Managing ultra-high complexity in real-time: some hints and ingredients

Fabrice NEYRET
CNRS / INRIA / Grenoble University, France
Pheno:

- Forests:
- Rivers:
- Ocean:
- Clouds:
- Smoke:
- Astro-imagery:
- Advected textures,
  Flow Noise:
- Bark, lava:
Representations:

- Textural world:
- Appearance filtering:
- SVO:

smart voxels:
Reproducing the **Natural Complexity**
Reproducing the Natural Complexity

ultra-detailed + ultra large
Reproducing the **Natural Complexity**

ultra-detailed + ultra large shape + animation + rendering
Reproducing the **Natural Complexity**

ultra-detailed + ultra large shape + animation + rendering

seamless + realistical
Reproducing **the Natural Complexity**

ultra-detailed + ultra large shape + animation + rendering realistical in real-time
Reproducing the **Natural Complexity**

ultra-detailed + ultra large
shape + animation + rendering
realistical
in real-time
controlable
Reproducing the Natural Complexity

ultra-detailed + ultra large
shape + animation + rendering
realistical
in real-time
controlable

NB: gaming more challenging than prod:
we don’t know where/what player will do
( and 1 / millionth of time budget )
So, what can we do? → some generalities

- Avoid wasting: often, realize el. useless after processed: v. frustum, hidden...
  → requires to structure data
So, what can we do? → some generalities

- **Avoid wasting**: often, realize el. useless after processed: v. frustum, hidden...
  → requires to structure data

- **Use available knowledge**:
  - a priori knowledge on content
  - assumptions and requirements on context
  - be consistent ( quality = worst parts, not best )
So, what can we do? → some generalities

- **Avoid wasting**: often, realize el. useless after processed: v. frustum, hidden...
  → requires to structure data

- **Use available knowledge**:  
  - a priori knowledge on content  
  - assumptions and requirements on context  
  - be consistent (quality = worst parts, not best)

- **Use relevant scale**: smallest image element = pixels
So, what can we do? → some generalities

- Avoid wasting: often, realize el. useless after processed: v. frustum, hidden...
  → requires to structure data

- Use available knowledge:
  - a priori knowledge on content
  - assumptions and requirements on context
  - be consistent (quality = worst parts, not best)

- Use relevant scale: smallest image element = pixels

→ **Minimalism**: do/store only what you need. ideally 1 sample / pix.
  → requires adapted representation
So, what can we do? → some generalities

- **Avoid wasting**: often, realize el. useless after processed: v. frustum, hidden...
  → requires to structure data

- **Use available knowledge**:
  - a priori knowledge on content
  - assumptions and requirements on context
  - be consistent (quality = worst parts, not best)

- **Use relevant scale**: smallest image element = pixels
  → **Minimalism**: do/store only what you need. ideally 1 sample / pix.
  → requires adapted representation
Minimalism: guides

- “Cost morality” : less info ⇒ should cost less, not more
  
e.g.: soft shadows, depth of field…
Minimalism: guides

- “Cost morality”: less info ⇒ should cost less, not more
  e.g.: soft shadows, depth of field…

- “What a painter would do”: 
Why MIPmap works so well* ? ( 1 sample / pix )

- backproj\(\text{pixel}\) on surface
- mapping : direct access to data
- texture pixel grid → direct access to neighborhood
- texture pixel grid → easy LOD hierarchy → precomputed filtering
→ General scheme for 1 sample / pix:

- differential cone tracing through 3D scene. (NB: DoF & soft shadow also cones. + tracing ray differentials).
- requires representing appearance at this scale
  & filtering detailed appearance down to this scale
Modelization / Choosing a representation

Content proxy + appearance model: some ingredients

- mesh, voxels, surfels, transp slices, textures, Zmaps, bump maps, NDF, flakes…
Modelization / Choosing a representation

Content proxy + appearance model: some ingredients

- mesh, voxels, surfels, transp slices, textures, Zmaps, bump maps, NDF, flakes…
- [Kajiya85] hierarchy of details: geom / bump texture / brdf, phase func
- ex: [KK89] “render. fur with 3D text.” : voxels + phase func + textural world
Volumetric Textures

“Rendering fur with three dimensional textures”
Kajiya & Kay [Sig’89]

- volume = impressionism illusion
- hierarchy of models [Kaj85]
  geom → texture → phase function
- mapping shapes onto shapes
  (shape as a 3D material)
Modelization / Choosing a representation

