
GameDevelopers Conference
MARCH 20-24
SAN JOSE, CALIFORNIA
WHAT'S NEXT
GDC:06
www.gdconf.com
GAME DEVELOPERS CHOICE AWARDS
INDEPENDENT GAMES FESTIVAL
GDC MOBILE
SERIOUS GAMES SUMMIT
GAME CONNECTION
CMP

Extremely Practical Shadows

Tom Forsyth

Programmer, RAD Games Tools
www.radgametools.com
www.eelpi.gotdns.org

Background

- Day job is “Granny” at RAD Game Tools
  - Animation middleware
  - This talk is nothing to do with that
- Previously a “real” game developer
  - Specialised in graphics
  - Particularly shadows and lighting
- This is the sequel to my GDC 2004 talk
  - Lots more depth
  - Bugs fixed!

Volume shadows suck

- Limits geometry – must be watertight
- Alpha test & pixel kill don't work
- High GPU/CPU cost to compute volumes
- Hard-edged
- Poor scaling with hardware trends
  - Fillrate increasing slower than computation
- My assertion: "Not The Future"
  - But prediction is hard ☺

Shadowbuffers suck too

- Spatial aliasing
  - Insufficient resolution
  - Non-uniform projection
- Depth aliasing
  - Surface acne
  - Peter-Pan syndrome
- FOV limits
  - Fundamental 180 degree limit
  - ~120 degrees in practice

The projection problem

- Shadowbuffer is rendered from light POV
- Then projected into camera POV
- The two do not agree
  - Can violently disagree – “duelling frustums”
- Too many texels in some places
  - Inefficient use of memory and fillrate
- Too few texels in others
  - Visible aliasing

Projection solutions

- Use smarter projections
  - Lots of freedom in the light-space projection
  - Perspective Shadow Maps (PSM)
  - Light-space Persp. Shadow Maps (LiSPSM)
  - Trapezoidal Shadow Maps (TSM)
- All help with spatial aliasing
- Only minor help with depth aliasing
  - Some make it worse!

Perspective Shadow Maps

- Stamminger and Drettakis, 2002
- Performs viewer perspective on scene
  - Objects close to camera get bigger
- Renders shadowbuffer from light POV
  - So objects close to camera get more texels
- Good redistribution of resolution
- Simple to implement basic algorithm

Perspective Shadow Maps

- Light rays no longer parallel
  - Directional lights become point lights
  - Points become directional or “inverse” points
  - Lights can cross div-by-zero plane
  - Shadow casters can too!
- Very difficult to make robust
  - Lots of special cases & discontinuities
  - Solutions are unintuitive and fragile
  - “Friends don’t let friends implement PSM”

Perspective Shadow Maps

- Demo

Light-Space PSM

- Wimmer, Scherzer, Purgathofer 2004
- Similar perspective distortion to PSM
  - But only perpendicular to light rays
  - So light rays are still parallel
  - No discontinuities to worry about
- Complex to implement
  - Needs volume/volume intersection code
  - But there's recipes to follow
  - And you can approximate

Light-Space PSM

- Demo

Trapezoidal Shadow Maps

- Martin and Tan, 2004
- Also distorts light space
  - Perpendicular to light direction like LiSPSM
- Visible frustum approximated by a trapezoid
- Trapezoid is distorted to square shadowbuffer
  - Not directly related to viewer perspective
- Over-distorts in many cases
  - Too much near detail
  - So they use multiple affine projections
- Can maintain some frame-to-frame continuity

Trapezoidal Shadow Maps

- Distortion is segmented, not affine
  - Solved by pixel-shader lookup (1D texture)
  - But depth is still tricky
- Finding the trapezoid is complex
  - Problems with “infinite” view frustum
  - Iterative process finds distortion segments
- Designed for large ground planes
  - Not very good at more “3D” worlds
  - Or odd-shaped ones (see later!)

Trapezoidal Shadow Maps

- Demo

Smart projection problems

- None of them solve duelling frustum case
  - Camera looking at light source
  - Two frustums are directly opposed
  - Projection has no degrees of freedom
  - All do the same as standard projection
- None of them solve omni lights
  - Cannot render FOV >180 degrees
  - Guaranteed duelling frustum case!

What I tried

- Simplified LiSPSM
  - Affine perspective distortion
  - Perpendicular to light rays
  - In direction of viewer Z
- I know more about my scene
  - Vanilla algorithm assumes near-infinite view
  - I have more control over what is in frustum
  - So I can use a simpler algo

    (see later for details)

Multi-frustum partitioning

- Splits scene into multiple light frustums
  - Each frustum rendered separately
- Solves the two big problems
  - Duelling frustums
  - Omni lights
- Helps in other ways
  - Copes gracefully when smart projection fails
  - More control over frustum contents
  - …allows “dumber” smart projection

MFP assumptions

- Each light has multiple frustums
- Each object is caster and/or receiver
- Receivers use only one frustum
  - Only care about receivers visible to viewer
- Casters can affect multiple frustums
  - Still care about casters not visible to viewer
- Algorithm mainly concerns receivers
  - Casters are easy to deal with
  - Standard GPU clipping does the right thing

