Texture Streaming in Titanfall 2

Chad Barb
Senior Software Engineer,
Respawn Entertainment

What is Texture Streaming?

- Dynamic loading to improve image quality
- Conceptually a form of compression
- If you notice it, it’s not working.
- Common approaches:
  - Manual Segmentation
  - Bounding Geometry Tests
  - GPU Feedback

What is Titanfall 2?

- Fast-paced first-person shooter
  - Move and turn quickly, take cover
  - Semi-linear (not open-world,) vertical gameplay
  - Lots of customized skins and weapons in MP
  - 60Hz! Avoid new GPU passes or non-threadable CPU
  - Atop a very mutated fork of Valve Source

What Platforms?

- Xbox One / PS4
  - HDDs, not optical! Better seek and bandwidth
  - Modern partially-resident texture features.
- Windows
  - DX11 (inclusive minspec—a few years back)
  - Variety of Resolutions and GPU RAM

Workflow Requirements

- Minimize manual work for design and art
  - Artists map textures freely (no fixed density)
  - Can add MIPs without hurting other textures
  - Preprocessing should be stable
  - Some manual hinting okay
  - Works with ‘Asset Bakery,’ including hotswap

Algorithm Overview

- Any MIP below 64kiB is permanent.
- MIPs can be added/dropped one-by-one.
- Use precomputed information to build list of what’s important/unimportant.
- Work toward that list each frame.

What is a ‘Histogram’?

- Want to prioritize MIPs by how many pixels they cover on the screen (coverage,) not just ‘yes/no’.
- ‘Histogram’ is coverage per MIP per material.
  - 16 scalar ‘bins’ (usually a float)– one per MIP.
  - Assume a 4k x 4k texture at 256 x 256 screen res.
- Shift and scale for resolutions and moving models.
- Appropriately weighs small, occluded, or backfacing triangles using low-density texture mapping.

Algorithm - Precomputation

- Compute histograms per material
  - static: per ‘column’ of world using GPU render, into file.
  - dynamic: per model at load time:
    - Compute texture gradients for each triangle.
    - Add area of triangle to histogram bin for MIP
    - Planned to project at various angles, but wasn’t worth it
    - Manually tweaked scale factor to match static data.

What Happens Each Frame?

- Stream player's ‘column’ from disk, add model coverage
- Divide coverage by texel count to get a ‘metric’
- Generate list of most and least important MIPs
  - Finer MIPs cascade down (coarser always >= finer)
- Load most important MIPs, drop least important
  - Cap on in-flight count and bytes dropped per frame
  - Do not drop something unless you are loading something more important!

How do we Choose Probes?

- Run “rstream.exe <*levelname*>”
- Instantiate models, compute bounds
- Chop geometry into 16ft^2 columns
- Probes are eye height above upward-facing triangles
- Add hint probes (Use Z in nearby columns, too)
- Use k-means to combine into max 8 probes per column
- Store probe locations in log file for debug use

How do we Render Probes?

- Upload static Geo to GPU(s) once
- Render N_{probes} UnorderedAccessViews:

```
float2 dx = ddx( interpolants.vTexCoord ) * STBSP_NOMINAL_TEX_RES; // STBSP_NOMINAL_TEX_RES is 4096.0
float2 dy = ddy( interpolants.vTexCoord ) * STBSP_NOMINAL_TEX_RES;
float d = max( dot( dx, dx ), dot( dy, dy ) );
// miplevel is log2 of sqrt of unclamped_d. (MATERIAL_HISTOGRAM_BIN_COUNT is 16.)
float mipLevel = floor( clamp( 0.5f * log2(d), 0.0f, (float)(MATERIAL_HISTOGRAM_BIN_COUNT - 1) ) );
InterlockedAdd( outHistogram[interpolants.vMaterialId * MATERIAL_HISTOGRAM_BIN_COUNT + (uint)mipLevel], 1 );
```

- Do once per cube face (accumulate results)
- Opaque pass writes depth, transparent only tests
  - No framebuffer!

Compiling Probe Data

- Now have coverage per material per MIP at probes
- ‘Max’ to combine probes within each column
- Make records: material ID, MIP#, coverage (4 bytes)
- Store the 512 most important records per column
- Group columns 4x4 into ~32kiB streamable pages
- Indexed to stable global material IDs and positions
- One ‘.stbsp’ file per level (Not a BSP though!)

Managing Texture Assets

- Each compressed (and swizzled) texture file may have a ‘streamable’ segment.
- When building fast-loading ‘rpak’ file for a level, we gather into a second ‘starpak’ file.
  - For shipping, we use a shared starpak for all levels.
  - (Only <64kiB MIPs duplicated on disk)
- Starpak contains aligned, ready-to-load data.