Content proxy + appearance model: some ingredients

- mesh, voxels, surfels, transp slices, textures, Zmaps, bump maps, NDF, flakes…
- [Kajiya85] hierarchy of details: geom / bump texture / brdf, phase func “anisotropic reflection models”
- ex: [KK89] “render. fur with 3D text.” : voxels + phase func + textural world
Content proxy + appearance model: some ingredients

- mesh, voxels, surfels, transp slices, textures, Zmaps, bump maps, NDF, flakes…
- [Kajiya85] hierarchy of details: geom / bump texture / brdf, phase func “anisotropic reflection models”
- ex: [KK89] “render. fur with 3D text”: voxels + phase func + textural world

We want dynamic LOD →

- hierarchical is important (closed repr.)
- seamless: transitions btw LOD (and repr.)
- filter appearance, not raw data
**PROLAND**, with Eric Bruneton

- whole Earth, all scales
- seamless, realistic
- animated

http://proland.inria.fr

<table>
<thead>
<tr>
<th>Image</th>
<th>Title</th>
<th>Authors</th>
<th>Journal</th>
<th>Page Numbers</th>
</tr>
</thead>
</table>
PROLAND, with Eric Bruneton

- whole Earth, all scales
- seamless, realistic
- animated

...only at useful pos & resol: quad-trees + out of core

http://proland.inria.fr

Real-time Realistic Rendering and Lighting of Forests
Bruneton Éric, Neyret Fabrice

Real-time Realistic Ocean Lighting using Seamless Transitions from Geometry to BRDF
Bruneton Éric, Neyret Fabrice, Holzschuch Nicolas

Scalable Real-Time Animation of Rivers
Yu Othm, Neyret Fabrice, Bruneton Éric,
Holzschuch Nicolas

Precomputed Atmospheric Scattering
Bruneton Éric, Neyret Fabrice

Real-time rendering and editing of vegetation and terrain
Bruneton Éric, Neyret Fabrice
Real-time all-scales ocean, with Eric Bruneton [ EG’10 / SCA’02 ]
simulate all waves …only at useful pos & resol

waves eqn: \( \Sigma \) trochoïds
+ oceanographic spectrum \( A(k) \)

\[
\begin{align*}
x - x_0 &= A e^{kz_0} \sin(\omega t - kx_0) \\
z - z_0 &= A e^{kz_0} \cos(\omega t - kx_0)
\end{align*}
\]
Real-time all-scales ocean, with Eric Bruneton [EG'10 / SCA'02]

Simulate all waves...only at useful pos & resol.

Waves eqn: $\Sigma$ trochoïds
+ oceanographic spectrum $A(k)$

\[
\begin{align*}
    x - x_0 &= A e^{kz_0} \sin(\omega t - kx_0) \\
    z - z_0 &= A e^{kz_0} \cos(\omega t - kx_0)
\end{align*}
\]

Appearance filtering: shape→<N>→BRDF

[Diagram of camera, screen space, and object space]
Endless forest, with Eric Bruneton [EG’12]

**in Proland** (⇒ all scales, real-time, seamless LOD)

realism: sun+sky, trees silverlining & transparency,
all-scales correlations (hot spot) + shadowing (ambient occlusion)
Endless forest, with Eric Bruneton [EG'12]

- several tree species
- Poisson-disk distribs
- gaussian params
- large scale: param maps

3 representations: near, mid, far

forest dyn texture tree / ground
forest dyn shader

individuals with parallax and detailed shading
Endless forest, with Eric Bruneton [EG’12]

- several tree species
- Poisson-disk distris
- gaussian params
- large scale: param maps

3 representations: near, mid, far

Near: ~Zmap IBR

Zv,Zl \to extinction

\|\n(\theta, \phi)
\|

light path

forest dyn texture

forest dyn shader

individuals with parallax and detailed shading
Endless forest, with Eric Bruneton [EG’12]

- several tree species
- Poisson-disk distribs
- gaussian params
- large scale: param maps

3 representations: near, mid, far

Near: ~Zmap IBR

Mid & far: masks*shaders (~“Fake Fur Rendering” [Sig97])

Zv,Zl → extinction

light path

individuals with parallax and detailed shading

k_g k_g k_t k_l

pdf

I(θ, φ)

G(θ_v, θ_i, φ, λ) D(λ)

T(θ_v, θ_i, φ, λ) E(θ, λ)
Deep a priori-knowledge allows deep LOD

What about more generic content?
Volumetric Textures

“Rendering fur with three dimensional textures”
Kajiya & Kay [ Sig’89 ]

- only for hairs
- not hierarchical / not filterable
- stochastic ray-tracing

→ static hierar. of details: not for dyn. LOD
→ PhD topic! :-)

