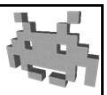
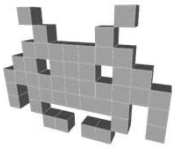
Course Plan



lec. 1: Introduction
lec. 2: Mathematics for 3D Games
lec. 3: Scene Graph
lec. 4: Game 3D Physics +
lec. 5: Game Particle Systems
lec. 6: Game 3D Models
lec. 7: Game Textures
lec. 8: Game 3D Animations
lec. 9: Game 3D Audio
lec. 10: Networking for 3D Games
lec. 11: Artificial Intelligence for 3D Games
lec. 12: Game 3D Rendering Techniques

22

How is a texture mapped over a given mesh?



23

How is a texture mapped over a given mesh?

- 3D Models
  i.e. tri-meshes with:
  - per vertex attrib
    - normals, color, AO, ...
  - LODs
  - uv-mapping
  - keyframes
    - cyclic animations
    - face-morphs, ...
  - "skinning"
- Materials
  - lighting model stats / flags
  - textures
    - RGB maps
    - normal maps
    - alpha maps ...
  - shaders
    - vertex, fragments, ...
- Animations
  - blend shapes
  - skeletal animations
  - kinematic animations
  - geometry caches
  - skeletons (rigs)
- Geometric proxies
  - hit-boxes
  - bounding objects
  - AI-meshes
- Particle systems
- Environments
  - 3d scenes
  - skydomes
  - env. maps
  - scene props

24

UV-Mapping of a mesh

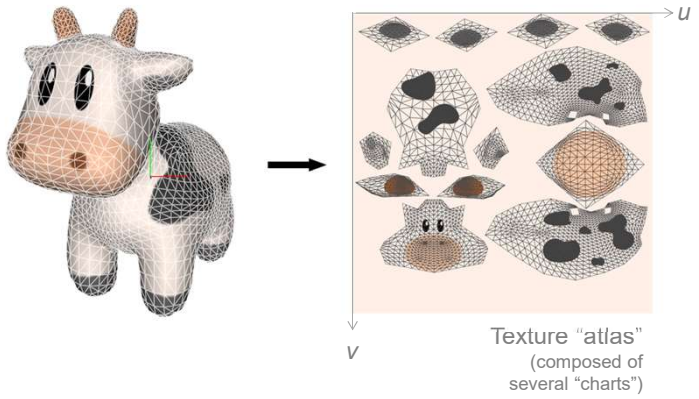
- A mapping : mesh surface → 2D texture space is needed
  - «parametrization» of the surface

$[0..1]^2$

- Store this mapping as per vertex attribute : (u,v)
  - The «u-v mapping» of the mesh

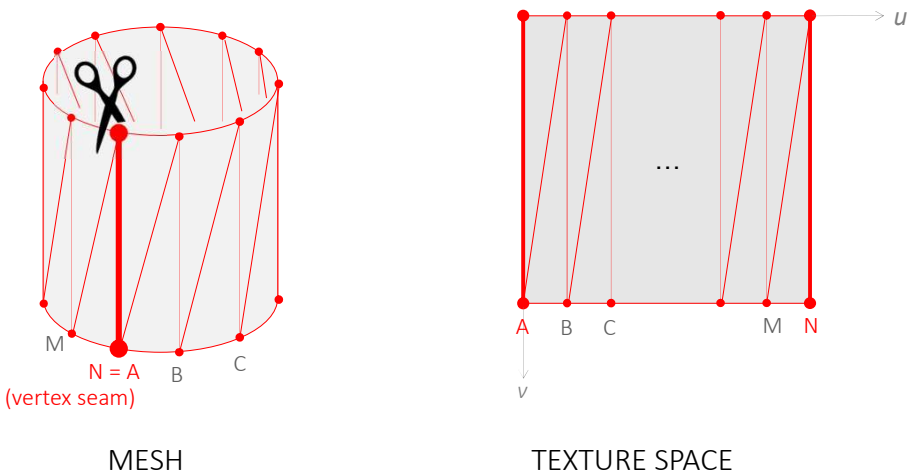
25

Modeling task: "u-v mapping" construction



26

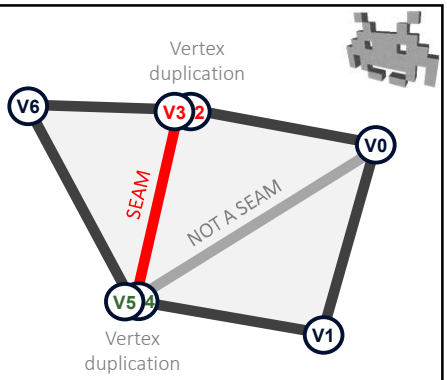
UV mapping: example



27

Texture seams (or cut)

- A vertex seam necessary to encode the UV-map



	X	Y	Z	U	V	...
V0	p_x0	p_y0	p_z0	$u0$	$v0$	...
V1	p_x1	p_y1	p_z1	$u1$	$v1$	...
V2	p_x2	p_y2	p_z2	$u2$	$v2$	...
V3	p_x2	p_y2	p_z2	$u3$	$v3$	...
V4	p_x3	p_y3	p_z3	$u4$	$v4$	...
V5	p_x3	p_y3	p_z3	$u5$	$v5$	...
V6	p_x4	p_y4	p_z4	$u6$	$v6$	...