MFP basic algorithm pt1

- For each receiver in scene
  - Find screen size of bounding volume
  - Texels-per-screen-pixel global quality setting
  - Result is texels-per-world-meter value
- Each light frustum has
  - Frustum direction/FOV
    - I use a circular cone for simplicity
  - Texels-per-degree variable
  - List of receivers that use frustum

MFP basic algorithm pt2

- Receiver calculates texel density for light
  - (ignore receivers not visible to viewer or light)
  - Find FOV angle relative to light position
  - So find texels per degree of light angle
- Receiver looks for suitable frustum
  - Must contain receiver volume in frustum
  - Must have at least enough texels-per-degree
  - But not too high - wastes fillrate & memory
    - Heuristic factor of ~4x allowed

MFP basic algorithm pt3

- If no frustum found, try to enlarge one
- Increase frustum FOV
  - …and alter direction of frustum
  - Union of new receiver and existing list
  - Texture size limit (texels per degree * FOV)
  - FOV distortion limit – around 120 degrees
- Increase texels per degree
  - Not past factor of 4x to any receiver in list
  - Prevents fillrate & memory waste

MFP basic algorithm pt4

- If still no suitable frustum, make new one
  - Exactly fits this receiver
  - Exactly matches texels-per-degree
  - Can be expanded to include other receivers
- Receiver may exceed single frustum
  - Too large to fit in 120 degree FOV
  - Too large for texture size restrictions
  - May need chopping

  (see later for more)

MFP + smart projections

- MFP can use any projection
- Different projection for each frustum
  - Can use PSM where no singularity problems
  - Use TSM/LiSPSM/standard elsewhere
- Each frustum is well-constrained
  - Can bias away from PSM singularities
  - TSM not good at large/odd-shaped frustums
  - LiSPSM gets more information for fine-tuning
- Guaranteed fallback to standard proj.

MFP results

- Consistent texel density
  - 4x best/worst ratio – filtering can cope
  - Objects close to viewer get more texels
  - Arbitrarily large scale variation in scene
- Distance from light is accounted for
  - Not perfect – surface may be edge-on
- Duelling frustums solved by partitioning
  - Usually only need 3-4 regions at extreme
  - Smart projections help, but not required

MFP results

- <120 degree FOV chunks
  - Wide/omni lights use multiple frustums
- Better than cube maps
  - “Cube sides” can be different resolutions
  - Unused/invisible “sides” never considered
    - Omnis with nothing above them but sky
    - Omnis beside viewer, shining into scene
- Omnis always have a duelling frustum
  - “Side” with duelling frustum is partitioned

MFP practical results

- Retrofitted to StarTopia (2001)
  - RTS/”god game”
  - Player-built world – no preprocessing
- 2D grid-aligned world
  - But shadowing system doesn’t know this
    - There’s no cheating!
  - Many objects larger than a grid square
  - Many non-aligned characters
  - World is bent into a ring

MFP practical results

- No gameplay or artwork changed
  - Alpha-test & clip planes used extensively
  - Already had lighting info, but no shadows
  - Many objects contain lights
  - All omnis – rampant duelling frustums!
- Very few lights altered
  - Some moved to nicer-looking places (lamps)
  - Shadows for “mood lights” turned off
- This implementation is slow
  - Engine is bad at multiple rendering passes

MFP results

- Demo
  - “Patch” downloadable from www.eelpi.gotdns.org

    (StarTopia is ~$3 from Amazon)
  - Email me for debug modes shown here

MFP problems - chopping

- Large/close receivers are problems
  - No single frustum can cover whole object
  - But first reject against light & viewer volumes
  - Problem part may be out of range/view
- May already have finer chunks
  - Material / bone limit boundaries
  - Procedural, e.g. landscape tiles
  - Precalculated split points (long walls & floors)
- But sometimes they still need chopping

MFP problems - chopping

- Not very common
  - So pick simplest method
- Split into arbitrary pieces
  - Need at worst six: cube-map chop planes
- Find frustum/SB for each piece
- Render object with all shadowbuffers
  - Ensure out-of-frustum reads = “dark”
  - Take maximum brightness of all results
  - Hardware limits may require multiple passes

MFP problems - popping

- Sudden aliasing change between frames
  - Object moves from frustum to frustum
  - Frustum changes direction/FOV
  - Frustum changes resolution
- Frustums change as objects move
  - Also as viewer or light moves
  - …but popping is far less visible then
- Frustum algorithm is greedy
  - Small movements can cause large changes
  - Changes are not localised to moving objects

MFP problems - popping

- Limit changes to existing frustums
  - Texel->world mapping cannot change
  - Can pan by whole texels
  - Can change texture size within limits
  - Can change scissor to avoid extra overdraw
- Make new frustums “steal” from old ones
  - Old ones removed when no objects in them
  - Limits number of old “fragmented” frustums