Figure 16
Volumetric Textures: generic, LOD

**Volume:**
- multiscale (MIPmap)
- compress void (SVO)
→ octree of voxels

**Voxel data:**
- “generic” reflectance
- viewdep opacity

**impressionism illusion:**

(→ SGGX, GGX)
Volumetric Textures: generic, LOD  

**Volume:**
- multiscale (MIPmap)
- compress void (SVO)
→ octree of voxels

**Voxel data:**
- “generic” reflectance
- viewdep opacity

**Imagery:**
- NDF of ellipsoid
  (→ SGGX, GGX)

**Differential cone tracing (3D MIP-mapping)**
1 ray per pixel. Is prefilttering shape appearance!

Impressionism illusion:
Volumetric Textures: real-time (Z-buff):

Volume as textured transparent slices → now real-time!
(no phase function. Only RGBA filtering)

Tiled pattern on volumetric layer:

As free objects:

with A Meyer [98],
Ph Decaudin [04,09]
GigaVoxels, with Cyril Crassin [ I3D’09 ]

GPU cone-tracing full volumetric scene

http://gigavoxels.inria.fr
GigaVoxels, with Cyril Crassin [ I3D’09 ]

GPU cone-tracing full volumetric scene

Volume:
- multiscale
- compress void (SVO)
  → octree of bricks
  + out of core
- ( hardcoded gradient phase function )

→ "volumetric PROLAND"

Differential cone tracing 1 ray per pixel.
GigaVoxels, with Cyril Crassin [I3D’09]

GPU cone-tracing full volumetric scene

Volume:
- multiscale
- compress void (SVO)
  → octree of bricks
  + out of core

Differential cone tracing 1 ray per pixel.
GigaVoxels, with Cyril Crassin [ I3D’09 ]

GPU cone-tracing full volumetric scene

Volume:
- multiscale
- compress void (SVO)
→ octree of bricks
+ out of core

Volume:
- multiscale
- compress void (SVO)
→ octree of bricks
+ out of core

Differential cone tracing 1 ray per pixel.

DoF and soft shadow = blurry
⇒ faster!
GI-Voxels, with Cyril Crassin [CGF'11]

+ reflectance + multiple scattering
GI-Voxels, with Cyril Crassin [CGF’11]

+ reflectance + multiple scattering
GI-Voxels, with Cyril Crassin [CGF’11]

+ reflectance + multiple scattering

Voxel data: (smart voxels)
- viewdep opacity (6 dir)
- col, reflectance (lobes) & light (6 dir)
- no reflectance LOD
Smarter voxels: correlation, with G. Loubet [EG’18]

- silhouette issue

renderer: Mitsuba
Smarter voxels: correlation, with G. Loubet [EG’18]

- silhouette issue
- self-shadowing issue

( renderer: Mitsuba )
Mixing mesh and volumes

- Because source data is often mesh
- Animated meshes
- Because host render engine is often mesh based
- For efficiency (walls... )
  and precision (walls... )
Mixing mesh and Volumes, with C. Crassin, G. Loubet

- Z-buffer + GI-Voxel in fragment shader
  CC [ CGF’11 ]
- dynamic voxelization
Mixing mesh and Volumes, with C. Crassin, G. Loubet

- Z-buffer + GI-Voxel in fragment shader
  - dynamic voxelization

- geom LOD + appearance filtering
  - continuous transition (progressive aggregation):
    - mesh + brdf (GGX) + col
    - voxels + phase-func (SGGX) + col

( renderer: Mitsuba )
LOD → Filtering shape into shading:

- That’s it for volumes.

- What can we do for surfaces?

→ heightfield
Appearance filtering of heightfields

- Small scale relief + visibility
→ all is view-dep and light-dep!
Appearance filtering of heightfields

- Small scale relief + visibility
  → all is view-dep and light-dep!

- Correlations everywhere!
  → light and colors → + content correlation
  → normals → missing in all bumps
  → visibility → microfacet models
  → occlusion
  ...

with Eric Heitz
[HPG’12, I3D’13, SIGA’13]
Appearance filtering of heightfields

content correlation

ex: color-height, color-orientation

⇒ pixel integral not separable

( or : why MIPmapping is so wrong )

$$I = \int_P L_i(x, \omega_i) C(x) p(n_x, \omega_o, \omega_i) V_o(x) V_i(x) w_P(x) \, dx
\int_P V_o(x) w_P(x) \, dx$$