GEOMETRY + ATTRIBUTES

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T0	0	1	4
T1	4	2	0
T2	5	3	6

CONNECTIVITY

29

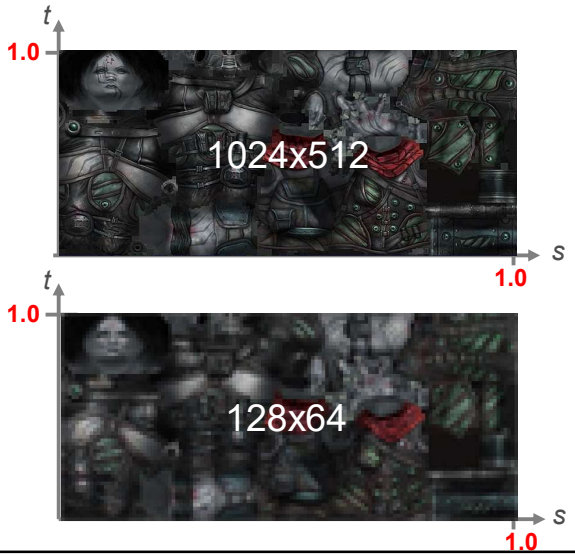
Texture space notation

Texture Space (or "parametric space" or "u-v space")

Texture Space = [0,1] x [0,1]

30

Note: Texture space independent from texture resolution (or aspect ratio)



Convenient!
We can reduce texture sheets res (balancing quality / memory) without affecting the mesh 'UV mapping.

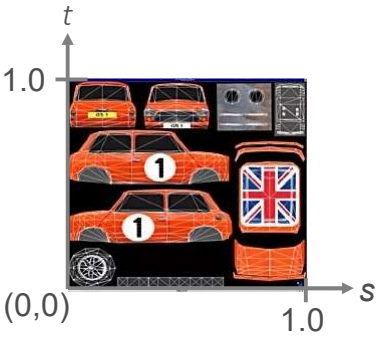
Eg: load in GPU RAM only
a few smaller MIP-map levels

31

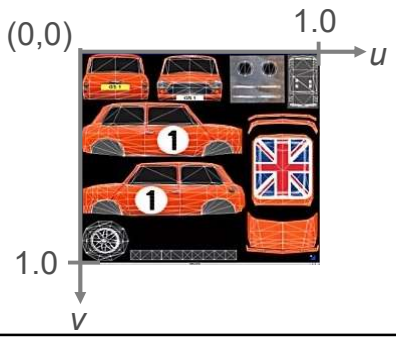
Two notations

Most used
(in game industry)

s-t
(es OpenGL)



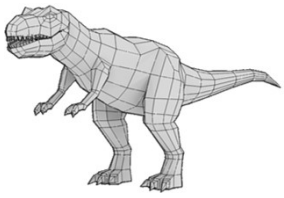
u-v
(es DirectX)



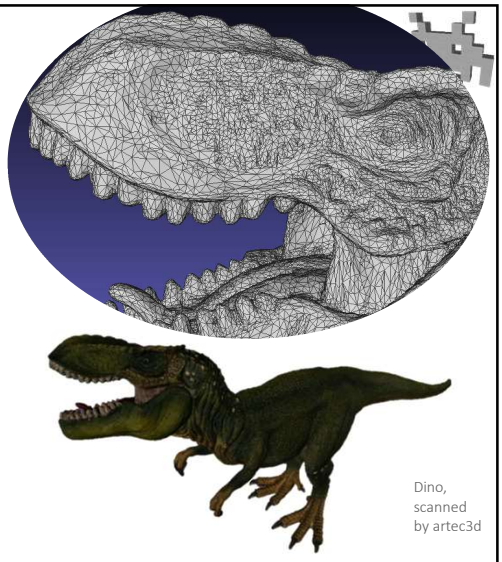
32

3D models sources: comparison

PERFECT for games!
(much easier to: animate, re-edit, uvmap, ...)



vs



Dino, scanned by artec3d

manually edited
low-poly mesh
(2K triangles)

scanned & cleaned
hi res mes
(30K triangles)

(sculpted meshes are similar)

33

Construction of a UV-map for a mesh (or, UV-mapping of a mesh)

- Typical task of the modeler (digital artists)
  - (semi-)automatic algorithms very studied
- We need to find a spot in the (2D) texture space for each (3D) mesh triangle
- Similar to to:
  - Peel an apple (cutting part)
  - Lay each produced peel in 2D (unfolding part)
  - Pack the peels inside a rectangular space (packing part)
- Cuts (or “texture seams”) are (almost) always required!
  - they are discontinuity of u,v attributes
  - stored in the mesh as vertex-seams (vertex duplications)

34

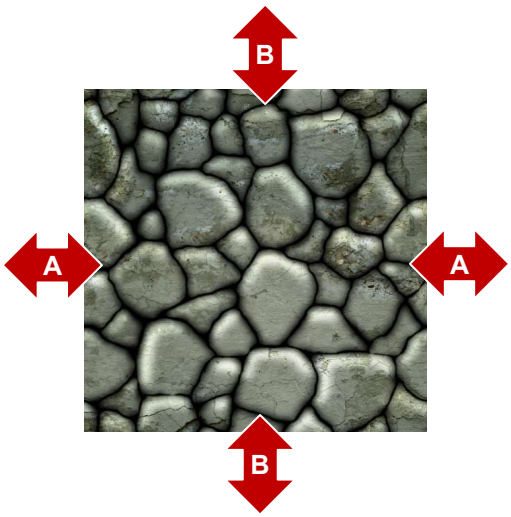
Modeling task: "u-v mapping"

- Strategies:
  - 1. select of the cutting edge
    ...or...
    1. assign faces to texture "charts"
    - either way, decide where "texture seams" are
  - 2. unfolding
    - minimizing "distortion" (by automatic algorithms)
  - 3. charts packing (again, often automatized)
    - Minimize the empty space in textures
    - Assign areas according to necessities
      (important parts → bigger texture space)
      (sampling of the texels becomes adaptive!)

DEMO!