Code – Crediting World Textures

```
Compute column (x,y integer), Ensure active page is resident (cache 4 MRU), or request it.
totalBinBias = Log$_2$(NOMINAL_SCREEN_RES * halfFovX / (NOMINAL_TEX_RES * viewWidthInPixels) )
For each material represented in column,
    For each texture in that material
        For each record (<material,bin,coverage>) in column (up to 16)
            If texture->lastFrame != thisFrame,
                texture->accum[0..15] = 0, and texture->lastFrame = thisFrame
            mipForBinF = totalBinBias + record->bin + Log2(textureWidthInPixels)
            mipForBint = floor( max( 0.0, mipForBucketF ) ), clamped to (16-1).
            texture->accum[mipForBin] += record->coverage * renormFactorForStbspPage;
```

Code – Crediting Models

```
float distInUnits = sqrtf( Max( VectorDistSqr( pos, *pViewOrigin ), 1.0f ) );
if ( distInUnits >= CUTOFF ) continue;
float textureUnitsPerRepeat = STREAMINGTEXTUREMESHINFO_HISTOGRAM_BIN_0_CAP; // 0.5f
float unitsPerScreen = tanOfHalfFov * distInUnits;
float perspectiveScaleFactor = 1.0f / unitsPerScreen;

// This is the rate of pixels per texel that maps to the cap on bin 0 of the mesh info.
// ( Exponentiate by STREAMINGTEXTUREMESHINFO_HISTOGRAM_BIN_CAP_EXPBASE for other slots )
float pixelsPerTextureRepeatBin0 = viewWidthPixels * textureUnitsPerRepeat * perspectiveScaleFactor;
Float perspectiveScaleAreaFactor = perspectiveScaleFactor * perspectiveScaleFactor;
pixelsPerTextureRepeatBinTerm0 = (int32)floorf(-Log2( pixelsPerTextureRepeatBin0 ); // Mip level for bin 0 if texture were 1x1.

For each texture t:
  if first use this frame, clear accum.
  if high priority, t->accum[clampedMipLevel] += HIGH_PRIORITY_CONSTANT (100000000.0f)
  For dim 0 and 1 (texture u,v):
    const int mipLevelForBinBase = (i32)FloorLog2( (u32)textureAsset->textureSize[dim] ) + pixelsPerTextureRepeatBinTerm0 ;
      For each bin
        // Log2 decreases by one per bin due to divide by two. (Each slot we double pixelsPerTextureRepeatBin0, which is in the denominator.)
        const int32 clampedMipLevel = clamp(mipLevelForBinBase - (i32)binIter, 0..15 )
        t->accum[clampedMipLevel] += modelMeshHistogram[binIter][dim] * perspectiveScaleAreaFactor;
  If accum exceeded a small ‘significance threshold’, update t’s last-used frame.
```

Code – Prioritization

```
For each texture mip,
    metric = accumulator * 65536.0f / (texelCount >> (2 * mipIndex));
    If used this frame:
        non-resident mips are added to ‘add list’, with metric.
        resident mips are added to ‘drop list’ with same metric.
    If not used this frame:
        all mips added to ‘drop list’ with metric of ( -metric + -frames_unused.)
          (also, clamped to finer mips’ metric + 0.01f, so coarser is always better)

Then partial_sort the add and drop lists by metric to get best & worst 16.
```

Code – Add/Drop

```
shift s_usedMemory queue
for ( ; ( (shouldDropAllUnused && tDrop->metric < 0.0f) || s_usedMemory[0] > s_memoryTarget) && droppedSoFar <
16MiB && tDrop != taskList.dropEnd; ++tDrop ) { drop tDrop, increase droppedSoFar; }
for ( TextureStreamMgr_Task_t* t = taskList.loadBegin; t != tLoadEnd; ++t ) { // t points into to add list
  if ( we have 8 textures queued || t->metric <= bestMetricDropped ) break;
  if ( s_usedMemory[STREAMING_TEXTURES_MEMORY_LATENCY_FRAME_COUNT - 1] + memoryNeeded <= s_memoryTarget ) {
   for ( u32 memIter = 0; memIter != STREAMING_TEXTURES_MEMORY_LATENCY_FRAME_COUNT; ++memIter ) {
    s_usedMemory[memIter] += memoryNeeded; }
   if ( !begin loading t ) { s_usedMemory[0] -= memoryNeeded; } // failure eventually gets the memory back
  } else for ( ;; ) { // Look for ‘drop items’ to get rid of until we'll have enough room.
    if ( planToLoadLater + memoryNeeded + s_usedMemory[0] <= s_memoryTarget ) {
     planToLoadLater += memoryNeeded; break; }
    if ( droppedSoFar >= 16MiB || tDrop >= taskList.dropEnd || t->metric <= tDrop->metric ) { break; }
    bestMetricDropped = Max( bestMetricDropped, tDrop->metric );
    drop tDrop, increase droppedSoFar;
    ++tDrop; } }
```

How do we Resize Textures?

- Windows/DirectX
  - Originally CPU-writable texture, map, read new MIPs
  - Create GPU texture, GPU copies new and old MIPs
  - Now just load into heap and pass to CreateTexture
- Console
  - Read new MIPs directly in
- Drops are queued 3 frames, to flush pipeline.