∫ color × masking × BRDF × shadowing

∞ × ∫ × ∫ ×
BTW, even `color(texture)` is an issue
→ also all `f(noise)`: LUT, clamp, abs...
since

\[
\text{average(LUT(text.))} \neq \text{LUT(average(text.))}
\]
Appearance filtering of heightfields

with Eric Heitz
[ HPG’12, I3D’13, SIGA’13 ]

BTW, even \textbf{color(texture)} is an issue

→ also all \texttt{f(noise)}: LUT, clamp, abs...

since

\begin{align*}
\text{average}(\text{LUT(text.})) & \neq \text{LUT(average(text.))}
\end{align*}

→ \textbf{Idea}: use stat distrib

1: \text{average} = < \text{LUT, histogram} >

\[
\overline{C_0} = \langle D_f, \rangle
\]
Appearance filtering of heightfields

BTW, even color(texture) is an issue
→ also all f(noise): LUT, clamp, abs...
since
average(LUT(text.)) # LUT(average(text.))

→ Idea: use stat distrib
1: average = < LUT, histogram >
2: histogram ~ gaussian
3: simply precompute iLUT(ν,σ)

NB: applies to any distrib e.g., heights ...
Appearance filtering of heightfields

heightfield content correlation

ex: color-height, pb: apparent heights distribution is view-dep and light-dep

\[ I = \frac{\int L_i(x, \omega_i) C(x) \rho(n_x, \omega_o, \omega_i) V_o(x) V_i(x) w_p(x) \, dx}{\int V_o(x) w_p(x) \, dx} \]

color-shape correlation ⇒ view+light correlation
Appearance filtering of heightfields

heightfield content correlation

pb: apparent heights distribution is view-dep and light-dep

color-shape correlation $\Rightarrow$ view+light correlation

4: effect = lobe offset $\rightarrow$ easy!

5: NB: for diffuse surface, effect of envmap = irradiance_map(N) $\rightarrow$ cf colormap(slope)
Appearance filtering of heightfields

heightfield $\rightarrow$ BRDF (microfacets)

$$I = \frac{\int_{\Omega} L_i(x, \omega_i) C(x) p(n_x, \omega_o, \omega_i) V_o(x) V_i(x) w_P(x) \, dx}{\int_{\Omega} V_o(x) w_P(x) \, dx}$$
Appearance filtering of heightfields

heightfield $\rightarrow$ BRDF (microfacets)

$$I = \frac{\int_P L_l(x, \omega_i) C(x) \rho(n_x, \omega_o, \omega_i) V_o(x) V_i(x) w_P(x) \, dx}{\int_P V_o(x) w_P(x) \, dx}$$

microfacets BRDF: Beckmann NDF $\leftarrow f (\text{slope variance})$
Appearance filtering of heightfields

heightfield → BRDF (microfacets)

\[
I = \int_{\Omega} L_i(x, \omega_i) C(x, p(n_x, \omega_o, \omega_i)) V_o(x) V_i(x) \ w_P(x) \ dx \\
\int_{\Omega} V_o(x) \ w_P(x) \ dx
\]

microfacets BRDF: Beckmann NDF ← f (full slope stat.)
→ tilted anisotropic lobe
Appearance filtering of heightfields

heightfield $\rightarrow$ BRDF (microfacets)

$$I = \int_P L_i(x, \omega_i) C(x, \theta(x, \omega_i) \cdot n(x)) V_o(x) V_i(x) w_P(x) \, dx$$

$$\int_P V_o(x) w_P(x) \, dx$$

LEAN maps $\rightarrow$ eval slope statistics

MIPmap = moments $\rightarrow$ cov matrix

$$h \xrightarrow{\text{LEAN maps}} \text{slopes} \xrightarrow{\text{cov matrix}} m_1, m_2, \Sigma$$

microfacets BRDF: Beckmann NDF $\leftarrow f(\text{full slope stat.})$

$\rightarrow$ tilted anisotropic lobe

point light + IBL
Appearance filtering of heightfields

heightfield → BRDF (microfacets)

\[ I = \int_{\mathcal{P}} L_1(x, \omega_t) C(x) \rho(n_x, \omega_o, \omega_t) V_o(x) V_i(x) w_P(x) \, dx \]
\[ + \int_{\mathcal{P}} V_o(x) w_P(x) \, dx \]

LEAN maps → eval slope statistics
MIPmap = moments → cov matrix

microfacets BRDF: Beckmann NDF
→ tilted anisotropic lobe
Managing ultra-high complexity:
so many other important stuffs...
Realistic clouds in real time, with Antoine Bouthors [ EGNP’06, I3D’08 ]
Realistic clouds in real time, with Antoine Bouthors [EGNP’06, I3D’08]

simulating all light paths: hard pb.
→ goal: real time!

reflectance: Mie :-s
absorption: 0
Realistic clouds in real time, with Antoine Bouthors [EGNP'06, I3D'08]

1: $\int$ Droplet Size Distrib → cancels Bessel oscillations
2: $n_{\text{scatter}} > 1$
  → - peak (50% $E) \approx$ no hit
  - high freq useless
  - no colored back-scatter

simulating all light paths: hard pb.

