Optimisation

An overview of
strategies and methods for
attaining sufficient performance

Contents

- Minimising lag
- Graphics pipeline
- Culling algorithms
- Managing the scene
- Polygon reduction
- Level of detail
- Mathematical representations
- Billboards
- Optimisation Tools

Minimising Lag

- Need to ensure that system lag is kept to a minimum to achieve interactive and comfortable 3d environments
- Lag is time it takes between a change being effected on a virtual environment and the result being reflected on the display
  - Transport delays (input-output)
    - system takes a while to respond to user input
  - Frame rate (rendering & processing)

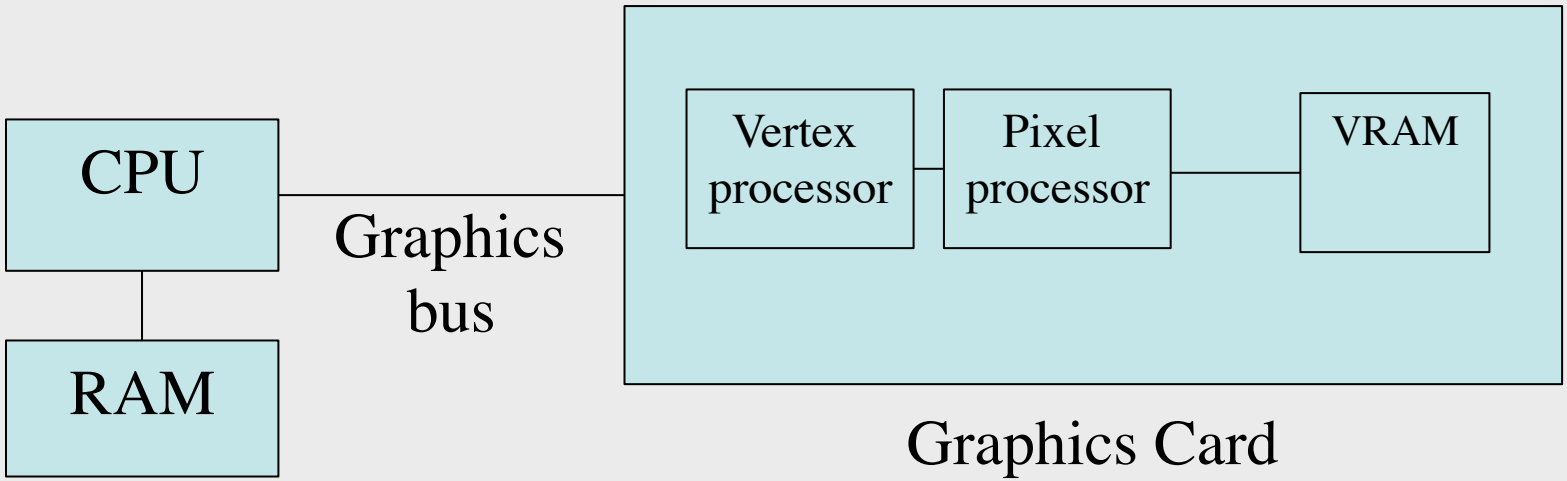
System lag

- System lag is cumulative effect of input-output, processing & rendering delays
- Effects of Lag
  - Degraded user performance
  - Degraded sense of presence
  - Motion sickness

Target platform considerations

- When designing a virtual environment consider what the minimum target platform is
  - Don't assume "high-end" graphics features!
- Decide what the minimum frame rate should be for the minimum platform
  - The do your best to achieve it!
  - A frame rate less than 8-10 fps on the minimum target platform is rarely acceptable
- Design to degrade gracefully in order to maximise the number of potential users

Graphics pipeline



- The parts of the pipeline runs in paralell
- Where is the bottleneck?

