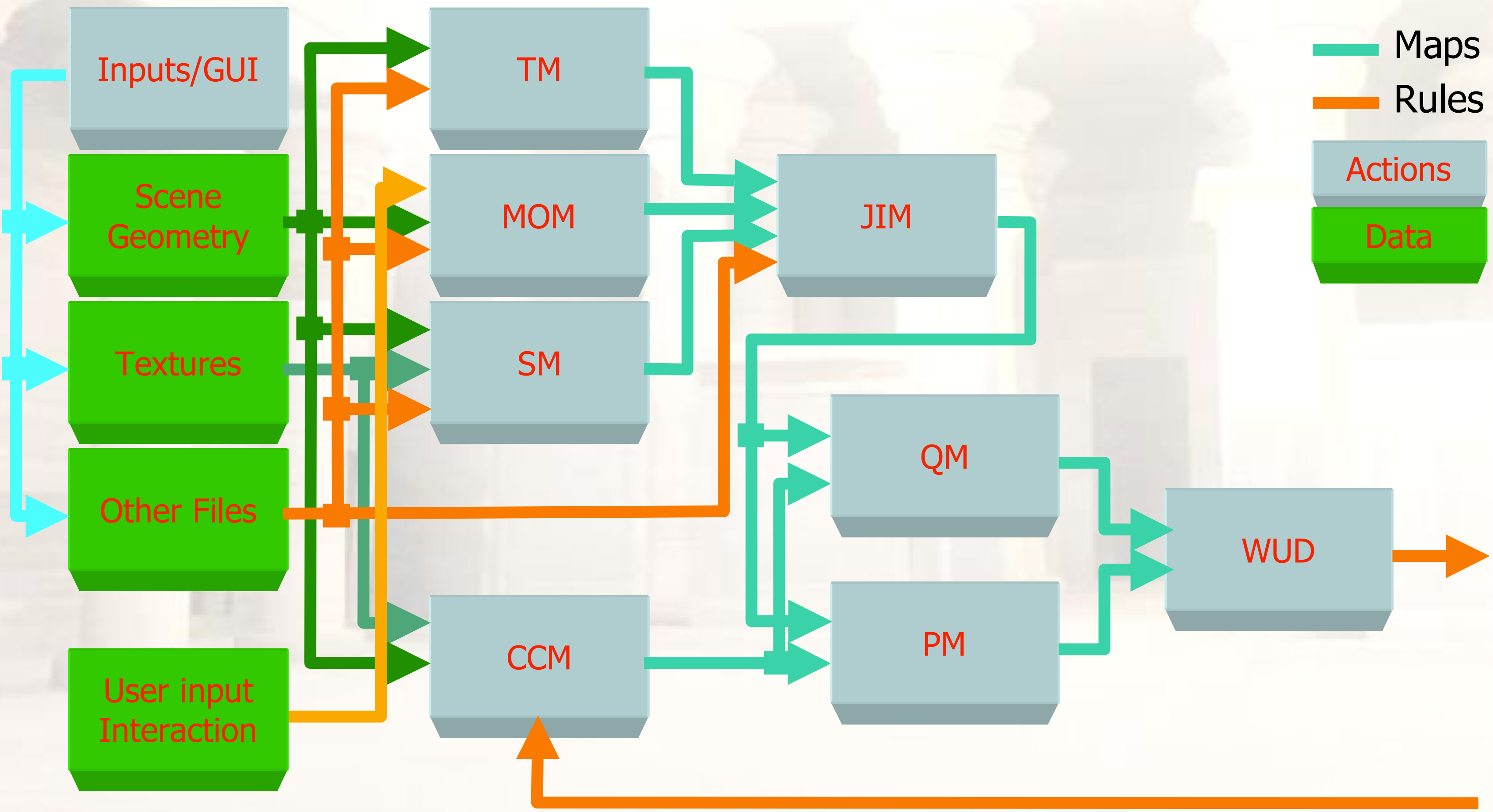
- SM – Saliency Map
- TM – Task Map
- CCM – Cost and Complexity Map
- MOM – Motion Map

Maps Level

- Create a rapid view in openGL of the image called "Snapshot" - used by some of the maps (e.g. SM)
- Reuse and overlap some operations between maps (such as pixel to object correspondence)
- Exploit some coherency at this level, namely in space domain (inside each frame) and temporal domain (between frames)
- Merge all the maps into a "quality" map and a priority map

