Texturing pitfalls in production

Fabrice NEYRET  May, 12th, 2015
1. Context

- Textures = key part of the realism/richness of the look
- Colors (i.e. shading parameters) + displacement + masks
- Hundreds x dozens x 4k² x multichannel
  → Huge human cost (amount of artist work)
  → Huge resource cost (storage / transfer / baking time / runtime memory)
1. **Context**

- **Textures** = key part of the realism/richness of the look
- **Colors** (i.e. shading parameters) + displacement + masks
- **Hundreds x dozens x 4k² x multichannels texture tiles**
  - → **Huge human cost** (amount of artist work)
  - → **Huge resource cost** (storage / transfer / baking time / runtime memory / rendering time)

- Approaches: Mapping vs triplanar vs procedural
- Plenty of filtering issues despite all MIP-mapped (displacement : Lean/Lead-R)
- No real workflow (paint → shader tree → bake)
1. Context

- Textures = key part of the realism/richness of the look
- Colors (i.e. shading parameters) + displacement + masks
- Hundreds x dozens x 4k² x multichannels texture tiles
  → Huge human cost (amount of artist work)
  → Huge resource cost (storage / transfer / baking time / runtime memory / rendering time)

- Approaches: Mapping vs triplanar vs procedural
- Plenty of filtering issues
despite all MIP-mapped (displacement : Lean/Lead-R)
- No real workflow (paint → shader tree → baking ; PRman workflow )
2. Mapping vs filtering issues

- interactive tools (Mari, Zbrush?): painting screenwise appearance
  - Multi-component footprint?
  - Which resolution / spatial spectrum in screen/texture/surface space?

- reconstruction issues: discont at atlas tile borders
  - Mag filter
  - Min filter:
    PB with displ filtering as roughness (Lean/Lead-R)

- Solutions:
  - Filter brush at painting. Accounting resol+jacobian all along workflow.
    PB: inside paint tool + no real workflow (pure image: no geom)
  - Painting paradigm: tangent space or screen space? (see artists workflows)
2. Mapping vs filtering issues

- interactive tools (Mari, Zbrush?): painting screenwise appearance
  - Multi-component footprint?
  - Which resolution / spatial spectrum in screen/texture/surface space?

- reconstruction issues: discont at atlas tile borders
  - Mag filter
  - Min filter:
    PB with displ filtering as roughness (Lean/Lead-R)

- Solutions:
  - Filter brush at painting. Accounting resol+jacobian all along workflow.
    PB: inside paint tool + no real workflow (pure image: no geom)
  - Painting paradigm: tangent space or screen space? (see artists workflows)
2. Mapping vs filtering issues

Filtering across atlas tile borders

The Reyes process-per-patch issue
- (The pb with displacement)
- Low filtering
- High filtering
  - and what about curvature?

Solutions:
- low: multi-footprint (data structure?) / Ptex format (but...)
- high: many-footprints / Mapping LOD?

PB: stick to old-school PRman workflow despite new usages and requirements!
3. Triplanar projection

- Solve all mapping issues (atlas tiles, mapping distortion…)
- Add many new issues:
  - repetition
    → pile-up layers (scale + look)
  - ghosting, overblending, varying contrast
    → more piling-up + masks (+adjust frame)
    → limit possible look

- May rely on mapping anyway:
  - intrinsic orientation (e.g., trees)
  - triplanar/procedural as shader tool vs editing tool
4. Procedural

- When editing/storing/loading are just too overwhelming (landscape)
- Or just editing (scales): editing tool vs shader tool

Issues:

- Which workflow for artists? (none → work blindly)
- Limited primitives / harsh PRman compatibility (100% forced locality)
- Displ: compatibility with other CG tools (collisions, flow, …)

→ little & specialized usage
  → more geometry + load + run-time memory

→ mix of procedural and maps/geom

Still, procedural placement is not procedural at rendering if any change (e.g., rocks)
(several notions of instancing)
  → more geometry and loading
5. Look-dev : controlling the overall look

- Several purposes: (NB: not only for textures)
  - Faithful interactive edition (filtering, shaders+combined…)
  - Preview
  - Look validation (client, directors, match real…)
  - Personal workflow (close view / far view, cycling pace)

- Crucial to preview using the final look
  (or artist will mess the data to see what he expect. e.g., displ)

- But too costly (filterings + shading + ambient occlusion + full rendering…)

→ Different tools:
  - Semi-realistic real-time rendering (with 1 or 2 focus: maps, material, light…)
  - “fast preview” (but “fast” might be quite slow)
  - Progressive rendering (good for lighting, bad for textures)
6. More filtering issues

Correlated and non-linear occlusion:
- Border of hidden parts (i.e. hiding through geometry occlusion)
- Colors vs orientation (color+displ, color+geom, displ+geom,...)
- Non-linear transforms (min-max, gamma, clamp...)

Very practical issues:
- Smooth derivatives (which filtering footprint paradigm ?)
- finite differences (displ normals)
6. More filtering issues

Far filtering: Curvature ? Silhouettes ?
→ deep filtering ⇒ rethink filtering integrating geometry

Filtering displacement map on silhouettes: how ?
→ alternate representation.
  Volume ? dedicated silhouette material ?
7. Memory footprint & Internal representation

- Globillu vs memory: Multiple views; can store only 1 version!
  = worst-case MIPmap → no prefiltering → even more rays
- Dewelded vertices, false instances, ½ deferred baking (params)
  → huge duplications, huge wastes of information

→ Rethink whole CG repr (high & low-level) / rendering
  in terms of memory occupancy (repr, compr, factor, param, lazy...)
→ Prod not at all convergent with gaming industry