Graphics pipeline

- CPU limited
  - Starving GPU while busy with physics, AI etc
- Graphics bus limited
  - Transferring coordinates and textures every frame
- Vertex limited
  - High triangle count
  - Expensive vertex shaders
- Fill-rate limited
  - Overdraw
  - Expensive pixel shaders

Graphics pipeline

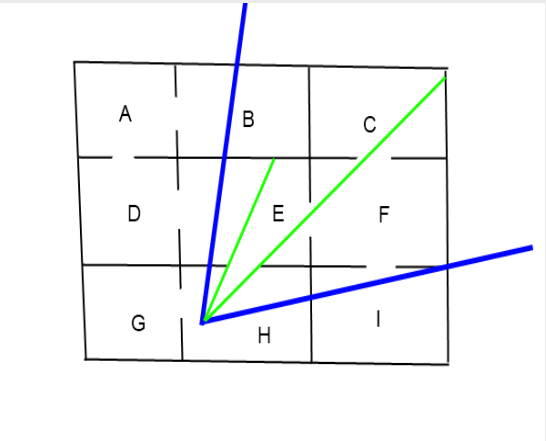
- GPU profilers can be used to determine bottlenecks
- Available from hardware manufacturers
  - AMD GPU PerfStudio
  - Intel Graphics Performance Analyzer
  - NVIDIA PerfHUD
- Some may only support D3D
- Tip to check if fill-rate limited
  - Change window size
  - At least partly fill-rate limited if frame-rate differs
  - Bottleneck can change within a single frame

Culling Algorithms

- Determines what faces or objects are invisible
  - Culled objects are not drawn
- Back-face culling
  - Triangles not facing the camera are culled
- View frustum culling
  - Checks bounding volume against view frustum
  - Implemented in jME and most scenegraph APIs

Culling Algorithms

- Portal culling
  - Cells connected threw portals
  - Cells are clipped against portals to determine which is visible



CEFH are visible

Culling algorithms

- Potentially visible set (PVS)
  - Each cell has a precalculated list of which other cells a potentially visible
- Occlusion culling
  - First draw occluders
  - Then draw simplified version of occluded object to determine if it is invisible

Think about Geometry

- The renderer does a lot of work for us in trying to update our virtual environments
  - by giving it hints, we can dramatically increase it's ability to run smoothly
  - most obvious place to start is by looking at the geometry
- In most cases should try to use as few polygons as possible in models
  - Exception to this is very large objects which should be spatially subdivided if they are typically only seen partially by the user
  - Models that are exported from CAD systems and some 3D modelling systems often have far more polygons than necessary and should be reduced

Managing the scene

- Assist the renderer
  - Organising the scene correctly to help the culling code to remove unseen objects
  - Minimising memory use through reuse of objects
  - (Using fog to hide culling activities if necessary)
- Help the renderer and reduce processing
  - Subdivide the world and minimise processing as appropriate
  - Turn off processing that has no effect on what the user can see (e.g. disable controls on objects that are not visible)
  - Implement level-of-detail techniques

Managing Textures

- Use textures instead of modelling lots of detail
  - but note that many large textures (1024x1024 or larger) will also affect performance
  - keep them as small as possible and note that most renderers work most efficiently with textures that have powers of 2 sizes
- Multiple-resolution textures (MIPMAPs) are slower to load because textures need to be processed, but offer more attractive textures and often run faster
  - Similar to level-of-detail (more on this later)

Polygon Reduction

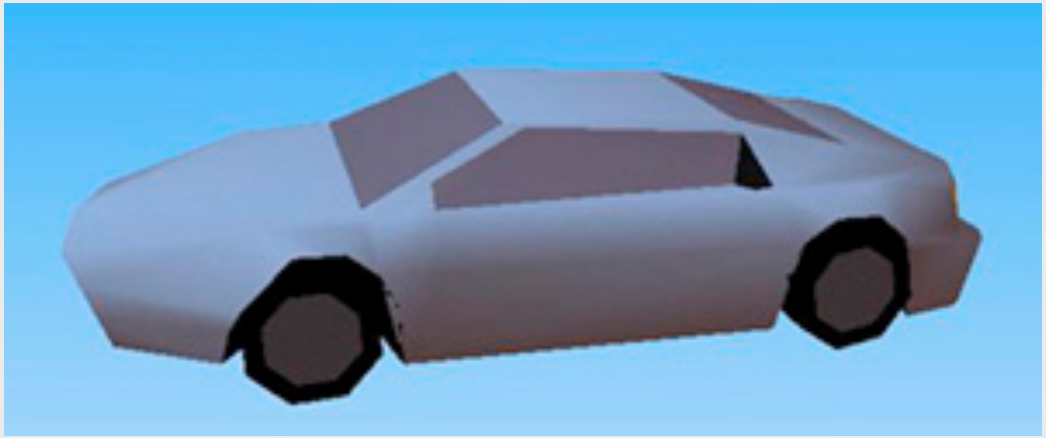
- Goal: Reduce the amount of geometry while maintaining a target level of detail

