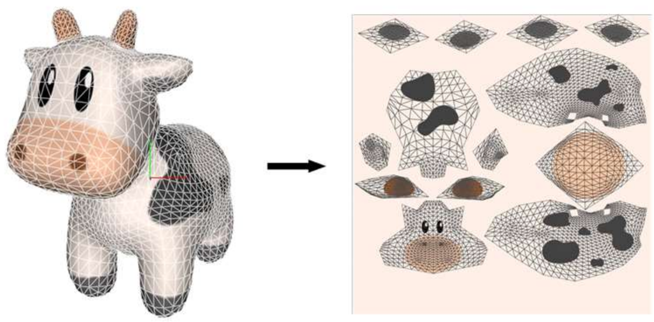
3D VideoGames

Textures in 3D Games

Marco Tarini



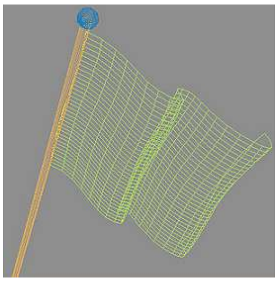
1

Course Plan

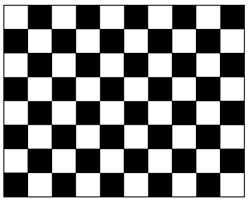
lec. 1: Introduction
lec. 2: Mathematics for 3D Games
lec. 3: Scene Graph
lec. 4: Game 3D Physics +
lec. 5: Game Particle Systems
lec. 6: Game 3D Models
lec. 7: Game Textures
lec. 8: Game 3D Animations
lec. 9: Game 3D Audio
lec. 10: Networking for 3D Games
lec. 11: Artificial Intelligence for 3D Games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

2

Texture mapping



+



=

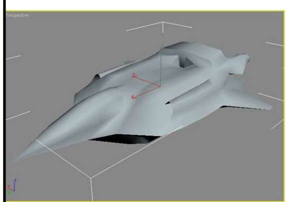
3D geometry
(set of quadrilaterals)

RGB texture 2D

(here: a color-map)

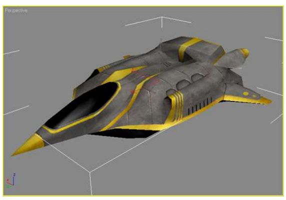
3

Example (color-map)