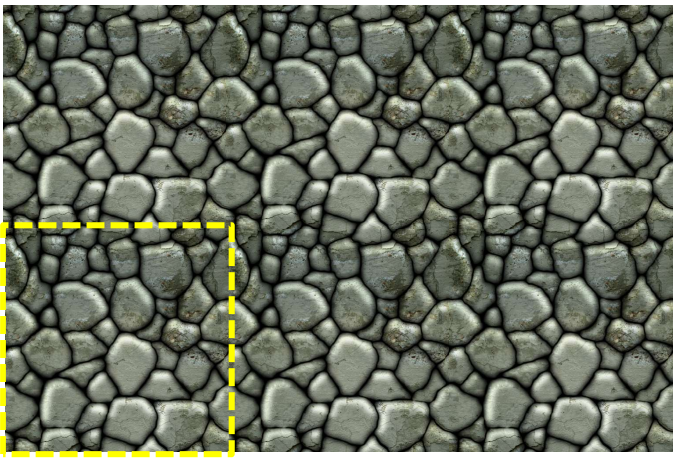
35

Tileable Textures



36

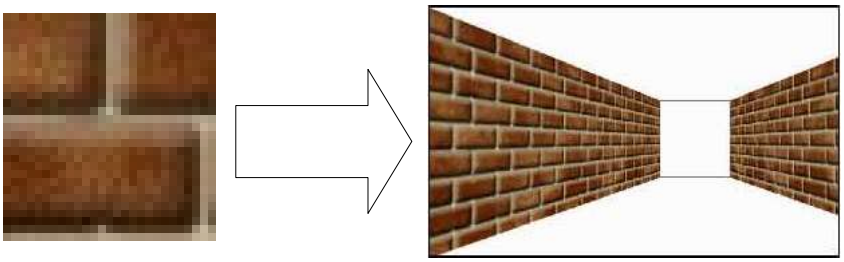
Tileable textures



38

Tileable textures

- Typical use



Very efficient in space

39

Two types of UV-maps

aka: "UV-map" (the standard)

- NOT injective UV mapping
  - Different zones of the mesh mapped to the same texture region
  - e.g.: with overlapping charts
  - ☺ Optimization of texture RAM
    - Can exploit of simmetries / repetitions
- Injective UV mapping
  - 1 (non empty) point on the texture = 1 point on the mesh
  - non-overlapping charts!
  - ☺ Generality / Flexibility
    - Used for several scopes (e.g. light baking)
- Different objectives
  - often, both are present: 2 distinct UV maps
  - 2 distinct UV attributes for each vertex

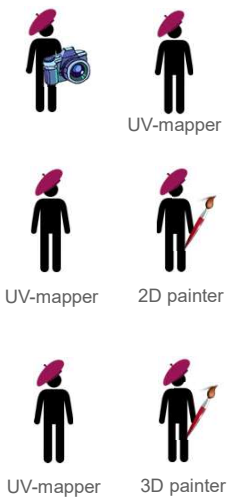
aka: "Unwrapping"
or: "Unwrapped UVs"
or: "1:1 UV-map"
or: "Lightmap" UV-map
or: "Non-overlapping" UV-map

Which is the type of the UV-maps shown in prev slides?

40

RGB maps: How are they obtained?

- Image *first, then* UV-mapping
  - e.g. images from photos
  - e.g. tileable images
- UV-mapping *first, then* paint 2D
  - paint with 2D app (e.g. photoshop)
- UV-mapping *first, then* paint 3D
  - paint within 3D modelling software,
  - *or:* 1. export 2D rendering,
    2. paint over with e.g. photoshop,
    3. reimport images
    4. goto 1



42

RGB maps: How are they obtained?

...or:

- *first* paint 3D
  - on hi-res model,
  - "paint" on vertex attributes
  - e.g. with Z brush...
- *then* coarsen
  - build / autobuild final low-poly version
- *then* UV-map
  - the low-poly model
  - must be a 1:1 mapping!
- *then* auto-texture
  - auto build texture

more about this later...

43

Cutout textures example
Texels = transparency level (0 or 1)

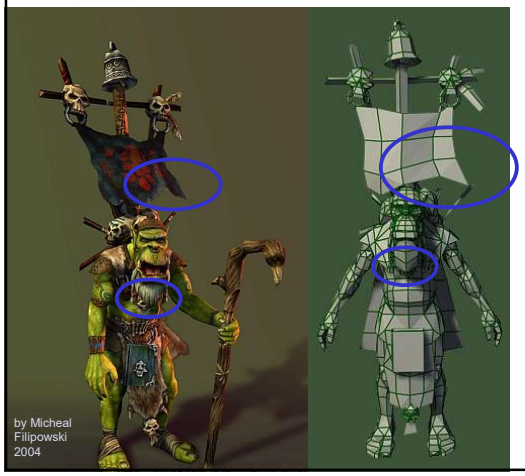
Alpha map

RGB map

44

Cutout textures
Texels = transparency level (0 or 1)

- e.g.: drapes, beard...