SharpEye

Merging Maps



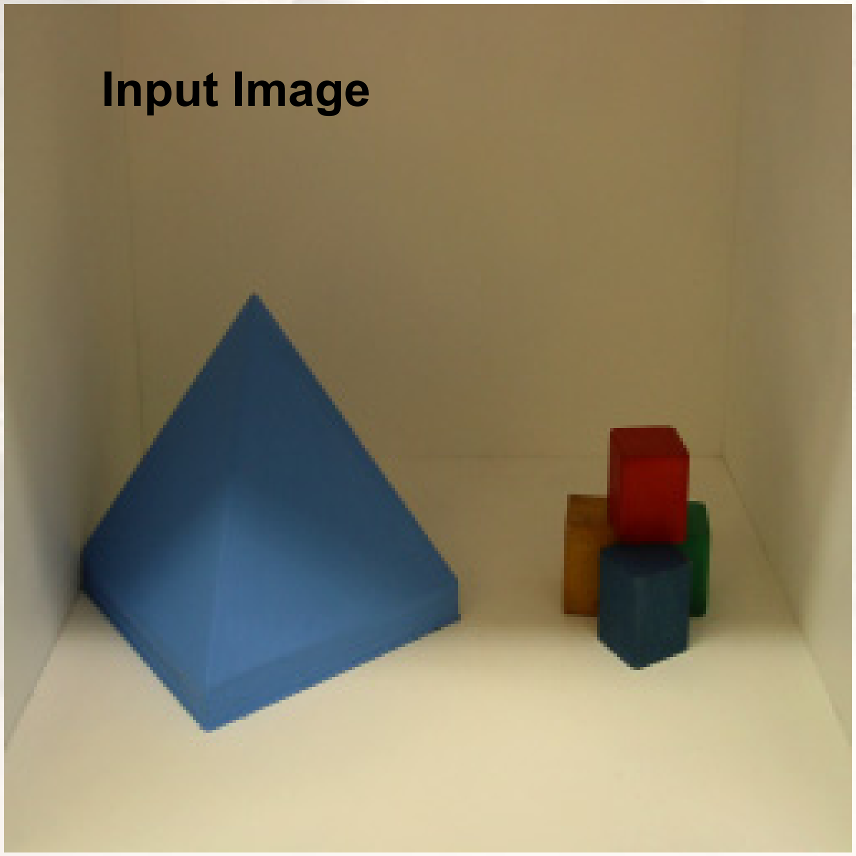
SALIENCY MAP

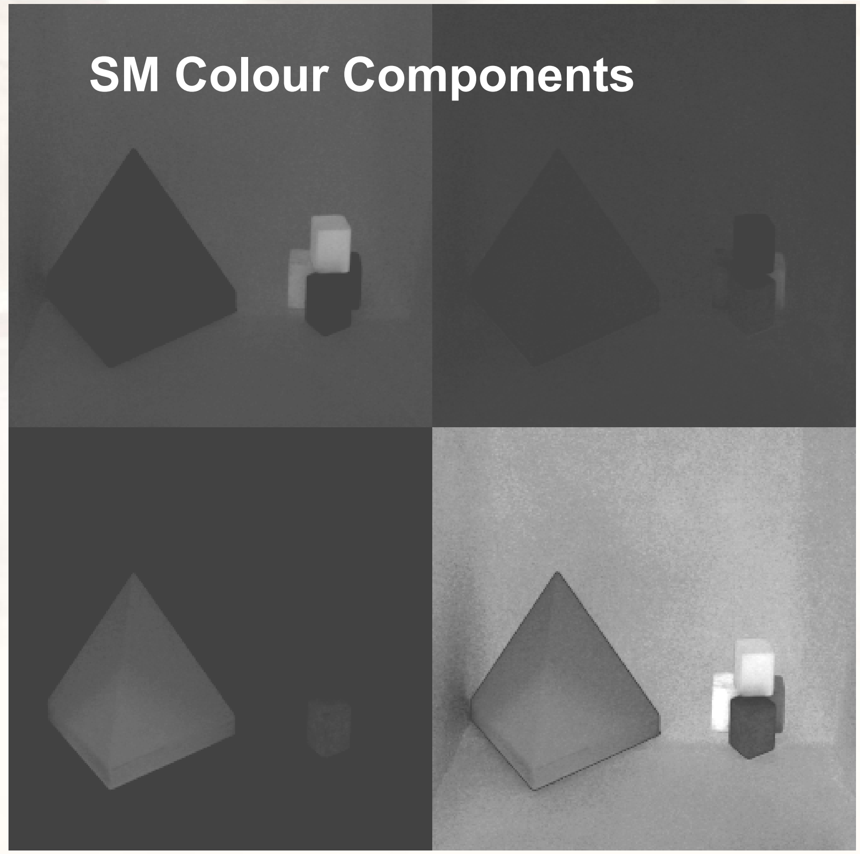
- In the short amount of time that a frame is displayed during the course of an animation not everything is witnessed
- Time can be saved if not everything is rendered to the same fidelity
- The Saliency Map (SM) identifies regions which are likely to be observed including those areas affected by aliasing (in conjunction with the CCM), these areas are targeted for greater computation than the rest of the image