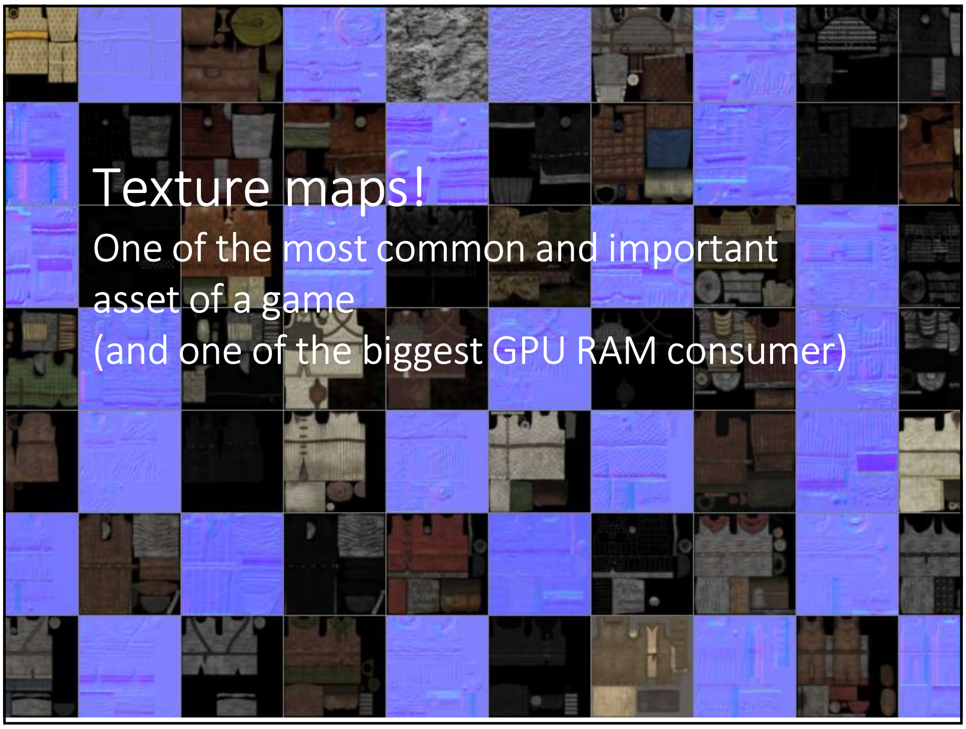


+

=



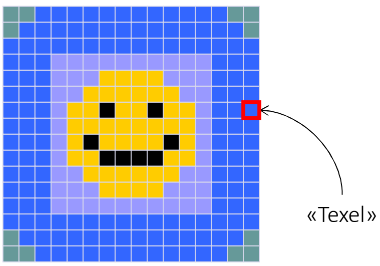
4



6

Texture maps: data structures

- In practice, a **rasterized image**



Texture sheet

7

Texutres (in games)

- Texture sheet =
  defines a signal over the mesh
  - Similar purpose to per-vertex attributes!
  - but…
    - # texels >> # vertices
    - More complex signals!

Texture: regular sampling, and dense

Attributes: irregular samplling (adaptive), and sparse

- A texel = a sample of that signal
  - Between samples: (bilinear) interpolation
- Signal sampling:
  - On a regular 2D grid (raster image)
  - At a given fixed resolution (NOT adaptive!)

8

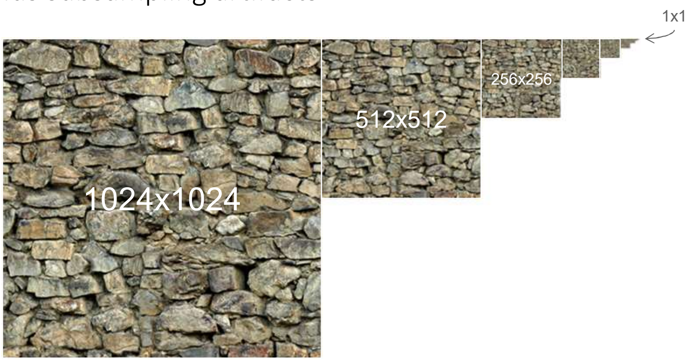
Signals stored in textures (in games)

- Each texel = a base-color (components: *r,g,b*)
  - The texture is a "diffuse-map" / "color-map" / "RGB-map"
- Each texel = a transparency factor (components: α)
  - The texture is a "alpha-map" or "cutout-texture" (exp. if 1bit)
- Each texel = a normal (versor, with components: *x,y,z*)
  - The texture is a "normal-map" or "bump-map"
- Each texel = a specular coefficient value
  - The texture is a "specular-map"
- Each texel = a glossiness value
  - The texture is a "glossiness-map"
- Each texel = a *baked* lighting value…
  - The texture is a (baked) "light-map"
- Each texel = a distance from a surface value
  - The texture is a "displacement map" or "height texture"

9

MIP map levels

- Pre-filtering of textures
- LOD pyramid, for images
- Hardware picks the right level (for each screen pixel)
- Avoids subsampling artifacts



10

Texture maps as assets: characteristics

- Size:
  - resolution
  - channels (1,2,3,4)
- MIP-map levels
  - are they present?
  - how many
- Compression?
  - e.g., color quantization (“color-map” or “palette”)
  - compression schemas designed specifically for textures such as: DXT1-5 (DirectX Texture – Microsoft)

HW imposed constraints:

- Power of 2 for side (U and V)
  - e.g.: 256x256 or 1024x512
  - not a strict requirement today today
- Hardwired upper-bound
  - today: 8K, 4K, 2K

11

Most of the visual richness perceived in the typical videogame is due to textures!

Texture resolution has a bigger impact impact on quality than Meshes resolution!

12

GPU rendering of a Mesh in a nutshell (reminder)

- Load…
  - store all data on GPU RAM
    - Geometry + Attributes
    - Connectivity
    - Textures
    - Shaders
    - Parameters / Settings
- …and Fire!
  - send the command: *"do it"* !