$\rightarrow$ goal: real time!

reflectance:
Mie :-s
absorption: 0
Realistic clouds in real time, with Antoine Bouthors [EGNP’06, I3D’08]

simulating all light paths: hard pb.
→ real time!

3: macro-material

(L,V,V_{pos},thick.)_{5D} → \text{collector}(pos,\sigma)_{3D}

~ ”Most Probable Path”
Realistic clouds in real time, with Antoine Bouthors [EGNP’06, I3D’08]

simulating all light paths: hard pb.
→ real time!

3: macro-material

\[ \psi(i, \sigma) \]

(L, V, V_{pos}, thick.)$_{5D}$ → collector(pos, \(\sigma\))$_{3D}$

3: separate scattering orders

→ shift Most Probable Path and scale of transport

10^7 simu

Fit
Realistic clouds in real time, with Antoine Bouthors [ EGNP’06, I3D’08 ]

simulating all light paths: hard pb.
→ real time!

3: macro-material (L,V,Vpos,thick.)\text{5D} → collector(pos,σ)\text{3D}

3:

separate scattering orders
→ shift Most Probable Path and scale of transport

4: solve i=collect(o) for cloud shape

5: ground ↔ cloud radiosity, sky illu
Animated details
Fluids as vortex filaments, with Alexis Angelidis [ SCA’05,06 ]
Fluids as vortex filaments, with Alexis Angelidis [ SCA’05,06 ]

- "soul" of fluid motion
- compact, highres, controlable…
- closer to std CG workflow
Fluids as vortex filaments, with Alexis Angelidis [ SCA’05,06 ]

- "soul" of fluid motion
- compact, highres, controlable…
- closer to std CG workflow

\[ w = \nabla \times v \]
\[ v = \iiint_x \frac{(p - x) \times w}{4\pi \|p - x\|^3} \, dx \]
\[ \frac{dw}{dt} = w \cdot \nabla v \]
\[ \Gamma = \int_L v \cdot dl = \iint_S \omega \cdot dS \]
Fluids as vortex filaments, with Alexis Angelidis [ SCA'05,06 ]

- "soul" of fluid motion
- compact, highres, controlable...
- closer to std CG workflow

\[ w = \nabla \times v \]

\[ v = \iiint \frac{(p - x) \times w}{4\pi\|p - x\|^3} \, dx \]

\[ \frac{dw}{dt} = w \cdot \nabla v \]

\[ \Gamma = \int_L v \cdot dl = \iint_S \omega \cdot dS \]

+ vortex noise
+ particles
= ellipsoid
Rivers & vector features, with NP, QY, EB, ... [ SCA’01, EG’09, ...]

grid ⇒ high res or aliasing
Rivers & vector features, with NP,QY,EB,... [SCA’01,EG’09,...]

grid ⇒ high res or aliasing

align mesh with features, or meshless

(images)
**Rivers & vector features**, with NP, QY, EB, ... [SCA’01, EG’09, ...]

- grid ⇒ high res or aliasing
- align mesh with features, or meshless
- analytic flow

(photos)
Fluids amplification: flow noise & advected texture

with K. Perlin, Q. Yu, ... [ SigSketch'01, SCA'03, Sig'07, TVCG'11 ]

1: given: coarse flow simu  
details: use texture

2: advect texture & preserve aspect

volume

height-field
Fluids amplification: flownoise & advected texture

with K. Perlin, Q. Yu, ... [ SigSketch’01, SCA’03, Sig’07, TVCG’11 ]

1: given: coarse flow simu
details: use texture

2: advect texture & preserve aspect

3: flownoise: swirling Perlin noise
**Fluids amplification: flownoise & advected texture**

with K.Perlin, Q.Yu, ... [SigSketch’01, SCA’03, Sig’07, TVCG’11]

1: given: coarse flow simu
details: use texture

3: flownoise: swirling Perlin noise

2: advect texture & preserve aspect

4: amplifying fluid: subscale turbulence

- Flownoise for sub-scales
  - rotations \(\equiv\) vorticity spectrum
  - Kolmogorov cascade
Life in texture space, with S. Lefebvre, … [ SIG’99, I3D’03, I3D’05, RR’06, GPU Gems2 ]