MFP problems - popping

- Blend when object changes frustum
  - Keep adding to both old and new
  - Render twice & crossfade
- But object may not fit in old frustum
  - New position, size, etc
  - There's a reason it changed frustum!
  - Will still pop, but it's rare
- Pop not as visible on animated objects
  - Only need this for moving non-anim, e.g. cars

MFP problems - seams

- Adjacent objects in different frustums
  - Slightly different aliasing between the two
- Only a problem with smooth joins
  - Floor/land tiles, wall sections
  - Hidden by noisy textures
- Discontinuity invisible over sharp joins
- Soft shadows help hide seams
  - Screen-space blur effective, but expensive
  - ATI “Parthenon” demo explores solutions

MFP problems - seams

- Store smooth connectivity between receivers
- Grow object volumes to include neighbours
  - Frustum can now cover both sides of seams
  - Does not create significantly larger frustums
- Both objects on a seam rendered twice
  - Once with each frustum
  - Blend at 50:50 at seam edge
- Requires vertex weights
  - Annoying preprocessing for general meshes
  - Easy with procedural geometry (floor tiles)

Render target management

- Allocate a few large textures
- Quadtree allocation inside each
  - Very fast, low fragmentation, easy to write
- Each allocation uses same texture as last
  - Minimise render-target changes
  - Only switch when you have to
  - Then stay switched
- Leave 1-texel borders
  - Careful with filtering, can't use CLAMP mode

Depth aliasing

- Lots of biases to use and tweak
  - 24 bits is still not enough
  - Low precision far from light, but may be near viewer!
- Bias too high
  - Shadows detach from feet – “Peter-Pan” syndrome
- Bias too low
  - Incorrectly self-shadows - “surface acne”
  - Render backfaces – terrible Peter-Panning!
  - Midpoint rendering – expensive, tricky
- No one bias value works for an entire scene

ID buffers

- Store object ID (not depth) in shadowbuffer
- Compare to ID of rendered object
- If they match, object is lit
- No math
  - No biases, no epsilons, no precision problems
- 256 IDs usually enough
  - Chances of collision are low
  - Collision causes no shadow, not extra shadows
- 65536 IDs enough
  - You have how many objects in your scene?

ID buffers

- Dark halos
  - Sampling at object edge picks up IDs behind
  - Can use “priority buffer” – requires ordering
- Point-sample nearest four texels
  - In shadow if no ID matches
  - Shadows shrink by a texel – not noticeable
- No self-shadowing
  - Could use ID per face
  - …needs lots of IDs & hardware support

Depth + ID buffers

- ID buffers used for inter-object shadows
- Depth used for self-shadowing

```
if ( buf.ID != obj.ID )
    in_shadow();
else if ( buf.depth < obj.depth )
    in_shadow();
else
    draw_lit();
```

Depth + ID buffers

- Depth range only spans a single object
  - Inter-object shadows done with IDs
- Few bits needed
  - Blade 2 on Xbox1 used only 8 bits
  - …one depth buffer per character
  - 8 bits over 2 meters = ~1cm resolution
  - Barely enough – 16 bits probably wiser
- Bias can be set differently for each object
  - Set once per object, no scene dependencies

Depth + ID buffers

- 8 bit ID + 8 bit depth = 16 bits
  - But probably need more depth bits
- Some effects need a global depth
  - Depth of field
  - Soft-edged shadows
  - All are more tolerant of precision/bias errors
- Can just use regular 24-bit Z buffer
  - 8-bit ID stored in stencil channel
  - If hardware supports it
  - Still use per-object bias

Depth + ID buffers

- Demo
- 8-bit ID - ID 0 reserved for floor tiles
- Other 255 IDs assigned in render order
  - Front to back (be nice to your Z buffer!)
  - Chance of collisions low – widely spaced
- Depth bias scaled by object radius
  - Could have artists fine-tune them per-object
  - But I have no artists ☺

Soft shadows

- PCF looks bad
  - Can get away with it on noisy surfaces
  - More taps are expensive, little extra quality
  - Not depth-dependent (blurry feet look wrong)
- Smoothies get good results
  - Chan and Durand, 2003
  - But needs geometric volumes!
- Willem de Boer's variant of smoothies
  - Entirely image-based

Questions…

tom.forsyth@eelpi.gotdns.org

www.eelpi.gotdns.org

References

- “Perspective Shadow Maps”, Stamminger and Drettakis, Proceedings of ACM SIGGRAPH 2002
- “Light Space Perspective Shadow Maps”, Wimmer, Scherzer, Purgathofer, Eurographics Symposium on Rendering 2004

References

- “Anti-aliasing and Continuity with Trapezoidal Shadow Maps”, Martin and Tan, Eurographics Symposium on Rendering 2004
- “Combined Depth and ID-Based Shadow Buffers”, Kurt Pelzer, Game Programming Gems 4, Charles River Media 2004

References

- “Rendering Fake Soft Shadows with Smoothies”, Chan and Durand, Proceedings of the Eurographics Symposium on Rendering 2003
- “Smooth Penumbra Transitions with Shadow Maps”, Willem de Boer, www.whdeboer.com

References

- “StarTopia”, developed by Muckyfoot Productions, published by Eidos Interactive, 2001