Asynchronous I/O

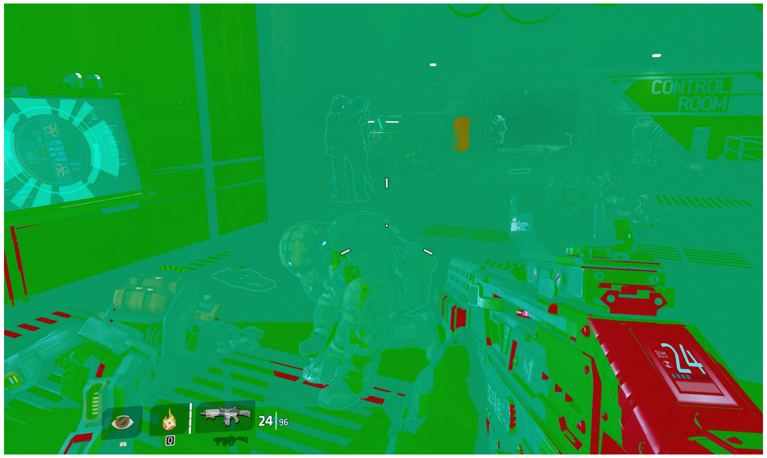
- Async thread
- 2 requests in flight
- Multiple priority queues
  - Textures low
  - Sound high
  - Reads occur in 64kiB chunks for interruptibility.

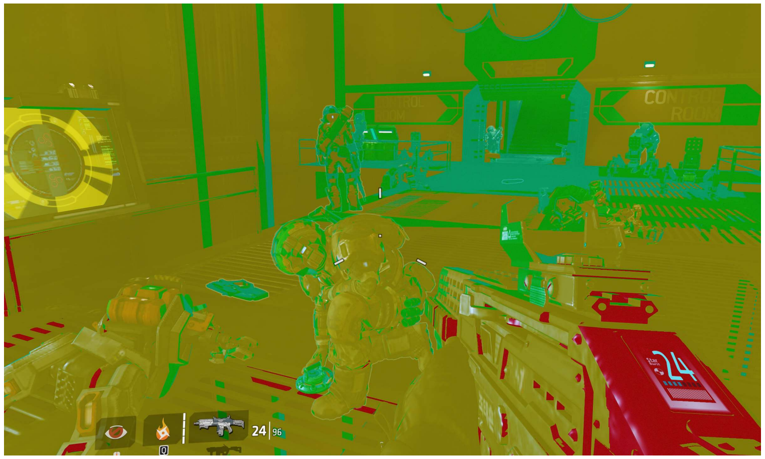
‘At a Glance’ Debug Shader

- Mip debug shader
  - Cyan = not using highest resident MIP (waste)
  - Green = using highest resident MIP
  - Yellow = could use higher streamable MIP
  - Red = could use higher MIP, none exists
- Best tool for quick test coverage.