Polygon Reduction

- Polygon reduction has a couple of important uses related to improving the performance of a geometry:
  - Create multiple “levels of detail”
  - Take large inefficient geometries and (hopefully) simplify them without sacrificing significant detail
  - Both are related since the measure of 'significant detail' is often a function of what can be perceived by the user
- Often there are details in a model that do not contribute to the perceived appearance of the model
  - Especially models exported from tools not designed for “low-polygon” modelling

Polygon Reduction

Polygon Reduction



Polygon Reduction

- Most polygon reduction algorithms attempt to collapse edges using various criteria to determine the order in which the edges should be collapsed
- By repeating the process many times on a model, it can be reduced to any target level
  - A couple of other techniques will be mentioned in the section on level of detail later in the lecture

Polygon Reduction

- Algorithms that are commonly used (often in combination) include:
  - Remove smallest triangles
  - Remove smallest edges
  - Merge parallel edges
- Good polygon reduction tools/algorithms will take into account colour boundaries and texture maps in addition to attempting to maintain the overall form of the geometry

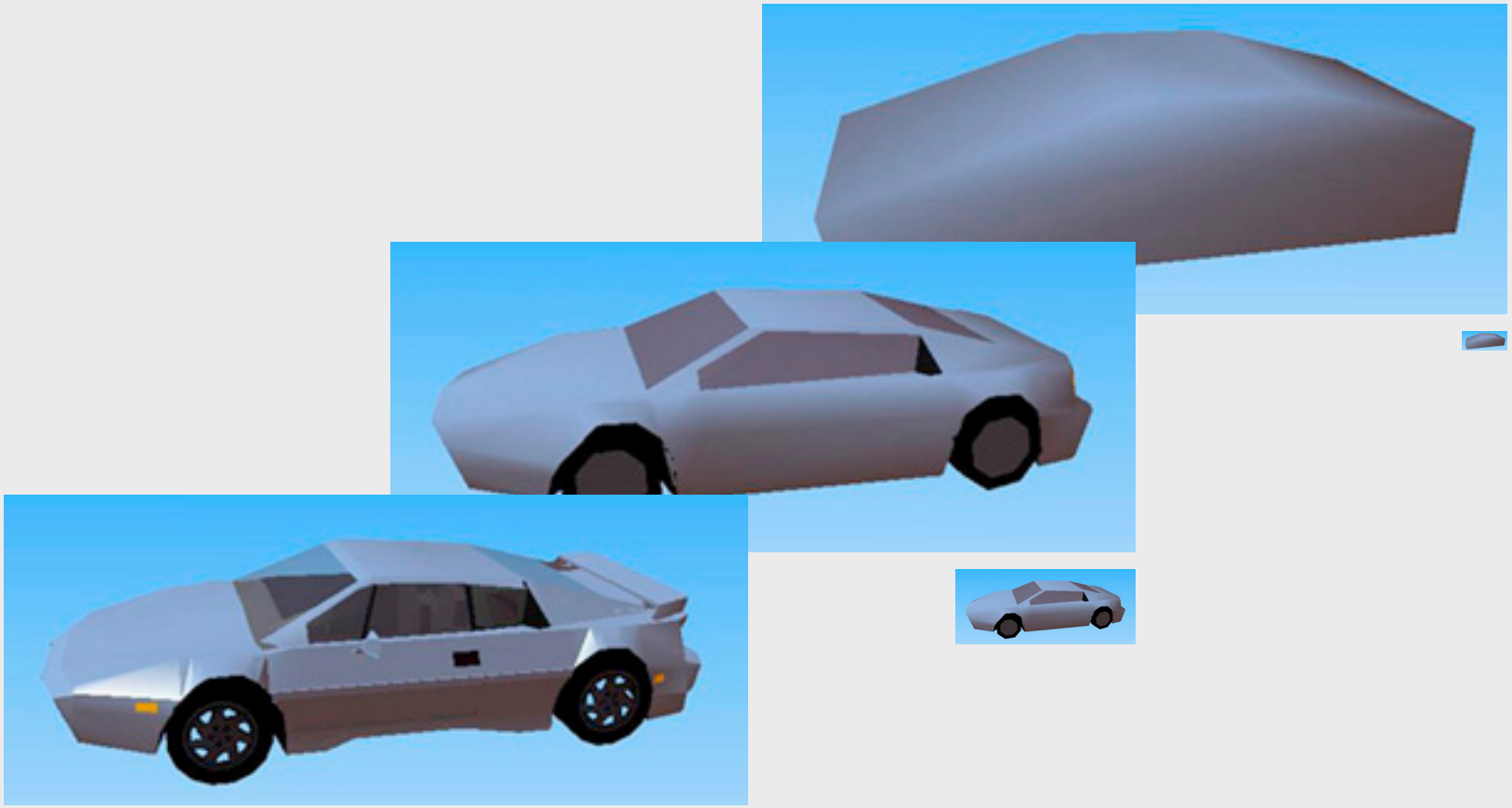
Polygon Reduction

- Shading also plays a significant purpose in helping us to keep the polygon count low
- A rounded object with smooth shading will often look much nicer (and use fewer polygons) than a similar flat-shaded object that attempts to use a large number of polygons to approximate roundness.

Level of Detail

- Can be implemented by a Node having children with different amounts of detail
- Then hide all but on of the children
- Use Spatial.setCullHint(CullHint) to hide or show a subgraph

Level of detail: Example



Level of detail

- There are five principle criteria in common use for modulating level of detail
  1. **Distance** - object's LOD is based on distance from observer
  2. **Size** - object's LOD is based on the pixel size on the display device
  3. **Eccentricity** - object's LOD is based upon the degree to which it exists in the periphery of the display
  4. **Velocity** - object's LOD is based on it's velocity relative to the observer (across the display device)
  5. **Fixed frame rate** - object's LOD is modulated to achieve and maintain a prescribed frame-rate for the simulation

Level of detail

- Ideally, the switch between levels of detail for an object should not be noticed by the user
  - This is quite difficult to achieve well in practice
    - Some try to morph between LODs or fade between them to reduce the "popping effect" of switching LODs
      - Fading can be achieved by animating transparency