THE MESH (Geometry + Attributes, Connectivity)

THE "MATERIAL" (Textures, Shaders, Parameters / Settings)

13

Texture maps assets and Mesh assets

- Several texture «sheets» associated to a mesh
  - or also: more meshes on the same sheet (bene)
- Typical structure :
  - each mesh associated to a material
  - each material:
    - 1 sheet di diffuse-map
    - 1 sheet bumpmap (if needed)
    - 1 sheet di alphamap (if needed)
    - 1 vertex shaders + fragment shader
    - Several parameters
      - (e.g., shininess, …)
  - If different parts of mesh associated to different textures: decompose the object in sub-mesh

14

How is a texture mapped over a given mesh?

- 3D Models
  i.e. tri-meshes with:
  - per vertex attrib
    - normals, color, AO, …
  - LODs
  - uv-map
  - 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

17

Marco Tarini
Università degli studi di Milano

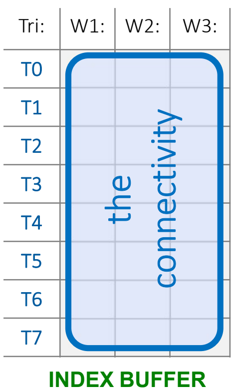
UV-Map of a mesh

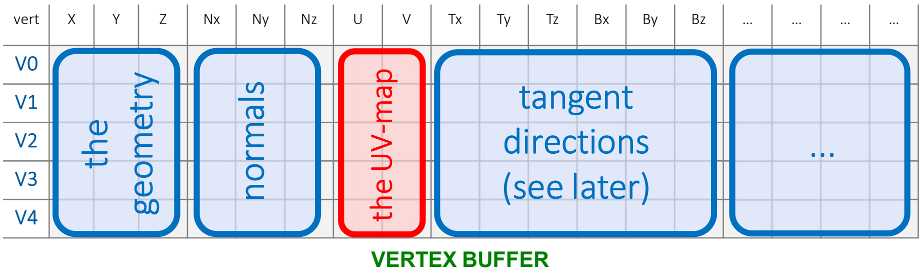
- A mapping :
  mesh surface→ 2D texture space is needed ($[0..1]^2$)
  - «parametrization» of the surface
- Store this mapping as per vertex attribute : (u,v)
  - The «u-v map» of the mesh

18

Mesh as buffers (tables in GPU ram)

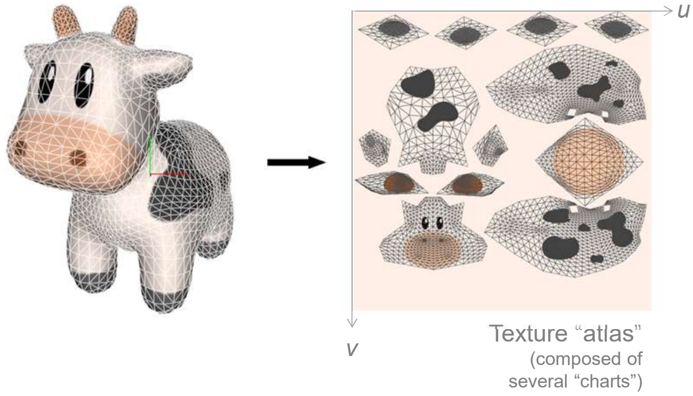
- Position
- Normal
- Color
- Texture Coordinate
- Tangent Directions
- Bone links…





19

Modeling task: u-v map construction



20

Texture space notation

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

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

21

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

Convenient!
We can reduce texture-sheet resolution (balancing quality / memory) without affecting the UV-map of the mesh.

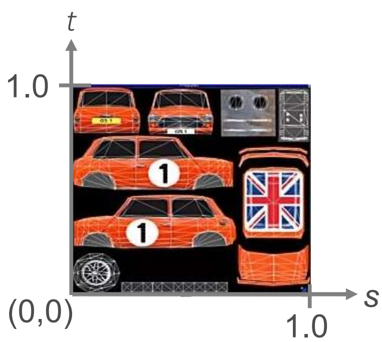
E.g.: load in GPU RAM only a few smaller MIP-map levels