‘At a Glance’ Debug Shader

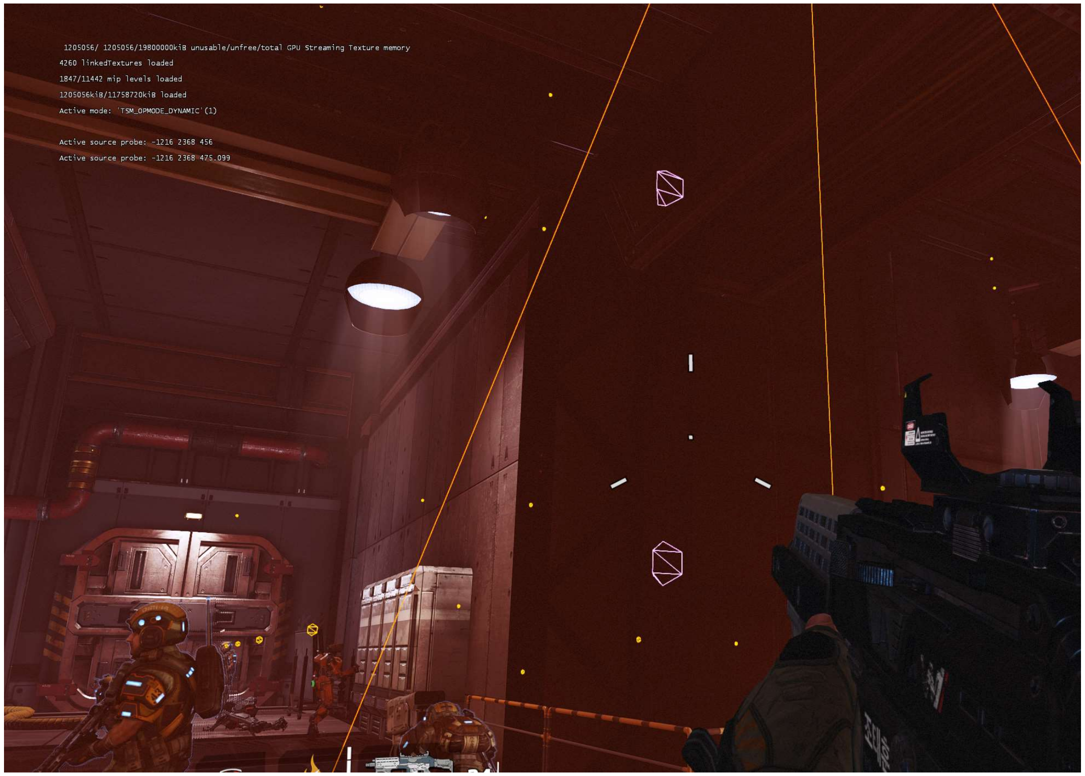
- Mip debug shader in action





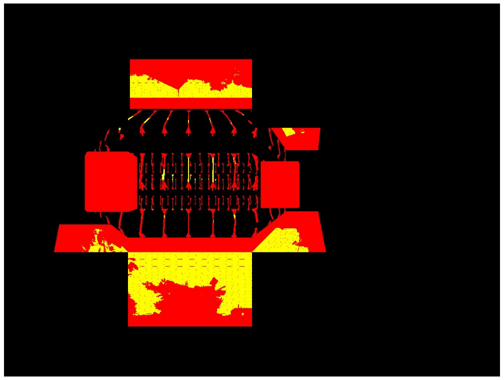
Debugging Probes

- See probe(s) used to generate current column
- Tool can render the cube maps for each material from that probe as PNGs
  - Track down what the histogram 'saw'
  - Red occluded, yellow not occluded
- Useful for particular materials not loading



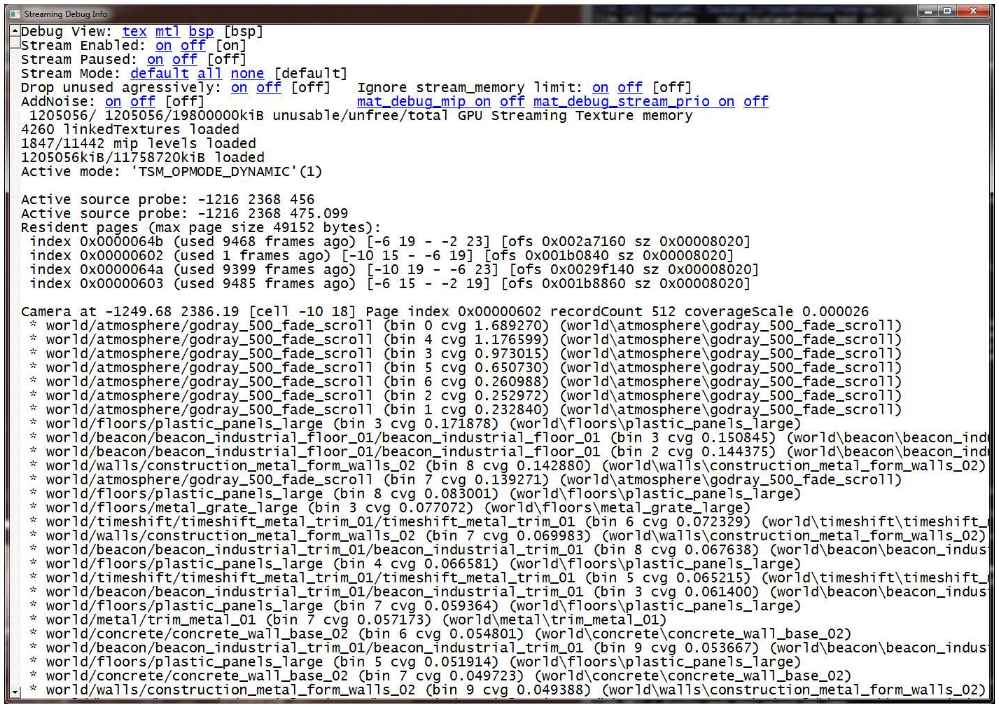
```
 1205056/ 1205056/19800000kiB unusable/unfree/total GPU Streaming Texture memory
4260 linkedTextures loaded
1847/11442 mip levels loaded
1205056kiB/11758720kiB loaded
Active mode: 'TSM_OPMODE_DYNAMIC'(1)

Active source probe: -1216 2368 456
Active source probe: -1216 2368 475.099
```