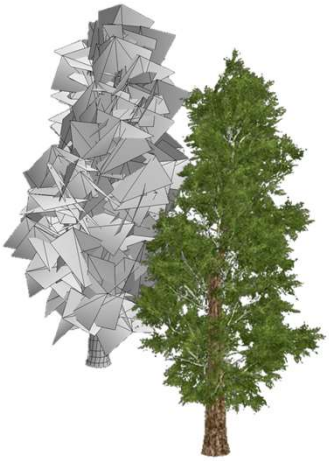


texture

45

Cutout textures
Texels = transparency level (0 or 1)

- e.g.: trees, foliage

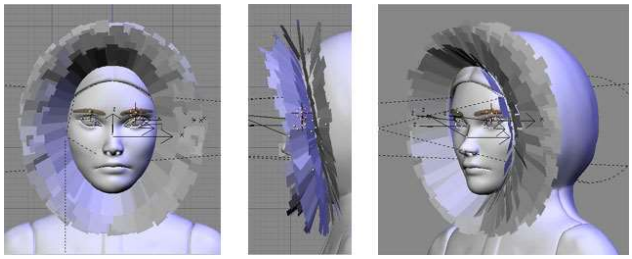


46

Texture mapping and Alpha Test

- Eg: fur, fur coats

(repeated) texture



47

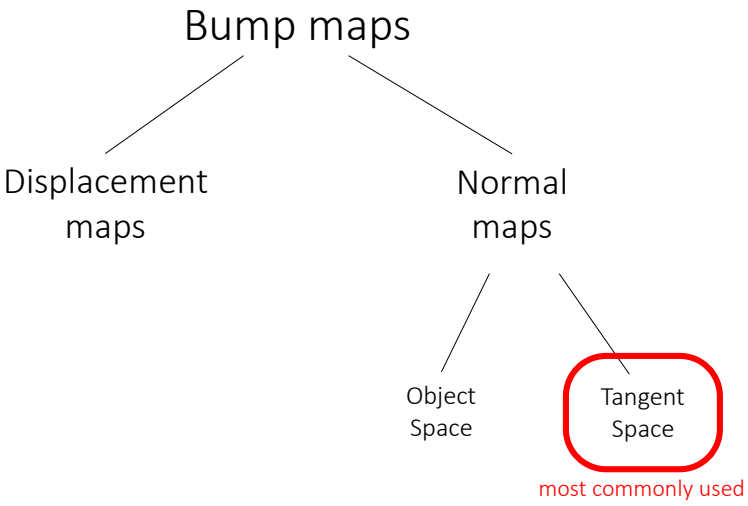
Bump-Map (*)

- a texture modelling (or, providing an illusion of) shape details (i.e., high-frequency geometric features)
  - details not modeled by the "real" geometry (the mesh)
  - remember: meshes tend to be low-poly
    - not much detail in them
  - approach also known as "Texture-for-Geometry"
  - rationale: texels are cheaper to render/store than vertices!
  - geometric details may extrude out or be engraved in the "real" (mesh) surface
  - in many cases: the detail affects lighting only
    - sufficient to trick the eye
    - especially true with dynamic lighting

(*) This terminology not universal: «Bump-map» can mean just «displacement map»