High res data on demand
low geometry cost

Simu in mix space
Galaxy Project, with RSA Cosmos & Paris-Meudon Observatory

- Real time walk-through
- ~ Hubble quality
- spectral

GigaVoxels++ & proceduralism
There is hope for ultra-high complexity in real-time :-}
Conclusion

There is hope for ultra-high complexity in real-time :-)  

Future:  
“converging” RT tools on GPU are cool, but don’t lose our soul:  
Gaming is more challenging than prod:  
\[ \text{ (1 / million}\text{th of time budget... )} \]  
→ we must keep being smart!
Conclusion

There is hope for ultra-high complexity in real-time :-)

Future:
“converging” RT tools on GPU are cool, but don’t lose our soul:
Gaming is more challenging than prod: (1 / millionth of time budget…)
→ we must keep being smart!

So:
- don’t get too picky with raw path tracing as light transport
- don’t get too picky with “no, it’s biased” argument
- keep our shader expressiveness
  (materials, proceduralism, alt. representations)

or LOD will be forbidden
  (then either perf or quality or complexity won’t be ok)
Conclusion

There is hope for ultra-high complexity in real-time :-)

Future:

“converging” RT tools on GPU are cool, but don’t lose our soul:
Gaming is more challenging than prod: (1 / millionth of time budget…)
→ we must keep being smart!

So:
- don’t get too picky with raw path tracing as light transport
- don’t get too picky with “no, it’s biased” argument
- keep our shader expressiveness
  ( materials, proceduralism, alt. representations )

or LOD will be forbidden
  ( then either perf or quality or complexity won’t be ok )

PS: all presented tools are usable in prod too ;-)
Questions ?
heap

( extra discussion material )
A few things I learned
Representations

Many tools on store!

raster (e.g., Photoshop) = grid
vs vector (e.g. Illustrator) = shape

grids: image texture. Voxels. Eulerian simu. BRDF table. SH.

- indeed, more continuous: amount of info:
  compressed data, base decomp., compr.sensing, fit, procedural, analytic
size matter:
- 4D table is cheap if interpolated low-res
- fitting or SH is not cheap if 798 coefs + transcend. math op

- opposed pro- and con-:
  - no universal one: choose the appropriate
→ can be mixed:
  - can change with scale or interaction length (local / long dist)
  - each box can use different one:
    shape, colors, shadowing & light transport, anim (space def)
Representations

Ones from Physics & maths:

Eulerian vs Lagrangian
Space vs Fourier
Velocity vs vorticity
Point-mechanics vs Finite elements / SPH
Color spaces

Point mechanics / statistic mechanics / fluids / waves / spectrums
energy lines

Photons / waves / rays / energy

( don’t forget validity domain & hypothesis )
Representations

Where to start:

- where is largest potential for improvement?
  ie, what worse part in the look / workflow?

- best improvement reachable for each bit of extra budget?
  think “differentials everywhere”: pixel=circle, occluder=slab, ray=spline.
  = 1st order Taylor approx
  \[ \text{better} = F(P) + PX \cdot \text{grad}(P), \ X \text{ in neighborhood.} \to \text{integrate}(f(Fb(P,X),X)) \]

- what constraints? preferences?
  time budget? storage budget? precision budget? hard or sloppy?

Have quality estim

→ faith → weighting, transition to backup to canonical approach

Reminder: quality = worst box, not best

so long “perfect equation” if no accurate parameter available

→ forgot nothing? Shannon-Nyquist ok? Large Numbers ok?
Differential everywhere!

= continuous integral everywhere

Points are not physical objects
differentials are. \( dS, dl, d\omega, \) cones… = local integral

differential domain \( \Rightarrow \) value=disturb.
\( \rightarrow \) Distributions everywhere!

Any scalar \( \rightarrow \) distribution (colors, mask …)

Any vector \( \rightarrow \) distribution (velocity, pos, …)

- minimal is a lot better than nothing
- can be cheap to have & store: Gaussian stddev, lobe width
- can be cheap to use

make well-posed many ill-posed problems

  e.g., aliasing and filtering issues

is a kind of LOD (subgrid model)


Reminder: **metrics = pixel color**

→ LOD is not “anything simpler”

**LOD ~ =** pre-integration over the pixel
   i.e., preparation of the colorfield pixel integral giving
   → compact magic atom renderable with 1 sample

Some LOD examples:

- **CG:** roughness, brdf, glossiness, surface.
  NDF, MIPmap, texture, impostors, particles.