Other Debugging Tools

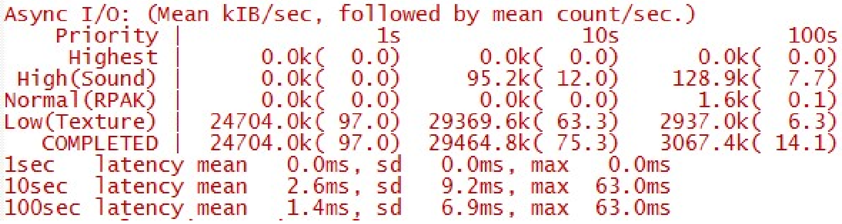
- All/None/Drop-*n*-finest modes
- Dynamic memory limit (or ignore)
- Add noise
- Aggressive drop
- Window with real-time reports
  - Metrics for materials and textures, column info
  - I/O: Mean/Max BW/Latency 1, 10, 100 seconds



```
Debug View: tex mtl bsp [bsp]
Stream Enabled: on off [on]
Stream Paused: on off [off]
Stream Mode: default all none [default]
Drop unused agressively: on off [off]   Ignore stream_memory limit: on off [off]
AddNoise: on off [off]                  mat_debug_mip on off mat_debug_stream_prio on off
 1205056/ 1205056/19800000kiB unusable/unfree/total GPU Streaming Texture memory
4260 linkedTextures loaded
1847/11442 mip levels loaded
1205056kiB/11758720kiB loaded
Active mode: 'TSM_OPMODE_DYNAMIC'(1)

Active source probe: -1216 2368 456
Active source probe: -1216 2368 475.099
Resident pages (max page size 49152 bytes):
 index 0x0000064b (used 9468 frames ago) [-6 19 - -2 23] [ofs 0x002a7160 sz 0x00008020]
 index 0x00000602 (used 1 frames ago) [-10 15 - -6 19] [ofs 0x001b0840 sz 0x00008020]
 index 0x0000064a (used 9399 frames ago) [-10 19 - -6 23] [ofs 0x0029f140 sz 0x00008020]
 index 0x00000603 (used 9485 frames ago) [-6 15 - -2 19] [ofs 0x001b8860 sz 0x00008020]

Camera at -1249.68 2386.19 [cell -10 18] Page index 0x00000602 recordCount 512 coverageScale 0.000026
 * world/atmosphere/godray_500_fade_scroll (bin 0 cvg 1.689270) (world\atmosphere\godray_500_fade_scroll)
 * world/atmosphere/godray_500_fade_scroll (bin 4 cvg 1.176599) (world\atmosphere\godray_500_fade_scroll)
 * world/atmosphere/godray_500_fade_scroll (bin 3 cvg 0.973015) (world\atmosphere\godray_500_fade_scroll)
 * world/atmosphere/godray_500_fade_scroll (bin 5 cvg 0.650730) (world\atmosphere\godray_500_fade_scroll)
 * world/atmosphere/godray_500_fade_scroll (bin 6 cvg 0.260988) (world\atmosphere\godray_500_fade_scroll)
 * world/atmosphere/godray_500_fade_scroll (bin 2 cvg 0.252972) (world\atmosphere\godray_500_fade_scroll)
 * world/atmosphere/godray_500_fade_scroll (bin 1 cvg 0.232840) (world\atmosphere\godray_500_fade_scroll)
 * world/floors/plastic_panels_large (bin 3 cvg 0.171878) (world\floors\plastic_panels_large)
 * world/beacon/beacon_industrial_floor_01/beacon_industrial_floor_01 (bin 3 cvg 0.150845) (world\beacon\beacon_indu
 * world/beacon/beacon_industrial_floor_01/beacon_industrial_floor_01 (bin 2 cvg 0.144375) (world\beacon\beacon_indu
 * world/walls/construction_metal_form_walls_02 (bin 8 cvg 0.142880) (world\walls\construction_metal_form_walls_02)
 * world/atmosphere/godray_500_fade_scroll (bin 7 cvg 0.139271) (world\atmosphere\godray_500_fade_scroll)
 * world/floors/plastic_panels_large (bin 8 cvg 0.083001) (world\floors\plastic_panels_large)
 * world/floors/metal_grate_large (bin 3 cvg 0.077072) (world\floors\metal_grate_large)
 * world/timeshift/timeshift_metal_trim_01/timeshift_metal_trim_01 (bin 6 cvg 0.072329) (world\timeshift\timeshift_
 * world/walls/construction_metal_form_walls_02 (bin 7 cvg 0.069983) (world\walls\construction_metal_form_walls_02)
 * world/beacon/beacon_industrial_trim_01/beacon_industrial_trim_01 (bin 8 cvg 0.067638) (world\beacon\beacon_indus
 * world/floors/plastic_panels_large (bin 4 cvg 0.066581) (world\floors\plastic_panels_large)
 * world/timeshift/timeshift_metal_trim_01/timeshift_metal_trim_01 (bin 5 cvg 0.065215) (world\timeshift\timeshift_
 * world/beacon/beacon_industrial_trim_01/beacon_industrial_trim_01 (bin 3 cvg 0.061400) (world\beacon\beacon_indus
 * world/floors/plastic_panels_large (bin 7 cvg 0.059364) (world\floors\plastic_panels_large)
 * world/metal/trim_metal_01 (bin 7 cvg 0.057173) (world\metal\trim_metal_01)
 * world/concrete/concrete_wall_base_02 (bin 6 cvg 0.054801) (world\concrete\concrete_wall_base_02)
 * world/beacon/beacon_industrial_trim_01/beacon_industrial_trim_01 (bin 9 cvg 0.053667) (world\beacon\beacon_indus
 * world/floors/plastic_panels_large (bin 5 cvg 0.051914) (world\floors\plastic_panels_large)
 * world/concrete/concrete_wall_base_02 (bin 7 cvg 0.049723) (world\concrete\concrete_wall_base_02)
 * world/walls/construction_metal_form_walls_02 (bin 9 cvg 0.049388) (world\walls\construction_metal_form_walls_02)
```



```
Async I/O: (Mean kIB/sec, followed by mean count/sec.)
    Priority |                1s                10s               100s
     Highest |       0.0k(  0.0)       0.0k(  0.0)       0.0k(  0.0)
 High(Sound) |       0.0k(  0.0)      95.2k( 12.0)     128.9k(  7.7)
Normal(RPAK) |       0.0k(  0.0)       0.0k(  0.0)       1.6k(  0.1)
Low(Texture) |   24704.0k( 97.0)   29369.6k( 63.3)    2937.0k(  6.3)
   COMPLETED |   24704.0k( 97.0)   29464.8k( 75.3)    3067.4k( 14.1)
1sec   latency mean   0.0ms, sd   0.0ms, max   0.0ms
10sec  latency mean   2.6ms, sd   9.2ms, max  63.0ms
100sec latency mean   1.4ms, sd   6.9ms, max  63.0ms
```

How Long Did it Take?