48

Types of Bump maps

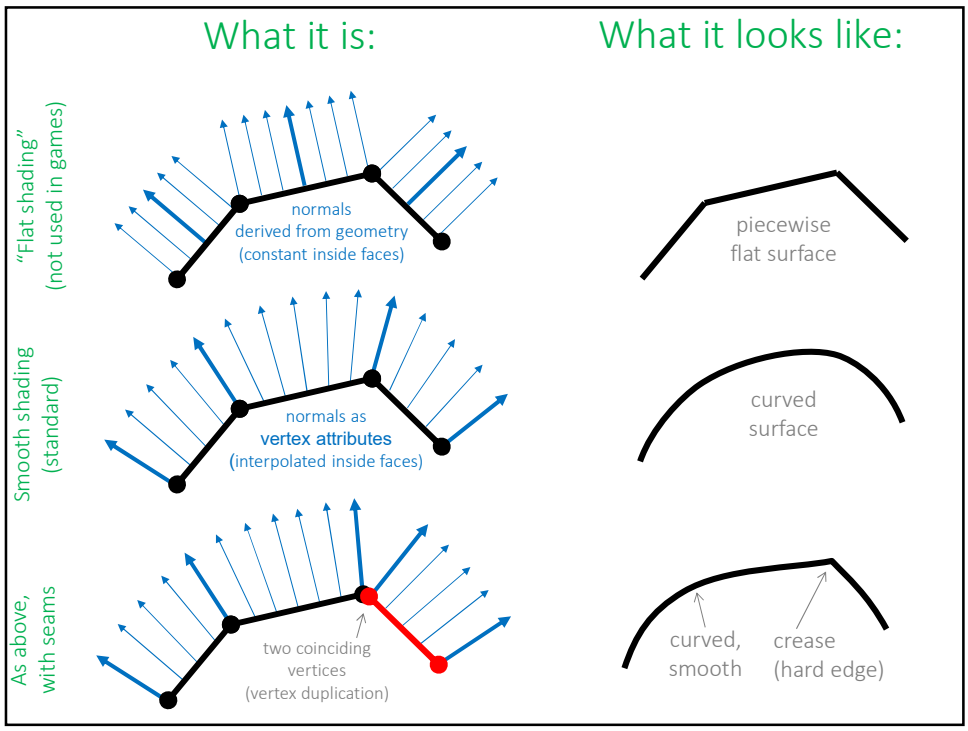


49

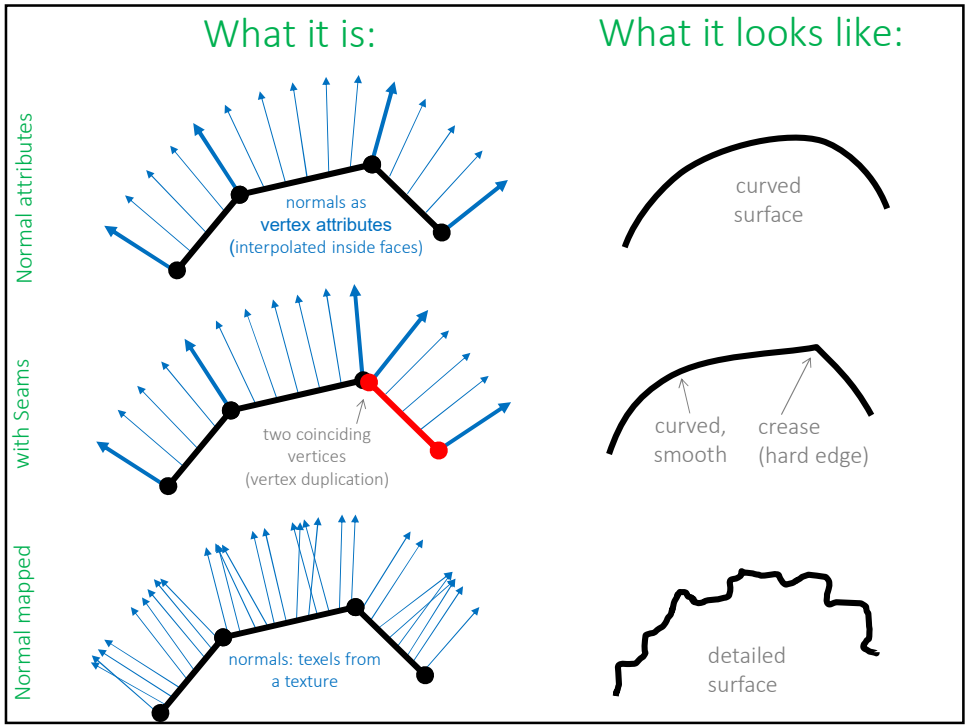
Types of Bump maps

- Bump map:
  - A texture encoding hi-frequency details
- Displacement Map:
  - Details are encoded by storing differences between mesh geometry and detailed surface:
  - as scalars (distance along the normal), or as vectors
  - used for: on-the-fly re-tessellation, and *parallax mapping* technique
- Normal Map:
  - Details are encoded by storing the normals of the detailed surface
  - used for: affecting the lighting
  - In which frame?
    - In Object Space: (Only for 1:1 UV-mapping)
    - In Tangent Space: (TBN space)
      - Usable on more surfaces independently from the orientation
      - Requires Tangent-Bitangent direction and normals on surface

51


What it is:
What it looks like:
"Flat shading"
(not used in games)
normals
derived from geometry
(constant inside faces)
piecewise
flat surface
Smooth shading
(standard)
normals as
vertex attributes
(interpolated inside faces)
curved
surface
As above,
with seams
two coinciding
vertices
(vertex duplication)
curved,
smooth
crease
(hard edge)

52


What it is:
What it looks like:
Normal attributes
normals as
vertex attributes
(interpolated inside faces)
curved
surface
with Seams
two coinciding
vertices
(vertex duplication)
curved,
smooth
crease
(hard edge)
Normal mapped
normals: texels from
a texture
detailed
surface

53

Bump-Map: from the modeler perspective

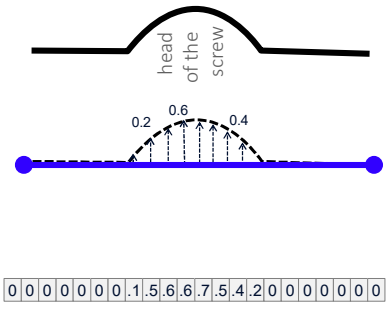
- macro-structure of the object → low-poly mesh
  - e.g.: the general shape of the horse
  - e.g.: the general shape of the face
  - e.g.: the general shape of the dragon
- meso-structure of the object → bump-map
  - e.g.: the musculature of the horse
  - e.g.: the wrinkles of the face
  - e.g.: the flakes of the dragon
- micro-structure of the object → material parameters
  - e.g.: the velvet-like fur of the horse
  - e.g.: the structure of the dermis / sebum
  - e.g.: the micro roughness / smoothnes of the flakes

54

Displacement map : concept

Stores the distance of the detailed surfaces from the plain geometry

- example: a bump-map for a screw-head  :



Detailed surfaces
(which I would like to represet)

low-poly mesh
(my approximation) (here: flat ☹)

displacement map
(scalars)

56

Displacement map: notes

- Each texel stores: a distance of the detailed surface
  - Along the normal direction (of low-poly mesh)
  - 1 scalar per texel –> 1 channel texture
- Which way:
  - outwards (*extrusions*)
  - inwards (*excavations*)
  - or both (signed displacements)
- Storage:
  - gray-scale image (1 scalar per pixel)
  - remap values within the interval [0..1]
  - global scale factor (on the fly)
- Possible uses:
  - Direct lighting of implied normals: "embossing" effect (old effect: it's a bad approximation, not common anymore)
  - Global illumination (ambient occlusion) See later
  - «Parallax mapping» technqie See later
  - Intermediate data for the construction of a normal map See later

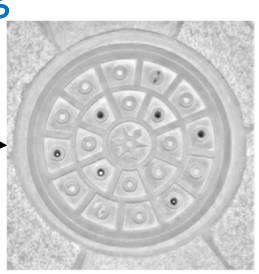
white = outwards
black = flat

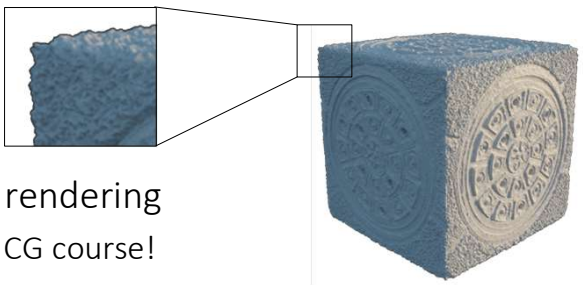
Easy to paint and manipulate!

57

(scalar) Displacement map: Rendering – parallax mapping

- Technique used render a mesh with a Displacement Map
  - Bonus: the silhouette of the object can be affected



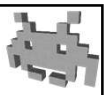


- See lecture on rendering
  - And Real time CG course!

