Rendering a Large Amount of Units

Silver Kirotar
Contents

● Overdraw
● Culling
● Draw calls
● Batching
  ○ Dynamic
  ○ Static
● Geometry Instancing
Rendering a large amount of units…?
Overdraw I - Definition

A pixel on the screen is being redrawn in a single frame. When 3D rendering:
● A pixel is replaced by a closer one.
● Distance is determined by Z coordinates towards the camera.

What are the problems?
● Rendering “empty” pixels/polygons == Wasting time.
● Redrawing non-transparent pixels.
Overdraw II - Minimizing (3D)

- Reducing transparent areas in meshes.
- Why? Triangles vs Big Unused Transparent Areas?
  - Triangles are cheaper
Overdraw III - Example (3D)
Overdraw IV - Example (2D)

- Reducing transparent areas in images.
- Not many options...
“Culling”

Selects objects for rendering operations
  ● in a defined region of interest.
Makes rendering quicker and more efficient.

What I mean:
  ● Frustum culling
  ● Back-face culling
  ● Occlusion culling
Frustum culling

View frustum - volume in space from a given viewpoint.

Only objects in view frustum are sent for rendering.

- “Potentially visible” objects.
Back-face culling

- Determines if a polygon is visible.
- Reduces the number of polygons to be drawn.
Occlusion Culling

Also
- Hidden surface determination
- hidden surface removal (HSR)
- visible surface determination (VSD)

Determines surfaces and parts of surfaces that are not visible from a certain viewpoint.
Culling illustrations

View Frustum Culling

- Geometry outside of the view frustum is removed.

Backface Culling

- Backfacing geometry is removed.

Occlusion Culling

- Occluded geometry is removed from further processing.
Draw calls I - Definition

- A number of materials drawn.
- For all objects.
  - Some objects have multiple materials.

(Also takes in count dynamic lighting)
Draw calls II - Minimizing

Shared material

- Create texture atlases.
- Single big vs several smaller textures.

- Separate textures which
  - use alpha values.
  - do not use alpha.
Draw calls III - Large objects

Large images with small amount of transparency.

- Separate areas with transparency.
  - Define subimages as alpha or no alpha.
- If possible, use
  - smaller mipmap levels of images.
  - meshes with smaller level of detail.
Draw calls III+ - Image separation example
Batching

- Multiple meshes are merged together.
- Reduces communication between CPU and GPU.
- Improves performance.
Dynamic batching

- Automatic*, used each frame.
- Reduces draw calls for objects that
  - share the same material.
  - can be moved.

Useful when transforming vertices is cheaper than doing these same draw calls.
Static batching

- Reduces draw calls for geometry that
  - shares the same material.
  - does not move.
- Usually more efficient than dynamic batching.
  - Pre-calculated
- Downside: Uses more memory.
- Bad examples: Trees in a dense forest.
Geometry Instancing

- Copies of mesh in different locations.
- Needs to know the position of each object.

- Especially useful for thousands of meshes.
- Used for repeated geometry, like
  - trees, grass, buildings,
  - Or characters.
Tips for better performance

● Simplify meshes.
● Use reasonable level of detail.

● Try vertex coloring.
● Try gradient mapping.
● Avoid dynamic lighting.
Thank you for listening!
See also

Overdraw in frontend development:
https://www.youtube.com/watch?v=T52v50r-JfE

Reducing polygon count:
https://blender.stackexchange.com/questions/78499/how-to-decrease-the-polygon-count-on-my-mesh

Optimizing graphics performance:
References

https://en.wiktionary.org/wiki/overdraw
http://polycount.com/discussion/162570/mobile-graphics-optimization
http://polycount.com/discussion/89154/overdraw-how-does-it-work-and-how-bad-is-it
https://stackoverflow.com/questions/2856448/how-to-prevent-overdrawing
https://en.wiktionary.org/wiki/mipmap
https://forum.unity.com/threads/what-are-draw-calls.27416/
https://docs.unity3d.com/Manual/DrawCallBatching.html
https://www.gamedev.net/articles/programming/graphics/opengl-batch-rendering-r3900/
https://www.khronos.org/opengl/wiki/Vertex_Rendering#Instancing
https://en.wikipedia.org/wiki/Viewing_frustum
http://slideplayer.com/slide/5268774/
Images borrowed from...

https://www.youtube.com/watch?v=lqWqRc7J0BU
http://polycount.com/discussion/162570/mobile-graphics-optimization
https://opengameart.org/content/lpc-tile-atlas
https://www.youtube.com/watch?v=c-UskAGQaBQ
https://en.wikipedia.org/wiki/Viewing_frustum
http://slideplayer.com/slide/5268774/