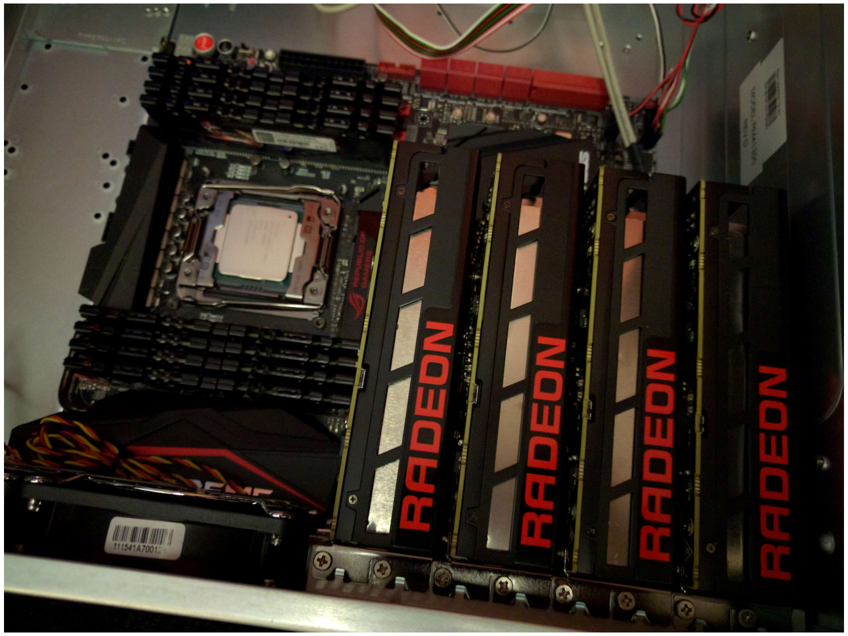
- Titanfall 2 in development roughly 2 years
- Streaming was one engineer, 10 months
  - 8 months solid, 2 months on-and-off
- Support – Related engine work
  - PC autodetect/minspec/driver work (Marton, Liu, Lambert)
  - Asset Bakery work, including patching (Hammon)
  - Console/Memory management work (Baker)

How Much RAM do we Need?

- Empirically, worked well with ~600MiB buffer, pretty hard to find fault ~1000MiB
- PC: 0, 375, 750, 1250, 5860MiB (“insane”)
- Console: 928MiB
- Plus permanent MIPs (~400MiB on typical level)
- Plus around 0.43MiB housekeeping overhead

How Much Data on Disk?

- Entire streamable set (starpak) is 21GiB
- ‘Effect and Cause’
  - 12GiB streamable (14011 MIPs,) 37MiB STBSP
- ‘mp_eden’
  - 13GiB streamable (14597 MIPs,) 130MiB STBSP
- 12.4GiB/1.4GiB ≈ a 9:1 ‘compression ratio’
  - Back-of-the-napkin average 1.6 more MIP levels



Precomputation Cruncher

4x AMD Radeon R9 Fury Nano

(Photo: Drew McCoy)

Precomputation Time

- Barely uses CPU
  - But heavy GPU use bogs down Windows UI!
- Cruncher (4x AMD Radeon R9 Fury Nano)
  - 15-109 minutes for each SP level, 54 minutes average
  - 10-36 minutes for MP, 18 minutes average
  - Very easy to divvy up a level’s probes to different GPUs
  - Got close to 4x speedup relative to single GPU

How Much Per-Frame CPU?

- 0.8 msec (on Console) typical per-frame on a busy scene
  - About half crediting BSP
  - Some additional time crediting models
  - About half generating add/drop list
- Opportunities:
  - Parallelization w/ jobs
  - Amortize across multiple frames

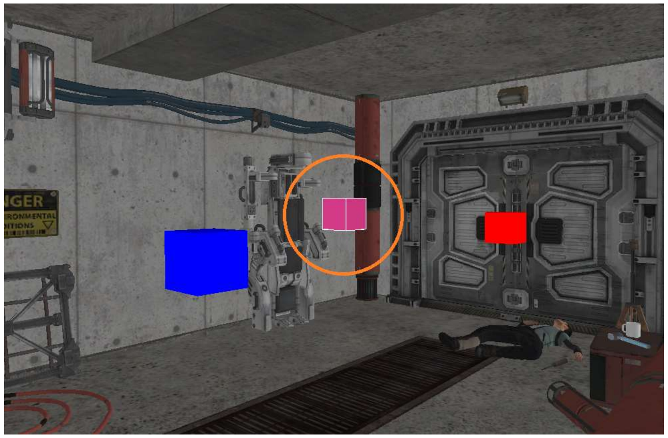
How did it Affect Artists?