Image courtesy of https://cgcookie.com/articles/normal-vs-displacement-mapping-why-games-use-norm

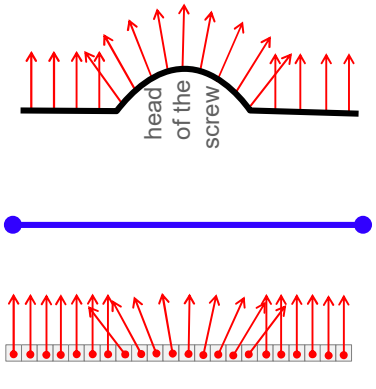
60

Normal Map: concept



Store the Normals of the detailed surfaces

- example -- a normal-map for a screw-head 



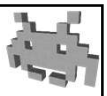
Detailed surface
(I would like to model)

low-poly mesh
(approximation of ^) (here: flat ☹)

normal map
(one normal per texel)

61

Normal Map: notes

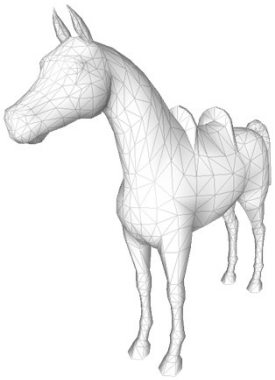


- Affects the lighting only
  - not the parallax
  - not the shape of the object
  - The lighting reflects the hi-freq detail of the object
    - dynamically (with variable lights!)
  - Total illusion: very convenient
    - If we are not trying to model a macro-structure
- In rendering: use the normal from the texture
  - (for lighting)
  - Instead of the interpolated per vertex normal
- Normals are expressed in cartesian coord
  - Often
    - But not always (∃ better ways to express unit vectors!)
  - Question: ok, but in which space??? *more later*

62

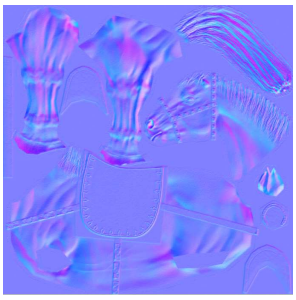
Normal-Mapping

see demo!



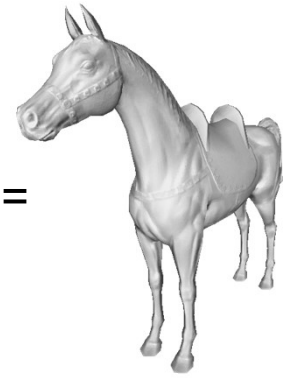
Low-poly mesh
(uv-mapped!)

+



Bump-map
(here: a
tangent space normal map)

=



lots of *cheap*
geometric detail
(apparently)

assets courtesy of "Mount&Blade" (Talesworlds)

63

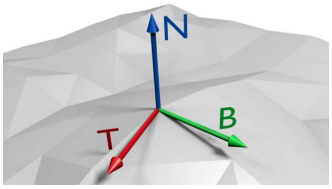
Normal Maps: in which space are the normals encoded?

- Object space: Object-Space Normal-Maps
  (The same in which I express the vertex pos)
  - ☺ the per-vertex normal becomes unnecessary!
    - The normal from texture substitute it
  - ☺ Trivial to apply (during rendering)
    - just use the normal fetched from the texture for lighting
  - ☹ normal-map is bound to a specific object
    - cannot be reused for different objects
  - ☹ Each region of the normal map is bounded to one specific area region of the object!
    - Injective UV-maps only!
    - e.g. no tiling, no exploitation of simmetries

65

Tangent space (aka TBN space)

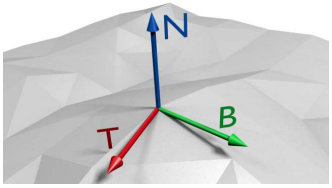
- A vector space defined ∀ point of the surface:
  - Z axis: Normal
    - orthogonal to surface
  - X and Y axis: tangent vectors
    - parallel to the surface
    - X = Tangent
    - Y = "Bi-Tangent"
      (sometimes, but inappropriately: *Bi-Normal)



66

Tangent space (aka TBN space)

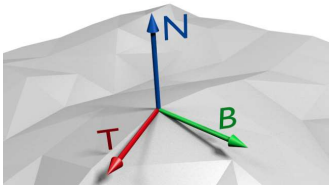
- How to store them?
  - As 3 versors stored as (per-vertex) **attributes**
    - So, they are interpolated inside faces (like any other attribute)
  - Optimizations are possible!
    - Not necessarily stored as 3 vectors (9 scalars)
    - E.g.: instead of storing B, we store N and T, then B = N × T
  - Note: they have discontinuities
    → seams (vertex duplications) are necessary
    - In first approximation, the same ones required by the UV-map (but non only! why?)



67

Tangent space (aka TBN space)

- How to compute them?
  - Normal
    - as usual (see lecture on mesh)
  - Tangent & Bi-Tangent
    - determined by the UV-map!
    - T = gradient of U coordinate
    - B = gradient of V coordinate



- details:
  - All three are defined and constant inside faces, then averaged at vertices (see per-vertex normal computation)
  - T,B,N can be *only approximatively* orthogonal to each other
  - T,B,N reference frame can be left-handed or right-handed (even different "handedness" in different parts of the same mesh)

68

Normal Maps: in which space are the normals encoded?