SALIENCY MAP

- The final map is created from several sub parts
  - Colour: selective chromaticity channel intensity, and centre surround differences
  - Intensity: luminance intensity
  - Orientation + Frequency: combination of maps related to human sensitivity

SALIENCY MAP IMAGES

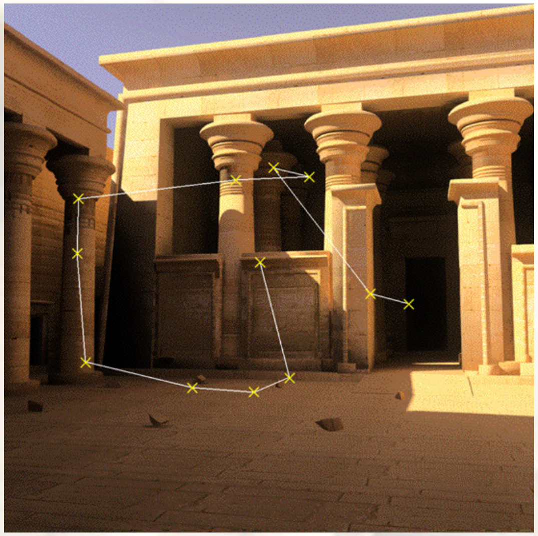




TASK MAP

- Find a model for determining the order of perceptual priority in a scene based on the viewer's task (Importance map)
- Important parts should be rendered first and at highest quality
- Provide the renderer an array with all pixels' importance values based on the task

TASK MAP



No task defined

With a task

Saliency map

Task map

COST AND COMPLEXITY MAP

- In order to maximise our resources utilisation we need to know the computational cost of the scene
- If the cost is low enough we may be able to render the entire frame at uniform high quality
- If there is a higher cost than the available resources then we need to adjust the qualities of each part of the image to most efficiently utilise resources

COST AND COMPLEXITY MAP

- The cost of casting rays in a scene can be determined to depend on:
  - The geometric complexity
  - The textural complexity of the objects
  - The lighting complexity

MOTION MAP

- Movement and path influence on rendering the scene
- Take in account the path in a scene in order to predict where a user's attention will be focused
- Take into account task map input to define probable path to be taken
- Output display specifications also influences this map (multiple displays or motion pod systems)

PARALLEL RENDER

- Based on the portable message passing library TPL*) with thread-safe communication primitives. TPL builds on MPI but hides the differences between vendor-specific MPI implementations (e.g. the support of thread-safety) from the application programmer

*) T. Plachetka: Event-Driven Message Passing and Parallel Simulation of Global Illumination, Dissertation, University of Paderborn, Germany, 2003

PARALLEL RENDER

- Traditional demand driven ray tracing
  - Dynamic load balancing
  - Exploiting coherence
  - MAP aware scheduling
- Parallelising the Irradiance Cache
  - Using memory based message passing (TPL)
  - Experimenting with different schemes
    - Local only irradiance cache
    - Centralised irradiance cache
    - Distributed irradiance cache

HARDWARE RESOURCE ALLOCATION

- Manage all the computational resources available in a node (one HRA per node)
- Provide the interface between Parallel Render and Hardware (CPU, GPU, FPGA, others)
- Maintain status and caching information (data/task) about every resource in use
- Data/task adaptation to each resource
- All work supplied to the HRA is to be done

HARDWARE RESOURCE ALLOCATION

- Works like a driver for the resources (all together). Provides basic primitives in each available resource with a standard interface
- Decides which work unit or block of work units should be dispatched to which resource based on resources status and scheduling
- May reorder and group work units in queue for optimization of resources usage