22

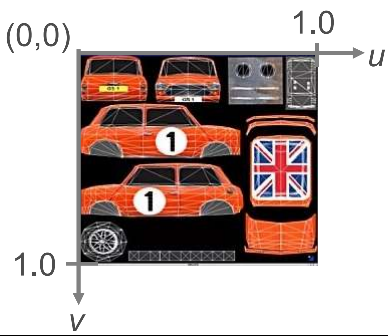
Two notations

Most used
(in game industry)

s-t
(es OpenGL)

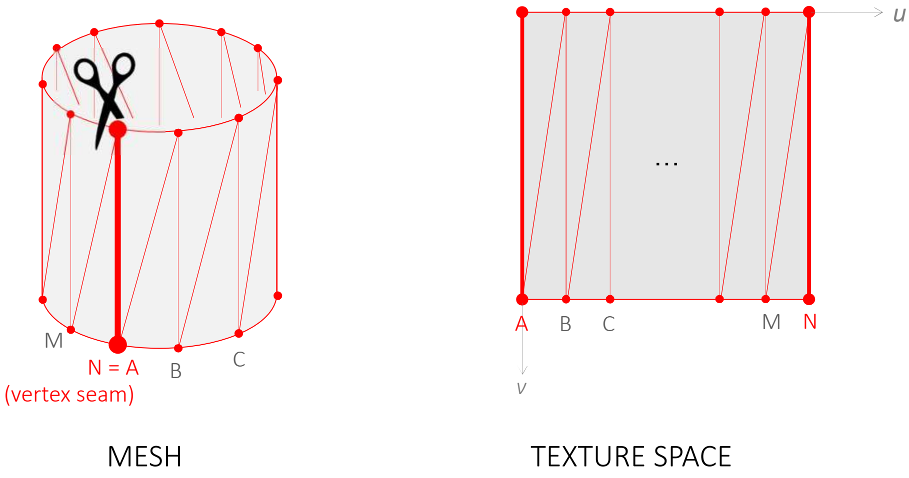


u-v
(es DirectX)



23

UV map: example



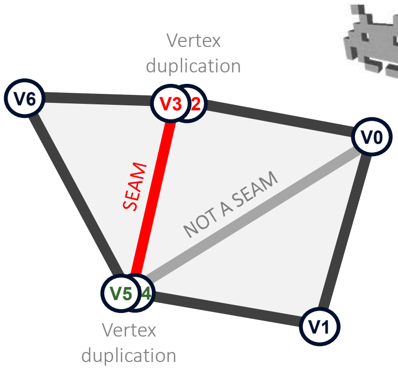
24

Texture seams
(or just texture "cuts")

- Seams are 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

26

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

- Typical task of the modeler (digital artists)
  - (semi-)automatic algorithms are 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)

28

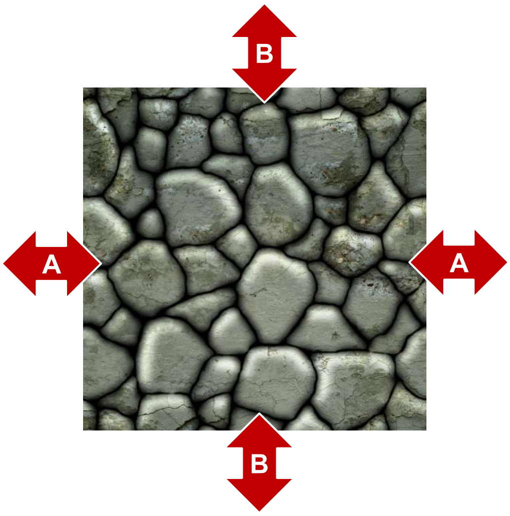
Modeling task: "u-v mapping" (verb)

DEMO!

- 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!)