- Tip: In general, do not switch to/from less detailed levels of detail too early
  - The popping effect that may result can be disorientating for the user

Level of detail

- Research issues also include methods for producing simplified geometry efficiently
- Automatically simplified geometries are typically created using one of the following strategies:
  - **geometry removal –** removing vertices or polygons from objects
  - **sampling –** attempting to create a model that fits a sample of an object's geometry (i.e. it' s "shape")
  - **adaptive subdivision –** refining a simplified model by subdividing the model where it varies from the original

Level of detail

- Not only geometry, but textures and materials can be varied at different detail levels to improve performance
  - e.g. removing specular highlights from materials and using low-resolution textures at a distance
- LOD techniques are especially important to terrain visualisation

Level of Detail References

- Book: Level of Detail for 3D Graphics
  - D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner
- Thesis: Perceptually Modulated Level of Detail for Virtual Environments
  - Martin Reddy's PhD thesis (freely available)
  - Especially Chapters 1 and 2
    - includes a general overview of system lag, level of detail strategies, and level of detail generation

Mathematical Representation

- NURBS, fractals, etc.
- Compact representation
  - Can load quickly
  - Use little memory when not in use
- Require effort to dynamically generate polygonal representations at run-time
  - Typically require faster hardware
    - Especially if continuously updated
  - Very effective if not too many of them

Billboards

- A billboard is a (typically flat) shape that is always oriented towards the user
  - it rotates automatically around a user-specified axis to face the user
- Much used to add scenery to models in a CPU-efficient manner (e.g. people and trees)
- Wide variety of other uses
  - help messages and “signposts”
  - text annotations
  - status or info displays

Billboards

Optimisation tools

- Tools exist to assist in the task of reducing polygons
  - for creating levels of detail
  - For simplifying models exported from CAD systems so the target system can cope with the model in real-time
- Some 3D modelling systems have polygon optimisation tools built-in

Summary

- Minimising system lag is important
- Try to find the bottleneck before optimising
- Managing the scene geometry can play an important role in reducing lag
- Polygon reduction often necessary to prepare models for interactive 3D use
- Switch-techniques including LOD are useful for scene geometry management
- Billboards can be a useful alternative to complex geometry in some situations