Physics:

- *pseudoforces:* buoyancy, coupling, ....
- *pseudo objects:* rays & optic geometry, Surfaces & solids
- *emerging numbers:* Temp, Pressure..., even Velocity...
  (probably even space & time)
LOD everywhere!

LOD \approx \text{pre-integration over the pixel}

\text{i.e., preparation of the colorfield pixel integral giving}

Not so easy:

\begin{itemize}
  \item non-linearity \quad \rightarrow \quad \text{average}(f(x)) \textbf{is not} f(\text{average}(x)). \quad \text{same for interpolation}
  \item correlations, non-separability \quad \rightarrow \quad \int f g \textbf{ is not } \int f \int g
  \item a cascade of wrongness & clandestine hypothesis: \quad \text{MIPmanning}
\end{itemize}

\[ I = \int_{\mathcal{P}} L_i(x, \omega_i) C(x) \rho(n_x, \omega_x, \omega_i) V_o(x) V_i(x) w_P(x) \, dx \]

\[ \int_{\mathcal{P}} V_o(x) w_P(x) \, dx \]

\rightarrow \text{Reformulate:}

\begin{itemize}
  \item other physics or math handle
  \item distributions. \quad \text{Stat momentums.}
  \item reparameterize: \quad \log, \sqrt{x}, ^2, 1/x, \text{equivalent set (e.e., polar)}
  \item change space: \quad uv \rightarrow uvw, \quad \text{or no uv}
\end{itemize}
LOD everywhere!

**hierarchical:**
- scalewise divide and conquer
- don’t forget upstream and downstream:
  - frequencies in data? frequencies once rendered?

**different scales might be totally different problems:**
- different purpose (scenario)
- different perception (river-way / flow / details)
- different knowledge
  → different controls

→ Choose best representation
● **Philosophical key questions**
  - What is an LOD? (metrics: screen, pixels)
  - What is a volume? a surface?
  - What is a normal? a transparency?
  - What is a sample? a texture?

● **Sampled scales along graphics pipeline → aliasing & bias**
  maths (integration calculus, signal processing)

<table>
<thead>
<tr>
<th>geometry/material/brdf</th>
<th>texture</th>
<th>render</th>
<th>geom</th>
<th>anim</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample span</td>
<td>fetch</td>
<td>pixel/intersect</td>
<td>vertex/polygon</td>
<td>vertex/voxel</td>
</tr>
<tr>
<td>multiscaling subgrid</td>
<td>footprint/kernel</td>
<td>kernel(Srate,DoF)</td>
<td>surf(kernel)</td>
<td>subgrid, motion blur</td>
</tr>
<tr>
<td>interpolation mag,min</td>
<td>LODfetch(aniso)</td>
<td>fragment(Abuff)</td>
<td>mag,LOD</td>
<td>mag, more blur</td>
</tr>
<tr>
<td>aliasing/oversmoothing</td>
<td>mag,min</td>
<td>Moire,noise</td>
<td>peak,jaggies</td>
<td>flick, pop, backturn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>jaggies,noise</td>
<td></td>
<td>(col, spec, shadow)</td>
</tr>
<tr>
<td>poor: (beside aliasing)</td>
<td>color change</td>
<td>shading change</td>
<td>silh, small feat.</td>
<td>ghosting, polymove</td>
</tr>
<tr>
<td>filtering (pre-integration)</td>
<td>shading change</td>
<td>silh, small feat.</td>
<td>ghosting, polymove</td>
<td></td>
</tr>
<tr>
<td>‘filtering’ means:</td>
<td>shading change</td>
<td>silh, small feat.</td>
<td>ghosting, polymove</td>
<td></td>
</tr>
<tr>
<td>Shannon-Nyquist obeing: no/poor filter</td>
<td>shading change</td>
<td>silh, small feat.</td>
<td>ghosting, polymove</td>
<td></td>
</tr>
</tbody>
</table>

Shannon-Nyquist obeing: no/poor filter  
ap after filter  

‘filtering’ means: lod+aniso  
mutisampling  
sampling anything  
having screen hifreq  

filtered image:  
Shannon-Nyquist obeing: no/poor filter  
ap after filter  

‘filtering’ means: lod+aniso  
mutisampling  
sampling anything  
having screen hifreq
About “physical models” (in CG tongue)
« ‘physical approach’, ‘exact’, ‘rigorous’ »

- There is no such thing like «exact» in physics
- «Physical» ≠ local (equa-diff)
- Local eqn vs macroscopic, «rigorous vs empirical»: subjective!
  - mecaQ → molecules → stat phys → thermodyn → NS → hydraulics/waves/atmo(oceano)sc
  - mecaQ → EM field → Huygens → geom optic → RT/radios/visibility
- Hypothesis, conditions, limits of validity
  ex, continuous fluids: notion of P,T, V, parcel (emergence)
- Border conditions, parameters
  one half of the problem is not or poorly known!
- continuous eqn → numerical engineering: resol issues
  - subgrid models: on-going research
  - sub-res → errors qualitatives and quantitatives [SAA00]
- Tool, inspiration. But don’t sacralize. Context is important!
What does users want?