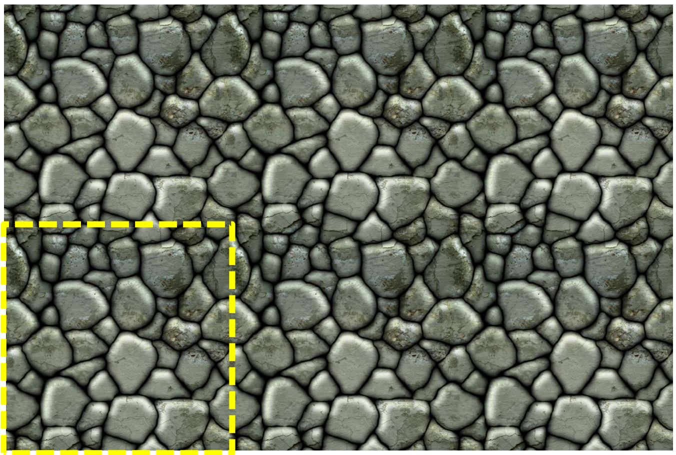
29

Tileable Textures



30

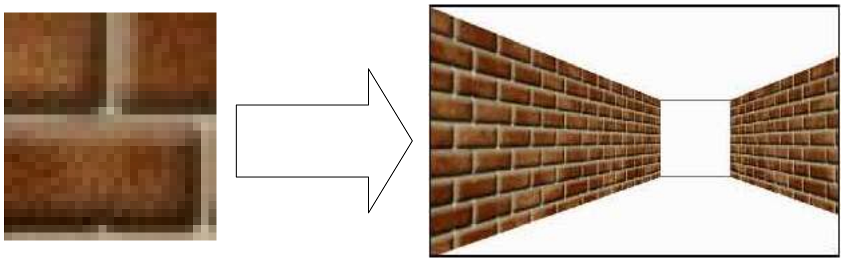
Tileable textures



32

Tileable textures

- Typical use



Very efficient in space

33

Two types of UV-maps

- NOT injective UV map
  - 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 map
  - 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: "UV-map" (the standard)

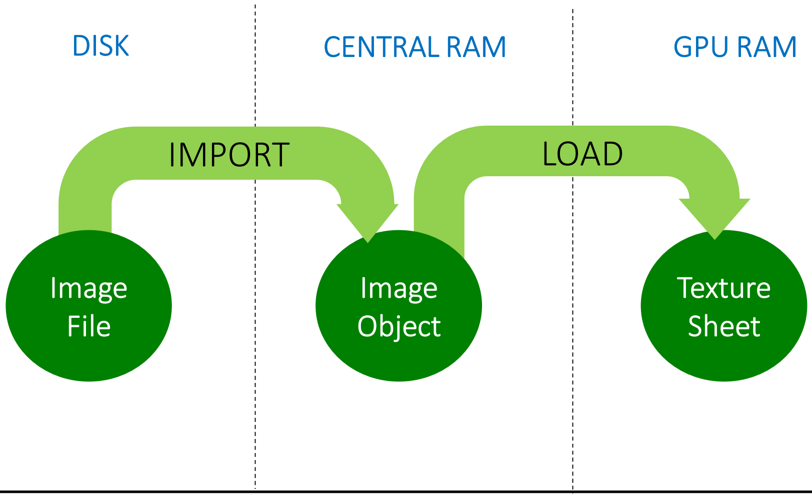
aka: "Unwrapping"
or: "Unwrapped UVs"
or: "1:1 UV-map"
or: "Lightmap" UV-map
or: "Non-overlapping" UV-map

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

34

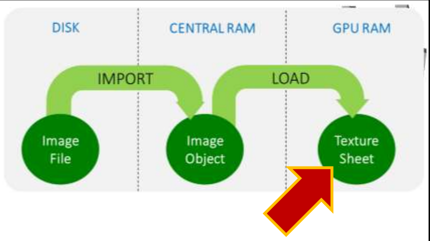
Life of a Texture in a Game Engine

the same as, basically, any 3D assets



35

Texture Sheets (in GPU RAM)



- Rasterized images, but with peculiarities ...
  - MIP-map levels
  - channels per texel: 1,2,3, or (most typically) 4
  - bits per channels:
    usually 8, fixed point
    floating textures supported
  - compression: specific texture schemas (see next)
  - resolution: powers of 2 per side

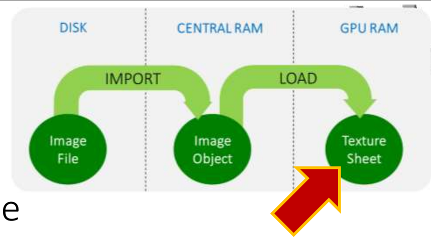
36

Per-fragment Texture fetch (during rendering, hardwired in GPU)