- Disabled for Skybox and FX materials
- 4k textures available on all platforms (esp. PC)
- No wrong way to make models, or busywork
- Disappointed when gun/cockpit not 100%
  - Being inside models not handled well by algorithm
  - Special priority for ‘viewmodels’
  - Breaks ‘artists can’t affect budget’ requirement
- Need ‘all’ mode sometimes, but try not to live there

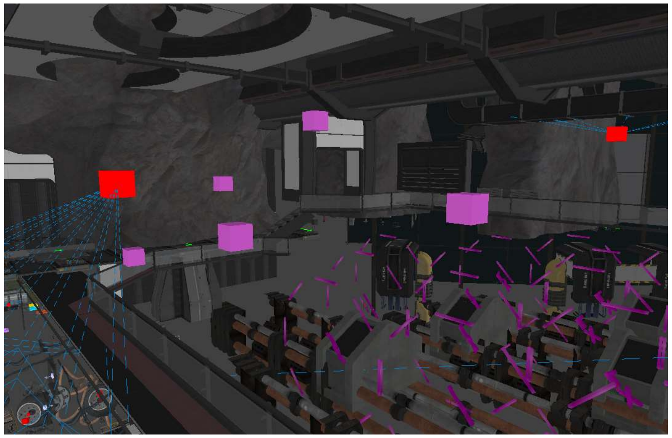
How did it Affect Designers?

- Only about 20 hints used in total; low impact
- Freed from manual texture optimization
  - (Searching for not-often-used textures)
- Out-of-date STBSPs worked a lot of the time
- Lots of customized skins/guns for MP
- Hacks to deal with swapping models (on vs off, etc)
  - Used hidden nearby models–should improve!

What are Hints Used for?



Default probe was inside wall.
Designer wanted to ensure the machine streamed in.



Probes were added for a model
with a number of small place-marker signs.

Production/QA Experience

- Cruncher machine using Jenkins
  - Perforce Update, do processing, commit stbsp.
- Many bugs due to out-of-date stbsp files
  - Cruncher stopped for various reasons
  - Levels had to be manually added to list
  - First step on bugs became ‘check Perforce dates’
- Console memory smaller in dev build
  - PC memory set to console size for QA

Some Gameplay Surprises

- ‘Effect and Cause’ - Teleportation between Z positions
  - Just worked due to columns, could handle models better
- ‘Ship to Ship’ – Massive moving geometry
  - Objects in precomputed world - move ‘virtual’ position
  - `SetStreamingRelativeEntity( ent id, base position )`
- MP menus need special handling
  - A ‘menu room’ for models, but world location locked
- Script shenanigans (pop & teleport) during fade in

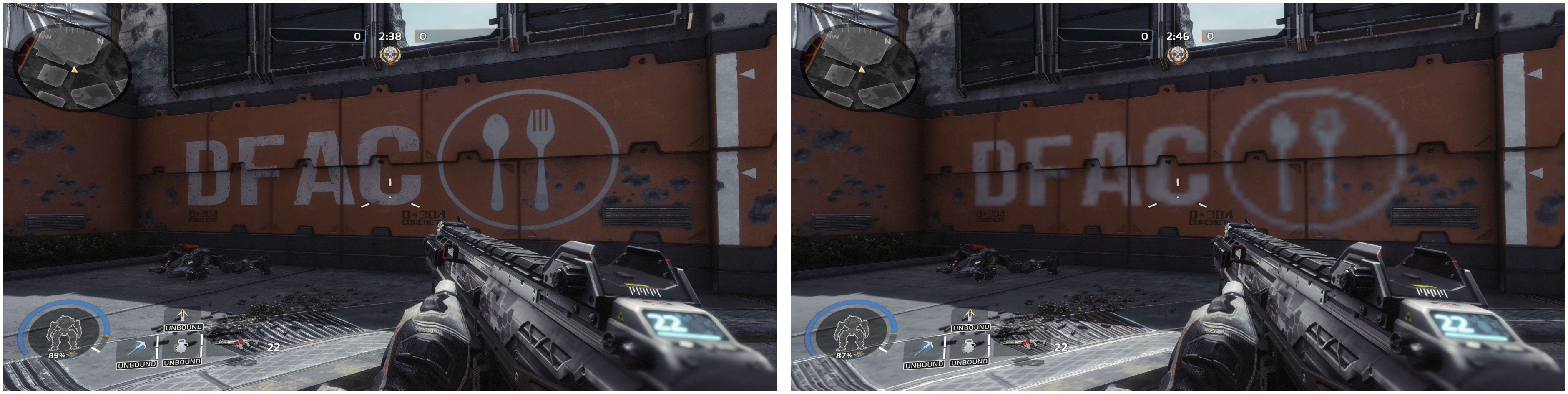
Some PC Surprises

- PCs Lowest buffer size setting was ZERO.
  - Didn’t really design for that!
  - Below minspec, but trying to be all-inclusive
- Hitches and overcommitting
  - Better API for memory detection
  - Moved texture creation to its own thread
  - Limit creation to 2 textures a frame
- Windows Async I/O often not asynchronous!