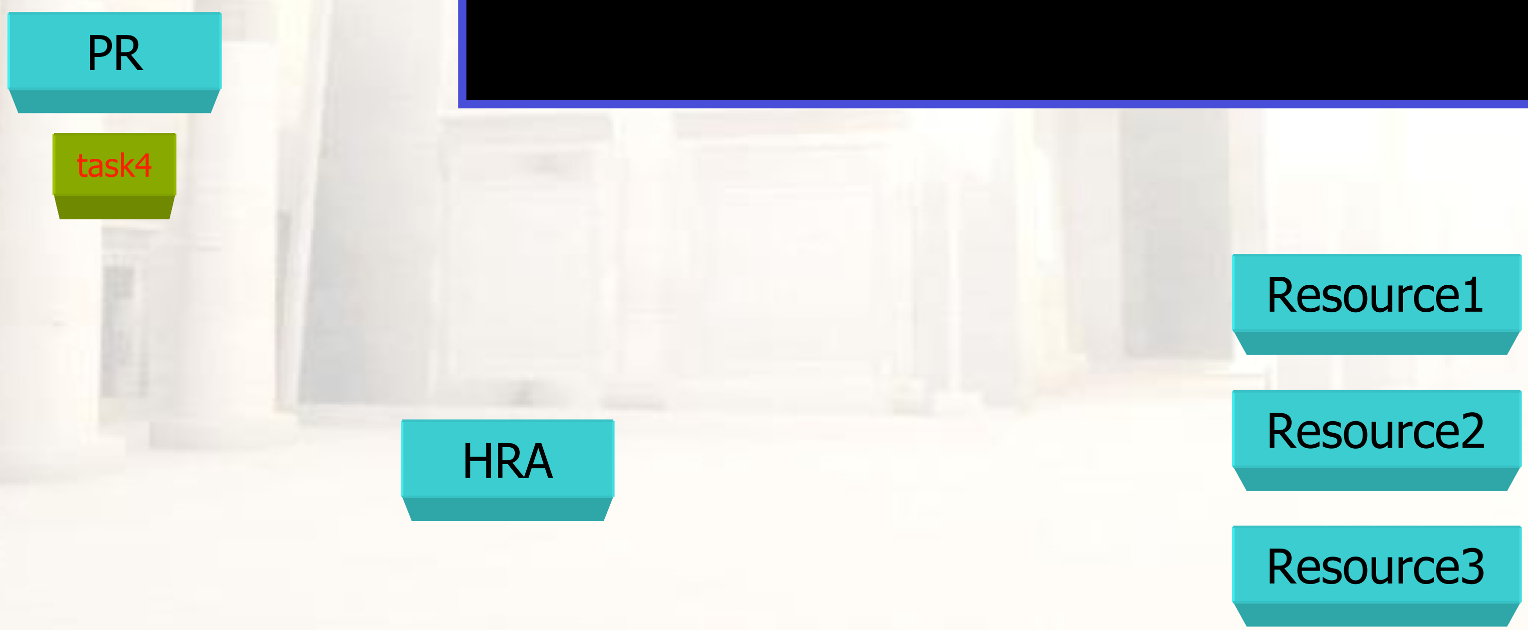
HARDWARE RESOURCE ALLOCATION

Catalogue and group hardware by several aspects:

- Latency (Local/Remote memory interface)
- Bandwidth (on PC Bus or via some other channel)
- Size and configuration of resource local memory
- Computational power for each type of basic primitive
- Capacity of reconfiguration and setup time on reconfiguration

HARDWARE RESOURCE ALLOCATION

```
PR : Request(task1, task2, task3)
PR : Request(task4, task5, task6)
HRA: Process()
HRA: Send(resource2, task1)
HRA: Send(resource3, task2, task5, task6)
HRA: Send(resource1, task3)
HRA: Send(resource2, task4)
```



FEATURES OF ROD

- Selective Rendering - SharpEye
- Cost Prediction - quotes to clients
- Automatic Perceptual Level of Detail
- Platform Independence Parallelism
  - Dedicated cluster
  - Grid
- Efficient Parallelism
  - Spatial and temporal coherence
- Multi-Sensory Rendering

It's all about Perceived Realism!

- RADIANCE in real time
  - ⇒ innovation
- Possibilities for multi-sensor, multi-user experiences
- ⇒ **Rendering on Demand**

THE END

Thanks...

- This presentation is available at:

http://www.cs.bris.ac.uk/home/pereira/docs/rw2004.pps

- RoD:

http://www.3cresearch.co.uk/3cprojects/rod

Thanks to:

Greg Ward – Ideas and support

Alan Chalmers, Gavin Ellis, Kurt Debattista, Peter Longhurst, Richard Gillibrand, Tomas Plachetka and Veronica Sundstedt