- Hard-wired GPU mechanisms to access the texture image at a given location: $(u, v) \rightarrow \mathbb{R}^4$ (number of channels)
- Includes many steps:
  1. Management of out-of-bound coordinates.
     E.g., repeat mode: $u \leftarrow \lfloor u \rfloor$ and $v \leftarrow \lfloor v \rfloor$
  2. De-normalization of coords, from normalized $[0..1]^2$ to texel coord $[0..res_X] \times [0..res_Y]$
  3. Selection of the appropriate MIP-map level (how?)
  4. On-the-fly decompression of compressed image data
  5. Bilinear interpolation between 4 texels, plus linear across MIP-map levels

37

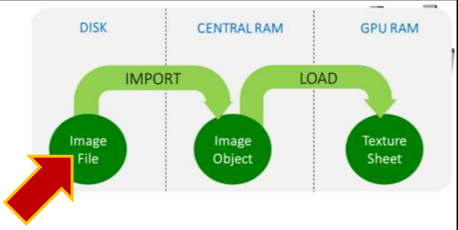
Texture compression (to save GPU RAM)



- Save RAM, but preserve the random-accessibility of texels
  - color quantization
    - e.g., 5 red 5 green 5 blue 1 alpha = 16 bits per texel
  - color-table, or "palette"
    - e.g., 256 color table for texture, an 8-bit index per texel
  - specialized image-compression schemas. They are:
    - Lossy (very much so)
    - Fixed compression rates (e.g. ¼)
    - Unfavorable compression/loss ratio ☹
    - Most diffuse scheme S3TC, with variants: DXT-1 (yes/no alphas) -2 -3 (uniform alphas) -4 -5 (smooth alphas)

38

Textures as assets: file formats



For generic images

(decompress the entire image before accessing any pixels)

- ☺ compression: excellent
- ☹ loading: heavy:
  - Decompress from RAM, (maybe) recompress in GPU-RAM
- ☹ MIP-map levels: Procedurally generated. Control by the engine
- ☺ Resolution: any (can pad on load)

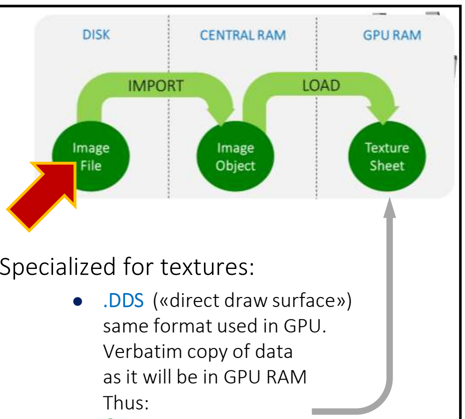
For textures

(random accessibility to texels, without uncompressing the entire image)

- ☹ compression: bad
- ☺ loading: light
  - direct streaming possible Disc => RAM => GPU RAM
- ☺ MIP-map levels: Baked. Control by the artist
- ☹ Resolution: must be a pow of 2

39

Textures as assets: file formats



For generic images:

- .JPG / .JPEG
  - ☹ lossy,
  - ☺ good compression rate
  - ☺ "photographic" images: best
  - ☹ only 3 channels (no choice)
  - ☹ 8 bit per channel (no choice)
- .PNG
  - ☺ lossless
  - ☹ compression ratio (for natural images)
  - ☺ good for artificial images (logos)
  - ☺ alpha channel: also possible
  - ☺ 16bits: possible
- .TIFF e .RAW (rare)
  - ☺ lossless
  - ☹ ☹ no compression
  - ☺ max flexibility for channels, image depth
- .PNM (rarer, but useful for toy projects)
  - ☹ ☹ ☹ compression: verbose
  - ☺ Very easy parsing! (no lib needed)

Specialized for textures:

- .DDS («direct draw surface») same format used in GPU. Verbatim copy of data as it will be in GPU RAM
  Thus:
  - ☺ includes MIPmap levels (if needed)
  - ☹ compression: very lossy And bad compression rate (and fixed)
  - ☺ GPU ready! Just read from disk & load on GPU memory (no decompress / recompress!)

40