- Tangent space: Tangent Space Normal-Maps (the standard «bump-map», in games)
  - ☹ extra attributes are now needed per vertex:

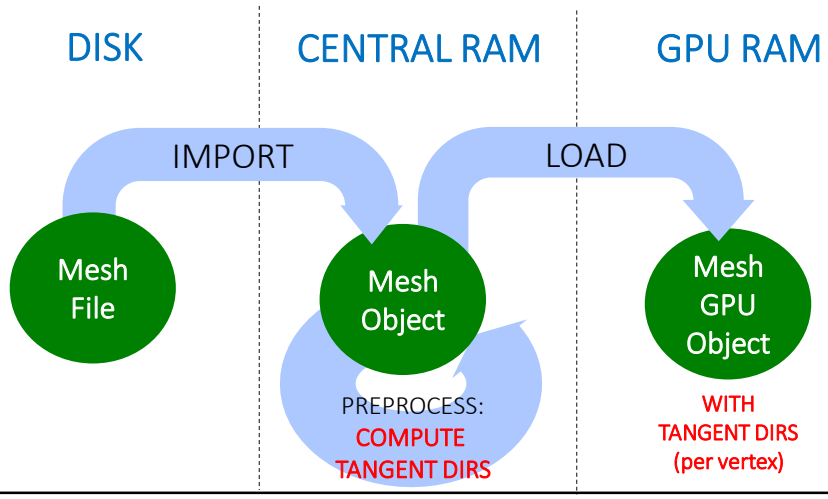
The tangent space
- Normal direction
- Tangent direction
- Bitangent direction

basically, a TS normal map specifies how to modify the per-vertex normal instead of replacing it

  - ☺ normal-map can be shared by different objects
  - ☺ non injective UV-maps can be used
    - e.g., the normal-map can be tiled
    - e.g., symmetries can be exploited
  - ☺ normal-map is independent from the mesh
    - e.g. can be constructed without knowing the mesh

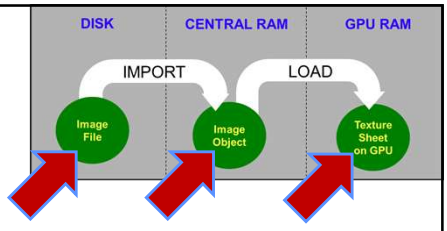
69

Tangent Dirs (Tangent and Bitangent) as per vertex attributes

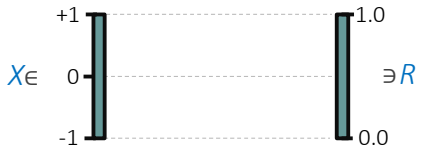


70

Normal-map: strorage



- Idea: store it as an RGB texture
  - $R \leftrightarrow X$
  - $G \leftrightarrow Y$
  - $B \leftrightarrow Z$
- but $X,Y,Z \in [-1,+1]$ and $R,G,B \in [0,+1]$ (normals are unit vectors)
  thus a linear mapping is needed:



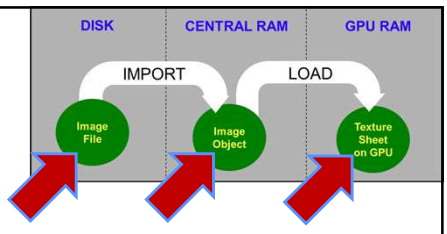
$R = ½ (X + 1)$

$X = 2 R - 1$

- Advantage: reuse compression of RGB textures/images
- Extra: store a (scalar) displacement map in 4th texture channel
- But, note: other, more efficient representations of versors exists

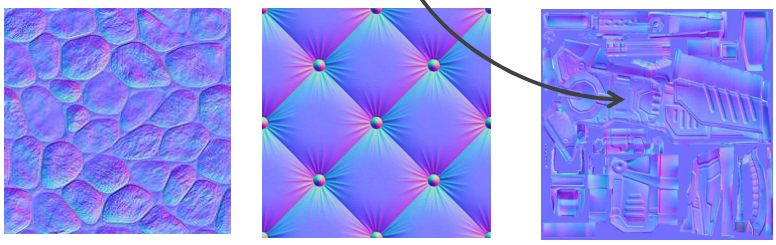
71

Normal-maps: Storage



- Examples of tangent space normal-maps

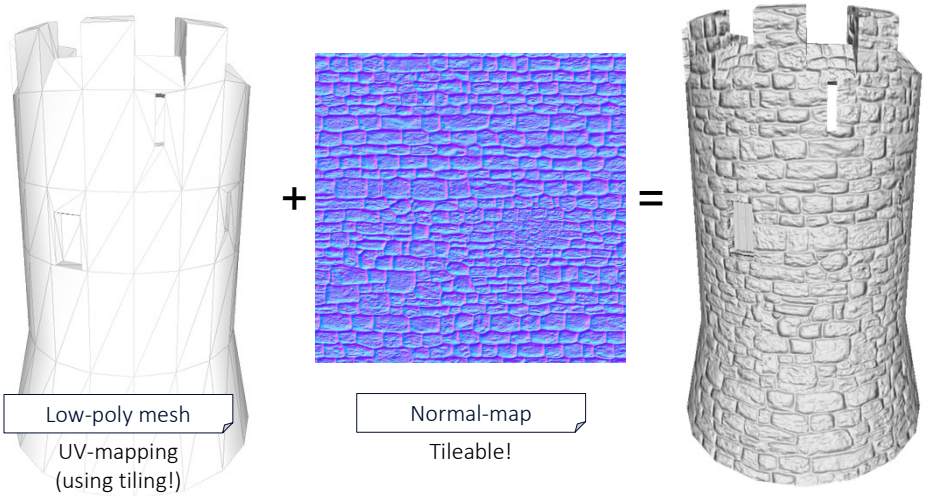
Prevailing normal : X=~0 , Y=~0 , Z=~1
⇒
Prevailing color: R =~0.5 , G=~0.5, B=~1
(~light blue)



72

Per e.g.: Tiled (tangent space) Normal Maps

not possible with object-space NM!