- **Graphist:**
  - Super-spectator
  - Scenario

- **Expressive tools:** not black box!
  - Usable
  - Controlable
  - Intuitive & predictables parameters
  - Generative space rich / useful enough
  - Feedback (→ fast is useful even for SFX)
  - For on scene, on shot.
    → All tools are on shell + full manual
Studying real world

Physics eqn vs the real Nature

– Structured vs ‘blurry’, known vs dirt & fluctuations
  Artificial symmetries, regularities, rigidities change the phenomenon (buckling, natural convection, silhouette BRDF)
– Clandestine hypothesis (Evil!)
– LC: borders, such a mysterious thing!
  (meso-shape, param value) e.g. “river bed”, “bark”
– Useless details vs uncontrolled emerging phenomena
– Simu: result change with resol [PDI-LF02]

A. Fournier: start from real images, end with real images
  (inspiration, validation)

– Observe. picture. film. touch. draw. Repeat.
– Learn how to see. Find the ‘meaning’ (the ‘structure’. of things & eye)
– Pb of subjective validation
Reproducing the **Natural Complexity**

Quality real-time rendering / animation is **sometime** reachable

- Choose the right representation
- Be smart rather than brute force
- Don’t get blinded by what you know
  → look through the window, Nature is right there ! :-)
Alternate representations

- **Scales:** (≠ meaning, perception, goal, data, simu)
  → coupling different models
- **Formes, surfaces:** subjectives notions!
- **How to representer the world?**
  - *What we know / what we see* (shape, relief…)
  - *Minimalist, impressionnist* approaches separate shape/relief, normals, shading
    Adaptive: hierarchy of models [Kaj85]
  - *Repr. of shapes*: meshes, surfels, voxels…
    Properties ≠: structuration, cost, filtering…
  - *Decoupling* (geom / texture space / light space / …)
Phenomenological simulation

• Large & detailed: physical simu out of reach. + [PDI-LF02]

• Some a priori knowledge usually exists!
  – values ranges, modes, dominant pheno…
  – at least: what the purpose is, what the scene is

• Emerging effects: instabil., waves, folds, equilibrium…
  – Equations: indirect, phys++. While predictable
  – Closer too meaning, macroscopic, intuition, user language

• Direct repr of emerging phenomenons

Macroscopic phys (phenomenological / empirical / analytical)
  • Available models / analytical / direct obs. / obs. ref simu

Macroscopic primitive
  – XVIIIth - XXth treasures
  – revisit, make yours, invent, generalize…
  – uneasy, sparsely explored…but results might pay.
Settling a problem

• **Purpose**
  (what are we aiming at ? why ?)
  goal: finalist (appli) vs constructive (fondam. tools)

• **Formalize data/knowledge**

• **Formalize hypothesis** (reasonned),

• **Goals** (list of requirements),

• **Criteria**

• **Proposal**
  – What already exist ? what to draw on, what’s inadapted and why ?
  – Your way (explicit and justified choices)
    goals →sub-goals →details (c/ code review!)
  – Validation, + & -, perfs, limitations, comparaisons
Texture filtering (interp & MIP-map)

• Clandestines hypothesis:
  – Linearity 1: N, courb., visibility, shadows, const params.
  – Linearity 2: fragment = \text{lin}(texture) \ , \ i.e.: \text{text} = \text{RGBA}
  – Continuity: neglect borders, holes, atlases, tiling
Texture filtering (interp &

• Clandestines hypothesis:
  – Linearity 1: N, courb., visibility, shadows, const params.
    ➢ pb: micro-geometry ! Ultimate filtering!
  – Linearity 2: fragment = lin(texture), i.e.: text = RGBA
    ➢ pb: textures for anything (Z,N,…)! 
  – Continuity: neglect borders, holes, atlases, tiling
    ➢ pb: indirections!

• Geometry filtering:
  – Polygons not antialiased
  – Get smaller and smaller
  – Not pre-filterable
  → repr alt, model transition [Kaj85]