Some Console Surprises

- Play from Bluray on Console during install
  - Too much load, especially with audio
  - Could do drop-*n*-finest, but way too late; abandoned
- Needed buffer space for debug mem on consoles
  - Issues on console debug builds were often invalid
  - Imperfect RAM accounting, so empirically set limits
- Needed to borrow memory for load/movies
  - Lower memory limit, wait a few frames

Signage Challenges

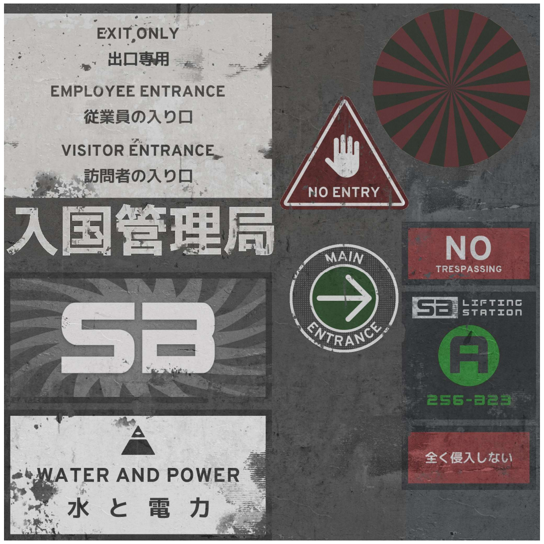


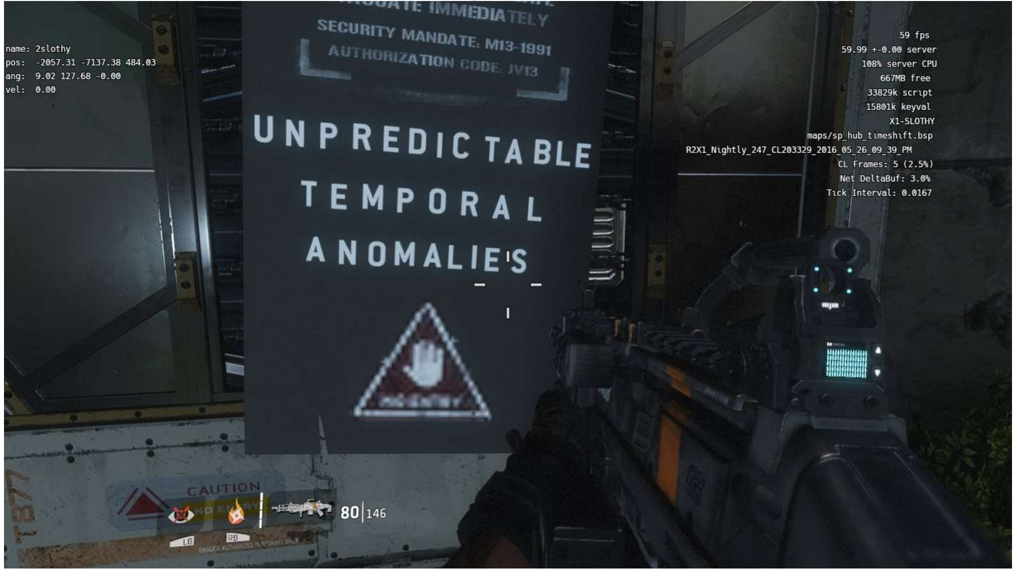
Cache RAM decreased on the right. The scene still looks reasonably good, *other* than the “DFAC” stencil!

Signage Challenges

- Signage (painted numbers, signs, etc.)
  - High contrast, worst-case
  - Built from pieces
    - Penalized heavily by cost (RAM) weighing
  - Could use distance fields, like we use for other UI?
  - Could use RMS error metric of MIPs to weigh metric?

More Signage Challenges





```
name: 2slothy
pos:  -2057.31 -7137.38 484.03
ang:  9.02 127.68 -0.00
vel:  0.00
```

```
                                      59 fps
                           59.99 +-0.00 server
                              108% server CPU
                                 667MB free
                               33829k script
                               15801k keyval
                                    X1-SLOTHY
                     maps/sp_hub_timeshift.bsp
R2X1_Nightly_247_CL203329_2016_05_26_09_39_PM
                        CL Frames: 5 (2.5%)
                        Net DeltaBuf: 3.0%
                      Tick Interval: 0.0167
```

Since the triangle on the right comes from a larger texture, it is disfavored by the metric.

Recurring Code Bugs

- Missed model drawing paths in engine
- Lifecycle of Textures/File Handles
  - Texture lifetime complicated by hotswap (live vs. backing)
- Tool (Editor and Model Previewer) parity
- NaNs
  - (Catch these early, exceptions ON!)

What’s next?

- Modern PRT APIs on PC
- Augment with some GPU Feedback?
- Is compression worth CPU cost and complexity?
- Generic moving geometry - multiple map ‘pieces’?
- Streaming effect textures, UI textures, geometry?
- Signage (Depth Fields?)

Thanks!

- Special thanks to Xin Liu, Earl Hammon Jr., Richard Baker, Steve Marton, and GDC mentor Julien Merceron.
- I'm chadbarb@gmail.com
- there's also jobs@respawn.com!

- Click to edit Master text styles
    - Second level
        - Third level
            - Fourth level
                - Fifth level