Low-poly mesh
UV-mapping
(using tiling!)
Tangent dirs.

Normal-map
Tileable!

assets courtesy of "Mount&Blade" (Talesworlds)

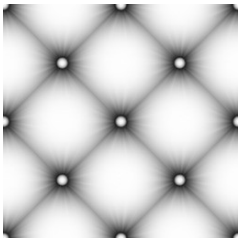
73

Bump-maps assets at a glance
(can you tell which is which?)

Tangent Space
Normal map

the default kind

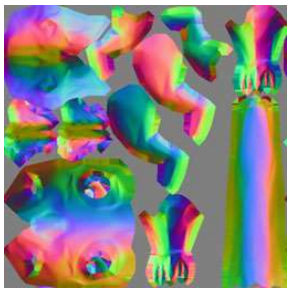
Object Space
Normal map



Displacement
Map (scalar)

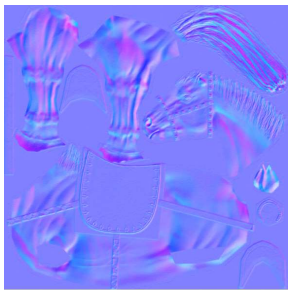
74

Observe



Object Space
Normal map

1:1 UV-map
right leg != left leg



(Tangent Space)
Normal map

UV-mapping NOT injective
Exploited symmetries!
Left side of head = right side of head

75

Normal map comparison: a summary

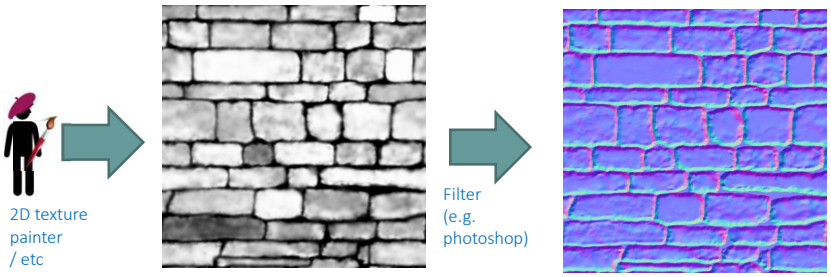
Object Space Normal map:	Tangent Space Normal map:
Replaces the normal of the objects	Modifies the normal of the objects
No normal attribute required on the mesh any more	Requires two extra attributes on the mesh: T an B versors (in addition to the normal)
Constructing the texture requires to know the mesh it will be applied to	Textures can be constructed independently from the mesh (just like a color map!)
E.g. a normal map cannot be constructed from a displacement map (w/o the mesh)	E.g. a normal map can be constructed from a displacement map
E.g. difficult to share a normal map between models	Normal maps can be shared between different models
"unwrapping" UV-maps required (unless few lucky cases)	Can be applied to non-injective UV-maps Eg: tiling, symmetry expolitation
E.g. no tiled textures. E.g. no symmetry exploitation	E.g. tiled textures ok, E.g. symmetry exploitation ok
E.g. east-wall and south-wall of a castle: different normal maps required	E.g. east wall and south wall of a castle: Same normal map possible
Looks colorful (if encoded as RGB)	Looks azure-ish (if encoded as RGB)

MUCH MORE USED IN GAMES

76

How are normal-maps obtained? (1/5) from a displacement map

see demo!



Displacement map come grayscale

Normal map

= extruded – outwards

= deep – carved in

77

How are normal-maps obtained? (1/5) from a displacement map

- Input: a scalar displacement map
  Output: a normal map
- Algorithm (basic image processing):
  - ∀ texel **t** of displacement map, compute best fitting plane around **t**
    - Consider all 3D points in a 3×3 patch surrounding **t**
    - Find plane minimizing the summed squared distance from them
    - It's a least-squares minimizatin problem
  - The normal of this plane is the normal for **t**
- Resulting normal map is expressed in tangent-space
  - By definition! (one advantage of Tangent Space Normal Maps)
  - Can be converted into Object-Space if needed (for a given UV-mapped mesh – injective maps only of course)

a texel at coords *u,v* corresponds to a 3D point (*u* , *v* , height[*u*,*v*])

or 5×5, or 7×7…

78

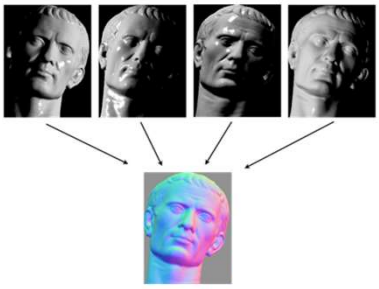
How are normal-maps obtained? (2/5) painting on 3D

- Direct painting of normal- on the model
  - (can be don, e.g., with Z-brush, Sculptris Alpha…)
  - Similar to a painting of color-maps
    - but artist paints geometric details not colors
  - Similar to mesh sculpting too
    - but, for each stroke, the system directly updates the normal on the texture-map, not the geometry on the mesh

79

How are normal-maps obtained? (3/5) captured from reality

- Captured form reality, using photos
- Example: "Photometric Stereo"
  - a form of "inverse lighting"
  - a computer vision technique
- Input: *n* real images
  - Same viewpoint
  - Different illumination
    - possibly, controlled and known
- Output: a Normal Map
  - expressed in image space
  - can be converted in object space, or in tangent space



80

How are normal-maps obtained? (3/5) captured from reality

- Normal map estimation from images
  - Traditionally, many pictures are required in input
  - Traditionally, controlled illumination is required (I must place lights in known position)
  - With Machine Learning, it's becoming possible to use a single image with natural illumination
- Idea:
  - input: a photo of a brickwall
  - output: a diffuse map + a normal map + a specular map
- It's an active area of research!

81